FBIM Transactions

MENADŽMENT KONTROLE KVALITETA SOFTVERA UPOTREBOM BLACK-BOX TESTIRANJA NA POSTOJEĆEM WEBSHOP-U TRINITISHOP

SOFTWARE QUALITY CONTROL MANAGEMENT USING BLACK-BOX TESTING ON AN EXISTING WEBSHOP TRINITISHOP

Pavle Dakić
Univerzitet Singidunum, Beograd, Srbija

Jelena Savić
Univerzitet Singidunum, Beograd, Srbija

Vladimir Todorović
Univerzitet "MB", Beograd, Srbija

JEL kategorija rada: **L15, M15**

Apstrakt

Stvaranje potrebe za stalnim rastom i napretkom svih uključenih i povezanih poslovnih entiteta, zahtevaju ispunjenost određenih web standarda podrškom za više različitih browsera i uređaja. Da bi smo se uverili u verziju objavljenog koda, potreban je tim koji voli izazove, poseduje kreativnost i želju za stalnim učenjem. Kod, kao osnova za uspešno poslovanje mora biti napisan na odgovarajući način sa minimalnim nedostacima u logici i pisanju. Ispravnost i validnost produkcionog koda najviše zavisi od samog programskog tima i njegove odgovornosti za napisani kod. Vitalni kod sa greškama može proizvesti ozbiljne probleme i nepredviđene posledice. Da bi se postigla potpuna tačnost svih delova pisanog koda, potrebno je koristiti testiranje softvera i QA tehnike - osiguranje kvaliteta. Fokus rada je na primeni i pisanju potrebnog programerskog koda koji koristi QA/QC i black-box metod testiranja postojećeg webshop-a.

Ključne reči: *QA, QC, test-case, black-box testing, automatsko testiranje, e-commerce*

Adresa autora zaduženog za korespodenciju
Pavle Dakić
pavledakic@yahoo.com

Abstract

Creating the need for continuous growth and progress of all involved and connected business entities. They require compliance with certain web standards and support for

multiple different browsers and devices. To be convinced of the version of the published code, we need a team that loves challenges, has the creativity and a desire for constant learning. Code, as a basis for a successful business, must be written appropriately with minimal deficiencies in logic and writing. The correctness and validity of the production code mostly depend on the program team itself and its responsibility for the written code. Vital code with errors can produce serious problems and unforeseen consequences. To achieve complete accuracy of all parts of the written code, it is necessary to use software testing and QA technique - quality assurance. The focus of the work is on the application and writing of the necessary programming code that uses QA/QC and the black-box method of testing the existing webshop.

Keywords: *QA, QC, test-case, black-box testing, automated testing, e-commerce*

1. UVOD

Izbijanje COVID-19 pandemije na svetskom nivou dovelo je do ozbiljnih promena uslova poslovanja za B2B i B2C kompanije. Na osnovu izvršenog istraživanja kompanije Deloitte za uticaj COVID-19 na primeru e-trgovine u Danskoj, moguće je videti da će se trajno promeniti način razmišljanja i funkcionisanja ljudskog društva (Nyrop, Nathan, Lindquist, & Karlsen, 2020). Sprovedene ankete u istraživanju (Nyrop, Nathan, Lindquist, & Karlsen, 2020) pokazuju naglo povećanje od 45% upotrebe e-commerce usled izbijanja pandemije. To je dovelo do povećanja potražnje robe i usluga unutar određenih kategorija onlajn prodavnica, koje do sada nisu u potpunosti koristile i razvijale svoje onlajn poslovanje. Kompanije sa fizički dostupnim lokalima moraju da odgovore na izazove štiteći svoje prihode, uprkos privremenim efektima ponašanja potrošača, zbog ne predviđene situacije prilagođavanja. Kompanije su sada primorane da krenu sa što bržom primenom besplatnih e-commerce rešenja, koja mogu imati dosta nepredviđenih problema. Primeri primene uključuju različite trgovce na malo i veliko. Nagle promene u načinu poslovanja u vidu neplaniranog pokretanja instant onlajn poslovanja stvara dosta problema za kompanije koje ne poseduju neophodne hardverske i ljudske resurse, stvarajući prisustvo na Internetu izuzetno važnim aspektom za trenutno poslovanje i opstanak na tržištu tokom trajanja pandemije. Fleksibilnost sistema i njegova sposobnost odgovora na bilo koji zahtev korisnika i treba da olakša svakodnevni život ljudi. Kreiranje konstantnog rasta kompanije i ostvarivanje profita svih uključenih i povezanih entiteta zahteva osiguranje kvaliteta standarda web stranica koje se koriste i prikazuju krajnjem korisniku. Kako bismo se uverili u verziju objavljenog koda, potreban je tim koji voli izazove, poseduje kreativnost i želju za stalnim učenjem što zahteva primenu najboljih tehnika menadžmenta. Glavni razlog za osiguranje i kontrolu kvaliteta se nalazi u sve većoj primeni mobilnih uređaja za različite namene.

