# SBFT Tool Competition 2026 - REST League

Michele Pasqua
michele.pasqua@univr.it
Dept. of Computer Science, University of Verona
Verona, Italy

Davide Corradini
davide.corradini@uni.lu
SnT, University of Luxembourg
Luxembourg, Luxembourg

Sofia Mari
sofia.mari@univr.it
Dept. of Computer Science, University of Verona
Verona, Italy

Mariano Ceccato
mariano.ceccato@univr.it
Dept. of Computer Science, University of Verona
Verona, Italy

## Abstract

REST APIs are the mainstream mechanism to expose remote resources and functionalities, enabling diverse software systems to communicate over the Internet via a common interface. Automated test generation for REST APIs is a valuable support for quickly exploring a large number of execution scenarios (reducing manual effort) for identifying bugs, leading to more reliable and secure systems. Generating tests for REST APIs is challenging, as the source code of the system under test is typically inaccessible, due to microservices architectures, third-party libraries, or proprietary components, and APIs must be treated as black boxes.

In this context, the *REST League* competition aims to foster research community and industry engagement, formalizing REST API test generation as a vital and emerging sector within software engineering. In this first edition, REST League received three industrial tools and two research tools for evaluation. These tools have been validated on a dataset of real-world REST APIs by using RESTgym as evaluation platform. Evaluation criteria include fault detection, API operation coverage, source code coverage, and testing efficiency, automatically collected by RESTgym.

## Keywords

REST APIs, Black-box testing, Tool competition

**ACM Reference Format:**
Michele Pasqua, Davide Corradini, Sofia Mari, and Mariano Ceccato. 2026. SBFT Tool Competition 2026 - REST League. In *19th Search-Based and Fuzz Testing Workshop (SBFT '26), April 12–18, 2026, Rio de Janeiro, Brazil.* ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/3786155.3795704

## 1 Introduction

The rapid growth of the web APIs diffusion, particularly those adhering to the *REST* (REpresentational State Transfer) architectural style [12], has necessitated the development of robust automated testing solutions. Due to the complex nature of web services, automated REST API testing is predominantly *black-box*: tools verify API correctness and security by interacting solely via the HTTP interface, without accessing source code or code-derived metrics, typically leveraging only its formal documentation. Such a documentation takes the form of an *OpenAPI specification* [14], that is, a structured file (YAML or JSON) that specifies how to reach the service using a URI, which authentication schema is adopted, and the details of all the operations available in the API. Figure 1 provides an excerpt of the OpenAPI specification for a simple REST API [11]. In particular, for each API operation, are described the input parameters (and their schema) to be used in requests, and the schema of responses. This information is used by testing tools to craft HTTP request sequences with the aim of spotting faults or security holes in the API under test.

```yaml
paths:
  /users:
    get:
      operationId: get_users
      responses:
        '200':
          description: See all details of the users
          content:
            application/json:
              schema:
                type: array
                items:
                  $ref: "#/components/schemas/User"
components:
  schemas:
    User:
      type: object
      properties:
        username:
          type: string
        password:
          type: string
```

**Figure 1: OpenAPI specification example (excerpt).**

While several research and industrial tools exist for black-box testing of REST APIs, there is often a lack of standardized benchmarks and comparative frameworks to evaluate their performance objectively. *Rest League* is a tool competition aiming to bridge the previously mentioned gap by providing a common platform where different testing tools are executed against a set of benchmark REST APIs. The primary goal is to foster innovation in the field of REST API testing and to identify the strengths and weaknesses of current state-of-the-art strategies, such as nominal testing, error testing, and security-oriented fuzzing. The competition is organized with the support of the Software Engineering research group at the University of Verona[1], that also maintains state-of-the-art software related to REST API testing, such as: RestTestGen Framework [7],

[1]UniVerSE Lab: https://universe-lab.pages.dev

This work is licensed under a Creative Commons Attribution 4.0 International License.
*SBFT '26, Rio de Janeiro, Brazil*
© 2026 Copyright held by the owner/author(s).
ACM ISBN 979-8-4007-2387-2/2026/04
https://doi.org/10.1145/3786155.3795704

