УДК 004.9

ПРИМЕНЕНИЕ ВЕБ-ТЕХНОЛОГИЙ ДЛЯ СОЗДАНИЯ ИНТУИТИВНО ПОНЯТНОЙ СИСТЕМЫ УПРАВЛЕНИЯ АВТОТЕСТАМИ

Тюленев Алексей Артёмович[1], Ушанкова Мария Юрьевна[2]

[1]*Тестировщик;*
ООО «Клауд Ком»;
Россия, 141983, Московская область, г. Дубна, ул. Программистов, 4;
e-mail: leksey.tyulenev@gmail.com.

[2]*Старший преподаватель;*
Государственный университет «Дубна»;
Россия, 141980, Московская обл., г. Дубна, ул. Университетская, 19;
e-mail: ushankova.m.ju@uni-dubna.ru.

В данной работе представлена разработка веб-приложения для управления автоматизированным тестированием, включающего функции запуска, редактирования и анализа автотестов. Система реализована с использованием современных технологий веб-разработки, включая Python, Django и PostgreSQL, что обеспечивает надежность и масштабируемость решения. Разработка охватывает проектирование пользовательского интерфейса, серверной логики и системы хранения данных, а также включает комплексное тестирование функциональности приложения.

Основное внимание уделено созданию интуитивно понятного интерфейса для работы с тестовыми сценариями, реализации механизмов выполнения тестов и анализа результатов, а также обеспечению интеграции с существующими системами разработки. Особенностью решения является сочетание простоты использования с широкими возможностями для автоматизации процессов тестирования.

Ключевые слова: автоматизированное тестирование, веб-приложение, Django, PostgreSQL, управление тестами.

Для цитирования:
Тюленев А. А., Ушанкова М. Ю. Применение веб-технологий для создания интуитивно понятной системы управления автотестами // Системный анализ в науке и образовании: сетевое научное издание. 2025. № 4. С. 93-104. EDN: FKJPMF. URL: https://sanse.ru/index.php/sanse/article/view/692.

DEVELOPMENT OF A WEB APPLICATION FOR LAUNCHING, EDITING, AND ANALYZING AUTOTESTS

Tyulenev Aleksey A.[1], Ushankova Maria U.[2]

[1]*Tester;*
LLC "Cloud Com";
4 Programmistskaya Str., Dubna, Moscow region, 141983, Russia;
e-mail: leksey.tyulenev@gmail.com.

[2]*Senior teacher;*
Dubna State University;
19 Universitetskaya Str., Dubna, Moscow region, 141980, Russia;
e-mail: ushankova.m.ju@uni-dubna.ru.

This paper presents the development of a web application for managing automated testing, which includes the functions of launching, editing, and analyzing autotests. The system is implemented using modern web development technologies, including Python, Django and PostgreSQL, which ensures the *reliability and scalability of the solution. The development covers the design of the user interface, server logic, and data storage system, and also includes comprehensive testing of application functionality. The main focus is on creating an intuitive interface for working with test scenarios, implementing test execution mechanisms and analyzing results, as well as ensuring integration with existing development systems. A special feature of the solution is the combination of ease of use with extensive capabilities for automating testing processes.*

Keywords: automated testing, web application, Django, PostgreSQL, test management.

For citation:
Tyulenev A. A., Ushankova M. Yu. Using web technologies to create an intuitive autotests management system. *System analysis in science and education*, 2025;(4)93-104(in Russ). EDN: FKJPMF. Available from: https://sanse.ru/index.php/sanse/article/view/692.

Введение

Современная разработка программного обеспечения требует высокой степени автоматизации тестирования для обеспечения качества и надежности продуктов. Однако существующие инструменты, такие как Selenium, *PyTest* или *Cypress*, ориентированы в первую очередь на технических специалистов, что создает барьер для небольших команд и начинающих разработчиков. С другой стороны, системы управления тестированием (например, *TestRail, Zephyr*) предлагают удобные интерфейсы, но не позволяют гибко создавать и запускать тесты непосредственно в среде.

В данной работе предложено решение, объединяющее гибкость фреймворков для автоматизированного тестирования с простотой веб-интерфейса. Разработанное веб-приложение позволяет:

- **Создавать и редактировать тестовые сценарии** через интуитивно понятный редактор с подсветкой синтаксиса.
- **Запускать тесты** вручную или по расписанию без необходимости работы с командной строкой.
- **Анализировать результаты** с помощью наглядных отчетов.