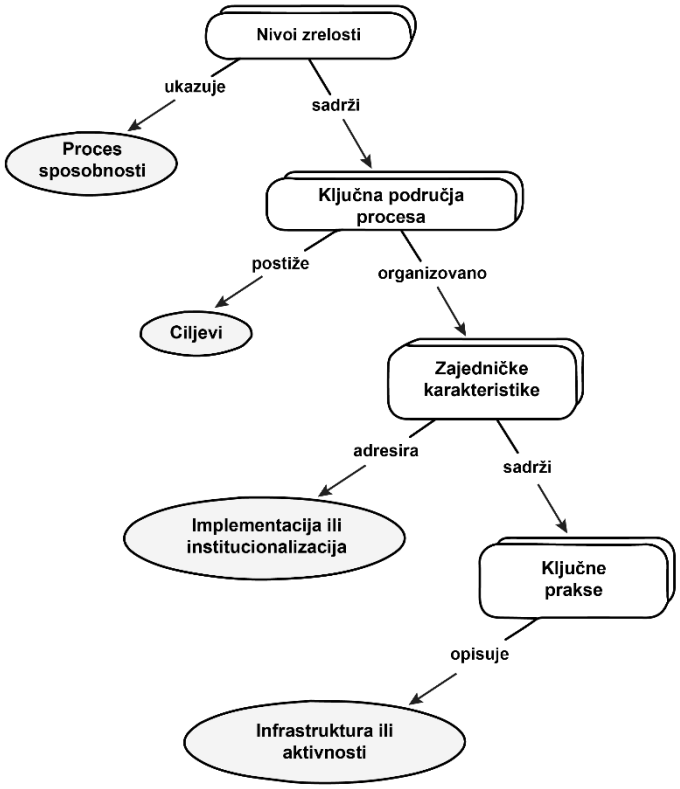
2. RAZVOJ SOFTVERA

U slučaju kreiranja kvalitetnog softvera proces testiranja je veoma naporan, zahtevan i skup zadatak za svaku kompaniju koja želi da bude sigurna u svoj distribuirani softverski proizvod. Osnovni korak u procesu razvoja je prikupljanje informacija od domenskih stručnjaka i/ili vlasnika proizvoda/procesa da bi se napravila lista specifikacija za dugoročni sistem. Tokom ove faze razvoja većinom se uključuju testeri da bi se uveo redosled potpunog razumevanja složenosti sistema. Razvoj softverskog paketa/celine nametnuo je strukturu koja je u sklopu procesa razvoja softvera. Postoji nekoliko modela ovog procesa i svaki opisuje pristupe različitim zadacima i aktivnostima koji se dešavaju tokom razvoja strategije. Sinonimi ovog procesa su „Softverski životni ciklus“ i „Softverski proces“. Razvijanje pouzdanog i upotrebljivog softvera koji se isporučuje na vreme i u okviru budžeta može biti težak poduhvat ako ne postoji jasan plan realizacije. Proizvodi koji kasne, premašuju budžet ili ne funkcionišu na predviđeni način, stvaraju probleme korisnicima krajnjeg proizvoda. Kako se softverski projekti i dalje povećavaju u veličini i značaju, povećava se i broj pitanja.

