СЪВРЕМЕННИ ИНСТРУМЕНТИ ЗА ТЕСТВАНЕ НА PHP БАЗИРАНИ УЕБ ПРИЛОЖЕНИЯ

Симеон Владков
Икономически университет – Варна/Катедра „Информатика“, Варна, България,
simeon.vladkov@ue-varna.bg

РЕЗЮМЕ

Създаването на уеб приложения е сред най-динамично развиващите се направления в сферата на софтуерните разработки. Всеки ден възникват нови технологии за разработка на уеб приложения, които надграждат и подобряват вече съществуващите технологии. Един от най-често използваните и предпочитани от програмистите езици за изграждане на уеб приложения е PHP. Езикът от години поддържа над три четвърти от всички съществуващи уеб сайтове и уеб приложения в интернет. Въпреки нарастващия интерес към други езици за уеб програмиране като JavaScript и Python, PHP остава любим избор на програмисти от цял свят, поради редовната поддръжка от страна на екипа разработчици, който поддържа езика и създава новите му версии. Лесната интеграция с различни уеб сървъри и системи за управление на бази от данни, както и богатото разнообразие от работни рамки и библиотеки, превръщат PHP в подходящ инструмент за изграждане на комплексни уеб приложения. При изграждането на комплексни софтуерни системи, подсигуряването на функционалноста, надеждността и цялостното качество на софтуерната разработка са от изключително значение. За целта, специалистите използват софтуерни тестове, които измерват множество показатели и дават обективна оценка за състоянието на тествания софтуер. За достигане до тази обективна оценка, тестващите използват набор от инструменти, наричан тестов пакет, който обединява различни софтуерни продукти за софтуерно тестване, с цел обстойно и цялостно тестване на разработвания софтуер. В доклада се разглеждат различните подходи и типове тестове при софтуерното тестване. Представени са съвременни библиотеки и работни рамки за софтуерно тестване на уеб приложения, създадени с PHP. Анализирани са различните аспекти на тези инструменти, дефинирайки силните и слабите им страни, и сравнявайки ефективността им в различни ситуации. Докладът има за цел да демонстрира стойността, която носи софтуерното тестване на крайния продукт, както за потребителите му, така и за разработчиците.

КЛЮЧОВИ ДУМИ: *уеб приложения, софтуерни тестове, PHP*

MODERN TESTING TOOLS FOR PHP WEB APPLICATIONS

Simeon Vladkov
University of Economics – Varna/Department of “Informatics”, Varna, Bulgaria,
simeon.vladkov@ue-varna.bg

ABSTRACT

The creation of web applications is among the most dynamically evolving fields in software development. Every day, new technologies for web application development emerge, building upon and enhancing existing ones. One of the most commonly used and preferred programming languages for building web applications is PHP. For years, PHP has supported over three-quarters of all existing websites and web applications on the internet. Despite the growing interest in other web programming languages like JavaScript and Python, PHP remains a favourite choice for developers worldwide due to the consistent support from its development team, which maintains the language and releases new versions. Its easy integration with various web servers and database management systems, as well as the rich variety of frameworks and libraries, make PHP a suitable tool for building complex web applications. While developing complex software systems, ensuring

functionality, reliability, and the overall quality of the software is crucial. For this purpose, specialists use software tests to measure various indicators and provide an objective evaluation of the tested software. To achieve this objective assessment, testers use a set of tools known as a testing suite, which combines various software testing products to thoroughly and comprehensively test the developed software. This report examines different approaches and types of tests in software testing. It introduces modern libraries and frameworks for testing PHP web applications. The various aspects of these tools are analyzed, defining their strengths and weaknesses, and comparing their effectiveness in different situations. The report aims to demonstrate the value that software testing brings to the final product, both for its users and its developers.

