Научно-практический электронный журнал
Оригинальные исследования (ОРИС), *ISSN 2222-0402*
www.ores.su
original-research.ru

УДК 004.415.2:004.738.5

СРАВНИТЕЛЬНЫЙ АНАЛИЗ ИНСТРУМЕНТОВ АВТОМАТИЗИРОВАННОГО ТЕСТИРОВАНИЯ ВЕБ-ПРИЛОЖЕНИЙ: SELENIUM, PLAYWRIGHT И CYPRESS

Курочкин Никита Алексеевич,
Студент, Поволжский Государственный Университет Телекоммуникаций и Информатики, г. Самара, Российская Федерация
E-mail: n.kurochkin6@gmail.com

Ахметшина Элеонора Газинуровна,
К.т.н., доцент каф. ИРС, Поволжский Государственный Университет Телекоммуникаций и Информатики, г. Самара, Российская Федерация
E-mail: e.ahmetshina@psuti.ru

Аннотация

Актуальность исследования связана с необходимостью обоснованного выбора инструмента автоматизированного end-to-end тестирования веб-приложений в условиях ограниченных ресурсов учебных проектов, когда сравнительные данные на идентичных UI-сценариях в учебной литературе представлены фрагментарно. Разработчик учебного или прототипного сервиса нередко выбирает E2E-фреймворк по популярности в сообществе, не имея количественных данных о времени прогона и стабильности на одном и том же приложении. Цель работы — выполнить сравнительный анализ Selenium WebDriver, Playwright и Cypress на едином наборе UI-сценариев и оценить время прогона и долю успешных запусков. Методика предусматривает реализацию пяти E2E-сценариев на небольшом веб-приложении (FastAPI + HTML-шаблоны), выполнение по три прогона для каждого фреймворка в headless-режиме и фиксацию суммарного времени выполнения набора тестов. Результаты: Playwright показал наименьшее среднее время (2,49 с) и полную стабильность (3 из 3); Selenium — средний результат (4,91 с) с одним неуспешным прогоном; Cypress — наибольшее время (8,43 с) при 100 % успешности. Выводы: для Python-проектов целесообразен Playwright, для JS-стека — Cypress, для корпоративной мультибраузерной инфраструктуры — Selenium; результаты применимы при выборе E2E-фреймворка в учебных и малых коммерческих веб-проектах.

Ключевые слова: автоматизированное тестирование; Selenium; Playwright; Cypress; веб-приложение; E2E; сравнительный анализ; качество программного обеспечения.

COMPARATIVE ANALYSIS OF AUTOMATED WEB APPLICATION TESTING TOOLS: SELENIUM, PLAYWRIGHT, AND CYPRESS

Kurochkin Nikita Alekseevich,
Student, Povolzhskiy State University of Telecommunications and Informatics, Samara, Russian Federation

E-mail: n.kurochkin6@gmail.com

Akhmetshina Eleonora Gazinurovna,
Candidate of Technical Sciences, Associate Professor, Department of IRS
Povolzhskiy State University of Telecommunications and Informatics, Samara, Russian Federation
E-mail: e.ahmetshina@psuti.ru

ABSTRACT

The relevance of the study is related to the need for an informed choice of a tool for automated end-to-end testing of web applications in the context of limited resources of educational projects, when comparative data on identical UI scenarios in the educational literature is presented fragmentarily. The developer of an educational or prototype service often chooses an E2E framework based on its popularity in the community, without having quantitative data on the runtime and stability on the same application. The aim of the work is to perform a comparative analysis of Selenium WebDriver, Playwright, and Cypress on a single set of UI scenarios and evaluate the runtime and the proportion of successful launches. The methodology involves implementing five E2E scenarios on a small web application (FastAPI + HTML templates), performing three runs for each framework in headless mode, and recording the total execution time of the test set. Results: Playwright showed the lowest average time (2.49 s) and complete stability (3 out of 3); Selenium showed an average result (4.91 s) with one unsuccessful run; Cypress showed the longest time (8.43 s) with a 100% success rate. Conclusions: Playwright is suitable for Python projects, Cypress for the JS stack, and Selenium for an enterprise multi-browser infrastructure. The results are applicable when choosing an E2E framework for educational and small commercial web projects.

Keywords: automated testing; Selenium; Playwright; Cypress; web application; E2E; comparative analysis; software quality.

