MINISTERUL EDUCAȚIEI ȘI CERCETĂRII AL REPUBLICII MOLDOVA

Universitatea Tehnică a Moldovei
Facultatea Calculatoare, Informatică și Microelectronică
Departamentul Informatică și Ingineria Sistemelor

Admis la susținere

Șef Departament:

Sudacevschi Viorica, conf. univ., dr.

_____

„___”_____2022

Sistem de automatizare a testării software bazat pe comportament

Teză de master

Student: **Iordan Ana-Maria, CRI-211M**

Conducător: **Moraru Victor, conf. univ., dr.**

Chișinău, 2022

ADNOTARE

La proiectul de master: „Sistem de automatizare a testării software bazat pe comportament”, elaborat de Iordan Ana-Maria. Chișinău, 2022.

Cuvinte cheie: Testare software, testare automatizată, dezvoltare definită de comportament, testare web, testare mobilă, raportare, conducte Jenkins.

În cadrul tezei de masterat s-a pus problema dezvoltării un sistem de automatizare a testarii software, bazat pe dezvoltarea definită de comportament, adică principiul „Behavior Driven Development” (BDD), ce are scopul de a minimiza discrepanța de comunicare dintre persoanele tehnice și non-tehnice ce participă la livrarea produsului software. Pentru implementarea proiectului s-a utilizat mediul integrat de dezvoltare IntelliJ IDEA. Proiectul părinte este un proiect bazat pe maven, precum și modulele derivate a acestuia. Sistemul este dezvoltat utilizând cadrul Spring-Boot și principiile de dezvoltare Spring. Cadrul de testare principal este Cucumber, ce suportă dezvoltarea definită de comportament, integrat cu cadrul TestNG, pentru execuția testelor și instrumentul de raportare Allure.

Modulul de testare Web utilizează cadrul Selenium, și face suport pentru sistemele de navigare web Chrome, Firefox și Microsoft Edge. Selenium creează o solicitare HTTP pentru fiecare comandă selenium depistată în clasele de definire a pașilor Cucumber, și o trimite la driverul browserului, apoi solicitarea se trimite la server. Starea de execuție este trimisă la serverul HTTP, care este apoi capturată de scriptul de automatizare și datele salvate pentru raportare.

Modulul de testare Mobile utilizează cadrul Appium care este o extensie a cadrului Selenium. La startarea execuției testelor, serverul Appium de pe mașină, expune un REST API, primește comenzile din clasele de definire a pașilor și le execută pe dispozitiv, care răspunde înapoi prin HTTP. Pentru executarea cererilor date, Appium folosește cadrele UI Automator și Apple Instruments. După execuție datele sunt salvate pentru realizarea rapoartelor de către instrumentul de raportare Allure.

În scopul automatizării procesului de rulare a suitelor de testare, s-a utilizat instrumentul de orchestrare CI – Jenkins. S-au creat două conducte Jenkins „Freestyle project”, pentru fiecare suită în parte, de testare web și de testare mobilă. Pentru crearea conductei s-a ales metoda declarativă, utilizând limbajul de scriptare Groovy.

Tehnologii utilizate: Java, IntelliJ IDEA, Maven, Spring-Boot, Cucumber, TestNG, Junit, AspectJ, Selenium, Appium, Allure, Jenkins, Groovy, Github.

ANNOTATION

On the Master thesis "Behavior-based software testing automation system"
elaborated by Iordan Ana-Maria. Chișinău, 2022.

Keywords: Software Testing, Automation Testing, Behavior Driven Development, Web Testing, Mobile Testing, Reporting, Jenkins Pipelines.

The scope of the master's thesis was to develop a software testing automation system, based on behavior-driven development, with the aim of minimizing the communication discrepancy between technical and non-technical people, who are involved in the delivery of the software product. The integrated development environment IntelliJ IDEA was used to implement the project. The parent project is a maven based project, as well as its derived modules. The system is developed using the Spring-Boot framework and Spring development principles. The main testing framework is Cucumber, which supports behavior-driven development, integrated with the TestNG framework for test execution, and the Allure reporting tool.

The Web test module uses the Selenium framework, and supports Chrome, Firefox and Microsoft Edge web browsers. Selenium creates an HTTP request for each selenium command depinned in the Cucumber step definition classes, and sends it to the browser driver, then the request is sent to the server. The execution status is sent to the HTTP server, which is then captured by the automation script and the data saved for reporting.

The Mobile test module uses the Appium framework, which is an extension of the Selenium framework. When starting test execution, the Appium server on the machine exposes a REST API, receives the commands from the step definition classes, and executes them on the device, which responds back via HTTP. To execute given requests, appium uses UI Automator and Apple Instruments frameworks. After execution the data is saved for reporting by the Allure reporting tool.

In order to automate the process of running the test suites, the CI – Jenkins orchestration tool was used. For each test suite, web and mobile, were created Jenkins Freestyle project pipelines. To create the pipeline, the declarative method was chosen, using the Groovy scripting language.

Technologies which were used: Java, IntelliJ IDEA, Maven, Spring-Boot, Cucumber, TestNG, Junit, AspectJ, Selenium, Appium, Allure, Jenkins, Groovy, Github.

