УДК 004.415.532 doi:10.20998/2413-4295.2019.05.12

АВТОМАТИЗОВАНЕ ТЕСТУВАННЯ ВЕБ-ДОДАТКІВ З РІЗНОРІВНЕВОЮ АРХІТЕКТУРОЮ

Г. М. КОДОЛА*, Н. С. ВОЛИНЕЦЬ, І. В. СЕРБУЛОВА

кафедра інформаційних систем, ДВНЗ УДХТУ, Дніпро, УКРАЇНА
**e-mail: gkodola@gmail.com*

АНОТАЦІЯ *Веб-додатки відіграють важливу роль в житті нашого суспільства. Вони застосовуються в таких секторах, як бізнес, охорона здоров'я та державне управління. Від якості таких додатків може залежити не лише зручність користувачів, але і функціонування організацій. Тестування є найбільш широко використовуваним і ефективним підходом для забезпечення якості та надійності програмного забезпечення, включаючи веб-додатки. Однак веб-додатки дуже відрізняються від традиційного програмного забезпечення, оскільки вони включають в себе динамічне створення та інтерпретацію коду, а також реалізацію конкретного режиму взаємодії на основі навігаційної структури веб-програми. Автоматизоване тестування – це автоматичне виконання набору тестів. Створивши цей набір один раз, його можна використовувати кожного разу після внесення деяких змін у веб-додаток. Крім того, сучасні веб-додатки побудовані на основі багаторівневої архітектури. Тому, щоб перевірити загальну поведінку веб-додатків, потрібно скласти комплекс методів тестування. Автоматизація тестування не може бути реалізована без відповідних інструментів. Саме вони визначають, як буде здійснюватися тестування та чи можуть бути досягнуті переваги автоматизації. Інструменти автоматизації тестування є найважливішим компонентом у інструментальному ланцюжку розробки. В статті було проаналізовано існуючі програмні засоби, які використовуються для автоматизованого тестування, та обрано серед них для кожного рівня веб-додатку ті, які зможуть забезпечити високий рівень безпеки і мінімізувати ймовірність помилок або збоїв в роботі програми. Для досягнення цієї цілі були розглянуті таки види програмних засобів: системи управління версіями; системи відстежування помилок; засоби автоматичного тестування; засоби для автоматизованого тестування навантаження; програмне забезпечення безперервної інтеграції. На їх основі було складено комплекс автоматизації тестування веб-додатку, який дозволить без зайвих складнощів проводити індивідуальні модифікації системи і значно зменшить кількість помилок в процесі доробки системи іншими спеціалістами. В результаті реалізації автоматизованого тестування веб-проекту був отриманий практичний досвід створення автоматизованої системи тестування веб-додатків за допомогою системи контролю версій (GIT) Bitbucket і системи безперервної інтеграції (CI) Jenkins.*

Ключові слова: *автоматизоване тестування веб-додатків; різнорівнева архітектура; безпека даних; інтеграція коду; мінімізація помилок*

AUTOMATED TESTING OF WEB APPLICATIONS WITH MULTILEVEL ARCHITECTURE

G. KODOLA*, N. VOLYNETS, I. SERBULOVA

Information Control Systems and Technology, The Ukrainian State Chemical-Technological University, Dnipro, UKRAINE

ABSTRACT *Web applications play an important role in the life of our society. They are applied in sectors such as business, health care and public administration. The quality of such applications can depend not only on user convenience but also on the functioning of organizations. Testing is the most widely used and effective approach to ensuring the quality and reliability of software, including web applications. However, web applications are very different from traditional software because they include dynamic creation and interpretation of the code, as well as the implementation of a specific interaction mode based on the navigation structure of the web application. Automated testing is an automatic execution of a set of tests. Having created this set once, you can use it every time after making some changes to the web application. In addition, modern web applications are built on the basis of multi-level architecture. Therefore, to test the overall behavior of web applications, you need to complete a set of testing methods. Automation testing cannot be implemented without the appropriate tools. It determines how they will be tested and whether the benefits of automation can be achieved. Test automation tools are the most important component in the development toolchain. The purpose of the work was to analyze the existing software tools used for automated testing, to apply among them for each level of the web application those that can provide a high level of security and minimize the likelihood of errors or failures in the program. To achieve this goal, the following kinds of software were considered: version control systems; error tracking systems; automatic testing tools; tools for automated load testing; continuous integration software. On their basis, a complex of testing automation of the web application was made, which would allow without any extra complexity to carry out individual modifications of the system and significantly reduce the number of errors in the process of updating the system by other specialists. As a result of the automated testing of the web project, practical experience was gained with the creation of an automated web application testing system using the Bitbucket version control system (GIT) and the Jenkins Continuous Integration System (CI).*

Keywords*: automated testing of web applications; multi-level architecture; data security; code integration; minimize errors*

Вступ

Важливість автоматизації тестування у веб-розробках походить від широкого використання веб-додатків та пов’язаної з нею потреби у якості коду. Автоматизація тестування вважається вирішальною для забезпечення рівня якості, що очікується

ISSN 2079-5459 (print)
ISSN 2413-4295 (online)

користувачами, оскільки вона може заощадити багато часу в тестуванні, а також допомагає розробникам випускати веб-програми з меншою кількістю дефектів. Основною перевагою автоматизації тестування є швидке, автоматичне виконання набору тестів після внесення деяких змін у веб-додаток. Крім того, сучасні веб-додатки використовують багаторівневу архітектуру, де реалізація розподіляється по різних шарах і запускається на різних машинах. З цієї причини, щоб перевірити загальну поведінку веб-додатків, потрібні наскрізні методи тестування.

Тестування веб-проектів необхідно застосовувати на всіх етапах життєвого циклу проекту (рис. 1). Набір методів тестування залежить від специфіки структури та призначення веб-проекту.

Рис. 1 – Схема життєвого циклу веб-додатку

Таким чином, комплекс автоматизації тестування веб-додатку дозволить без зайвих складнощів проводити індивідуальні модифікації системи і дозволить мінімізувати кількість помилок в процесі доробки системи іншими спеціалістами.

Мета роботи

Розробити методику автоматизованого тестування веб-додатків з різнорівневою архітектурою для забезпечення високого рівня безпеки і мінімізації ймовірності помилок або збоїв в роботі програми, що базується на використанні сучасних засобів і технологій розробки і супроводу веб-проектів, які розробляються командою розробників з розподіленою спільною роботою.

Викладення основного матеріалу

Тестування веб-додатків – це складний процес, тому що його труднощі примножені усіма розподіленими компонентами системи, що взаємодіють з додатком.