**Table 1: Details of the REST APIs used in the competion.**

| API | LoC | Ops | Source |
|---|---|---|---|
| *blog* | 1,188 | 52 | RESTgym [24] |
| *erc20* | 1,161 | 13 | GitHub [26] |
| *feature-service* | 1,489 | 18 | RESTgym [24] |
| *flight-search* | 11,368 | 40 | GitHub [21] |
| *gestao-hospital* | 2,652 | 20 | GitHub [25] |
| *kafka-rest-proxy* | 71,496 | 50 | GitHub [3] |
| *market* | 8,302 | 13 | RESTgym [24] |
| *notebook-manager* | 634 | 5 | GitHub [2] |
| *person-controller* | 913 | 12 | RESTgym [24] |
| *pet-clinic* | 6,930 | 35 | GitHub [22] |
| *poject-tracking-system* | 4,072 | 59 | RESTgym [24] |

to quickly develop black-box REST API testing tools; Restats [6], to collect black-box REST API testing metrics; and RESTgym [5], to orchestrate the execution of REST API testing tools.

In this first edition of the competition, we received five submissions, comprising three industry-ready tools and two research prototypes. Participants have been instructed on how to make their tool compatible with the evaluation infrastructure, following the RESTgym documentation[2]. Tools have been evaluated on a dataset of 11 REST APIs, having different level of complexity, not disclosed in advance to the participants. Competing tools have been run multiple times on each API in order to account for non-deterministic behaviors. Tools have been evaluated on three performance dimensions: fault detection, testing efficiency, and testing effectiveness.

## 2 Experimental Setting

To ensure a fair and comprehensive comparison, Rest League employs a standardized evaluation infrastructure, leveraging components like RESTgym [5] for environment management and Restats [6] for metric computation. The evaluation setting is characterized by the following components.

### 2.1 Benchmark APIs

Tools have been evaluated against a suite of diverse REST APIs, consisting in 11 case studies with a total of 317 operations. These comprise 5 APIs from RESTgym, already considered in the literature, and 6 APIs taken from GitHub, never used in previous literature work. Case studies vary in complexity, domain, and the quality of their OpenAPI specifications. Details of the APIs used in the competition are reported in Table 1, which shows the lines of source code and the number of operations documented for each API[3].

### 2.2 Evaluation Metrics

The performance of the tools has been measured using several quantitative metrics: *Source Code Coverage*, to determine how much of the API's internal logic was exercised; *Specification Coverage*, to determine how much of the API's documentation (i.e., its OpenAPI specification) was exercised; and *Fault Detection*, to determine the number of unique bugs identified. Specifically, we collected the following metrics to assess the *effectiveness* of the tools.

**Brach Coverage** Percentage of source code branches covered.
**Line Coverage** Percentage of source code lines covered.
**Method Coverage** Percentage of source code methods covered.
**Operations Covered** Absolute value of operations listed in the specification successfully tested with `2XX` status code [17].
**Fault Detection** Absolute value of unique server-side failures triggered, that is, test interactions returning `5XX` status code having unique error message[4].

Additionally, we also assessed the efficiency of the tools by measuring the evolution over time of the previous metrics. Tools achieving high effectiveness in less time are considered more efficient. Hence, we also collected the following *efficiency* metrics.

**Brach Coverage AUC** The area under curve of Branch Coverage evolution over one hour time budget.
**Line Coverage AUC** The area under curve of Line Coverage evolution over one hour time budget.
**Method Coverage AUC** The area under curve of Method Coverage evolution over one hour time budget.
**Operations Covered AUC** The area under curve of Operations Covered evolution over one hour time budget.
**Fault Detection AUC** The area under curve of Fault Detection evolution over one hour time budget.

Effectiveness metrics are automatically collected by Restats, embedded in the RESTgym evaluation infrastructure, while efficiency metrics have been computed in a posteriori analysis.

