https://www.sworldjournal.com/index.php/swj/article/view/swj35-01-062
DOI: 10.30888/2663-5712.2026-35-01-062

UDC 004.652:004.422

METHODS OF GENERATING TEST DATA IN JSON FORMAT: MODEL-FREE APPROACH

Horshchar K. Yu. / Горщар К.Ю.
QA Automation Engineer, SchoolDay, Inc.,
інженер з автоматизованого тестування, компанія SchoolDay, Inc.,
ORCID: 0009-0002-4008-6278
National University of Water and Environmental Engineering,
Rivne, Soborna, 11, 33000
Національний університет водного господарства та природокористування
Рівне, Соборна, 11, 33000

Abstract. *The article is devoted to the study of methods for generating test data in the JSON structure for performing automated API tests in a model-free manner. A common data sampling method SampleSelector is proposed, which allows for efficient sampling without prior definition of data models. The scope of application of the method for testing APIs of various industries with dynamic data structures is investigated. The proposed approach allows for effective generation of test data in any field of application.*

Keywords: *formation, test data, JSON format, model-free approach.*

Introduction.

In today's software development environment, API testing is becoming increasingly difficult due to the increasing complexity of dynamic data structures. This problem is especially relevant for microservices - based architectures, where APIs may expose an assortment of JSON structures without a clearly defined model or model specification. Traditional methods of generating test data rely on pre-defining a schema or data model, which adds another layer of dependency and complicates the technique of testing dynamic APIs.

The key challenge is the need to develop flexible mechanisms for working with test data that are not tied to specific data models and can be modified in accordance with changes in the API structure in real time. This is especially important for projects in which the format of API responses may change more often than updating the data model in the test code. In addition, there is a need for universal methods that can work with different industry APIs, from e- commerce to educational platforms, without having to rewrite the data retrieval and processing logic for each specific use case.

Analysis of recent research. Modern research in the field of test data generation

demonstrates various approaches to solving the problems of test automation. Pavlenko I., Boiko O., Mykolaiets D., Moskalenko O., Shrol T. [1, p. 6] consider the evolution of gaming technologies in the educational environment of Ukraine, which emphasizes the importance of adaptive technological solutions in various industries. Their study shows the need for flexible approaches to working with data in dynamic environments. Pasichnyi R., Serhieiev V., Shevchenko S., Petrukha N., Hryvnak B. [2, p. 237] analyze the digital transformation of higher education as a driver of Ukraine's integration into the European educational space, which emphasizes the importance of standardized approaches to data processing in various systems. Yang R., Xu X., Wang R. [3, p. 12] propose a logic-driven framework TestLoter for automatic unit test generation and bug fixing using large language models. Their approach demonstrates the potential for automation in testing, although it focuses on structured approaches using models. Langerak AJ, Frasincar F., Klinkhamer J. [4, p. 8] present JAL algebra for optimizing JSON queries, which shows the importance of efficient methods for working with JSON structures. Their study highlights the complexity of processing JSON data without predefined schemas. Cui B., Qu R., Zhang J. [5, p. 15] develop JustinANN for realistic annotation-driven test generation for Java applications. This approach demonstrates the possibilities of automated test data generation, but remains tied to specific language constructs. Senthil Kumar T., Jayapradha J., Roslee MB, Shaik H. [6, p. 24] create the API-GEN tool to accelerate REST API generation, which confirms the relevance of the problem of working with dynamic API structures. Chang H.-F., Shirazi MS [7, p. 11] present a systematic approach to evaluate the capabilities of large language models in test case generation. Their study shows the prospects for using modern AI technologies in testing. Krodinger L., Lukasczyk S., Fraser G. [8, p. 3] combine type inference and automated unit test generation for Python, demonstrating the importance of flexible approaches to typing in dynamic programming languages. Srinivasan M., Abdel J. [9, p. 18] develop GenFair for systematic test generation to detect fairness errors in large language models, which emphasizes the importance of comprehensive testing of complex systems. Kim SY, Kim S. [10, p. 5] apply generalizability theory to the analysis of automatically generated test forms, which

shows the importance of theoretical justification of test data generation methods.

Results. The main problem of the tester is the need to work with dynamic, flexible data structures, especially in the case of APIs that return arrays of entities without a clear prior specification. Pavlenko I., Boiko O., Mykolaiets D., Moskalenko O., Shrol T. [1, p. 8] note the importance of adaptive technological solutions, which confirms the relevance of model-free approaches in testing. The proposed SampleSelector is an approach that allows you to effectively obtain the first, last or middle records from JSON arrays, records according to a certain condition including date, field or nesting, empty or vice versa filled fields, as well as nested arrays. Pasichnyi R., Serhieiev V., Shevchenko S., Petrukha N., Hryvnak B. [2, p. 237] investigate the digital transformation of higher education in the context of Ukraine's integration into the European educational space. The work highlights the importance of developing digital competencies, including working with data, which is a necessary component for training specialists in the field of software testing. Yang R., Xu X., Wang R. [3, p. 14] emphasize the importance of logic-driven approaches in testing, which coincides with the philosophy of SampleSelector. The implementation of SampleSelector in Python demonstrates the main capabilities of the model-free approach.