Огляд літератури виявив велику кількість досліджень та розробок присвячених автоматизованому тестуванню веб-додатків. Універсальність веб-додатків є переважною особливістю, яка робить тестування веб-додатків складною задачею [1]. Основною особливістю тестування веб-додатків, що відрізняє його від традиційного тестування десктопних програм, є те, що веб-програми абсолютно різнорідні за природою на різних рівнях [2]. В роботах [3,4] наведено переваги автоматизованого тестування. Огляд сучасних методів та видів тестування веб-додатків детально розглянуто в роботах [5,6]. Також запропоновано використання системних сценаріїв для оптимізації процесу тестування в [7] і розглянуто питання тестування безпеки та наявності вразливостей веб-додатків [8]. В роботах [6,9] докладно розглянуто існуючі на ринку програмні засоби для автоматизації тестування. Проаналізовано методики створення тестів окремих модулів, їх відтворення та автоматизація в [10,11]. Однак цілий ряд питань, пов'язаних з практичною реалізацією зазначених підходів, як і раніше залишається актуальним.

Розглянемо складнощі, що можуть виникнути в ході розробки проекту, та способи їх вирішення за допомогою тестів [6,12]:

– безпека персональних даних і даних яка обробляє і зберігає програмний продукт. Тестування дозволяє здійснювати додаткові перевірки в можливих вразливих містах в коді програми, запускати тестові сценарії роботи користувача і спроби отримати закриту інформацію;

– вірогідність відмови в роботі певних компонентів програми при роботі різних спеціалістів над однаковими фрагментами коду – коли штат спеціалістів великий, існує велика вірогідність що інший програміст, який вносить зміни в існуючий код може змінити логіку роботи таким чином, що цей код буде працювати некоректно, або буде негативно впливати на роботу інших програмних вузлів. Тестування допомагає додатково перевіряти чи не змінився тип оброблюваних даних, призначення методу і в цілому механізм роботи функції, особливо якщо компонент активно взаємодіє з іншими частинами програми;

– вірогідність збоїв при навантаженні додатку. Перевірка програми на стійкість до навантажень особливо важлива у випадку якщо проектом буде користуватись велика кількість людей. При розробці всі компоненти можуть працювати без помилок і досить швидко, але стрес-тести допомагають виявити слабкі і неоптимізовані місця проекту і уникнути труднощів в експлуатації великою кількістю користувачів;

– обробка документації тестів і оцінка ефективності роботи програми з урахуванням змін. По тестовим сценаріям можливо оцінювати глибину та ефективність тестів. У випадку ручного тестування неможливо оцінити ефективність роботи і дати точну гарантію стабільної роботи після оновлення веб-додатку;

– розходження функціоналу і можливостей фінального релізу програмного продукту і умов

вказаних в технічному завданні до проекту. Якщо проект складний і дуже об'ємний, можливі ситуації коли виникає різниця між технічним завданням і самим проектом. Написання тестів дозволяє заздалегідь спроектувати найважливіші аспекти проекту і уникнути при його подальшому розвитку серйозних розходжень в плані функціоналу і роботи продукту.

Для покриття різних варіантів сценаріїв збоїв і складнощів при розширенні можливостей веб-додатку з різнорівневою архітектурою (рис. 2), зазвичай використовують наступні види тестування програмного забезпечення [6,13]:

– Unit-тести (одиничне тестування, або модульне тестування) – це автоматичний вид тестування програмного забезпечення (виконується скриптом) тест для певної частини програми. Автоматичні тести дозволяють значно збільшити обсяг тестування порівняно з ручним тестуванням компонентів;

– тести сумісності – подібні тести дозволяють впевнитись що інтерфейс додатку буде сумісний з відповідними версіями браузерів. Можливо перевірити коректність роботи CSS-стилів, HTML-розмітки, Java Script коду в різних браузерах та їх версіях;

– тести бази даних – цей вид тестування надає можливість контролювати стабільну роботу бази даних додатку, цілісності даних таблиць, індексів і функцій;

– функціональне тестування – це тестування програмного забезпечення для перевірки реалізації функціональних вимог, тобто можливості в певних умовах вирішувати певні задачі які необхідні користувачам програми;

– стрес-тести – це тестування веб-додатку в умовах перевищення кордонів для нормального функціонування додатку. Стрес-тест призначений для визначення продуктивності системи і передбачення поведінки системи в умовах стресових навантажень.

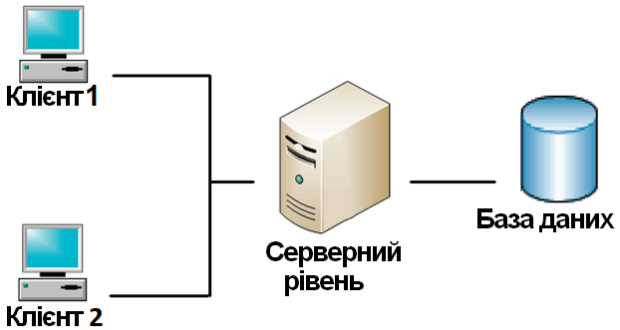

Рис. 2 – Схема трьохрівневої архітектури

Таким чином, на кожному із рівнів архітектури проводиться свій вид тестів. На серверному рівні проводяться стрес-тести, тести на навантаження серверу і додаткового програмного забезпечення яке його обслуговує. Клієнтський рівень це безпосередньо робота самого додатку – на цьому рівні проводяться функціональні і модульні тести, а також тести на сумісність. На рівні бази даних проводиться окремо тестування бази даних додатку, перевірка цілісності даних, їх зміна, видалення або збереження.

Функціональні, модульні тести і тести бази даних тісно пов'язані один з одним, і разом використовуються при тестуванні окремих компонентів додатку.

Розглянемо етапи автоматизації тестування з використанням систем підтримки процесу розробки веб-проектів.

Контроль за змінами, що вносяться в проект допомагають забезпечити системи управління версіями (Version Control System – VCS), які зберігають попередні версії вихідних файлів проекту, відстежують вироблені в файлах зміни, забезпечують спільну командну роботу над проектом та ін. До найбільш популярних на поточний момент VCS відносяться: SVN, GIT, Microsoft VSS. Використання системи контролю версій піднімає загальний рівень якості розробки.

Для управління супроводу проекту, внесення в нього змін, виправлення помилок та іншими аспектами розробки та поліпшення якості менеджменту використовуються системи відстежування помилок (Bug Tracking System – BTS). Головний компонент BTS являє собою базу даних з віддаленим доступом, що забезпечує централізований доступ до всіх необхідних файлів, специфікаціям, графіками, планам, зауважень та ін. Існує широке розмаїття систем стеження за вадами: Basecamp, Bugzilla, Trac, MantisBT, Redmine, Jenkins тощо.