### 2.3 Experimental Procedure

Tools have been executed within Docker containers to ensure isolation and reproducibility. Orchestration of the tool execution and metrics collection have been automatically handled by RESTgym. Each tool has been allocated a fixed time budget of one hour per API to generate and execute test cases. To mitigate bias from non-deterministic tool behaviors, all testing sessions have been executed ten times, taking the average results. Tools and APIs in the competition have been run on a 128-core machine with 386GB of RAM, enabling parallel testing sessions. To each tool container, 8 cores and 16GB of RAM have been reserved. The machine used in the experiments is equipped with an Nvidia RTX 4090 video card.

### 2.4 Competing Tools

To participate in the competition, tools had to satisfy some minimal requirements, detailed in the call for tools. Specifically, they must operate in a black-box fashion, interacting with the APIs solely via HTTP requests based on the provided OpenAPI specification; and, they must be compatible with RESTgym, the evaluation infrastructure adopted in the competition. We did not impose constraints on the use of small and large language models for this run of the competition. All submissions to REST League fulfilled these requirements, resulting in the following five tools (listed in alphabetic order).

[2]RESTgym: https://github.com/SeUniVr/RESTgym
[3]Lines of code have been computed by using the online tool https://ghloc.vercel.app on the APIs' GitHub repository.
[4]An error message is considered unique when it is sufficiently different from other messages, as previously defined in the literature [4, 15].

**AutoRestTest** An open-source tool that integrates the Semantic Property Dependency Graph (SPDG) with Multi-Agent Reinforcement Learning (MARL) and large language models (LLMs) for effective REST API testing [23].

**CATS** An open-source tool for automated fuzzing and negative testing of REST APIs combining a large catalogue fuzzers with a self-healing engine that adapts test cases to changes in the API specification [10].

**EvoMaster** An an open-source, search-based fuzzer aimed at Web APIs, including REST, GraphQL and RPC APIs [1].

**RESTest** An open-source model-based testing framework that supports black-box test case generation for REST APIs [18].

**Schemathesis** An open-source tool for automated black-box testing of REST APIs using property-based testing and stateful exploration [13].

We added to the competition **ResTestGen**, an open-source tool that uses an Operation Dependency Graph to prioritize API operations calls based on data dependencies to test REST APIs [8]. This tool was developed by the Software Engineering research group at the University of Verona, and is already available in RESTgym. RestTestGen has been added to serve as a baseline only, and thus excluded from the competition results.

## 3 Empirical Results

The competition uses a system of digital badges to represent a tool's achievements across different testing dimensions. Specifically, we individuated three challenges: *Fault Detection Challenge*, awarding tools showing higher detection of server-side failures; *Effectiveness Challenge*, awarding tools showing higher effectiveness (in terms of source code and specification coverage); and *Efficiency Challenge*, awarding tools showing higher balance between effectiveness and testing time. This results in the following badges.

- **`Bug Hunter`** badge, given to the tool achieving the best score in the Fault Detection Challenge
- **`Roadrunner`** badge, given to the tool achieving the best score in the Efficiency Challenge
- **`Gold API Tester`** badge, given to the tool achieving the best score in the Effectiveness Challenge
- **`Silver API Tester`** badge, given to the tool achieving the second-best score in the Effectiveness Challenge

The tool earning the **`Gold API Tester`** badge is also the ultimate champion of the competition. To assign badges, we developed a scoring system based on the collected testing metrics. The score for the Fault Detection Challenge is based on the Fault Detection metric, averaged on all APIs and uniformly scaled using Z-Score normalization [16]. The score for the Efficiency Challenge is based on the Fault Detection AUC, Operations Covered AUC, and Branch Coverage AUC metrics, averaged on all APIs and uniformly scaled using Z-Score normalization. The score of a tool is the unweighted sum of these normalized metrics. Finally, the score for the Effectiveness Challenge is based on the Fault Detection, Operations Covered, and Branch Coverage metrics, averaged on all APIs and uniformly scaled using Z-Score normalization. The score for a tool is the unweighted sum of these normalized metrics.