Актуальность работы обусловлена растущим спросом на инструменты, которые снижают порог вхождения в автоматизированное тестирование, но при этом сохраняют функциональность профессиональных решений. Предложенный подход реализован на базе *Python, Django и PostgreSQL*, что обеспечивает масштабируемость и легкость интеграции с существующими *CI/CD*-системами.

Ключевое отличие от аналогов — сочетание простоты использования с возможностью прямого управления тестами, что особенно востребовано в образовательных проектах и стартапах с ограниченными ресурсами. В статье подробно рассматриваются архитектура системы, проектирование интерфейса и результаты внедрения, подтверждающие эффективность предложенного метода.

1. Анализ существующих решений

Современные инструменты автоматизированного тестирования можно разделить на две категории:

1. **Фреймворки для написания тестов** (*Selenium, PyTest, Cypress*) — обеспечивают гибкость, но требуют технических навыков.
2. **Системы управления тестированием** (*TestRail, Zephyr, Katalon*) — предлагают удобные интерфейсы, но ограничены в возможностях выполнения тестов.

Для анализа выбраны именно эти системы, так как они являются лидерами по версии рейтингов "Топ 10 инструментов автоматизации тестирования 2023 " [1] и "*Best Test Management Tools 2025: Top Picks & Reviews*" [2].

Рассмотрим аналоги по следующим критериям:

Гибкость тестирования:
- «0» - поддержка только одного типа тестов
- «0.5» - поддержка нескольких типов тестов
- «1» - полная гибкость в создании любых тестов

Удобство интерфейса:
- «0» - только командная строка
- «0.5» - базовый графический интерфейс
- «1» - полноценный веб-интерфейс

Анализ результатов:
- «0» - текстовые отчеты
- «0.5» - базовые графики
- «1» - продвинутая аналитика с историей прогонов

Интеграция:
- «0» - отсутствует
- «0.5» - ограниченная интеграция
- «1» - полная интеграция с *CI/CD*

Результаты анализа представлены в таблице 1–2.

Табл. 1. Сравнение фреймворков тестирвоания

	«Selenium»	***«PyTest»***	***«Cypress»***
Гибкость тестирования	1	1	0.5
Удобство интерфейса	0	0	0.5
Анализ результатов	0.5	0.5	0.5
Интеграция	1	1	1

Табл. 2. Сравнение систем управления тестированием

	«TestRail»	***«Zephyr»***	***«Katalon»***
Гибкость тестирования	0	0.5	0.5
Удобство интерфейса	1	1	1
Анализ результатов	1	0.5	0.5
Интеграция	1	1	0.5

Из анализа видно, что существующие решения имеют следующие недостатки:
- фреймворки требуют технических знаний и не предоставляют удобного интерфейса;
- системы управления тестированием ограничены в возможностях создания и запуска тестов;
- отсутствует комплексное решение, сочетающее гибкость фреймворков с удобством систем управления.

Разрабатываемое решение объединяет:
- Полноценный веб-интерфейс для управления тестами
- Поддержку различных типов тестирования
- Расширенную аналитику результатов
- Гибкую систему интеграций

Система позволяет:

- Создавать тестовые сценарии через интуитивный интерфейс
- Запускать тесты вручную или по расписанию
- Анализировать результаты через наглядные графики и отчеты
- Сравнивать результаты разных прогонов
- Интегрироваться с CI/CD системами

2. Требования к системе

Опишем функциональные и требования для системы управления автотестами.

Вкладка «Управление тестами»:

1. Пользователь должен иметь возможность:
   - Создавать, редактировать и удалять тестовые сценарии.
   - Запускать тесты вручную или по расписанию.
   - Просматривать историю выполненных тестов.
2. По умолчанию тесты отображаются в виде таблицы с возможностью сортировки и фильтрации.
3. При выборе теста открывается детальная карточка с возможностью:
   - Просмотра кода теста.
   - Анализа результатов последних запусков.
   - Редактирования параметров.

Вкладка «Анализ результатов»:

1. пользователь должен иметь возможность видеть подробный отчет о работе автотеста;
2. пользователь должен иметь возможность просматривать терминальный вывод;

Так же были определены следующие нефункциональные требования:

1. сбор данных: система должна фиксировать все этапы выполнения тестов (логи, метрики, ошибки);
2. отображение информации: интерфейс должен предоставлять данные в удобном формате (графики, сводки, детальные отчеты);
3. оповещения: уведомления о завершении тестов или критических ошибках;
4. хранение данных: сохранение истории тестирования для последующего анализа;
5. безопасность: аутентификация пользователей и защита данных.