По завершенні етапу активного програмування починається етап тестування коректності функціонування створеного веб-додатку: перевірки на наявність граматичних помилок, пропущених картинок, непрацюючих посилань та ін., а також перевірки функціонування сайту в різних веб-браузерах. Даний етап може бути автоматизований за допомогою засобів автоматичного тестування таких, як IBM Rational AppScan, Empirix E-TEST Suite, XSpider, WAS, Selenium WebDriver і ін.

Таким чином, основні етапи автоматизації тестування з використанням систем підтримки процесу розробки веб-проектів можна представити у вигляді наступної схеми (рис. 3).

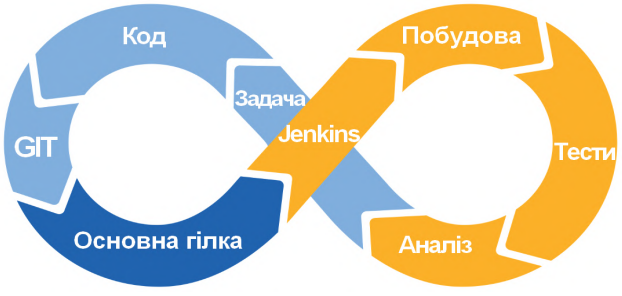

Рис. 3 – Схема етапів автоматизації тестування з використанням систем підтримки процесу розробки

ISSN 2079-5459 (print)
ISSN 2413-4295 (online)

Проведений аналіз сучасних засобів для автоматизованого тестування навантаження серед інструментів Apache JMeter, AB (Apache Benchmark) і HP LoadRunner [14] показав, що:

– AB (Apache Benchmark) – однопоточна програма для командного рядка, що використовується для вимірювання продуктивності HTTP веб-серверів. Спочатку розроблена для тестування HTTP-сервера Apache, в основному підходить для тестування будь-якого веб-сервера. Утиліта ab поставляється разом зі стандартною дистрибуцією Apache, і як сам Apache, є вільним програмним забезпеченням і розповсюджується під Ліцензією Apache. Недоліком цієї програми є обмежений функціонал і можливість повноцінно працювати лише при використанні веб-серверу Apache.

– HP LoadRunner (також HPE LoadRunner) – програма для автоматизованого тестування навантаження. Програма може виконувати тестування як різних додатків, так і сайтів різного рівня складності. При тестуванні емулює паралельну роботу великої кількості так званих віртуальних користувачів (у вигляді процесів або потоків), що виконують різні скрипти (дії) за різними сценаріями. Програма має відповідні набори інструментів для проведення тестування. Також до складу HP LoadRunner входить набір інструментів для роботи з додатком по різних протоколах (віддалено, через проксі-сервер та ін.). В порівнянні з Apache Benchmark має більший функціонал але не поширюється безкоштовно.

Якщо порівняти Apache JMeter і HP LoadRunner проаналізувавши статистику використання, то як результат JMeter використовується набагато частіше ніж HP LoadRunner. Процентне порівняння зображено на рисунку 4.

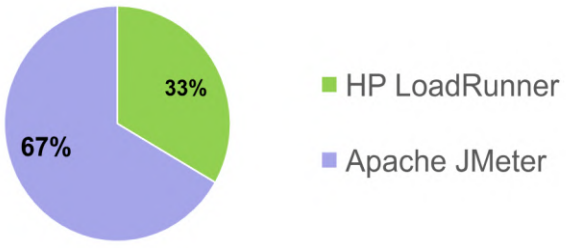

Рис. 4 – Порівняння програм тестування навантаження за популярністю

Також був проведений аналіз ефективності роботи програм і їх можливостях в імітації навантаження. Тестування проводилось на одному сервері, для імітації навантаження виконувались SQL-запити select, insert і update в базу даних додатку. По результатам, HP LoadRunner має більші можливості для здійснення великого навантаження за певний обсяг часу. Результати тестування зображені на рисунку 5.

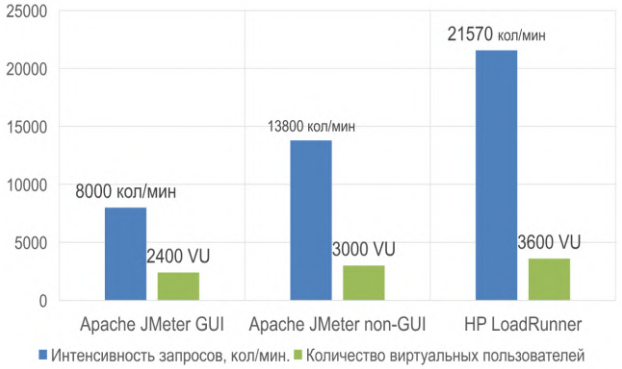

Рис. 5 – Результат порівняння програм по максимальному навантаженню сервера за обмежений обсяг часу

Як результат, HP LoadRunner є лідером враховуючи його функціональні можливості і більшу ефективність при тестах навантаження. Але, враховуючи що це платний продукт, і функціональних можливостей і ефективності навантаження Apache JMeter вистачає для проекту, був обраний саме Apache JMeter.

Враховуючи що велика кількість спеціалістів працює над одним проектом, використовується система контролю версій GIT – подібна система дозволяє контролювати процес розробки і надає можливість клонувати проект для кожного працівника у цілях організації окремого простору для створення нового і редагування старого функціоналу програми без шкоди для основного проекту. На рисунку 6 зображений механізм роботи системи контролю версій.

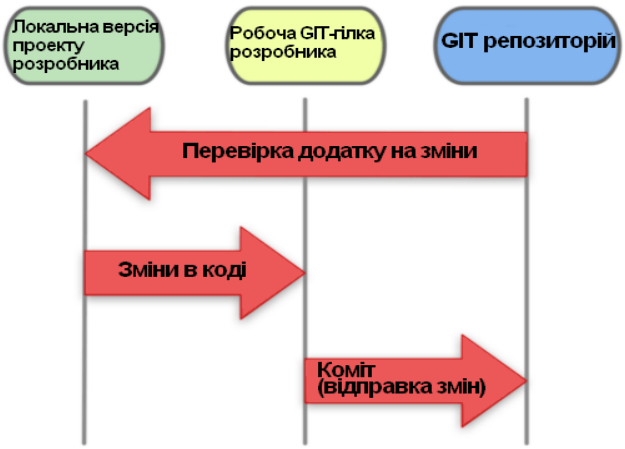

Рис. 6 – Схема роботи з GIT

Розглянуті веб-сервіси Bitbucket, GitHub і GitLab – це основні конкуренти подібних систем [15]. Основна перевага Bitbucket – це безкоштовні приватні репозиторії. На інших платформах безкоштовно можливо розмістити тільки відкритий проект, вихідний код якого буде доступний всім користувачам. Також у порівнянні з GitLab, Bitbucket

працює більш стабільно, з мінімальним відсотком відмов у роботі і завантаженні змін на сервер.