Введение

Качество веб-приложений во многом определяется полнотой проверки пользовательских сценариев. Ручное регрессионное тестирование трудоёмко и плохо масштабируется при частых изменениях интерфейса; автоматизация E2E (end-to-end) позволяет воспроизводить действия пользователя в браузере и выявлять дефекты до выпуска новой версии [1; 3].

Современные веб-сервисы сочетают серверную часть (REST API или шаблонный рендеринг) и клиентский интерфейс; ошибки могут проявляться только при полном прохождении сценария — авторизация, навигация, отправка формы, обновление списка. Модульные и интеграционные тесты не заменяют проверку поведения в реальном браузере: они не учитывают задержки рендеринга, работу JavaScript и совместимость с DOM [3; 6]. При этом внедрение E2E-автоматизации требует выбора фреймворка, от которого зависят скорость написания тестов, стабильность прогонов в CI/CD и трудозатраты на сопровождение тестовой базы [6; 7].

На рынке представлены Selenium WebDriver как отраслевой стандарт с поддержкой множества языков, Microsoft Playwright с современным API и встроенными ожиданиями, а также Cypress как специализированный фреймворк для фронтенд-команд [2; 4; 5]. Формальное сравнение этих инструментов на идентичном прикладном коде в учебной

литературе представлено фрагментарно: чаще описывается один фреймворк или приводятся качественные рекомендации без измерения времени прогона на одном приложении [3; 10].

Объект исследования — небольшое веб-приложение на стеке FastAPI с HTML-шаблонами и три E2E-фреймворка (Selenium, Playwright, Cypress). Предмет исследования — влияние выбора E2E-инструмента на суммарное время прогона набора UI-сценариев и долю успешных запусков при фиксированной конфигурации headless-браузера. Научная новизна заключается в эмпирическом сопоставлении трёх актуальных фреймворков на едином небольшом веб-приложении с количественными метриками времени и стабильности при одинаковом наборе из пяти сценариев.

Практическая значимость работы состоит в том, что полученные результаты и рекомендации могут быть использованы при выборе E2E-инструмента в рамках учебной практики, дипломных проектов и прототипов веб-сервисов, где ограничены ресурсы на развёртывание тестовой инфраструктуры, но требуется обоснованное решение о стеке автоматизации.

Обзор инструментов E2E-тестирования.

Selenium WebDriver — открытый стандарт автоматизации браузеров с поддержкой Python, Java, C# и других языков [2]. Архитектура основана на взаимодействии с драйвером браузера через W3C WebDriver Protocol [8]. Преимущества — зрелость экосистемы, большое число интеграций, поддержка Selenium Grid для распределённого запуска; недостатки — необходимость явно задавать ожидания элементов и возможная нестабильность тестов при асинхронной загрузке страницы [6]. В учебных проектах Selenium часто выбирают из-за обилия примеров и совместимости с unittest/pytest, однако начинающему разработчику приходится отдельно настраивать драйвер и синхронизацию [2; 6].

Playwright — фреймворк Microsoft с единым API для Chromium, Firefox и WebKit [4]. Встроенные auto-wait механизмы автоматически ожидают появления и готовности элементов, что снижает число «плавающих» (flaky) тестов. Playwright поддерживает Python, JavaScript, C# и Java, что делает его удобным для backend-разработчиков, уже использующих pytest [4]. Генератор тестов (codegen) и трассировка (trace) упрощают отладку, а единый бинарный дистрибутив браузеров сокращает время первичной настройки [4].

Cypress — JavaScript-фреймворк, выполняющий тесты внутри браузера [5]. Он обеспечивает наглядную отладку, автоматические скриншоты и time-travel при просмотре шагов теста. Ограничения — ориентация на JS/TS-стек и меньшая гибкость при работе с несколькими вкладками и доменами по сравнению с Selenium [3; 10]. Для команд, где фронтенд и тесты пишутся на одном языке, Cypress снижает порог входа, но накладные расходы Electron и Node.js-окружения могут увеличивать время прогона [5].

В обзорных работах подчёркивается, что выбор инструмента автоматизации должен учитывать не только скорость выполнения тестов, но и стоимость сопровождения, интеграцию с pipeline CI и компетенции команды [1; 10]. Для небольших учебных проектов критичны простота первого запуска и предсказуемость результатов при повторных прогонах на одной машине [7; 12].

Постановка задачи.

Цель работы — выполнить сравнительный анализ Selenium, Playwright и Cypress на идентичном наборе UI-сценариев и оценить время прогона и стабильность.