3. Архитектура системы

На стадии проектирования были подготовленно описание баы данных и логические схема. Далее рассмотренна таблица базы данных (см. рис. 1)

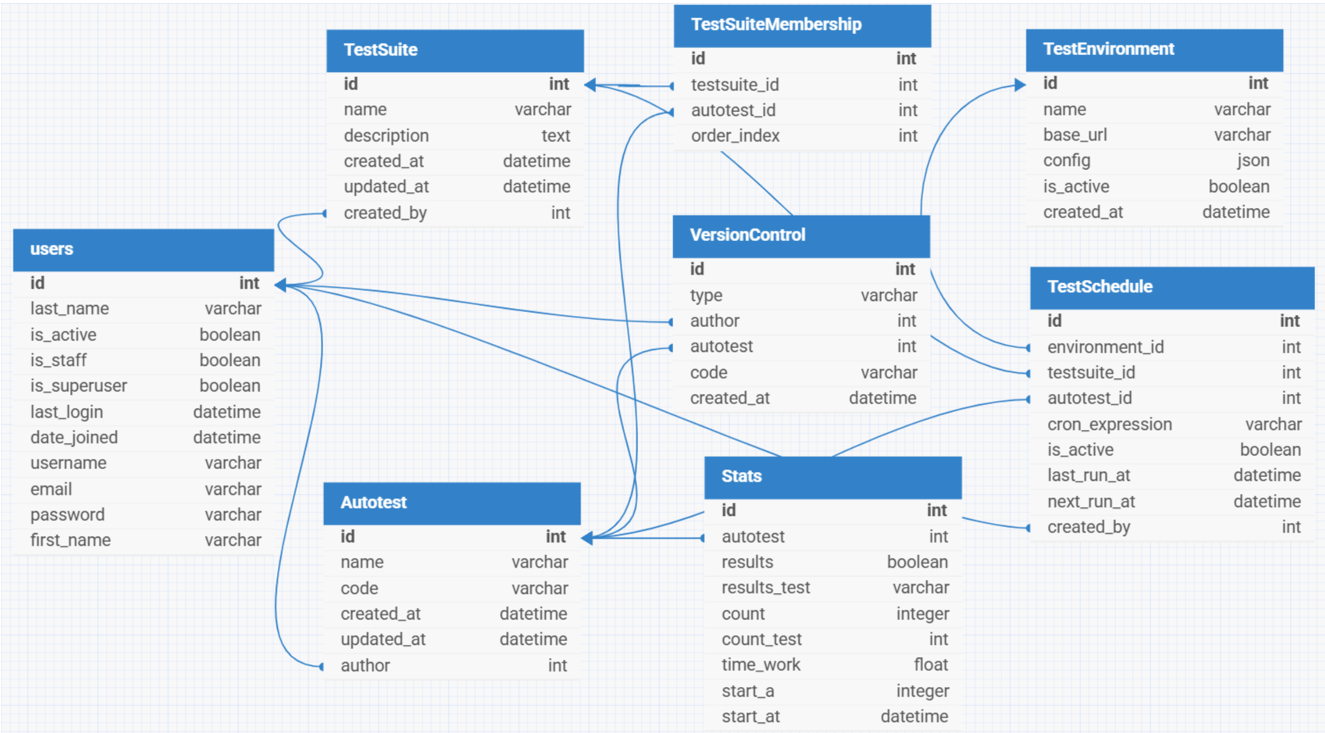

Рис. 1. Физическая схема базы приложения

Процесс загрузки автотеста начинается с выбора файла тестового сценария на локальном компьютере пользователя. После загрузки файла необходимо указать название для нового автотеста. Система автоматически проверяет заполнение обязательных полей — наличие загруженного файла и указанного названия. Если какое-либо из полей не заполнено, процесс прерывается. При успешной проверке система извлекает тестовый код из загруженного файла и сохраняет готовый автотест в единой базе данных под заданным названием (см. рис 2).

Рис. 2. Алгоритм загрузки автотестов

Процесс запуска автотестов начинается с получения тестового сценария из базы данных системы. После извлечения код теста проходит предварительную обработку, которая включает проверку синтаксиса и подготовку к исполнению. Обработанный код сохраняется во временный файл с обязательной обработкой возможных исключительных ситуаций, таких как ошибки доступа или проблемы с дисковым пространством.