Оптимальний варіант роботи з проектом – це повна автоматизація тестових та зборочних процесів програми, використання інструментів для безперервної інтеграції, яке необхідне для можливості зручної і більш швидкої розробки. За допомогою Bitbucket можливо провести інтеграцію з системами стеження за помилками (Jira, Trello, Slack), що дозволяє оптимізувати роботу між керівниками розробки і спеціалістами які займаються написанням коду, тестуванням і написанням тестів.

Безперервна інтеграція і розгортання (CI) – це стратегії, які допомагають збільшити швидкість розробки і прискорити випуск добре протестованих, готових до використання програмних продуктів. Безперервна інтеграція дозволяє розробникам тестувати і інтегрувати свої зміни в загальну кодову базу для мінімізації конфліктів інтеграції [14]. Безперервна доставка усуває труднощі розгортання або випуску програмного забезпечення. Безперервне розгортання – ще один крок, він розгортає кожну збірку, яка автоматично проходить тестування. На рисунку 7 представлена детальна схема роботи CI.

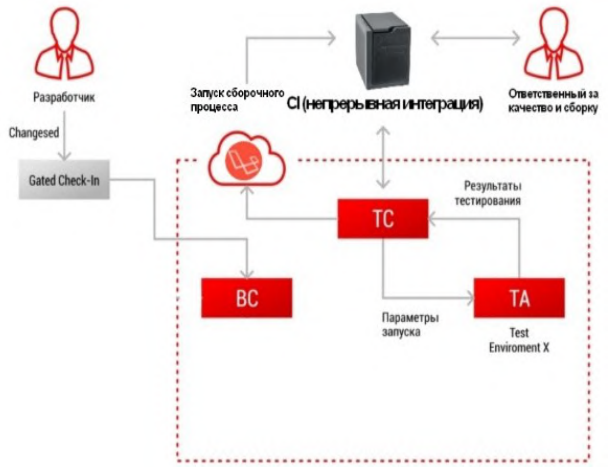

Рис. 7 – Схема роботи механізму безперервної інтеграції

Програмні засоби відіграють велику роль в тому, наскільки швидко і успішно будуть виконуватися перераховані вище етапи. Програмне забезпечення безперервної інтеграції може допомогти розробникам автоматично вносити зміни, щоб скоротити час зворотного зв'язку.

Перед вибором подібної системи, був проведений аналіз найпопулярніших засобів безперервної інтеграції, доставки і розгортання: Jenkins, GitLab CI, Buildbot, Drone і Concourse [14]:

– Jenkins є одним з перших серверів безперервної інтеграції з відкритим вихідним кодом і залишається найбільш поширеним на сьогодні варіантом. Спочатку входить в проект Hudson співтовариство і кодова база розділилися в ході конфліктів з Oracle після придбання ними Sun Microsystems, початкових розробників. Hudson був випущений в 2005 році, а перший реліз Jenkins – в 2011 році.

Згодом сервер Jenkins перетворився в потужну і гнучку систему автоматизації завдань, пов'язаних з програмним забезпеченням. Сам Jenkins використовується в основному як фреймворк автоматизації, а більшість логіки реалізується через бібліотеки і плагіни. Все – від прослуховування вебхуків і перегляду репозиторіїв до створення середовища і підтримки мов – обробляється плагінами. Процес безперервної інтеграції може залежати від численних сторонніх плагінів, що з одного боку забезпечує гнучкість, з іншого – робить середу занадто крихкою.

Робочий процес конвеєра Jenkins, який також доступний через плагін, є відносно новою функцією, доступною з 2016 року. Процес безперервної інтеграції можна декларувати або використовувати мову Groovy в файлах сховища або текстових полях в веб-інтерфейсі Jenkins. Одним із загальних зауважень до Jenkins є те, що заснована на плагінах модель конфігурації і можливість визначати процеси конвеєра або збірки поза репозиторіїв іноді ускладнює реплікацію на іншому екземплярі Jenkins. Система Jenkins написана на Java і випущена під ліцензією MIT [16].

– GitLab CI – це інструмент безперервної інтеграції, вбудований в GitLab, платформу для розміщення сховищ і засобів розробки git. Спочатку випущений як самостійний проект, GitLab CI був інтегрований в основне програмне забезпечення GitLab з випуском GitLab 8.0 в 2015 році.

Процес HI в GitLab CI визначається всередині файлу в самому репозиторії коду за допомогою синтаксису YAML. Потім робота передається на машини під назвою runners, які легко настроюються і можуть використовувати різні ОС. При налаштуванні runner-серверів можна вибрати Docker, shell, VirtualBox або Kubernetes, щоб визначити, як виконуються завдання.

Тісний зв'язок GitLab CI з платформою GitLab має певні наслідки для використання програмного забезпечення. GitLab CI не підходить розробникам, які використовують інші платформи для розміщення сховищ. З одного боку, інтегрована функціональність дозволяє користувачам GitLab настроювати середовище безп, не встановлюючи додаткових інструментів. Автоматичне тестування можна запустити за допомогою веб-інтерфейсу, реєстрації runner-машин і додати файл визначення конвеєра в репозиторій. Тісний взаємозв'язок також дозволяє розподіляти runner-сервери між проектами, автоматично переглядати поточний статус збірки в репозиторії і зберігати артефакти збірки з кодом, який їх створив. GitLab і GitLab CI написані на Ruby і Go і випущені під ліцензією MIT.

ISSN 2079-5459 (print)
ISSN 2413-4295 (online)

– Buildbot – це гнучкий фреймворк безперервної інтеграції. Випущений в 2003 році в якості альтернативи проекту Tinderbox від Mozilla, Buildbot розроблявся в основному як спосіб автоматизації тестування збірки на широкому спектрі платформ.

Buildbot випускається за ліцензією GPL і написаний на Python з бібліотекою Twisted. Конфігурація Buildbot повністю написана на Python. Це означає, що конфігурація значно складніше, ніж в інших системах, але це дає адміністраторам можливість для розробки свого ідеального робочого процесу і конвеєра. Кожен етап збірки чітко розділяється. Buildbot позиціонує себе як структуру з інструментами для створення призначених для користувача процесів, подібно до того, як веб-фреймворки дозволяють створювати призначені для користувача сайти.

Платформа тестування збірки Buildbot підтримує безліч різних операційних систем і систем управління версіями. Оскільки Buildbot розроблений з урахуванням тестування відкритого виходячи коду, його архітектура дозволяє користувачам легко створювати робочі процеси з індивідуальними платформами і розширювати базу проекту. Користувачеві потрібно тільки встановити кілька пакетів Python, а потім надати облікові дані.

– Drone – це сучасна платформа безперервної інтеграції з архітектурою на основі контейнерів.