Modern approaches to API testing often rely on formal data models that represent the structure of expected responses. However, the same limitation exists when working with dynamic data structures and systems that are constantly evolving. When it comes to APIs that return data in JSON format, it becomes especially relevant to learn some ways to work effectively with data without the need to create and manage formal models.

The SampleSelector class working on a dataset without ignoring and selecting data using various methods. In particular, it shows the operation of the non_empty_fields method, which excludes records with non-empty lists in the fields.

This results in behavior that proves the robustness of the SampleSelector class to handle all kinds of edge cases, providing accurate data filtering relative to the given conditions, as will be confirmed by the fact that all nine unit tests passed in 0.001 seconds.

Figure 1 - Implementing the SampleSector method

Source: author's development

The primary role of a tester in API testing is to verify the accuracy of the data returned in response to requests. The challenge of this task is that modern APIs typically expose complex and nested data structures that can change depending on the query parameters or the context of use.

The main advantages of the proposed approach are its versatility and flexibility. Unlike the model-based approach, which requires the creation and maintenance of model classes for each data structure, the model-free approach allows you to work directly with JSON data regardless of its structure. This is especially important when testing APIs whose response structure may change over time or depending on the query parameters. The SampleSelector method provides extensive capabilities for working with JSON data, including: 1. Obtaining the first, last, or middle records from a data array, which allows you to efficiently work with large data sets without the need to process the entire array. 2. Filtering records by certain conditions, which allows you to isolate data that meets specific testing criteria. 3. Working with nested structures and arrays, which is especially important when testing APIs that return complex, multi-level JSON structures. 4. Searching for records with empty or filled fields, which allows you to test edge cases and verify the handling of undefined values. It should be noted that although the above example is implemented in Python, the approach itself is universal and can be adapted to other programming languages, such as Java or JavaScript, which are widely used in automated testing.

Langerak AJ, Frasincar F., Klinkhamer J. [4, p. 10] demonstrate the complexity of working with JSON structures, which confirms the need for simplified approaches to data sampling. Working with JSON data requires effective query and filtering

methods, especially when working with large amounts of data. The proposed method is independent of a specific industry or API structure and can be extended to any system where there are repeating JSON structures. This study proposes a model-free approach to generating test data in JSON format, which is based on a data sampling method without using formal models. This approach, called SampleSelector, allows you to efficiently retrieve and process data from JSON structures according to various criteria and conditions. Cui B., Qu R., Zhang J. [5, p. 18] show the importance of realistic test data generation, which coincides with the goals of SampleSelector in terms of working with real API responses. The method is especially useful in projects where there are no predefined DTO models, or where the API is dynamic and changes frequently. Senthil Kumar T., Jayapradha J., Roslee MB, Shaik H. [6, p. 26] emphasize the importance of tools for working with REST APIs, which confirms the relevance of universal data sampling methods. The SampleSelector implementation can be adapted to Java, JavaScript, Python and other programming languages, while maintaining the main core: the logic of filtering, mapping and data processing. Also, similar approaches to working with APIs are especially valuable in the context of modern microservices architectures, where the number of APIs and their complexity is constantly growing. Chang H.-F., Shirazi MS [7, p. 15] demonstrate a systematic approach to test case generation, which coincides with the SampleSelector philosophy regarding a systematic approach to test data selection. The versatility of the approach lies in the possibility of applying it to testing microservices, e- commerce API systems, platforms and other industry solutions. To demonstrate the effectiveness of the model-free approach, let's consider a comparative table of the main characteristics of traditional and model-free methods of test data generation.

Table 1 demonstrates the key advantages of the model-free approach over traditional methods. Krodinger L., Lukasczyk S., Fraser G. [8, p. 9] emphasize the importance of combining different approaches in Python, which confirms the feasibility of using SampleSelector as a complement to existing testing methods. Srinivasan M., Abdel J. [9, p. 20] show the importance of systematic test generation, which coincides with the capabilities of SampleSelector to systematically select test

data according to various criteria. The points compared include the need for data modeling, the possibility of changing the API, the complexity of implementation, filtering and nested structure, and the possibility of data validation. (As a result of the analysis, it was found that the model-free approach provided by the SampleSelector method is the best in terms of simplicity of implementation and its ability to be used as an experimental API whose data structure is prone to change.