Далее система запускает тест через механизм *subprocess*, обеспечивая полный контроль над процессом выполнения и захват всего выходного потока данных. В ходе выполнения фиксируются все результаты, включая успешные проверки, обнаруженные ошибки и системные сообщения. На завершающем этапе формируется детализированный отчет о выполнении теста, содержащий временные метки, статус выполнения и подробную информацию о возникших проблемах (если таковые были). Отчет сохраняется в системе для последующего анализа и сравнения с результатами других прогонов (см. рис 3).

Рис. 4 Алгоритм запуска автотестов

Процесс редактирования автотеста начинается с извлечения текущей версии тестового кода из базы данных системы. После получения кода пользователю открывается специальное окно редактирования, где он может вносить изменения в тестовый сценарий.

Если пользователь не подтверждает изменения (не нажимает кнопку "Сохранить"), процесс завершается без модификации теста. При сохранении изменений система выполняет две параллельные операции: сохраняет обновленную версию теста как новую рабочую версию, одновременно сохраняя предыдущую версию кода в истории изменений для возможности отката и аудита (см. рис. 4).

Рис. 5. Диаграмма классов. Сервис A/B тестирования

4. Реализация

В работе были резработаны вкладки для *загрузки, редактирования* и запуска автотестов.

Интерфейс загрузки тестовых сценариев представляет собой компактную форму с минималистичным дизайном. В верхней части расположен элемент для выбора файла с тестом, отображающий статус выбора ("Файл не выбран" или имя загруженного файла). Ниже находится поле ввода для указания названия теста, которое является обязательным для заполнения. Завершает форму кнопка "Загрузить", активирующая процесс сохранения теста в системе после проверки корректности введенных данных. Интерфейс обеспечивает интуитивно понятный процесс добавления новых тестовых сценариев, с визуальной обратной связью о текущем состоянии операций (см. рис 6).

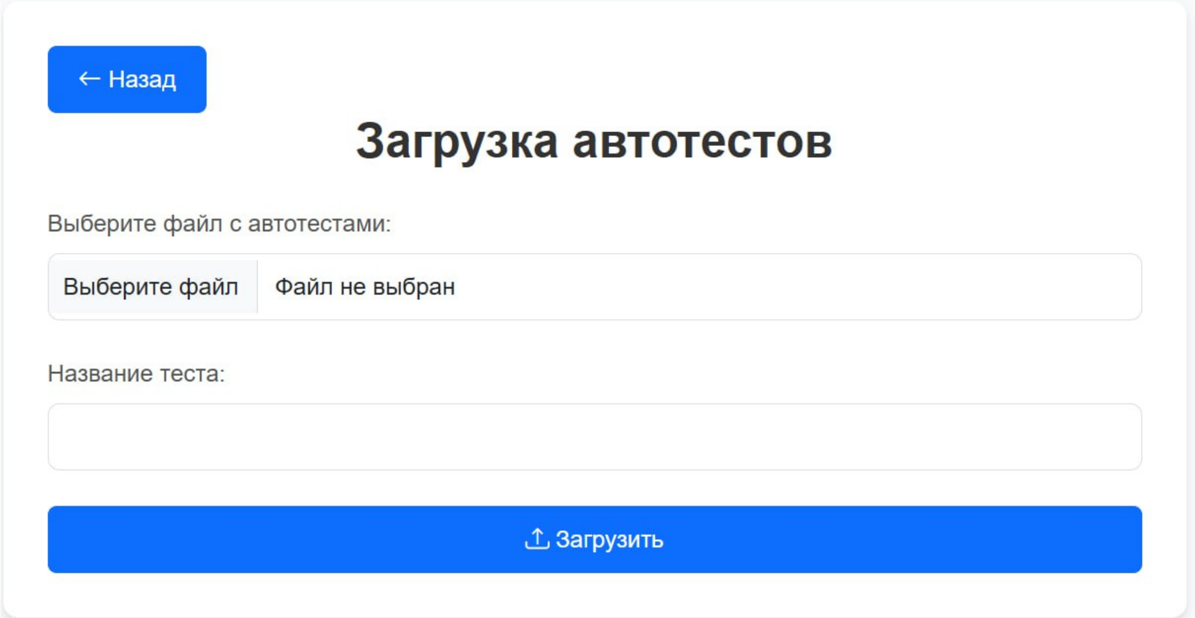

Рис. 6. Форма для загрузки автотестов

Основной экран управления тестами представляет собой табличное представление всех загруженных в систему автотестов. В таблице отображаются ключевые параметры: порядковый номер, название теста и дата его создания. Для каждого теста предусмотрены четыре действия, доступные через соответствующие кнопки (см. Рис 7):