Хоча розглянуті вище інструменти теж дозволяють запускати збірку за допомогою Docker, контейнери лежать в основі конструкції Drone. Drone написаний на Go і був вперше випущений в 2014 році під ліцензією Apache.

Drone працює як середній рівень між Docker і провайдером сховища. Замість того, щоб запускати сервер безперервної інтеграції і підключатися до системи управління версіями, Drone вимагає інформацію облікового запису для завантаження власних моделей перевірки автентичності, користувача і дозволів. Drone запускається як контейнер. Він підтримує кілька баз даних і провайдерів репозиторіїв і має вбудовану підтримку для настройки сертифікатів TLS / SSL.

Drone шукає в репозиторіях спеціальні файли YAML для визначення конвеєра. Синтаксис чіткий і простий, щоб будь-хто, хто використовує репозиторій, міг зрозуміти процес безперервної інтеграції. Drone надає плагіновую систему, але вона працює не так, як в Jenkins. У Drone плагіни є спеціальними контейнерами Docker, які використовуються для додавання попередньо налаштованих завдань в звичайний робочий процес. Це спрощує виконання спільних завдань, адже немає необхідності обробляти весь процес вручну. У цьому сенсі плагіни Drone дещо схожі з командами Unix, які призначені для обробки вузькопрофільних завдань.

– Concourse – відносно нова платформа безперервної інтеграції, випущена в 2014 році. Підхід Concourse до системи безперервної інтеграції значно відрізняється від інших інструментів тим, що Concourse намагається абстрагувати все зовнішні чинники за допомогою так званих ресурсів. Мета цього підходу полягає в підвищенні доступності сервера інтеграції. Це дозволяє запускати одні і ті ж процеси на будь-якому сервері Concourse.

Кожна частина процесу безперервної інтеграції складається з основних примітивів, які моделюють різні елементи системи. Залежно кожного етапу визначаються явно. Наприклад, для першого завдання може знадобитися останній Комміт в репозиторії VCS, а на наступних етапах процесу може знадобитися останній Комміт, що пройшов попередні етапи. Цей метод побудови конвеєрів шляхом зіставлення точних залежностей кожного кроку призводить до строго певної поведінки.

Concourse не бере участь в передачі даних між завданнями і не надає ніякого внутрішнього способу зберігання артефактів збірки. Вся інформація, необхідна для наступного етапу, повинна бути явно визначена і потенційно перенесена у зовнішнє сховище. За допомогою явних визначень Concourse намагається мінімізувати кількість невідомих змінних, які повинна враховувати система. Система Concourse написана на Go і доступна за ліцензією Apache.

Програмне забезпечення безперервної інтеграції, доставки і розгортання – це складні системи автоматизації, призначені для забезпечення адаптованості і повторюваності процесів. Існує безліч різних ідей і підходів до автоматичного тестування.

Обговорення результатів

Для реалізації автоматизації сценаріїв тестування використовувалась розроблена веб-базована підсистема підтримки електронної торгівлі. Відповідно розвитку програмної частини було необхідно реалізувати систему контролю написання нових функціональних можливостей і безпеки та стабільності роботи панелі керування для всіх працівників проекту. Для цього було прийнято рішення описати існуючий функціонал різними видами тестів і виконати подальшу автоматизацію цього процесу.

З урахуванням того що проект базується як веб-додаток і реалізований на мові програмування PHP за допомогою фреймворку Laravel, а також з використанням Java Script, JQuery, HTML і CSS, для реалізації комплексу тестування використаний PHP-фреймворк спеціально створений для організації тестового процесу – PHPUnit. Цей фреймворк дозволяє здійснити модульне і функціональне тестування додатку, а також тестування роботи бази даних. Додатково необхідно враховувати можливі навантаження на сервер, і його стійкість до непередбачуваних ситуацій – великий наплив користувачів, робота неоптимізованих компонентів

системи, навантаження на базу даних в майбутньому, з ростом обсягів даних які зберігаються в базі даних.

В даний час PHPUnit найбільш популярний фреймворк для юніт-тестування в PHP. Крім наявності таких можливостей, як підробки (mocking) об'єктів, він також може аналізувати покриття коду, логування даних і надає тисячі інших можливостей. Враховуючи те, що Laravel побудований з урахуванням тестування, то фактично, підтримка PHPUnit доступна за замовчуванням. Проект містить в собі всі необхідні бібліотеки, конфігураційні файли і команди для успішного запуску тестування. Також Laravel надає дуже зручний API для створення HTTP-запитів до додатка, перевірки виведення, і навіть заповнення форм, і надає кілька допоміжних функцій для тестування JSON API і їх відгуків. Наприклад, методи get(), post(), put(), patch() і delete() використовуються для виконання різних HTTP-запитів.

PHPUnit дозволяє створювати необмежену кількість тестів, і надає можливість запускати перед внесенням змін до проекту як один із тестів, так і певну кількість або повністю всі тести додатку. Це дозволяє насамперед впевнитись в тому, що після внесення змін, система надалі буде працювати стабільно і не виникнуть непередбачувані складнощі в роботі, в першу чергу для користувачів додатку. При запуску тестів, через інтерфейс командної строки відображається процес тестування, і його результат. У випадку якщо запускається більше одного тесту, і один із тестів не пройшов, команда виводить звіт про помилку і автоматично завершить роботу тестів (рис. 8).

```
$ phpunit -c app
PHPUnit 4.3.5 by Sebastian Bergmann.

Configuration read from /Users/javier/Desktop/symfony_demo_2/app/phpunit.xml.dist

..................

Time: 3.44 seconds, Memory: 49.75Mb

OK (17 tests, 21 assertions)

Remaining deprecation notices (2)

getEntityManager is deprecated since Symfony 2.1. Use getManager instead: 2x
    1x in DefaultControllerTest::testPublicUrls from AppBundle\Tests\Controller
    1x in BlogControllerTest::testIndex from AppBundle\Tests\Controller
```

Рис. 8 – Звіт з помилками в одному із тестів

У випадку, коли всі тести були успішно пройдені, то буде сформований звіт з врахуванням часу, який був потрібний для повного тестування і повідомлення про успішне проходження всіх тестів (рис. 9).

