УДК 004.415.533:004.774

СРАВНИТЕЛЬНЫЙ АНАЛИЗ ПОДХОДОВ К АВТОМАТИЗАЦИИ ТЕСТИРОВАНИЯ ВЕБ-API: POSTMAN/NEWMAN И КОД-ОРИЕНТИРОВАННЫЕ ФРЕЙМВОРКИ

Кривоносова М.А.

Белорусский государственный университет информатики и радиоэлектроники, г. Минск, Республика Беларусь

Научный руководитель: Косарева Е.М. – ассистент кафедры ПИКС

Аннотация. Выполнен сравнительный анализ подходов к автоматизации тестирования веб-API на основе Postman/Newman и код-ориентированных фреймворков (REST Assured для Java, pytest для Python). Рассмотрены особенности организации тестовых сценариев, управления окружениями и данных, а также применения проверок на разных этапах жизненного цикла разработки. Установлено, что подход на основе Postman/Newman обеспечивает минимальное время внедрения и ориентирован на оперативную сценарную верификацию с возможностью интеграции запусков в CI/CD, тогда как код-ориентированный подход демонстрирует более высокую масштабируемость и сопровождаемость при росте тестового контура за счёт построения тестов как части программного репозитория и использования повторно применяемых компонентов.

Ключевые слова: веб-API, автоматизация тестирования, Postman, Newman, REST Assured, CI/CD, контрактное тестирование, негативные сценарии

Введение. Современные программные системы все чаще строятся по принципу сервис-ориентированной архитектуры, где взаимодействие компонентов и внешних потребителей реализуется посредством веб-интерфейсов прикладного программирования (API). В таких условиях API выступает «контрактом» взаимодействия между подсистемами, поэтому качество и корректность его функционирования становится одним из ключевых факторов надежности и устойчивости всей системы. Ошибки на уровне интерфейса приводят не только к сбоям отдельных операций, но и к нарушению работы связанных сервисов и клиентских приложений. Это повышает требования к регулярной проверке и воспроизводимости тестирования.

На сегодняшний день в практике автоматизации тестирования веб-API сформировались два подхода к решению данной задачи. Первый подход является инструментальным и представлен платформой Postman, а также её консольным компонентом Newman, позволяющим выполнять коллекции запросов в автоматическом режиме. Второй подход является код-ориентированным и опирается на специализированные фреймворки и библиотеки, такие как REST Assured для Java или библиотека requests в сочетании с pytest для Python, где тестовые сценарии реализуются в виде программного кода и развиваются вместе с проектом. Настоящая работа посвящена сравнительному анализу указанных подходов и выявлению их особенностей с точки зрения практического применения в задачах автоматизации тестирования веб-API [1].

Основная часть. Подход Postman/Newman можно описать как сценарно-коллекционный. Тестировщик формирует коллекцию запросов к API, настраивает окружения, добавляет предусловия и проверки в виде скриптов. В Postman проверки реализуются в песочнице на JavaScript через объект pm, а ключевой механизм спецификаций тестов – pm.test, который связывает читаемое имя проверки и функцию-утверждение. Результаты отображаются в выводе выполнения запросов. Этот формат удобен на ранних этапах, когда API ещё меняется, а команда параллельно уточняет требования и ожидаемое поведение [2].

Newman продолжает эту же модель, но переносит запуск коллекций в командную строку. По сути, Newman запускает Postman-коллекции без графического интерфейса и позиционируется как инструмент для интеграции с системами сборки и CI/CD, где важны воспроизводимость и автоматический запуск по расписанию или на каждый коммит. Благодаря этому один и тот же набор сценариев может работать как локально у инженера, так и в конвейере поставки [3].

Код-ориентированный подход строится иначе: тесты являются частью программного репозитория и пишутся на различных языках программирования с использованием библиотек для HTTP-вызовов и фреймворков для запуска тестов. В практике для задач автоматизированного тестирования REST-сервисов выделяется библиотека REST Assured, ориентированная на удобочитаемую проверку HTTP-взаимодействий в Java и поддерживающая построение запросов и валидацию ответов. Назначение такого инструмента заключается в переводе тестирования веб-API в плоскость декларативно и семантически выразимого кода, где предусловия, выполняемое действие и критерии корректности результата формализуются как взаимосвязанные элементы единого тестового сценария [4].

В Python схожую роль играет связка библиотек HTTP-клиента с тестовым раннером. Pytest позиционируется как фреймворк, который помогает писать небольшие читаемые тесты и при необходимости масштабировать набор до сложного функционального тестирования. В автоматизации тестирования API это обеспечивает поддержку параметризации, применение фикстур для подготовки окружения и тестовых данных, расширяемость за счёт плагинов, а также интеграцию с процессами разработки программного обеспечения (версионирование, ревью, CI/CD) [5].