**Table 2: Fault Detection Challenge results and scores.**

| Tool | Fault Detection | Score | Rank |
|---|---|---|---|
| AutoRestTest | 67.09 | 70.90 | 1 |
| Schemathesis | 24.65 | 49.71 | 2 |
| RestTestGen | 22.90 | 48.83 | – baseline |
| CATS | 21.21 | 47.99 | 3 |
| RESTest | 11.25 | 43.02 | 4 |
| EvoMaster | 4.31 | 39.55 | 5 |

**Table 3: Efficiency Challenge results and scores.**

| Tool | Fault Detection AUC | Operations Covered AUC | Branch Coverage AUC | Score | Rank |
|---|---|---|---|---|---|
| AutoRestTest | 151,490.00 | 60224.11 | 744.41 | 61.33 | 1 |
| RestTestGen | 57,785.23 | 56,590.65 | 780.10 | 55.07 | – baseline |
| EvoMaster | 14,212.70 | 58,307.26 | 671.54 | 50.85 | 2 |
| Schemathesis | 58,822.56 | 34,539.22 | 707.13 | 50.60 | 3 |
| CATS | 66,760.82 | 27,848.08 | 385.96 | 45.33 | 4 |
| RESTest | 17,972.24 | 8,322.85 | 236.66 | 36.82 | 5 |

**Table 4: Effectiveness Challenge results and scores.**

| Tool | Fault Detection | Operations Covered | Branch Coverage | Score | Rank |
|---|---|---|---|---|---|
| AutoRestTest | 67.09 | 17.27 | 21% | 61.54 | 1 |
| RestTestGen | 22.90 | 16.33 | 23% | 55.50 | – baseline |
| EvoMaster | 4.31 | 16.73 | 19% | 50.14 | 2 |
| Schemathesis | 24.65 | 10.61 | 20% | 48.48 | 3 |
| RESTest | 11.25 | 10.89 | 18% | 44.75 | 4 |
| CATS | 21.21 | 9.56 | 12% | 39.59 | 5 |

### 3.1 Final Scores

The results and the final scores of the competition are reported in Table 2, Table 3, and Table 4. For presentation purposes, the scores have been shifted to a positive-only interval. Table 2 reports the results for the Fault Detection Challenge, awarding AutoRestTest with the **`Bug Hunter`** badge, indicating its superior capability in triggering server-side failures. Table 3 reports the results for the Efficiency Challenge, awarding AutoRestTest with the **`Roadrunner`** badge, indicating that its generation strategy explores the API test space in less time than the competitors. Finally, Table 4 reports the results for the Effectiveness Challenge, awarding AutoRestTest with the **`Gold API Tester`** badge, indicating its superior capability in generating test cases that more extensively explore the API behavior. In the same challenge, EvoMaster achieved the **`Silver API Tester`** badge, as the tool with the second-best testing performance.

*Remark on language models usage.* Among the competing tools, only one tool, RESTest, use a small language model during testing, running locally on the benchmarking machine, and only one tool, AutoRestTest, uses a large language model remotely accessed. Specifically, RESTtest employs a locally deployed Llama 3.2 3B model [20], while AutoRestTest employs a Gemini 2.5 Flash-Lite model [9] proxied via OpenRouter[5].

[5] OpenRouter: https://openrouter.ai

## 3.2 Success Rate

The competing tools comprise three industry-ready tools (CATS, EvoMaster, and Schemathesis) and two research prototypes (AutoRestTest, and RESTest), with the baseline tool (RestTestGen) which is also a research prototype. During the evaluation, we noted that all tools exhibit an high success rate. Indeed, all tools managed to test all APIs in the dataset except one (that varies from tool to tool), achieving a remarkable success rate of 91%.

In detail, AutoRestTest was not able to test the *flight-service* API due to a crash of the tool related to an error during JSON decoding. CATS stopped sending requests after a couple of minutes when testing the *erc20* API. EvoMaster made the *feature-service* API container to crash (the crash only happens with this tool). RESTest was not able to test the *kafka-rest-proxy* and *flight-search* APIs due to a crash of the tool related to an error concerning a not supported object parameter in the query or path. Schemathesis was not able to test the *pet-clinic* API due to a crash of the tool related to a failure when parsing a pattern from the API specification. RestTestGen made the *kafka-rest-proxy* API container to crash (the crash only happens with this tool). All failures happened in all ten repetitions of the testing sessions, thus excluding non-deterministic bias.