За допомогою фреймворку PHPUnit можливо проводити окремий аналіз покриття коду (code coverage, tests coverage) – це міра, яка використовується при тестуванні програмного забезпечення. Вона визначається відсотком тестованого вихідного коду (сирцевий код) програми. Це дозволяє розуміти, який відсоток додатку перевіряється – чим вищий відсоток покриття, тим більша вірогідність мінімізації помилок при внесенні змін, так як повне покриття дозволяє виключити майже будь-які непередбачувані збої і порушення логіки роботи компонентів.

```
[root@Alvin-CentOS6 trio]# ./vendor/bin/phpunit --coverage-text tests/AppBundle/Entity/ClientTest.php
PHPUnit 6.1.1 by Sebastian Bergmann and contributors.

..                                                                  2 / 2 (100%)

Time: 67 ms, Memory: 4.00MB

OK (2 tests, 6 assertions)


Code Coverage Report:
  2017-04-21 14:14:37

 Summary:
  Classes:  0.00% (0/30)
  Methods:  2.65% (6/226)
  Lines:    0.22% (9/4153)

\AppBundle\Entity::Client
  Methods:  85.71% ( 6/ 7)   Lines:  90.00% ( 9/ 10)
[root@Alvin-CentOS6 trio]#
```

Рис. 9 – Звіт про успішне проходження всіх тестів

За допомогою цього фреймворку і набору бібліотек до нього можливо повноцінно провести функціональне, модульне тестування і тести бази даних.

Для стрес-тестів був використаний інструмент Apache JMeter – це інструмент для проведення навантажувального тестування, що розробляється Apache Software Foundation. Хоча спочатку JMeter розроблявся як засіб тестування веб-додатків, тепер він здатний проводити навантажувальні тести для JDBC-з'єднань, FTP, LDAP, SOAP, JMS, POP3, IMAP, HTTP і TCP. Це дозволяє здійнити тестування на серверному рівні і отримати детальну інформацію наскільки стабільно сервер буде працювати враховуючи різні фактори навантаження.

Для зручного аналізу і проектування для всіх працівників і зокрема спеціалістів які відповідальні за контроль якості коду і внесенням нових змін в програму використовується веб-сервіс Bitbucket – для сумісної розробки на основі системи контролю версій GIT. Bitbucket дозволяє створити репозиторій у якому буде зберігатись вихідний код проекту, і також гілки проекту кожного працівника, який задіяний в розробці.

Враховуючи дослідження і аналіз існуючих рішень для систем безперервної інтеграції, було прийнято рішення використовувати Jenkins – так як він розповсюджується безкоштовно, є дуже гнучким у налаштуваннях, має велику кількість додаткових плагінів яки значно можуть розширити роботу системи, а також дозволяє отримати повний контроль над всією системою проекту, у порівнянні з конкурентними продуктами. На рис. 10 зображена конфігурація системи безперервної інтеграції, яка налаштована для комплексу, що реалізується.

Інформаційна система орієнтована виключно на працівників відділу розробки. Доступ до системи контролю версій мають всі програмісти, спеціалісти з тестування і керівник відділу. Кожний працівник має в Bitbucket свої права доступу. Наприклад керівник має можливість повністю контролювати всі гілки і весь проект загалом. А певний програміст має доступ лише до своєї гілки і може передивлятись зміст основної гілки додатку. Кожному користувачеві

ISSN 2079-5459 (print)
ISSN 2413-4295 (online)

можливо призначити права доступу як на перегляд, так і на запис даних но певної гілки, а також надати або зняти певні функціональні можливості. Це дозволяє здійснити точне та індивідуальне налаштування кожного користувача, що забезпечує покращення безпеки роботи сервісу і виключає можливості отримання несанкціонованих даних.

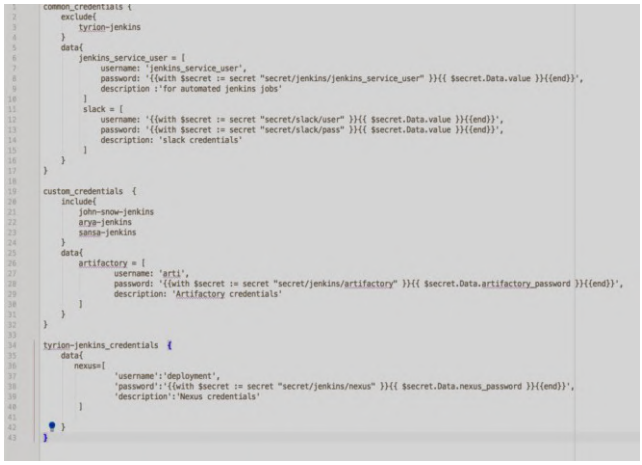

Рис. 10 – Приклад конфігурації системи безперервної інтеграції

Тестовий комплекс реалізований за допомогою фреймворків Laravel і PHPUnit має файлову архітектуру зображену на рис. 11. По спеціалізованим директоріям розподілені різнопланові тести під певний компонент і функціонал додатку.

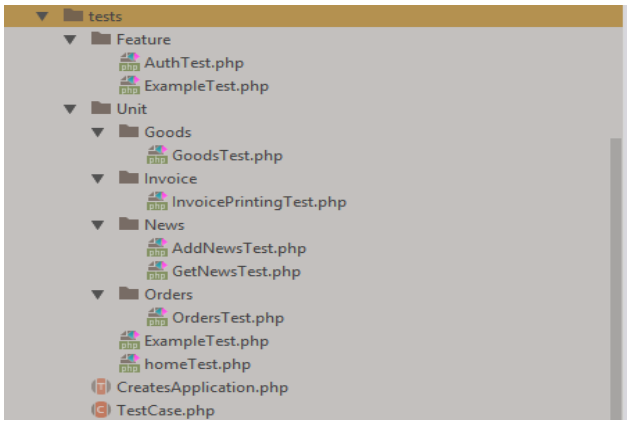

Рис. 11 – Приклад архітектури тестів компонентів і функцій додатку

Самі тести представляють функції в яких виконуються певні дії – одна функція за стандартом має здійснювати одну дію, наприклад отримання одного товару. Інша функція може вже отримувати наприклад 10 товарів. Це необхідно для більшої деталізації тестування і можливості описати максимально велику кількість ситуацій які можуть виникнути в реальних умовах використання. Приклад реалізації тесту на отримання одного товару із бази даних показаний на рис. 12.

Рис. 12 – Приклад тесту на отримання одного товару із бази даних системи

Висновки

В результаті реалізації автоматизованого тестування веб-проекту був отриманий практичний досвід створення автоматизованої системи тестування веб-додатків за допомогою системи контролю версій (GIT) Bitbucket і системи безперервної інтеграції (CI) Jenkins.

Розроблена система тестування дозволяє швидко і ефективно адмініструвати всі компоненти системи, отримувати інформаційні повідомлення і здійснювати аналіз змін, помилок, ресурсів серверної частини і загалом оптимізувати роботу розробників проекту.

Система створювалась з розрахунком на подальший швидкий розвиток веб-проекту у різних напрямках, що дозволить проводити оновлення стабільно і без зайвих ризиків в роботі програми.

В запропонований підхід, по можливості, були включені переваги розповсюджених підходів та інструментів, призначених для автоматизованого тестування веб-додатків.

Одним з найбільш важливих напрямків розвитку проекту є автоматизація підтримки тестового набору в актуальному стані при зміні веб-додатку, а також розширення видів тестування.