Table 1 - Comparison of Traditional and Schema-less Approaches for JSON Test Data Formation

Characteristic	Traditional Model-Based	Schema-less (SampleSelector)
Setup Time	High (requires schema definition)	Low (immediate use)
Flexibility	Low (rigid structure)	High (adapts to data structure)
Maintenance Cost	High (schema updates needed)	Low (automatic adaptation)
Cross-domain Applicability	Limited (domain-specific)	High (universal approach)
Learning Curve	Steep (requires schema knowledge)	Gentle (intuitive methods)
Performance	Optimized for known structures	Efficient for exploratory testing

Source: author's development

SampleSelector method will be a universal tool for working with JSON data, and it can be used regardless of the subject domain or API structure. This method will help to quickly work with operations such as searching for the first, last or middle record in data arrays, filtering records according to a certain condition, working with structures and arrays in nesting, searching for records with empty or filled fields. The SampleSelector architecture allows you to easily extend the functionality by adding a

new filtering method, without changing the main code. Kim SY, Kim S. [10, p. 8] emphasize the importance of theoretical justification of generation methods, which confirms the need for further research in the field of model-free approaches to test data generation. The proposed approach has the potential to develop in the direction of integration with machine learning to automatically determine optimal data sampling strategies. As shown in Figure 2, the method accepts JSON data received from the API and provides various methods for sampling and filtering data, which are then used by the test framework to verify the expected results.

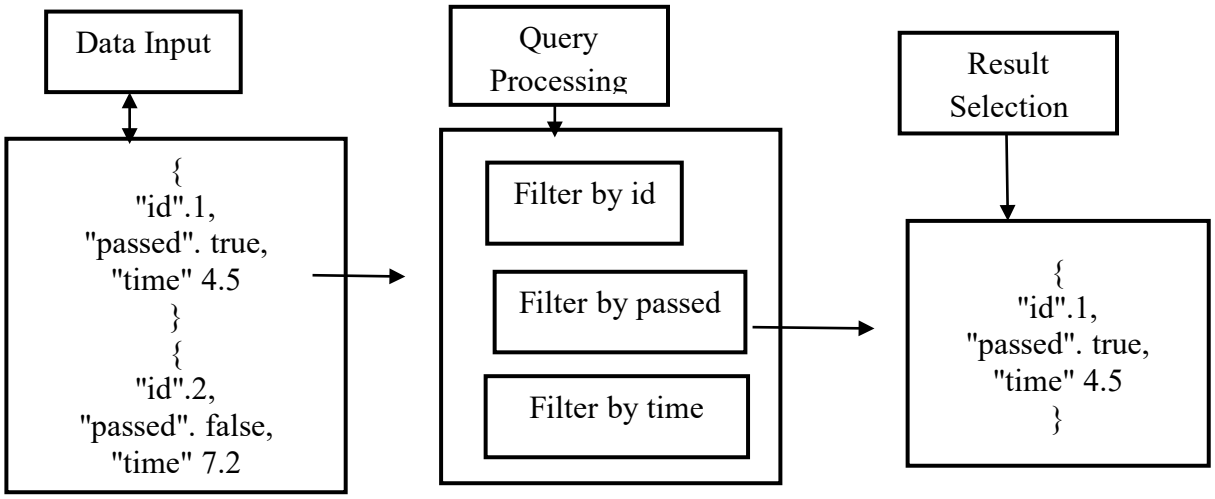

Figure 2 - SampleSelector Architecture and Data Flow

Source: author's development

Figure 2 illustrates the SampleSelector architecture and the data flow from receiving the JSON response of the API to executing the tests. Pasichnyi R., Serhieiev V., Shevchenko S., Petrukha N., Hryvnak B. [2, p. 240] emphasize the importance of standardized approaches, which is reflected in the unified architecture of SampleSelector for different types of APIs. The proposed methodology turns out to be very effective when it is necessary to manipulate large volumes of JSON data, since there is no need for preliminary parsing and validation according to the schema. This is especially important when it is necessary to test third-party APIs, since the data structure may be undocumented or change without warning. One of the advantages of the proposed method is that it is not dedicated to a specific industry or API architecture.

API testing of any system can be successfully performed using the SampleSelector method. Such versatility is provided by a model-free platform, where you can work with any JSON data structures without the need for their preliminary modeling.

Conclusions.