Для проведения сравнительного анализа подходов Postman/Newman и код-ориентированных фреймворков выбраны критерии, отражающие ключевые аспекты жизненного цикла автоматизации тестирования веб-API. В качестве основы использованы параметры, непосредственно влияющие на практическую применимость решения в проекте (таблица 1).

Таблица 1 – Обоснование выбора критериев сравнительного анализа

Критерий	**Описание критерия**
Время первоначального развёртывания автоматизации	характеризует скорость подготовки и запуска первых автотестов, включая настройку окружения и получение воспроизводимых результатов
Масштабируемость и сопровождаемость автоматизации в длительной перспективе	отражает способность подхода поддерживать рост тестового контура с минимальными издержками за счёт повторного использования компонентов и единообразной структуры
Практики версионирования и совместной разработки	определяет удобство коллективного внесения изменений и их контроля в системе версионирования, включая прозрачность диффов и разрешение конфликтов
Управляемость тестовыми данными и окружениями	описывает средства подготовки, хранения и воспроизводимого использования параметров окружения и тестовых данных при выполнении сценариев
Контроль соответствия API формальному описанию интерфейса	оценивает возможность системной проверки соблюдения правил интерфейса (структуры ответов, типов полей, кодов и формата ошибок) при изменениях API
Поддержка безопасности и «негативных» сценариев	характеризует пригодность подхода для реализации и масштабирования проверок ошибок, авторизации/аутентификации, пограничных значений и иных негативных воздействий

При сопоставлении рассматриваемых подходов по критерию времени первоначального развёртывания автоматизации преимущество демонстрирует Postman. Наличие графического интерфейса, механизма коллекций и типовой модели формирования запросов и проверок снижает порог входа и ускоряет прототипирование тестовых сценариев. Это востребовано на практике при необходимости быстро сформировать минимальный регрессионный контур либо воспроизвести дефект в контролируемых условиях. Но вместе с этим проявляются характерные ограничения коллекционного подхода: усложняется организация повторного использования общих компонентов, затрудняется поддержание единообразия оформления сценариев, а проверочная логика концентрируется в виде множества встроенных скриптов. В код-ориентированной парадигме предпосылки масштабируемости закладываются исходно: общие HTTP-клиенты, абстракции доменных операций, генераторы тестовых данных и унифицированные механизмы логирования. Они формируются средствами языка программирования и принятых процессов командной разработки. Postman целесообразно рассматривать как инструмент быстрого накопления и верификации сценариев, а кодовые

фреймворки обеспечивают более устойчивую управляемость и сопровождаемость автоматизации в длительной перспективе [6].

Практики версионирования и совместной разработки также выявляют различие подходов. В Postman предусмотрена возможность экспорта коллекций и их хранения в системе контроля версий, а также выполнения в CI посредством Newman. Но при параллельном редактировании коллекций несколькими участниками возникающие конфликты при слиянии версий в системе контроля версий часто сложнее анализировать и корректно разрешать, чем аналогичные конфликты в исходном коде. В код-ориентированном подходе тесты включаются в стандартный контур командной разработки. Изменения проходят процедуру code review, доступны средства статического анализа. Инструменты разработки (IDE) обеспечивают поддерживаемый рефакторинг и унификацию структуры проекта [7].

Отдельный критерий – управляемость тестовыми данными и окружениями. В Postman это решается переменными окружений, пред- и пост-скриптами и цепочками запросов, где данные извлекаются из ответа и подставляются в следующий шаг. В код- ориентированном подходе тот же сценарий обычно оформляется как явная подготовка данных через фикстуры, фабрики и клиенты. Это делает зависимости более видимыми и облегчает локализацию причины падения [8].

Ещё один критерий сопоставления – контроль соответствия API формальному описанию интерфейса (контракту). Это проверка того, что структура запросов и ответов, обязательность и типы полей, коды HTTP-ответов и формат ошибок соответствуют установленным правилам. В Postman/Newman подобные проверки обычно задаются на уровне отдельных запросов (через локальные утверждения и скрипты, привязанные к конкретным сценариям выполнения). Такой подход удобен для быстрой верификации, но при расширении покрытия возрастает риск дублирования однотипных проверок и усложняется поддержание единообразия при изменениях API.

В код-ориентированной парадигме контрактные требования проще формализовать централизованно. Общие правила интерфейса могут быть вынесены в повторно используемые компоненты (унифицированные ассёрты, валидаторы структуры ответов, единые модели ошибок), после чего применяться во всех тестах системно и согласованно. Это повышает управляемость регрессионного контура. Изменения в интерфейсе отражаются в общих проверках и автоматически распространяются на весь набор тестов. Это снижает вероятность рассогласования между фактическим поведением сервиса и ожидаемыми условиями корректности [9].