Для достижения поставленной цели сформулированы следующие задачи исследования:

Проанализировать особенности Selenium, Playwright и Cypress в контексте автоматизации тестирования небольших веб-приложений.

Разработать единый набор из пяти E2E-сценариев на небольшом веб-приложении с атрибутами data-testid.

Реализовать сценарии на Selenium (Python + unittest), Playwright (Python + pytest) и Cypress (JavaScript).

Выполнить по три прогона для каждого фреймворка в headless-режиме и зафиксировать суммарное время выполнения и долю успешных запусков.

Сформулировать рекомендации по выбору E2E-инструмента для учебных и малых коммерческих проектов.

Гипотеза исследования: Playwright обеспечит наименьшее среднее время прогона за счёт встроенных auto-wait и оптимизированного запуска Chromium; Selenium покажет промежуточный результат, но возможны нестабильные прогоны без явных ожиданий; Cypress продемонстрирует наибольшее время из-за накладных расходов Electron при сохранении полной успешности.

Архитектура прототипа.

Прототип построен по схеме «браузерный автомат — HTTP-клиент — сервер приложений — хранилище данных». Серверная часть реализована на Python с использованием FastAPI и Jinja2-шаблонов; приложение предоставляет форму авторизации, список записей и операции создания и удаления [11]. Клиентская часть — HTML-страницы с семантической разметкой; для устойчивости селекторов в элементы интерфейса добавлены атрибуты data-testid, рекомендуемые в практиках тестируемой вёрстки [1; 7; 12].

E2E-тесты организованы в три независимых каталога: selenium_tests (unittest + Chrome WebDriver), playwright_tests (pytest + встроенный Chromium) и cypress (npm + Electron). Один и тот же набор из пяти сценариев воспроизведён на каждом стеке: проверка заголовка страницы; ошибка входа при неверных данных; успешная авторизация; создание новой записи; удаление записи из списка. Такой подход обеспечивает сопоставимость результатов: отличия измерений обусловлены фреймворком, а не различиями в бизнес-логике приложения [6].

Скрипт последовательно запускает три серии прогонов, фиксируя wall-clock время выполнения всей команды (unittest discover, pytest или npm test) и код возврата процесса. Результаты сохраняются в JSON для построения таблиц и графиков.

Методика эксперимента.

Объект тестирования — небольшое веб-приложение на FastAPI по адресу http://127.0.0.1:8000/. Приложение реализует авторизацию по логину и паролю, отображение списка записей, создание и удаление элементов. Перед серией измерений сервер приложения запускался на порту 8000 и оставался активным между прогонами одного фреймворка.

Параметры экспериментального стенда приведены в таблице 1. Прогоны выполнялись в headless-режиме: Playwright — Chromium, Selenium — Chrome через WebDriver, Cypress — встроенный Electron. Для каждого фреймворка выполнено 3 последовательных прогона полного набора из 5 тестов; между прогонами разных фреймворков окружение не менялось.

Таблица 1. Параметры экспериментального стенда

Параметр	Значение
ОС	Linux
Python	3.14
FastAPI	0.115

Параметр	Значение
Selenium	Chrome + WebDriver
Playwright	Chromium (headless)
Cypress	Electron (headless)
Число сценариев	5
Прогонов на фреймворк	3

Метрики эксперимента: суммарное время прогона (с) — от запуска команды до её завершения; стабильность — число успешных прогонов из 3; объём кода — число строк тестовых файлов (качественно, таблица 3). Время включает инициализацию браузера, выполнение пяти сценариев и завершение процесса; сетевые задержки localhost минимальны, основная доля — работа браузера и фреймворка [2; 4; 5].

Результаты эксперимента.

Количественные результаты сравнения инструментов представлены в таблице 2. Качественные характеристики по критериям удобства разработки и отладки — в таблице 3. Сравнение среднего времени прогона иллюстрируется на рисунке 1.