Problemi se mogu prevazići usmerenim naporima u stvaranju infrastrukture primenom efikasnih i dobrih praksi za razvoj softvera (Paulk, Weber, Garcia, Chrissis, & Bush, 1993). Vizuelni elementi su sastavni deo našeg svakodnevnog života i sve je relevantnije razumevanje njihovog načina komunikacija i komunikativnih efekata koji se ostvaruju prilikom testiranja. Orijentacija prema raznovrsnoj komunikaciji ne preovladava samo u

marketingu, već i u svim vrstama profesionalne komunikacije, uključujući web stranicu kao komunikacioni medijum. Konkretno, deca i odrasli najbolje komuniciraju upotrebom vizuelnih simbola i elemenata kroz njihovu interaktivnost (Dakic, Kocić, Paspalj, & Popović, 2016). Web lokacija predstavlja prozor na mreži koji ljudima predstavlja stvarni svet. Svaki pojedinačni entitet kao deo vizuelnog prikaza na web lokaciji, počevši od boje, dizajna, funkcije koja uključuje lakoću navigacije, pa čak i sadržaja, do vremena učitavanja poseduje određenu važnost za krajnjeg korisnika. Pravilna analiza i testiranje funkcionalnosti nezaobilazni je deo razvojnog ciklusa projekta, gde tim za osiguranje kvaliteta ulazi u svaki definisani scenario, proveravajući ispunjenost zadatih kriterijuma.

Radeći ruku pod ruku sa timom za osiguranje kvaliteta u svakoj fazi izrade web stranica postići će se manje grešaka. Samim tim može se osmisliti i strategija za iskazivanje jedinstvenosti i profesionalnosti u onome što se radi. Proizvođači softvera da bi ispunili preduslove i da bi se podržao model procesa razvoja, koriste određene metode razvoja softvera koji uključuju standarde ISO 12207, CMM, CMMI, ISO 9000, SPICE i mnoge druge.



Slika 1. Struktura modela zrelosti sposobnosti – CMM
Izvor: prilagođena izrada prevoda (Paulk, Weber, Garcia, Chrissis, & Bush, 1993)

Međunarodni standard ISO 12207 (ISO/IEC 12207:2008 - Systems and software engineering — Software life cycle processes, n.d.) se može koristiti za opis strategije, implementaciju i praćenje životnog ciklusa softvera. Model potencijalnog sazrevanja CMM (engl. Capability Maturity Model) je jedan od vodećih modela primene u softveru koji se danas razvija (Paulk, Weber, Garcia, Chrissis, & Bush, 1993). Drugim rečima model zrelosti (CMM) (Paulk, Weber, Garcia, Chrissis, & Bush, 1993) je okvir koji opisuje ključne elemente efikasnog softverskog procesa. Navedeni model opisuje put kontinuiranog poboljšanja od ad hoc, ne planiranog procesa do zrelog, disciplinovanog/uređenog procesa koji može menadžment da planira i kontroliše. Nezavisni revizori ocenjuju organizacije koje primenjuju i podržavaju standarde prilikom razvoja softvera, definišući svoje interne procese u cilju osiguranja kvaliteta softvera.