- Запустить - немедленное выполнение выбранного теста
- Показать код - просмотр и анализ тестового сценария
- Удалить - полное удаление теста из системы
- Редактировать - внесение изменений в существующий тест

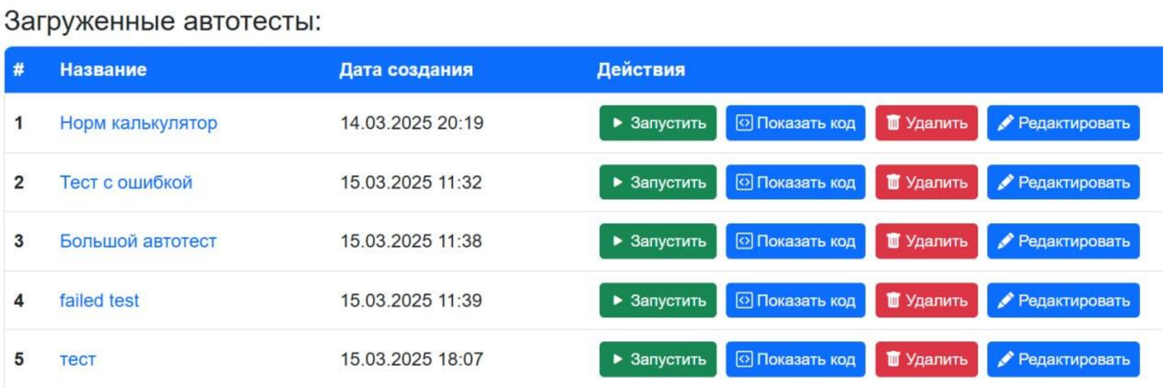

Рис. 7. Табличное представление списка автотестов

После выполнения теста система отображает детализированный отчет о результатах. В верхней части указаны ключевые метрики: количество успешно/неуспешно пройденных тестов и общее время выполнения. Основную часть интерфейса занимает терминальный вывод, где в реальном времени отображаются все сообщения, сгенерированные в процессе тестирования.