Таблица 2. Результаты сравнительного прогона E2E-тестов

Инструмент	Тестов	Прогонов	Успешных	Среднее, с	Мин., с	Макс., с
Selenium	5	3	2	4.91	4.60	5.08
Playwright	5	3	3	2.49	2.42	2.57
Cypress	5	3	3	8.43	8.40	8.45

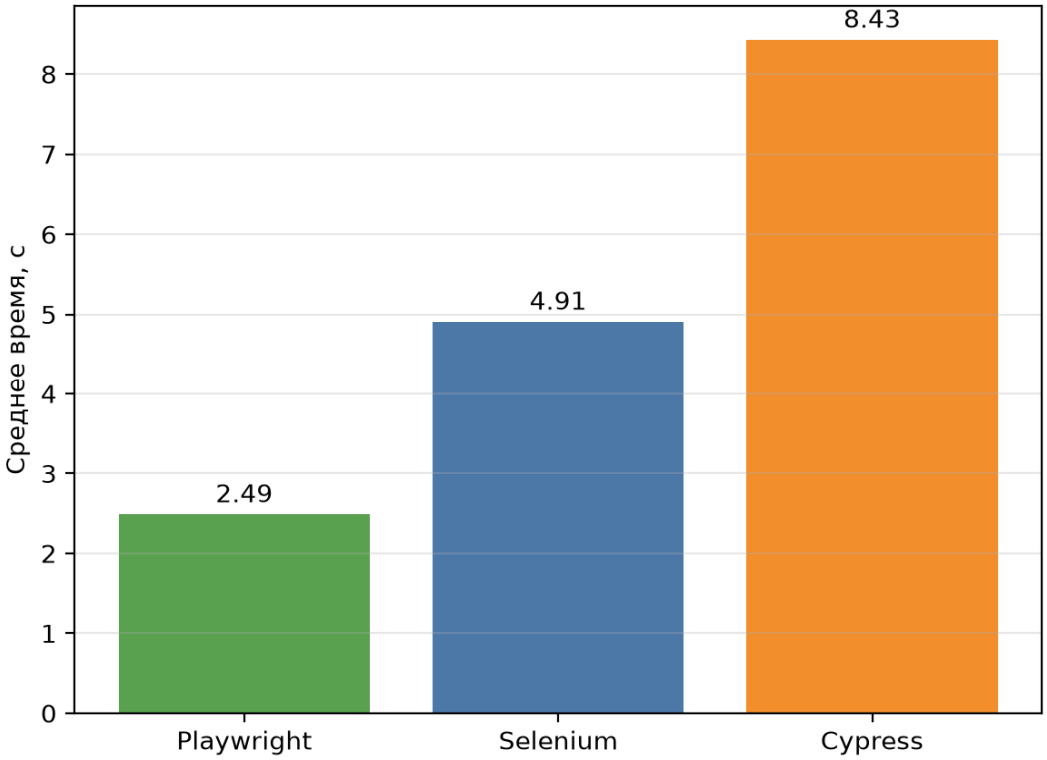

Рисунок 1. Среднее время прогона E2E-тестов по инструментам.

Таблица 3. Качественное сравнение E2E-инструментов

Критерий	Selenium	Playwright	Cypress
Языки	Много	Много	JS/TS
Auto-wait	Частично	Да	Да
Отладка	Средняя	Хорошая	Отличная
Порог входа	Средний	Низкий	Низкий
Зрелость	Высокая	Растущая	Высокая

Анализ времени прогона. Playwright показал наименьшее среднее время — 2,49 с — и полную стабильность (3 успешных прогона из 3) [4]. Разброс между минимальным и максимальным временем минимален (2,42–2,57 с), что свидетельствует о воспроизводимости результатов. Selenium продемонстрировал среднее время 4,91 с, но один из трёх прогонов завершился с ошибкой, вероятно из-за гонки при запуске нового экземпляра браузера или недостаточного ожидания элемента [2; 6].

Cypress обеспечил 100 % успешных прогонов, однако среднее время (8,43 с) оказалось наибольшим — примерно в 3,4 раза больше, чем у Playwright. Это объясняется накладными расходами Electron, инициализацией Node.js-окружения и особенностями архитектуры Cypress, выполняющего тесты внутри браузера [5]. Разброс времени Cypress узкий (8,40–8,45 с), что указывает на стабильную, но более медленную инициализацию.

Сводная оценка. Таблица 4 фиксирует лидирующий инструмент по каждой метрике. Playwright лидирует по времени и стабильности; Cypress — по доле успешных прогонов наравне с Playwright, но с наибольшим временем; Selenium занимает промежуточное положение по скорости, но уступает по стабильности в данном эксперименте.