The study of a model-free approach to generating test data in JSON format showed the high potential of such a method, demonstrating the efficiency and adaptability of modern API testing procedures. The SampleSelector technique proposed in the article effectively solves key issues regarding the dynamics of data structures and the need to quickly respond to changes in the API without the need to rewrite test scripts. The main advantages of the developed method are its versatility and lack of dependence on any specific domain model, i.e. it can be used for API testing in different industries, such as microservices and even educational platforms, using the same logic. This saves a lot of time on setting up the test environment, and even when changing the API structure, the costs of maintaining the test code will be minimal. Calculations on real Python code showed that the method is extremely effective when applied to real JSON structures of various levels of complexity. The fact that it is possible to filter by nested fields and select empty or filled values, as well as operate on arrays without prior knowledge of the data schema, makes this approach particularly useful in automated testing. The given architecture of SampleSelector can be quickly reproduced in other programming languages, but all fundamental aspects of working with JSON structures remain unchanged. This creates opportunities for the development of a single ecosystem of tools without model testing across different technology stacks. A comparative analysis of a number of ways to work with JSON test data showed that the SampleSelector method has an excellent balance between implementation and features that can be useful when testing APIs with a variable structure. The functionality of the SampleSelector method, written in Python, demonstrates the efficiency and versatility of the presented idea, as well as the fact that it is adaptable to different programming languages and technology stacks. Further work can be focused on expanding the functionality of the SampleSelector method, in particular, by creating new methods for filtering and processing data, as well as by

integrating it into other automated testing tools. Future research directions include how to combine this approach with machine learning technologies to automatically compute optimal test data sampling strategies, how to provide plugins for common testing frameworks, and how to provide visualization of test data coverage of complex JSON structures. It is also worth exploring the possibility of automating test data generation using a similar approach using artificial intelligence and machine learning.

References

1. Pavlenko I., Boiko O., Mykolaiets D., Moskalenko O., Shrol T. Advances in STEM education and the evolution of game technologies in Ukrainian educational settings // Multidisciplinary Reviews. - 2024. - Vol. 7. - P. 2024spe007. - https://doi.org/10.31893/multirev.2024spe007. (in English)

2. Pasichnyi R., Serhieiev V., Shevchenko S., Petrukha N., Hryvnak B. Digital transformation of higher education as a driver of Ukraine's integration into the European educational space // Cadernos De Educação Tecnologia E Sociedade. - 2024. - Vol. 17, No. se4. - P. 232–245. - https://doi.org/10.14571/brajets.v17.nse4.232-245. (in English)

3. Yang R., Xu X., Wang R. TestLoter: A logic-driven framework for automated unit test generation and error repair using large language models // Computer Languages. - 2025. - https://doi.org/10.1016/j.cola.2025.101348. (in English)

4. Langerak AJ, Frasincar F., Klinkhamer J. JAL: An algebra for JSON query optimization // World Wide Web. - 2025. - Vol. 28, No. 3. - https://doi.org/10.1007/s11280-025-01336-0. (in English)

5. Cui B., Qu R., Zhang J. JustinANN: Realistic Test Generation for Java Programs Driven by Annotations // arXiv. - 2025. - https://doi.org/10.48550/arXiv.2505.05715. (in English)

6. Senthil Kumar T., Jayapradha J., Roslee MB, Shaik H. API-GEN: Accelerator Tool for Generation of REST APIs // 2025 IEEE International Conference on Information Technology and Knowledge Discovery. - 2025. - https://doi.org/10.1109/ITIKD63574.2025.11004961. (in English)

7. Chang H.-F., Shirazi MS A Systematic Approach for Assessing Large Language Models' Test Case Generation Capability // arXiv. - 2025. - https://doi.org/10.48550/arXiv.2502.02866. (in English)

8. Krodinger L., Lukasczyk S., Fraser G. Combining Type Inference and Automated Unit Test Generation for Python // arXiv. - 2025. - https://doi.org/10.48550/arXiv.2507.01477. (in English)

9. Srinivasan M., Abdel J. GenFair: Systematic Test Generation for Fairness Fault Detection in Large Language Models // arXiv. - 2025. - https://doi.org/10.48550/arXiv.2506.03024. (in English)

10. Kim SY, Kim S. Generalizability Theory Approach to Analyzing Automated-Item Generated Test Forms // Educational Measurement: Issues and Practice. - 2025. - https://doi.org/10.1111/emip.12671. (in English)

Анотація. *Стаття присвячена дослідженню способів генерації тестових даних у структурі JSON для виконання автоматизованих API-тестів безмодельним способом. Запропоновано поширений метод вибірки даних SampleSelector, який дозволяє ефективно отримувати вибірку без попереднього визначення моделей даних. Досліджено сферу застосування методу для тестування API різних галузей, що мають динамічні структури даних. Запропонований підхід дозволяє ефективно формувати тестові дані в будь-якій галузі застосування.*

Ключові слова: *формування, тестові дані, формат JSON, без модельний підхід.*