Также при оценке подходов следует учитывать поддержку безопасности и «негативных» сценариев. Даже при фокусе на функциональной корректности автоматизированное тестирование API практически неизбежно включает проверки механизмов аутентификации и авторизации, корректности обработки некорректных входных данных, соблюдения ограничений на ресурсы и устойчивости к повторным запросам. В рамках Postman/Newman реализация подобных проверок осуществляется на уровне сценариев коллекции (через последовательности запросов, переменные окружения и локальные скрипты утверждений). Такой подход удобен для быстрой верификации типовых security-сценариев и воспроизведения уязвимых последовательностей. Но при росте числа проверок усложняется управление наборами входов и систематизация негативных кейсов, так как логика генерации и вариативности часто распределяется по множеству запросов и скриптов. В код-ориентированном подходе негативное и security-тестирование легче масштабируется за счёт параметризации, фабрик тестовых данных и повторно используемых модулей авторизации/клиентов. Это позволяет запускать большие серии проверок с контролируемым покрытием (комбинации ролей, токенов, границ значений) [10].

Для обобщения результатов сравнительного анализа и наглядного представления различий между инструментальным подходом Postman/Newman и код-ориентированными фреймворками выполнено сопоставление по ранее выбранным критериям. В таблице 2 представлено обобщение различий, связанных с организацией тестов, их развитием, совместной работой, управлением данными, контролем контрактной согласованности и реализацией негативных и security-сценариев.

Таблица 2 – Сопоставление подходов Postman/Newman и код-ориентированных фреймворков по выбранным критериям

Критерий	***Postman/Newman***	**Код-ориентированные фреймворки**
Время первоначального развёртывания автоматизации	меньшие временные затраты на стартовую настройку и получение первых сценарных проверок за счёт графического интерфейса и механизма коллекций	большие временные затраты на стартовую настройку, обусловленные необходимостью реализации тестов в виде программного кода и организации запуска
Масштабируемость и сопровождаемость автоматизации в длительной перспективе	при росте тестового набора возрастает трудоёмкость сопровождения из-за усложнения повторного использования общих компонентов, поддержания единообразия оформления и концентрации логики во встроенных скриптах	масштабируемость обеспечивается исходной организацией общих HTTP-клиентов, абстракций доменных операций, генераторов тестовых данных и унифицированных механизмов логирования
Практики версионирования и совместной разработки	поддерживается экспорт коллекций и хранение в системе контроля версий; при параллельном редактировании усложняется анализ и корректное разрешение конфликтов слияния	тесты включаются в стандартный контур командной разработки; изменения проходят code review, применимы средства статического анализа и поддерживаемый рефакторинг в IDE
Управляемость тестовыми данными и окружениями	управление реализуется через переменные окружений, пред- и пост-скрипты и цепочки запросов, где данные извлекаются из ответа и подставляются в следующий шаг	управление реализуется через фикстуры, фабрики и клиентов, что обеспечивает явную подготовку данных и делает зависимости более видимыми
Контроль соответствия API формальному описанию интерфейса	проверки соответствия задаются на уровне отдельных запросов через локальные утверждения и скрипты, привязанные к конкретным сценариям выполнения	контрактные требования формализуются централизованно в повторно используемых компонентах и применяются системно и согласованно ко всему тестовому набору
Поддержка безопасности и «негативных» сценариев	реализация проверок выполняется в рамках сценариев коллекции (последовательности запросов, переменные окружения, локальные скрипты утверждений); при росте числа проверок усложняется управление наборами входов и систематизация негативных кейсов	масштабирование негативных и security-проверок обеспечивается параметризацией, фабриками тестовых данных и повторно используемыми модулями авторизации/клиентов, что позволяет выполнять большие серии проверок с контролируемым покрытием

Выполненное сопоставление подходов Postman/Newman и код-ориентированных фреймворков по выбранным критериям позволило охарактеризовать особенности их применения в задачах автоматизации тестирования веб-API. В ходе анализа показано, что различия между подходами проявляются не только на этапе формирования первых сценариев, но и при дальнейшем развитии тестового контура: в организации повторного использования общих компонентов, поддержании единообразия проверок, управлении тестовыми данными и окружениями, а также в способах формализации требований к интерфейсу и реализации негативных и security-сценариев. Табличная форма обеспечила структурирование выявленных различий и позволила представить их в едином формате, пригодном для практического применения при проектировании автоматизации на конкретном проекте. Полученные положения формируют основу для последующей формулировки выводов о целесообразности применения каждого из подходов в зависимости от масштаба системы, динамики изменений API и требований к управляемости и воспроизводимости тестирования.