KEYWORDS: *web applications, software tests, PHP*

ВЪВЕДЕНИЕ

При разработката на уеб приложения, качеството на разработваните приложения е от изключителна важност. За да осигурят високо качество на разработките си, специалистите използват софтуерни тестове, благодарение на които създават надеждни и функционални приложения. Тестването гарантира, че едно приложение отговаря на изискванията, функционира според очакванията и предоставя оптимално потребителско изживяване, дори при непредсказуеми условия. Програмистите и тестерите изследват нуждите на разработваните приложения и според тях подбират подходящи методи за тестване и типове тестове, с които да подсигурят функционалността и устойчивостта на софтуера. На база избраните методи и типове тестове, специалистите селектират различни инструменти за реализиране на тестовете, чрез които се създава тестовият пакет на приложението. В контекста на PHP уеб приложенията, разработчиците разполагат с богат набор от инструменти и библиотеки, чрез които могат да реализират различни типове софтуерни тестове. Тези инструменти се допълват като функционалност, като по този начин гарантират ефикасността на тестовия пакет, който изграждат и реализират.

1. ОСНОВНИ ТИПОВЕ СОФТУЕРНИ ТЕСТОВЕ

Софтуерните тестове се разделят в две основни категории – функционални и нефункционални тестове (Elijah, 2019). Функционалните софтуерни тестове подсигуряват правилното функциониране на софтуерната разработка според зададените към нея изисквания. Нефункционалните тестове не тестват пряко функционалността на приложенията, а се фокусират над други фактори, които влияят върху качеството на софтуерната разработка.

Функционалните тестове валидират начинът на работа на приложението и го сравняват с функционалните му спецификации. При функционалния подход, тестовете варират според обхвата си и се надграждат йерархично с цел тестване на цялата система (Tahvili, 2022). Най-често се използват следните типове функционални тестове:

- **Единични тестове** (unit tests) – изпълняват се върху най-малките градивни единици от кодовата база – единични функции или методи на класове. Позволяват на разработчиците да тестват изолирано всеки компонент, гарантирайки правилното му функциониране.
- **Интеграционни тестове** (integration tests) – тестват взаимодействието между отделните компоненти на приложението. Подсигуряват правилното движение на данни през приложението и успешната комуникация между различните части от кодовата база.
- **Регресионни тестове** (regression tests) – тестват функционалността на приложението след промени и добавяне на нови елементи към кодовата база. Предотвратяват неочаквани ефекти при обновяване на кода и гарантират качеството на приложението при

продължаваща разработка.

- **Тестове за разум** (sanity tests) – изпълняват се най-често след отстраняване на неизправности и бъгове в кодовата база. Позволяват да се оцени ефикасността и функционалността на приложението след изчистване на откритите грешки.
- **Системни тестове** (system tests) – тестват цялостната функционалност на софтуерната разработка, като проверяват дали приложението изпълнява напълно дефинираните изисквания. Изпълняват се след тестване на единичните компоненти на приложението.
- **Тестове за приемане** (acceptance tests) – определят дали приложението отговаря на потребителските нужди. Изпълняват се най-често от крайни потребители, в последните фази на функционално тестване, преди въвеждане на софтуера в експлоатация.

За разлика от функционалните тестове, нефункционалните тестове не следват стриктна йерархия или строга последователност при изпълнение. Всеки тип нефункционални тестове отговаря за различен аспект, свързан с качеството на приложението (Najihi, 2022). В практиката се използват разнообразни подходи и типове нефункционални тестове, сред които:

- **Тестове за представяне** (performance tests) – тестват скоростта и стабилността на приложението както при нормално натоварване, така и при стресови обстоятелства. Подпомагат идентифицирането на слаби (тесни) места в софтуерната разработка.
- **Тестове за натоварване** (load tests) – симулират различни типове и обеми натоварване за приложението, като по този начин тестват способността му да работи в различни ситуации, които могат да възникнат в процеса на експлоатация.
- **Тестове за скалируемост** (scalability tests) – тестват възможността за надграждане на приложението при нужда, породена от промяна в изискванията. Използват се от специалистите за изграждане на стратегия за развитие при приложения, чиято използваемост нараства бързо.
- **Тестове за сигурност** (security tests) – тестват устойчивостта на приложението към хакерски атаки. Чрез тях се идентифицират и отстраняват слабости в сигурността на приложението, като по този начин се гарантира безопасността на данните в системата.
- **Тестове за съответствие** (compliance tests) – валидират съответствието на приложението с различни закони и легални норми, както и съответствието с браншови стандарти и вътрешни политики на крайния клиент.
- **Тестове за използваемост** (usability tests) – тестват колко лесно и удобно за използване е приложението от крайния потребител. Тестват се основно потребителските интерфейси и цялостното потребителско изживяване при работа със софтуера.