CMM je postepeno zamenjen sa CMMI (engl. Capability Maturity Model Integration). CMMI model se koristi prilikom pokretanja procesa procene softvera sa akcentom na dizajnu i performansama koje se mogu ostvariti prilikom samog procesa analize zahteva. Model se sastoji iz tri celine koje se obrađuju prilikom procene: razvoj i popravke softvera, pokretanje razvoja sa menadžmentom i upravljanje softverskim modulima i njihovim ispravkama. ISO 9000 opisuje standarde za formalno organizovane procese dokumentacije. ISO 15504 (ISO, 2012), nazvan SPICE određuje mogućnosti unapređenja softverskog procesa (engl. SPICE - Software process improvement capability determination) i može biti okvir za procenu softverskih procesa. Upotrebom određenih standarda stvara se i kreira procena procesa, detaljan opis strukture, ključne komponente modelovanog sistema. Svaki sistem bi trebalo da sadrži u svojoj celini dve dimenzije koje uključuju i sadrže: dimenziju procesa i dimenziju sposobnosti sa pokazateljima procene uspešnosti. Glavni razlog i cilj primene standarda je u utvrđivanju jasnih modela za uporedne procese (ISO, 2012).

3. QA, QC

QA (engl. Quality Assurance) procedura zahteva upotrebu dobre prakse kako bi se dobio objektivni pregled u slučaju ovog rada, vezan za standard vođenja inventara. Praksa je da firme, koje se bave inventarom, izvrše osnovni pregled pre unošenja zaliha, kako bi uočile potencijalne probleme i izvršile korekcije gde je to moguće. Takođe je dobra praksa da se ovaj pregled koristi za bilo koju ili sve kategorije izvora u okviru inventara proizvoda. QA/QC (engl. QC - Quality Control) plan može biti osnovni element QA/QC sistema i dobra je praksa za razvijanje daljeg softvera. U planu bi trebalo da budu predstavljene QA/QC aktivnosti koje su sprovedene zajedno sa planiranim vremenskim okvirom koji prati pripremu inventara od njegovog početnog stanja do konačnog izveštavanja u bilo kojoj verziji. Zahteva postojanost opisa metoda i rasporeda svih kategorija proizvoda i njihovih specifikacija (IPCC, 2006). Osiguranje kvaliteta (QA) omogućava aktivno učešće u organizaciji programskog koda i kao rezultat dovodi do uspešnog procesa prijave u osiguranju kvaliteta. QA se više fokusira na prilagođeni proces proizvoda za kupca. Organizacija mora da obezbedi efikasnost i efektivnost kako bi uključivala standarde kvaliteta za novi softver. Osiguranje kvaliteta se često naziva i kontrola kvaliteta. QA testiranje se može objasniti kao postupak za obezbeđivanje softverskog proizvoda najvišeg kvaliteta za krajnje kupce. QA su isključivo tehnike kojima se sprečavaju problemi sa uslugama i proizvodima, kako bi se osiguralo sjajno korisničko iskustvo za kupce. Prema IEEE standardu za dokumentaciju o testiranju softvera (IEEE Standard for Software Test Documentation, n.d.), dokument plana testa po standardu bi trebalo da sadrži određene definisane informacije.

Rešenje za elektronsku trgovinu niskog kvaliteta je rezultat nezadovoljnih klijenata, što može doneti izgubljeni prihod i lošu reputaciju brenda. Efikasno testiranje e-trgovine je pouzdana metoda za sprečavanje takvih rizika. Najčešći primer provere upotrebom testiranja QA/QC je pisanje Test slučaj scenarija koji pokrivaju celokupan proces kupovine od početka do isporuke. Iako nije odgovornost direktora za e-trgovinu da upravlja načinom na koji se rešenje testira, oni će se direktno suočiti sa posledicama pokretanja greške web lokacije za e-trgovinu ili zanemariti redovna testiranja sa ciljem provere sigurnosti, funkcionalnosti, performansi i drugih aspekata rešenja. Važne tačke na koje se direktor/menadžer e-trgovine može usredsrediti je da proveri da li je postupak testiranja koji sprovodi interni ili spoljni QA tim na pravom putu. Pre same kupovine verovatno će posetioci stranice pregledati artikle u ponudi. Slaba funkcionalnost pretrage i mala brzina učitavanja definitivno će razočarati potencijalne kupce i rezultirati niskom konverzijom rešenja za e-trgovinu (Director's Guide to Effective Ecommerce Testing, n.d.).