Список літератури

1. **Arora, A.** Web Application Testing: A Review on Techniques, Tools and State of Art / **A. Arora, M. Sinha** // *International Journal of Scientific & Engineering Research*. – 2012. – Vol. 3, Issue 2. – P. 1-5.
2. **Girgis, Moheb R.** An Automated Web Application Testing System / **Moheb R. Girgis, Tarek M. Mahmoud, Bahgat A. Abdullatif, Alaa M. Zaki** // *International Journal of Computer Applications* (0975 – 8887). – 2014. – Vol. 99, No.7. – P. 37-44. – doi: 10.5120/17387-7926.
3. **Троян, А. М.** Доцільність автоматизованого тестування для забезпечення якості програмних продуктів / **А. М. Троян, Ю. Б. Моденов** // *Проблеми*

інформатизації та управління. – Київ: НДІ ІТТ НАУ, 2017. – Т. 1, № 57-58. – С. 86-89. – doi: 10.18372/2073-4751.1.12798.

4. **Янгунаева, Е. А.** Сравнение автоматизированного и ручного подхода в тестировании веб-приложений / **Е. А. Янгунаева, В. М. Янгунаев** // *Научный альманах*. – Тамбов: ООО «Консалтинговая компания Юком», 2016. – № 1-1 (15). – С. 546-549. – doi: 10.17117/na.2016.01.01.546.
5. **Shakti Kundu**. Web Testing: Tool, Challenges and Methods / **Shakti Kundu** // *IJCSI International Journal of Computer Science Issues*. – 2012. – Vol. 9, Issue 2, No 3. – P. 481-486.
6. **Lakshmi, D. Rajya.** A Review on Web Application Testing and its Current Research Directions / **D. Rajya Lakshmi, S. Suguna Mallika** // *International Journal of Electrical and Computer Engineering (IJECE)*. – 2017. – Vol. 7, No. 4. – P. 2132-2141. – doi: 10.11591/ijece.v7i4.pp2132-2141.
7. **Животова, А. А.** Методи та засоби тестування web-додатків / **А. А. Животова, Ю. Б. Моденов** // // *Проблеми інформатизації та управлінню*. – Київ: НДІ ІТТ НАУ, 2014. – Т. 2 № 46. – С. 27-30. – doi: 10.18372/2073-4751.2.7711.
8. **Лапонина, О. Р.** Использование сканера уязвимостей ZAP для тестирования веб-приложений / **О. Р. Лапонина, С. А. Малаховский** // *International Journal of Open Information Technologies*. – 2017. – Vol. 5, №.8. – P. 18-26.
9. Comparing Automated Testing Tools: Selenium, TestComplete, Ranorex, and more. URL: https://www.altexsoft.com/blog/engineering/comparing-automated-testing-tools-selenium-testcomplete-ranorex-and-more/ 04.02.2019.
10. **Soni, P.** Testing Web Applications Using UIO with GA / **Poonam Soni, Dr. Sanjay Tyagi** // *The International Journal of Soft Computing and Software Engineering*. – 2013. – Vol. 3, Issue 5. – P. 636-640.
11. **Silva, R. A.** A Systematic Review on Search Based Mutation Testing / **Rodolfo Adamshuk Silva, Simone do Rocio Senger de Souza,Paulo Sergio Lopes de Souza** // *Information and Software Technology*. – 2017. – Vol. 81, Issue C. – P. 19-35. – doi: 10.1016/j.infsof.2016.01.017.
12. **Myers, G. J.** The Art Of Software Testing. / **G. J. Myers. –** Wiley & Sons, Inc. 2004. – 254 p.
13. **Burns, D.** Selenium 1.0 Testing Tools / **D. Burns.** – London: Packt Publishing, 2011. – 248 p.
14. **Хамбл, Д.** Непрерывное развертывание ПО: автоматизация процессов сборки, тестирования и внедрения новых версий / **Д. Хамбл, Д. Фарли.** – М.: Диалектика-Вильямс, 2018. – 432 с.
15. **Штрауб, Б.** Git для профессионального программиста / **Б. Штрауб, С. Чакон**. – М.: Питер, 2016. – 496 с.
16. **Laster, B.** Jenkins 2: Up and Running: Evolve Your Deployment Pipeline for Next Generation Automation 1st Edition / **B. Laster**. – M.: O'ReillyMedia , 2018. – 600 p.

References (transliterated)

1. **Arora, A.**, **Sinha, M.** Web Application Testing: A Review on Techniques, Tools and State of Art. *International Journal of Scientific & Engineering Research*, 2012, **3**(2), 1-5.
2. **Girgis, M. R., Mahmoud, T. M., Abdullatif, B. A., Zaki, A. M.** An Automated Web Application Testing System. *International Journal of Computer Applications* (0975 – 8887), 2014, **99**(7), 37-44, doi: 10.5120/17387-7926.
3. **Troyan, A. M, Modenov, Yu. B.** Dotsil′nist′ avtomatyzovanoho testuvannya dlya zabezpechennya yakosti prohramnykh produktiv [The expediency of automated testing to ensure the quality of software products]. *Problemy informatyzatsiyi ta upravlinnya [Problems of informatization and management]*. – Kyiv: Research Institute of ITT NAU, 2017, **1**(57-58), 86-89, doi: 10.18372/2073-4751.1.12798.
4. **Yanhunaeva, E. A., Yanhunaev, V. M.** Sravneniye avtomatizirovannogo i ruchnogo podkhoda v testirovanii veb-prilozheniy [Comparison of automated and manual approach to testing web applications]. *Nauchnyy al'manakh [Scientific Almanac]*. Tambov: Consulting Company Ucom LLC, 2016, **1-1** (15), 546-549, doi: 10.17117/na.2016.01.01.546.
5. **Kundu, Shakti**. Web Testing: Tool, Challenges and Methods. *IJCSI International Journal of Computer Science Issues*, 2012, **9**(2), 481-486.
6. **Lakshmi, D. Rajya, Mallika, S. S.** A Review on Web Application Testing and its Current Research Directions. *International Journal of Electrical and Computer Engineering (IJECE)*, 2017, **7**(4), 2132-2141, doi: 10.11591/ijece.v7i4.pp2132-2141.
7. **Zhivotova, A. A., Modenov, Yu. B.** Metody ta zasoby testuvannya web-dodatkiv [Methods and tools for testing web-applications] *Problemy informatyzatsiyi ta upravlinnya [Problems of informatization and management]*. – Kyiv: Research Institute of ITT NAU, 2014, **2**(46), 27-30, doi: 10.18372/2073-4751.2.7711.
8. **Laponina, O. R., Malakhovskiy, S. A.** Ispol'zovaniye skanera uyazvimostey ZAP dlya testirovaniya veb-prilozheniy [Using ZAP Vulnerability Scanner to test web applications]. *International Journal of Open Information Technologies*, 2017, **5**(8), 18-26.
9. Comparing Automated Testing Tools: Selenium, TestComplete, Ranorex, and more. Available at: https://www.altexsoft.com/blog/engineering/comparing-automated-testing-tools-selenium-testcomplete-ranorex-and-more/.
10. **Soni, P. Dr. Tyagi, S.** Testing Web Applications Using UIO with GA. *The International Journal of Soft Computing and Software Engineering*, 2013, **3**(5), 636-640.
11. **Silva, R. A., Senger de Souza, S. R., Lopes de Souza, P. S.** A Systematic Review on Search Based Mutation Testing. *Information and Software Technology*, 2017, **81**(C), 19-35, doi: 10.1016/j.infsof.2016.01.017.
12. **Myers, G. J.** The Art Of Software Testing. Wiley & Sons, *Inc*. 2004, 254.
13. **Burns, D.** Selenium 1.0 Testing Tools. London: Packt Publishing, 2011, 248.
14. **Khambl, D., Farly, D.** Nepreryvnoye razvertyvaniye PO: avtomatizatsiya protsessov sborki, testirovaniya i vnedreniya novykh versiy [Continuous software deployment: automate the build process, test and implement new versions] Moscow: Dialectics-Williams, 2018, 432.
15. **Shtraub, B., Chakon, S.** Git dlya professional'nogo programmista [Git for a professional programmer]. Moscow.: Piter, 2016, 496.
16. **Laster, B.** Jenkins 2: Up and Running: Evolve Your Deployment Pipeline for Next Generation Automation 1st Edition. Moscow: O'ReillyMedia, 2018, 600.
17.