# 4 Final Remarks

Rest League represents a significant step toward standardizing the assessment of REST API testing tools. By providing a transparent and rigorous evaluation framework, it encourages the academic and industrial communities to refine their automated testing strategies. The insights gained from the competition not only highlight the current state-of-the-art but also point toward future research directions, such as improved handling of inter-parameter dependencies and more effective stateful testing of complex REST API workflows. Indeed, while effective in triggering server-side failures, the relatively low code coverage across all tools (none exceeded 23%) recorded during the competition suggests that black-box testing still struggles to exercise deep business logic. Moreover, the competition results show that the only tool leveraging a large language model outperforms the other testing tools in all evaluation dimensions. This suggests that the semantic reasoning capabilities of large language models are currently outperforming traditional search-based or property-based fuzzing techniques, at least in the black-box setting. Finally, despite the high success rates (91% overall), the competition revealed that even state-of-the-art and industry-ready tools are prone to failures when faced with complex APIs.

*Data availability.* All experiment logs and detailed results are available in the competition replication package [19].

# Acknowledgments

We thank the participants for their invaluable contributions and feedback, which support the evolution and maturity of automated testing strategies for REST APIs. Davide Corradini was supported by the Luxembourg National Research Fund under the Industrial Partnership Block Grant (IPBG), ref. IPBG19/14016225/INSTRUCT.

# References

[1] Andrea Arcuri. 2019. RESTful API Automated Test Case Generation with EvoMaster. *ACM Trans. Softw. Eng. Methodol.* 28, 1, Article 3 (2019), 37 pages. doi:10.1145/3293455

[2] bird-developer. 2026. *A Real World Example for Open API Restful Spring boot Application + Swagger + MySQL + Docker.* https://github.com/birddevelper/NoteBookManager

[3] confluentic. 2026. *Confluent REST Proxy for Kafka.* https://github.com/confluentinc/kafka-rest

[4] Davide Corradini, Zeno Montolli, Michele Pasqua, and Mariano Ceccato. 2024. DeepREST: Automated Test Case Generation for REST APIs Exploiting Deep Reinforcement Learning. In *Proceedings of the 39th IEEE/ACM International Conference on Automated Software Engineering (ASE '24).* Association for Computing Machinery, New York, NY, USA, 1383–1394. doi:10.1145/3691620.3695511

[5] Davide Corradini, Michele Pasqua, and Mariano Ceccato. 2025. RESTgym: A Flexible Infrastructure for Empirical Assessment of Automated REST API Testing Tools. In *2025 IEEE Conference on Software Testing, Verification and Validation (ICST).* 757–761. doi:10.1109/ICST62969.2025.10988956

[6] Davide Corradini, Amedeo Zampieri, Michele Pasqua, and Mariano Ceccato. 2021. Restats: A Test Coverage Tool for RESTful APIs. In *2021 IEEE International Conference on Software Maintenance and Evolution (ICSME).* 594–598. doi:10.1109/ICSME52107.2021.00063

[7] Davide Corradini, Amedeo Zampieri, Michele Pasqua, and Mariano Ceccato. 2022. RestTestGen: An Extensible Framework for Automated Black-box Testing of RESTful APIs. In *2022 IEEE International Conference on Software Maintenance and Evolution (ICSME).* IEEE, 504–508.