Заключение. Выполнен сравнительный анализ подходов к автоматизации тестирования веб-API на базе Postman/Newman и код-ориентированных фреймворков. Установлено, что его применение обеспечивает минимальные затраты на первоначальное развёртывание и позволяет быстро формировать сценарные наборы проверок с воспроизводимым запуском в средах CI/CD.

Код-ориентированный подход обеспечивает улучшенную управляемость автоматизации в длительной перспективе за счёт применения повторно используемых абстракций, централизованной формализации контрактных требований и эффективного масштабирования негативных и security-сценариев.

Список литературы

1. Тестирование программного обеспечения : учеб. пособие / С. С. Куликов, Г. В. Данилова, О. Г. Смолякова, М. М. Меженная. – Минск : БГУИР, 2019. – 276 с. : ил. – ISBN 978-985-543-462-8.2.

2. Writing tests and assertions in scripts (pm.test, pm.expect) / Postman // Postman Learning Center [Электронный ресурс]. – Режим доступа: https://learning.postman.com/docs/tests-and-scripts/write-scripts/postman-sandbox-reference/pm-test-expect (дата обращения: 02.03.2026).

3. Run and test collections from the command line using Newman CLI / Postman // Postman Learning Center [Электронный ресурс]. – Режим доступа: https://learning.postman.com/docs/collections/using-newman-cli/command-line-integration-with-newman (дата обращения: 02.03.2026).

4. Лось, Н. А. Автоматизация тестирования REST сервисов с помощью REST Assured / Н. А. Лось, А. Л. Ярошенко // 54-я научная конференция аспирантов, магистрантов и студентов БГУИР. – Минск : БГУИР, 2018. – С. 74–75.

5. How to use fixtures / pytest Development Team // pytest documentation [Электронный ресурс]. – Режим доступа: https://docs.pytest.org/en/stable/how-to/fixtures.html (дата обращения: 02.03.2026).

6. Козак, У. М. Автоматизация процесса тестирования и отчетности на проекте Web-Insync в ЗАО «Альфа-Банк» = Automation of the testing and reporting process on the Web-Insync project at Alfa-Bank / У. М. Козак, В. Г. Лунин // «BIG DATA and Advanced Analytics. BIG DATA и анализ высокого уровня» : сборник научных статей X Междунар. науч.-практ. конф., Минск, 13 марта 2024 г. : в 2 ч. Ч. 2 / БГУИР. – Минск, 2024. – С. 54–65.

7. Лось, Н. А. Автоматизация тестирования web-приложений как часть процесса continius integration / Н. А. Лось, А. Л. Ярошенко // 55-я юбилейная научная конференция аспирантов, магистрантов и студентов БГУИР. – Минск : БГУИР, 2019. – С. 165.

8. Store and reuse values using variables / Postman // Postman Learning Center [Электронный ресурс]. – Режим доступа: https://learning.postman.com/docs/sending-requests/variables/variables/ (дата обращения: 02.03.2026).

9. OpenAPI Specification. Version 3.1.0 / OpenAPI Initiative [Электронный ресурс]. – 2021. – Режим доступа: https://spec.openapis.org/oas/v3.1.0.html (дата обращения: 02.03.2026).

10. OWASP Top 10 API Security Risks – 2023 / OWASP Foundation [Электронный ресурс]. – 2023. – Режим доступа: https://owasp.org/API-Security/editions/2023/en/0x11-t10/ (дата обращения: 02.03.2026)

UDC 004.415.533:004.774

COMPARATIVE ANALYSIS OF WEB API TEST AUTOMATION APPROACHES: POSTMAN/NEWMAN AND CODE-ORIENTED FRAMEWORKS

Krivonosova M.A.

Belarusian State University of Informatics and Radioelectronics, Minsk, Republic of Belarus

Scientific advisor: Kosareva E.M. – Assistant Lecturer, Department of ICSD

Annotation. A comparative analysis of web API test automation approaches based on Postman/Newman and code-oriented frameworks (REST Assured for Java, pytest for Python) is presented. The study examines the specifics of organizing test scenarios, managing environments and test data, and applying checks at different stages of the software development lifecycle. It is established that the Postman/Newman-based approach requires minimal initial setup time and is aimed at rapid scenario verification with the possibility of integrating runs into CI/CD, whereas the code-oriented approach demonstrates higher scalability and maintainability as the test suite grows due to embedding tests into the project repository and using reusable components.

Keywords: web API, test automation, Postman, Newman, REST Assured, CI/CD, contract testing, negative scenarios