Функционалните и нефункционалните тестове се различават по множество критерии (Таблица 1). Именно тези разлики позволяват на специалистите да тестват обстойно софтуерните си разработки. Тестването на множество различни аспекти на приложението води до цялостно подобряване на качеството на софтуера. В зависимост от размерите и целите на приложението, програмистите разработват подходяща стратегия за тестване, която се реализира от различни подходи и инструменти. В екосистемата на PHP, разработчиците разполагат с богато разнообразие от инструменти, които позволяват изпълняването на различни типове функционални и нефункционални тестове. Подборът на инструменти за тестовия пакет зависи както от разработвания софтуер, така и от възможностите и ограниченията на специалистите, които го създават.

Таблица 1

Критерий	Функционални тестове	Нефункционални тестове
Основна цел	Тестват дали приложението функционира според зададените изискванията	Тестват качествени атрибути на приложението като сигурност, скалируемост и използваемост
Насока	Функционалности и операции на приложението	Състоянието на приложението в различни условия
Създаване на тестове	На база функционалните и потребителските изисквания към приложението	На база стандарти за ползваемост, легални рамки и добри практики в сигурността
Начин на изпълняване на тестове	Основно ръчно, автоматизирано за по-комплексните подходи (системни тестове, тестове за приемственост)	Основно автоматизирано
Моменти на изпълняване	По време на разработката, в тестови фази след разработката	След функционалното тестване, преди въвеждане в експлоатация

Източник: Собствена разработка

2. СОФТУЕРНО ТЕСТВАНЕ НА PHP УЕБ ПРИЛОЖЕНИЯ

PHP е сред най-използваните програмни езици за създаване на динамични уеб приложения. Гъвкавостта на PHP, възможността му за интеграция с различни уеб сървъри и системи за управление на бази данни, както и богатата му екосистема от библиотеки и инструменти, го превръщат в любим избор на програмистите при разработка на комплексни уеб базирани системи.

Комплексността на съвременните уеб приложения увеличава възможността за възникване на бъгове и уязвимости. Това превръща обстойното софтуерно тестване в неизменна част от жизнения цикъл на разработка на приложението. За да се създаде качествен тестов пакет, специалистите трябва да изберат инструменти за тестване, които от една страна да тестват успешно всички функционални изисквания към разработваното приложение, и от друга страна да дават адекватна оценка на ключови качествени показатели като сигурност и използваемост. В екосистемата на PHP съществуват множество утвърдени библиотеки и работни рамки за софтуерно тестване, с големи потребителски бази и редовна поддръжка от страна на разработчиците. Тези инструменти се използват в различни комбинации с цел оптимално тестово покритие на приложението. Сред най-често използваните са:

- **PHPUnit** – библиотека за създаване на unit тестове. Позволява на разработчиците да създават и изпълняват тестове за единичните компоненти на приложението (Bergmann, 2024). Предлага лесно организиране на тестовете в тестови пакети и генерира детайлни отчети при тестване. Тестовете могат да се изпълняват ръчно, както и автоматизирано, благодарение на лесната интеграция във вече съществуващи схеми за продължителна интеграция (continuous integration) по време на разработка. Библиотеката също така се интегрира лесно с други инструменти за софтуерно тестване, които я използват като основа и надграждат върху функционалността и.
- **Codeception** – работна рамка за тестване на PHP приложения. Поддържа множество типове функционални тестове и позволява цялостно тестване на функционалностите на разработката, от unit тестове до системни тестове. Предлага добри възможности за симулация на потребителска интеракция с приложението, което я превръща в подходящ инструмент за изпълняване на acceptance тестове (Codeception, 2024). При създаване на

тестовете се използва обектно-ориентиран подход, със семантичен синтаксис за удобство на програмистите. Интегрира се лесно с работни рамки за създаване на PHP уеб приложения като Laravel и Symphony, както и с други инструменти за тестване като PHPUnit.