[8] Davide Corradini, Amedeo Zampieri, Michele Pasqua, Emanuele Viglianisi, Michael Dallago, and Mariano Ceccato. 2022. Automated black-box testing of nominal and error scenarios in RESTful APIs. *Software Testing, Verification and Reliability* 32, 5 (2022), e1808. doi:10.1002/stvr.1808

[9] DeepMind. 2026. *Introducing 2.5 Flash-Lite, a thinking model for those looking for low cost and latency.* https://deepmind.google/models/gemini/flash-lite

[10] Endava. 2026. *CATS: REST API fuzzer and negative testing tool.* https://endava.github.io/cats

[11] erev0s. 2026. *Vulnerable REST API with OWASP top 10 vulnerabilities for security testing.* https://github.com/erev0s/VAmPI

[12] Roy Thomas Fielding. 2000. *Architectural styles and the design of network-based software architectures.* University of California, Irvine.

[13] Zac Hatfield-Dodds and Dmitry Dygalo. 2022. Deriving semantics-aware fuzzers from web API schemas. In *Proceedings of the ACM/IEEE 44th International Conference on Software Engineering: Companion Proceedings (ICSE '22).* Ass. for Computing Machinery, New York, NY, USA, 345–346. doi:10.1145/3510454.3528637

[14] OpenAPI Initiative. 2025. *OpenAPI Specifications.* https://spec.openapis.org/oas

[15] Myeongsoo Kim, Saurabh Sinha, and Alessandro Orso. 2024. Adaptive REST API Testing with Reinforcement Learning. In *Proceedings of the 38th IEEE/ACM International Conference on Automated Software Engineering (ASE '23).* IEEE Press, 446–458. doi:10.1109/ASE56229.2023.00218

[16] Erwin Kreyszig, Herbert Kreyszig, and E. J. Norminton. 2011. *Advanced Engineering Mathematics* (tenth ed.). Wiley, Hoboken, NJ.

[17] Alberto Martin-Lopez, Sergio Segura, and Antonio Ruiz-Cortés. 2019. Test coverage criteria for RESTful web APIs. In *Proceedings of the 10th ACM SIGSOFT International Workshop on Automating TEST Case Design, Selection, and Evaluation (A-TEST 2019).* Association for Computing Machinery, New York, NY, USA, 15–21. doi:10.1145/3340433.3342822

[18] Alberto Martin-Lopez, Sergio Segura, and Antonio Ruiz-Cortés. 2021. RESTest: automated black-box testing of RESTful web APIs *(ISSTA '21).* Association for Computing Machinery, New York, NY, USA, 682–685. doi:10.1145/3460319.3469082

[19] Michele Pasqua, Davide Corradini, Sofia Mari, and Mariano Ceccato. 2026. *Replication package for the SBFT Tool Competition 2026 - REST League.* https://doi.org/10.5281/zenodo.18434145

[20] Meta Platforms. 2026. *Llama 3.2: Revolutionizing edge AI and vision with open, customizable models.* https://ai.meta.com/blog/llama-3-2-connect-2024-vision-edge-mobile-devices

[21] Rapter1990. 2026. *Case Study - Flight Search Api (Java 21, Spring Boot, Mongodb, JUnit, Spring Security, JWT, Docker, AOP, Kubernetes, Prometheus, Grafana, Github Actions (CI/CD), Jenkins).* https://github.com/Rapter1990/flightsearchapi

[22] spring-petclinic. 2026. *REST version of the Spring Petclinic sample application.* https://github.com/spring-petclinic/spring-petclinic-rest

[23] Tyler Stennett, Myeongsoo Kim, Saurabh Sinha, and Alessandro Orso. 2025. AutoRestTest: A Tool for Automated REST API Testing Using LLMs and MARL. In *Proceedings of the IEEE/ACM 47th International Conference on Software Engineering: Companion Proceedings* (Ottawa, Ontario, Canada) *(ICSE '25).* IEEE Press, 21–24. doi:10.1109/ICSE-Companion66252.2025.00015

[24] UniVerSE Lab. 2026. *RESTgym benchmarking infrastructure.* https://github.com/SeUniVr/RESTgym

[25] ValchanOficial. 2026. *Code:Nation - Sistema de Gestão Hospitalar.* https://github.com/ValchanOficial/GestaoHospital

[26] web3labs. 2026. *ERC-20 token standard RESTful service using Spring Boot and web3j.* https://github.com/web3labs/erc20-rest-service