Таблица 4. Сводка лидеров по метрикам

Метрика	Selenium	Playwright	Cypress	Лидер
Среднее время, с	4,91	2,49	8,43	Playwright
Успешных прогонов	2/3	3/3	3/3	Playwright, Cypress
Разброс (макс. − мин.), с	0,48	0,15	0,05	Cypress

Согласно таблице 3, Playwright сочетает низкий порог входа и встроенные ожидания; Cypress выигрывает по удобству отладки; Selenium — по зрелости экосистемы и поддержке legacy-инфраструктуры [2; 10]. При росте числа сценариев и интеграции с CI/CD абсолютное время прогона будет масштабироваться линейно, но соотношение между фреймворками, вероятно, сохранится, поскольку накладные расходы инициализации доминируют на малых наборах тестов [6; 10].

Обсуждение и рекомендации.

Для учебных проектов на Python целесообразен Playwright: быстрый старт, стабильные тесты, интеграция с pytest и компактный объём кода [4]. Студент может написать первый работающий E2E-тест за одного занятия, не настраивая отдельно WebDriver и явные ожидания для каждого элемента. Структурирование тестов по принципам читаемого кода и явных селекторов data-testid снижает стоимость сопровождения [12].

Для фронтенд-команды на JavaScript предпочтителен Cypress: наглядная отладка, интерактивный режим и минимальная конфигурация [5]. Selenium рекомендуется при требованиях к мультибраузерности, интеграции с корпоративным grid (Selenium Grid) и

наличии готовых Page Object библиотек в Java- или C#-проектах [2; 10]. Выбор между Playwright и Cypress в условиях данного эксперимента определяется прежде всего стеком языка тестов, а не разницей в успешности прогонов.

Сравнение по критериям выбора для малых веб-проектов: Playwright выигрывает по времени и стабильности на Python-стеке; Cypress — по опыту отладки для JS-разработчиков; Selenium — по совместимости с существующей корпоративной инфраструктурой, но требует дополнительных усилий по синхронизации. Альтернативы без смены фреймворка включают Page Object Model, явные WebDriverWait и вынос общих шагов в фикстуры pytest [6; 9].

Тестирование выполнено на одном приложении и одной конфигурации браузеров в headless-режиме; не исследованы мобильные WebView, параллельный запуск тестов, Firefox/WebKit в Playwright и интеграция с pipeline CI на удалённом сервере. Не измерялись метрики flakiness при сотнях повторных прогонов и стоимость сопровождения тестовой инфраструктуры. Результаты отражают класс нагрузки «учебное приложение с пятью последовательными сценариями», а не промышленный регрессионный контур. Дальнейшая работа может включить сравнение на большем числе сценариев, оценку параллельного запуска в CI и сопоставление с инструментами вроде Puppeteer [10].

Заключение.

В работе выполнен сравнительный анализ Selenium, Playwright и Cypress на едином наборе E2E-сценариев небольшого веб-приложения на стеке FastAPI. Реализован воспроизводимый стенд с автоматизированным прогоном через пользовательский скрипт и фиксацией времени и успешности для трёх серий на каждый фреймворк.

Экспериментально подтверждено, что Playwright обеспечивает наименьшее среднее время прогона (2,49 с) и наивысшую стабильность (3 из 3) по таблице 2, таблице 4 и рисунку 1. Selenium показал средний результат (4,91 с) с одним неуспешным прогоном; Cypress — наибольшее время (8,43 с) при полной успешности.

Рекомендуется использовать Playwright для учебных и прототипных проектов на Python, Cypress — для JS/TS-команд с акцентом на отладку, Selenium — при требованиях мультибраузерной корпоративной инфраструктуры. Полученные результаты могут использоваться при выборе E2E-фреймворка в рамках учебной практики и при проектировании малых веб-сервисов.

Список литературы:

1. ГОСТ Р 56920-2016/ISO/IEC/IEEE 29119-1:2013. Системная и программная инженерия. Тестирование программного обеспечения. Часть 1. Понятия и определения. — Москва : Стандартинформ, 2016.
2. Документация Selenium WebDriver [Электронный ресурс] / Selenium Project. — Режим доступа: https://www.selenium.dev/documentation/ (дата обращения: 30.06.2026).
3. Майерс, Г. Дж. Искусство тестирования программ / Г. Дж. Майерс, К. Сандлер, Т. Баджет ; пер. с англ. — 3-е изд. — Москва : Диалектика, 2019. — 272 с. — ISBN 978-5-907144-37-8.
4. Документация Playwright [Электронный ресурс] / Microsoft Corporation. — Режим доступа: https://playwright.dev/docs/intro (дата обращения: 30.06.2026).
5. Документация Cypress [Электронный ресурс] / Cypress.io, Inc. — Режим доступа: https://docs.cypress.io/ (дата обращения: 30.06.2026).