- **Selenium** – инструмент за автоматизирано тестване на уеб приложения в различни уеб браузери (Software Freedom Conservancy, 2024). Използва се за автоматизация комплексни части от тестовите пакети, които най-често симулират потребителско поведение. Това прави Selenium подходящ инструмент за изпълняване на acceptance тестове, тестове за натоварване и тестове за използваемост в различни среди, като по този начин се гарантира съвместимостта на приложението с всички съвременни уеб браузери.
- **Behat** – инструмент за създаване на софтуерни тестове, който използва поведенческия подход на разработка (behavior-driven development). Този подход позволява писането на тестове на език, който лесно се чете от неспециалисти. Използва се синтаксисът Gherkin, чрез който на човешки четим език се описват различни сценарии, на база на които се генерират тестове (Kudryashov, 2024). По този начин се улеснява колаборацията между програмистите, инженерите по качеството и нетехническите лица.
- **Kahlan** – работна рамка за създаване на функционални и нефункционални тестове, която също използва поведенчески подход при разработка на тестовете. Предлага удобен синтаксис наречен describe-it и лесна за навигиране файлова структура (Kahlan team, 2024). Позволява на разработчиците да използват динамично имитиране (dynamic mocking), както и да заместват части от тествания код по време на изпълнение на тестовете. Инструментът генерира детайлни тестови артефакти, в които се съдържа информация за изпълнените тестове.
- **BrowserStack** – облачно-базирана платформа за софтуерно тестване. Позволява използването на облачни ресурси за тестване на уеб приложения в различни браузери, на различни устройства и операционни системи (Browserstack, 2024). Инструментът позволява на екипи с ограничени хардуерни ресурси да създават комплексни стратегии за тестване, които гарантират правилно функциониране на разработваните приложения и повишават качеството на потребителското изживяване.

Съвременните инструменти и рамки за тестване на PHP уеб приложения предлагат разнообразие от функционалности, които обхващат различни аспекти на софтуерното качество. От тестване на отделни компоненти с PHPUnit, до създаване на комплексни автоматизирани тестови пакети със Selenium или BrowserStack, тези инструменти дават възможност на разработчиците да създават устойчиви, сигурни и високопроизводителни приложения. Чрез интегрирането на такива инструменти в процеса на разработка, екипите могат да минимизират рисковете, да подобрят надеждността и да предоставят оптимално потребителски изживявания.

ЗАКЛЮЧЕНИЕ

На база представените методи за софтуерно тестване и инструменти за реализацията им, можем да заключим, че интегрирането на стратегия за тестване в процеса на разработка на уеб приложения с PHP, е задължително за успешното изграждане на функционални софтуерни разработки с високо качество. Софтуерното тестване е основен фактор за успешна софтуерна разработка, като гарантира, че PHP уеб приложенията са не само функционални, но и устойчиви, скалируеми и лесни за използване от потребителите. Чрез комбинирането на различните

видове тестове и използването на подходящите инструменти за реализирането им, разработчиците и експертите по качество могат да създават приложения с дългогодишен експлоатационен период, които отговарят на високи стандарти за качество и предоставят стойност на потребителите.

REFERENCES / ИЗПОЛЗВАНА ЛИТЕРАТУРА

1. Bergmann, S., (2024) *PHPUnit.* [Online]
   Available at: https://phpunit.de/documentation.html
   [Accessed 18 11 2024].
2. Browserstack, (2024) *Browserstack.* [Online]
   Available at: https://www.browserstack.com/docs/
   [Accessed 18 11 2024].
3. Codeception, (2024) *Codeception.* [Online]
   Available at: https://codeception.com/docs/Introduction
   [Accessed 18 11 2024].
4. Elijah, A., (2019) Functional vs. Non-Functional Testing: Balancing Quality in Agile Teams. *International Journal of Advanced Engineering Technologies and Innovations,* 1(4), pp. 251-257.
5. Kahlan team, (2024) *Kahlan.* [Online]
   Available at: https://kahlan.github.io/docs/
   [Accessed 18 11 2024].
6. Kudryashov, K., 2024. *Behat.* [Online]
   Available at: https://docs.behat.org/en/latest/guides.html
   [Accessed 18 11 2024].
7. Najihi, S., (2022) Software Testing from an Agile and Traditional view. *Procedia Computer Science,* Volume 203, pp. 755-782.
8. Software Freedom Conservancy, (2024) *Selenium.* [Online]
   Available at: https://www.selenium.dev/documentation/
   [Accessed 18 11 2024].
9. Tahvili, S., (2022) *Artificial Intelligence Methods for Optimization of the Software Testing Process.* s.l.:Academic Press.