CUPRINS

INTRODUCERE

Lumea se află în pragul perturbărilor tehnologice și va rămâne sub acest statut încă pentru o perioadă lungă de timp. Industria software-ului a evoluat de-a lungul deceniilor și astfel în consecință, testarea software, ce este o parte integrantă a ciclului de dezvoltare a fost de asemenea martoră la mai multe schimbări în ultimele decenii. Conceptul de testare software a evoluat în decursul anilor la o parte componentă indispensabilă din ciclul de dezvoltare a softului, deoarece anume prin testare se poate de îmbunătățit calitatea sistemului furnizat, depistând erori și defecte în timpul dezvoltării. Potrivit definiției Ilene Burnstein [1], „Testarea software-ului poate fi descrisă ca un proces utilizat pentru a descoperi defectele software-ului și pentru a stabili că software-ul a atins un anumit grad de calitate în raport cu atributele selectate”. Deci conchidem că testarea software reprezintă un proces de evaluare și verificare a faptului că un produs sau o aplicație software face ceea ce trebuie să facă conform specificațiilor indicate de către clienți și părțile interesate.

Teza de masterat este dedicată dezvoltării unui sistem de automatizare a proceselor de testare bazat pe dezvoltarea definită de comportament, ce va fi integrat cu tehnologiile cele mai actuale din domeniul testării, în scopul de a micșora timpul de testarea și a spori calitatea procesului de testare a unui proiect software.

În mare parte procesul de testare a produselor software are loc manual, rulând testele unul câte unul. Toate rapoartele cu rezultatele obținute la fel sunt generate manual fără ajutorul cărorva instrumente. Executarea testelor manual asigură un nivel ridicat de control asupra procesului, astfel inginerii de testare au mai multă vizibilitate în timpul procesului și se pot concentra pe fiecare problemă în mod precis. Însă testarea manuală ajunge să fie consumatoare de timp, și respectiv de resurse umane. De asemenea este greu de testat manual un produs software mare, îndeosebi acum când majoritatea echipelor de dezvoltate lucrează conform metodologiilor dinamice, ce necesită testarea repetată la fiecare ciclu nou. Deoarece testarea automatizată se rulează mult mai rapid decât echivalentele lor manuale, devine fezabil să se testeze mult mai frecvent și mai eficient, economisind astfel cât și timp și resurse umane necesare pentru efectuarea testării manuale. Testarea automatizată presupune crearea unui proiect software, integrat cu alte instrumente și aplicații software, ce va permite rularea scripturilor de testare în mod automatizat și de asemenea personalizat. Proiectul trebuie să suporte și programarea unui orar prestabilit regulat de rulare a testelor, de generare automatizată a raporturilor cu rezultatele rulării, indicând inclusiv capturile de ecran și jurnalele aplicației, necesare pentru depanarea problemei defectului, astfel ușurând procesul de fixare pentru echipa de dezvoltatori, dezvoltatorii văd erorile mai repede atunci când folosesc teste automate, pentru fiecare versiune de cod, și astfel le pot

remedia mai devreme și la un cost mai mic. Astfel, în pragul perturbărilor tehnologice, investiția și dezvoltarea unui proiect de automatizare a proceselor de testare, devine o soluție optimă, eficientă și comodă.

BIBLIOGRAFIE

1. BURNSTEIN, Ilene. *Practical Software Testing: A Process-Oriented Approach. Springer Professional Computing.* Springer Science & Business Media, 2003. ISBN 9780387951317.
2. HETZEL, Bill, HETZEL, William C. *The Complete Guide to Software Testing*. University of Michigan: QED Information Sciences, 1988. ISBN 9780894352423.
3. *InDevLab: What is Agile Development.* [citat 09.10.2022].
   Disponibil: https://indevlab.com/blog/what-is-agile-development/
4. *Medium: Kanban in Software Development*. Diaame Consulting Services, 2018. [citat 09.11.2022].
   Disponibil: https://medium.com/@teamdiaame/kanban-in-software-development-c47828c49a87
5. *Software Testing Class: Find, Fix and prevent Vulnerabilities*. 2012. [citat 09.11.2022].
   Disponibil: https://www.softwaretestingclass.com/system-testing-what-why-how/
6. *Tricentis: Intelligent test automation*. Tricentis, 2022. [citat 09.15.2022].
   Disponibil: https://www.tricentis.com/products/automate-continuous-testing-tosca#
7. *Smartbear: TestComplete*. SmartBear Software, 2022. [citat 09.15.2022].
   Disponibil: https://smartbear.com/product/testcomplete/
8. SINGH, Manish. API platform Postman valued at $5.6 billion in $225 million fundraise [online]. *Techcrunch*, 2021. [citat 09.15.2022].
   Disponibil: https://techcrunch.com/api-platform-postman-valued/
9. *Postman: Build APIs together*. Postman, Inc., 2022. [citat 09.15.2022].
   Disponibil: https://www.postman.com/
10. *Cloudfront: Aqtrace User Manual.* SmartBear Software, 2011. [citat 09.17.2022].
    Disponibil: https://cloudfront.net/docs/aqtrace_user_manual.pdf
11. *Spring: Spring boot.* VMware, Inc., 2022. [citat 10.06.2022].
    Disponibil: https://spring.io/projects/spring-boot

12. SHRIVASTAVA, Anish. Automation Framework Architecture for Enterprise Products: Design and Development Strategy [online]. *Oracle*, 2012. [citat 10.17.2022].
Disponibil: https://www.oracle.com/technical-resources/articles/enterprise-architecture.

13. HOMBERGS, Tom. Testing with Spring Boot and @SpringBootTest [online]. *Reflectorium*, 2021. [citat 11.06.2022].
Disponibil: https://reflectoring.io/spring-boot-test/

14. SURMANN, Thomas. Best practices for multi-module projects with Spring Boot [online]. *Bootify,* 2022. [citat 11.08.2022].
Disponibil: https://bootify.io/multi-module/best-practices-for-spring-boot-multi-module.html

15. *Istqb-glossary.* Interational Software Testing Qualifications Board, 2018. [citat 11.27.2022]
Disponibil: https://istqb-glossary.page/test-report/