6. Месарош, Г. Шаблоны тестирования xUnit : рефакторинг кода тестов / Г. Месарош ; пер. с англ. — Москва : Вильямс, 2009. — 831 с. — ISBN 978-5-8459-1448-4.
7. Гагарина, Л. Г. Технология разработки программного обеспечения : учеб. пособие / Л. Г. Гагарина, Е. В. Кокорева, Б. Д. Сидорова-Виснадул ; под ред. Л. Г. Гагариной. — Москва : ИД «Форум» : ИНФРА-М, 2018. — 400 с. — ISBN 978-5-8199-0812-9.
8. WebDriver : W3C Recommendation [Электронный ресурс] / World Wide Web Consortium (W3C). — Режим доступа: https://www.w3.org/TR/webdriver/ (дата обращения: 30.06.2026).
9. Copeland, L. A Practitioner's Guide to Software Test Design [Электронный ресурс] / L. Copeland. — Boston : Artech House, 2004. — 320 с.— Режим доступа: https://us.artechhouse.com/A-Practitioners-Guide-to-Software-Test-Design-P756.aspx (дата обращения: 30.06.2026).
10. Криспин, Л. Гибкое тестирование : практическое руководство для тестировщиков ПО и гибких команд / Л. Криспин, Д. Грегори ; пер. с англ. — Москва : Вильямс, 2010. — 464 с. — ISBN 978-5-8459-1625-9.
11. Документация FastAPI [Электронный ресурс] / Tiago Angelo Pinto, Sebastián Ramírez. — Режим доступа: https://fastapi.tiangolo.com/ (дата обращения: 30.06.2026).
12. Мартин, Р. Чистый код : создание, анализ и рефакторинг / Р. Мартин ; пер. с англ. — Санкт-Петербург : Питер, 2020. — 464 с. — ISBN 978-5-4461-0960-9.

References:

1. GOST R 56920-2016/ISO/IEC/IEEE 29119-1:2013. Systems and software engineering. Software testing. Part 1. Concepts and definitions. Moscow: Standartinform, 2016.
2. Selenium WebDriver Documentation [Electronic resource] / Selenium Project. Access mode: https://www.selenium.dev/documentation/ (accessed: 30.06.2026).
3. Myers, G. J. The art of software testing / G. J. Myers, K. Sandler, T. Budgett; translated from English. - 3rd ed. - Moscow: Dialectika, 2019. - 272 p. - ISBN 978-5-907144-37-8.
4. Playwright Documentation [Electronic resource] / Microsoft Corporation. — Access mode: https://playwright.dev/docs/intro (date accessed: 06/30/2026).
5. Cypress Documentation [Electronic resource] / Cypress.io, Inc. — Access mode: https://docs.cypress.io/ (date accessed: 06/30/2026).
6. Meszarosh, G. xUnit Testing Patterns: Test Code Refactoring / G. Meszarosh; trans. from English. — Moscow: Williams, 2009. — 831 p. — ISBN 978-5-8459-1448-4.
7. Gagarina, L. G. Software Development Technology: textbook / L. G. Gagarina, E. V. Kokoreva, B. D. Sidorova-Visnadul; ed. L. G. Gagarina. - Moscow: ID "Forum": INFRA-M, 2018. - 400 p. - ISBN 978-5-8199-0812-9.
8. WebDriver: W3C Recommendation [Electronic resource] / World Wide Web Consortium (W3C). - Access mode: https://www.w3.org/TR/webdriver/ (date accessed: 30.06.2026).
9. Copeland, L. A Practitioner's Guide to Software Test Design [Electronic resource] / L. Copeland. — Boston : Artech House, 2004. — 320 p. — Available at: https://us.artechhouse.com/A-Practitioners-Guide-to-Software-Test-Design-P756.aspx (Accessed: 30.06.2026).

10. Crispin, L. Agile Testing: A Practical Guide for Software Testers and Agile Teams / L. Crispin, D. Gregory; translated from English. — Moscow : Williams, 2010. — 464 p. — ISBN 978-5-8459-1625-9.
11. FastAPI Documentation [Electronic resource] / Tiago Angelo Pinto, Sebastián Ramírez. — Available at: https://fastapi.tiangolo.com/ (Accessed: 30.06.2026).
12. Martin, R. Clean Code: Creation, Analysis, and Refactoring / R. Martin; trans. from English. - St. Petersburg: Piter, 2020. - 464 p. - ISBN 978-5-4461-0960-9.