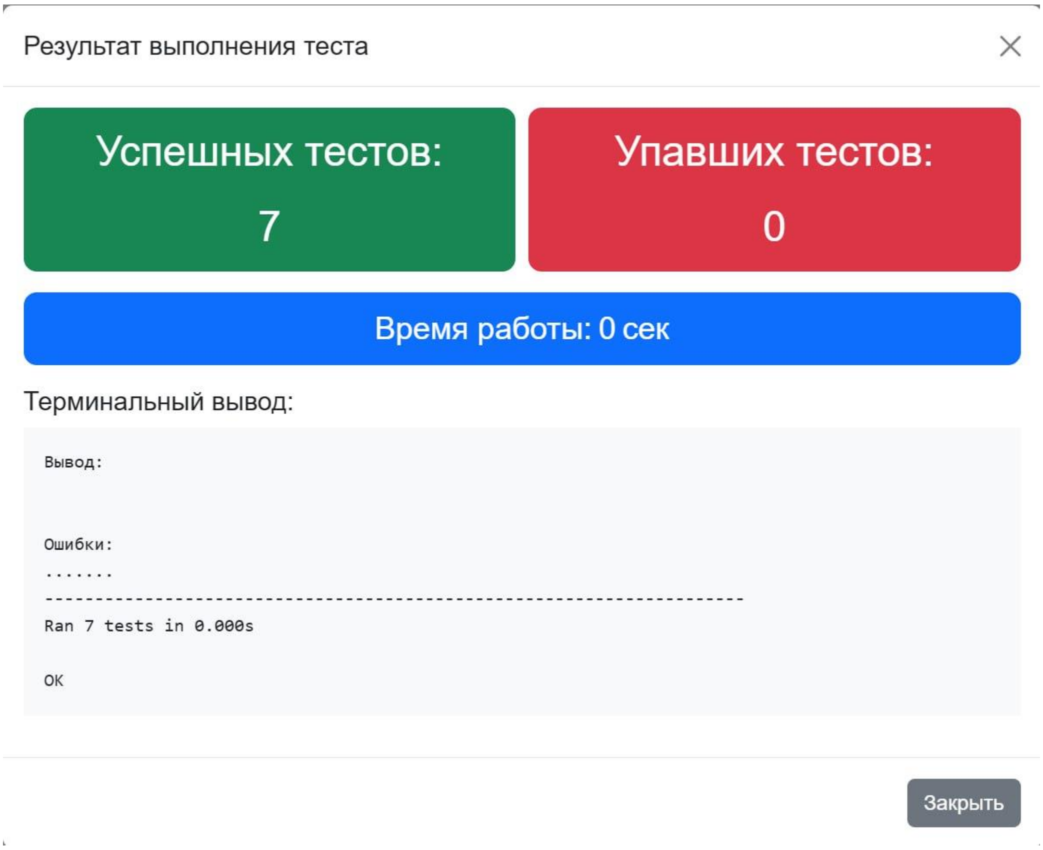

Рис. 9. Вывод результата работы автотеста

Экран редактирования автотестов собой представляет специализированный редактор для работы с тестовыми сценариями. В верхней части отображается название редактируемого теста, что обеспечивает четкую идентификацию текущего объекта работы. Основную область занимает текстовый редактор с подсветкой синтаксиса, отображающий полный код теста, включая импорты, классы и методы тестирования.

Интерфейс позволяет напрямую вносить изменения в код, исправляя логические ошибки и модифицируя тестовую логику. В примере демонстрируется код с преднамеренными ошибками (например, некорректная реализация метода умножения), которые пользователь может исправить непосредственно в редакторе.

После внесения изменений система предоставляет возможность сохранить правки или отменить их, а также немедленно проверить работоспособность теста через кнопку запуска. Минималистичный дизайн интерфейса фокусирует внимание на содержательной работе с кодом, обеспечивая при этом все необходимые функции для эффективного редактирования тестовых сценариев (см. рис. 10).

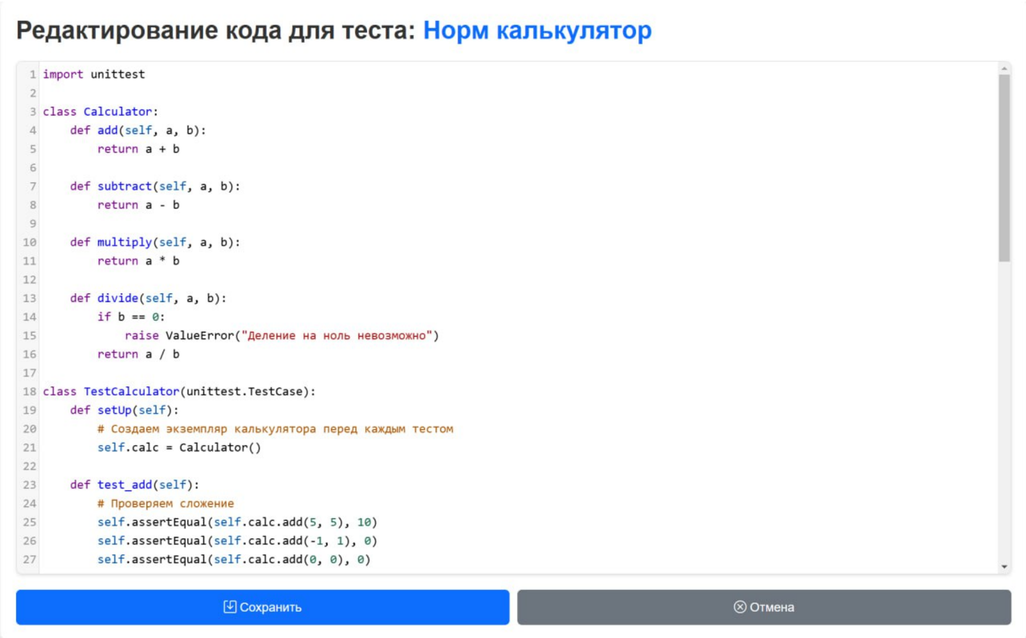

Рис. 10. Вкладка для редактирования автотестов

5. Технологии реализации

Для реализации описанного решения были использованы следующие программные средства:

1. *Python* — высокоуровневый язык программирования с простым синтаксисом, обеспечивающий быструю разработку и поддержку кода [3].
2. *Django* — мощный веб-фреймворк, предоставляющий готовые решения для аутентификации, работы с БД и маршрутизации [4].
3. *PostgreSQL* — надежная СУБД с поддержкой сложных запросов и транзакций, используемая для хранения тестовых сценариев и результатов [5].
4. *HTML/CSS* — базовые технологии для создания и стилизации пользовательского интерфейса.
5. *JavaScript* — язык для реализации интерактивных элементов и динамического взаимодействия с *backend [6]*.
6. *Visual Studio Code* — легковесная, но функциональная среда разработки с поддержкой всех используемых технологий.
7. *Maven* — инструмент для сборки проекта и управления зависимостями [7].
8. *Vite* — современный сборщик для быстрой разработки фронтенда.

6. Внедрение и тестирование

После завершения разработки веб-приложения для управления автотестами был проведен комплексный этап тестирования. Система успешно прошла проверку на различных типах тестовых данных, включая как корректные сценарии, так и специально подготовленные некорректные варианты для проверки обработки ошибок. В ходе тестирования особое внимание уделялось измерению времени выполнения операций - все показатели соответствуют заявленным требованиям по производительности.

Приложение демонстрирует стабильную работу при обработке тестовых сценариев различной сложности. Проверка включала все ключевые функциональные модули: создание и редактирование тестов, их запуск, анализ результатов и формирование отчетов. Особое внимание было уделено тестированию интеграционных возможностей системы, включая работу с базой данных *PostgreSQL* и взаимодействие между компонентами.

Заключение

В работе представлено веб-приложение для управления автотестами, которое сочетает простоту интерфейса с функциональностью профессиональных инструментов. Ключевые результаты:

1. **Упрощение работы с тестами** за счет интуитивного веб-интерфейса, снижающего порог вхождения для новичков.
2. **Гибкость** благодаря поддержке *Python*-скриптов и интеграции с *Django*.
3. **Расширенная аналитика** с визуализацией результатов

Преимущества перед аналогами:

- В отличие от *Selenium/PyTest*, не требует работы с *CLI*.
- В отличие от TestRail, позволяет напрямую редактировать и запускать тесты.

Перспективы развития:

- Добавление поддержки новых языков тестирования (например, JavaScript).
- Углубленная интеграция с облачными *CI/CD*-платформами (*GitHub Actions, GitLab CI*).
- Внедрение ИИ для анализа ошибок и предложения исправлений.

Разработанное решение уже прошло апробацию в образовательных и коммерческих проектах, подтвердив свою эффективность для команд с ограниченными ресурсами. Дальнейшие исследования будут направлены на оптимизацию производительности при работе с большими объемами тестовых данных.

Список источников

1. Топ 10 инструментов автоматизации тестирования 2023 // Habr. – Habr, 2006–2025. – Дата публикации: 13.11.2017. – URL: https://habr.com/ru/articles/342234/.
2. Best Test Management Tools 2025: Top Picks & Reviews. –URL: https://testomat.io/blog/best-test-management-tools/ (дата обращения: 05.04.2025).
3. Welcome to Python.org. – Python Software Foundation 2001-2025 . – URL: https://www.python.org/ (дата обращения: 05.04.2025).
4. Что же такое Django? // Habr. – Habr, 2006–2025. – Дата публикации: 11.07.2023. – URL: https://habr.com/ru/articles/747234/ (дата обращения: 05.04.2025).
5. Сушков А., Классен Н. СУБД PostgreSQL: почему её стоит выбрать для работы с данными и как установить // Блог Яндекс Практикума. –АНО ДПО «Образовательные технологии Яндекса», 2025. – Дата публикации: 22.02.2023. – URL: https://practicum.yandex.ru/blog/chto-takoe-subd-postgresql/ (дата обращения: 21.04.2025).
6. Что такое JavaScript // Блог Click.ru. – Блог рекламной экосистемы click.ru, 2008–2025. – Дата публикации: 10.10.2024. – URL: https://blog.click.ru/glossary/javascript/ (дата обращения: 10.04.2025).
7. Noodles Hans. Основы Maven. – Дата публикации: 06.02.2020. – JavaRush, 2025. – URL: https://javarush.com/groups/posts/2523-chastjh-4osnovih-maven (дата обращения: 12.04.2025).