ISSN 2079-5459 (print)
ISSN 2413-4295 (online)

Відомості про авторів (About authors)

Кодола Галина Миколаївна – Державний вищий навчальний заклад «Український державний хіміко-технологічний університет», викладач кафедри Інформаційних систем; м. Дніпро, Україна; ORCID: http://orcid.org/0000-0001-9403-1462; e-mail: galina_kodola@udhtu.edu.ua.

Galyna Kodola – The Ukrainian State Chemical-Technological University, Information Control Systems and Technology, Dnipro, Ukraine; ORCID: http://orcid.org/0000-0001-9403-1462; e-mail: galina_kodola@udhtu.edu.ua.

Волинець Наталія Сергіївна – Державний вищий навчальний заклад «Український державний хіміко-технологічний університет», викладач кафедри Інформаційних систем; м. Дніпро, Україна; e-mail: natalia.wolynec@gmail.com.

Natalia Volynets – The Ukrainian State Chemical-Technological University, Information Control Systems and Technology, Dnipro, Ukraine; e-mail: natalia.wolynec@gmail.com.

Сербулова Інна Валеріївна – Державний вищий навчальний заклад «Український державний хіміко-технологічний університет», старший викладач кафедри Інформаційних систем; м. Дніпро, Україна; e-mail: innasrb20@gmail.com.

Inna Serbulova – The Ukrainian State Chemical-Technological University, Information Control Systems and Technology, Dnipro, Ukraine; e-mail: innasrb20@gmail.com.

Будь ласка, посилайтесь на цю статтю наступним чином:

Кодола, Г. М. Автоматизоване тестування веб-додатків з різнорівневою архітектурою / **Г. М. Кодола, Н. С. Волинець, І. В. Сербулова** // *Вісник НТУ «ХПІ», Серія: Нові рішення в сучасних технологіях*. – Харків: НТУ «ХПІ». – 2019. – № 5 (1330). – С. 91-100. – doi:10.20998/2413-4295.2019.05.12.

Please cite this article as:

Kodola, G., Volynets, N., Serbulova, I. Automated testing of web applications with multilevel architecture. *Bulletin of NTU "KhPI". Series: New solutions in modern technologies*. – Kharkiv: NTU "KhPI", 2019, **5** (1330), 91-100, doi:10.20998/2413-4295.2019.05.12.

Пожалуйста, ссылайтесь на эту статью следующим образом:

Кодола, Г. Н. Автоматизированное тестирование веб-приложений с разноуровневой архитектурой / **Г. Н. Кодола, Н. С. Волинец, И. В. Сербулова** // *Вестник НТУ «ХПИ», Серия: Новые решения в современных технологиях*. – Харьков: НТУ «ХПИ». – 2019. – № 5 (1330). – С. 91-100. – doi:10.20998/2413-4295.2019.05.12.

АННОТАЦИЯ *Веб-приложения играют важную роль в жизни общества. Они применяются в таких секторах, как бизнес, здравоохранение и государственное управление. От качества таких приложений может зависеть не только удобство пользователей, но и функционирования организаций. Тестирование является наиболее широко используемым и эффективным подходом для обеспечения качества и надежности программного обеспечения, включая веб-приложения. Однако веб-приложения очень отличаются от традиционного программного обеспечения, поскольку они включают в себя динамическое создание и интерпретацию кода, а также реализацию конкретного режима взаимодействия на основе навигационной структуры веб-приложения. Автоматизированное тестирование - это автоматическое выполнение набора тестов. Создав этот набор один раз, его можно использовать каждый раз после внесения некоторых изменений в веб-приложение. Кроме того, современные веб-приложения построены на основе многоуровневой архитектуры. Поэтому, чтобы проверить общее поведение веб-приложений, нужно составить комплекс методов тестирования. Автоматизация тестирования не может быть реализована без соответствующих инструментов. Именно они определяют, как будет осуществляться тестирование и могут быть достигнуты преимущества автоматизации. Инструменты автоматизации тестирования является важнейшим компонентом в инструментальной цепочке разработки. В статье было проанализировано существующие программные средства, используемые для автоматизированного тестирования, и выбраны среди них для каждого уровня веб-приложения те, которые смогут обеспечить высокий уровень безопасности и минимизировать вероятность ошибок или сбоев в работе программы. Для достижения этой цели были рассмотрены такие виды программных средств: системы управления версиями; системы отслеживания ошибок; средства автоматического тестирования; средства для автоматизированного тестирования нагрузки; программное обеспечение непрерывной интеграции. На их основе был составлен комплекс автоматизации тестирования веб-приложения, которое позволит без лишних сложностей проводить индивидуальные модификации системы и значительно уменьшит количество ошибок в процессе доработки системы другими специалистами. В результате реализации автоматизированного тестирования веб-проекта был получен практический опыт создания автоматизированной системы тестирования веб-приложений с помощью системы контроля версий (GIT) Bitbucket и системы непрерывной интеграции (CI) Jenkins.*

Ключевые слова: *автоматизированное тестирование веб-приложений; разноуровневая архитектура; безопасность данных; интеграция кода минимизация ошибок*

Надійшла (received) 27.02.2019