4. TEST SLUČAJ

Test slučaj (engl. Test case) može biti skup radnji koje se izvršavaju radi verifikacije odabrane funkcije ili funkcionalnosti softverske aplikacije. Test slučaj sadrži niz radnji: korake ispitivanja, podatke o testiranju, preduslove. Izvršavanje unutar programerskog koda proverava da li će biti izvršeni definisani uslovi i koji je njihov krajnji ishod. Izvršavanjem testa vrši se verifikacija slučaja ispitivanja primer Tabela 1. Proces definisanih test akcija uključuje određene varijable ili uslove, pomoću kojih test inženjer ili testing tim

može uporediti očekivane i dobijene stvarne rezultate da bi se proverio način funkcionisanja softverskog paketa prema postavljenim zahtevima klijenta i autora Test slučaja. Većina testera koristi Excel za pisanje testova, zbog mogućnosti lakog grupisanja test slučajeva po tipovima testova sa mogućnošću prikaza potrebnih grafikona.

Glavni nedostatak napisanih Excel test-case scenarija je u otežanom praćenju i ubrzanom povećavanju broja napisanih testova, kada je veoma teško upravljati i nadgledati buduće pisanje. Iz tog razloga za upravljanje i izvršavanje potrebnih testova je bolja opcija primena plaćenih alata ili softver otvorenog kôda. U cilju jednostavnosti i efikasnosti testiranja navedenog veb sajta TrinitiShop, korišćen je Excel i napisani su osnovni TestCase scenariji (SoftwareTestingHelp, 2020).

Tabela 1. Amount Test Case

Test Case		
Author	Jelena Savić	
Test Case ID	1	
Test Case Name	Changing the amount	
Description	This test case is used to verify the ability to change the amount in the cart	
Step #	**Steps**	**Expected Result**
1	Open Google Chrome	
2	Navigate to https://trinitishop.pdpro.media	
3	Click on: "AUTOMOBILI"	
4	Mouseover on the product item	
5	Click on the card icon	
6	Tick on check box: "Zatamnjena stakla"	
7	Choosefromdropdown "Veličina felni": "22"	
8	Click on button: "DODAJ U KORPU"	
9	Click on: "PREGLED KORPE"	
10	Click on combobox "Količina"	
11	Clear ComboBox	
12	Enter new value: "2"	
13	Click on: "AŽURIRANJE KORPE"	The amount is updated

5. BLACK-BOX TESTIRANJE

Funkcionalno testiranje je softverski program ili sistem koji se testira i uzima se u obzir "crna kutija" (engl. Black-Box). Izbor test slučajeva za funkcionalno testiranje zasniva se na potrebi ili specifikaciji dizajna softverskog subjekta koji se testira. Uzorci očekivanih rezultata ponekad se spominju kao predviđanja, uključujući zahteve / specifikacije dizajna, vrednosti koje se računaju na osnovu simuliranih testiranja i dobijenih rezultata. Funkcionalno testiranje se uglavnom fokusira na eksterno ponašanje softverskog entiteta. Black box testiranje poznato kao ispitivanje ponašanja (engl. Behavioral Testing) je metoda testiranja softvera u kojoj unutrašnja struktura / dizajn / primena softvera, koji se testira nije poznata testeru. Testiranje u crnoj kutiji može biti metoda testiranja softvera koja ispituje funkcionalnost aplikacije bez poznavanja njene unutrašnje strukture ili načina rada.

6. AUTOMATSKO TESTIRANJE I PRIMENA TESTOVA – JAVA

Automatsko testiranje i pisanje odgovarajućih testova, zahteva određene softverske pakete, koji poseduju funkcionalnosti, koje omogućavanju pisanje i izvršavanje TestCase scenarija. Za potrebe uspešnog izvršavanje testova i

mogućnosti testiranja korišćeni su sledeći Maven paketi:

- Selenium - omogućava upravljanje elementima stranice
- WebDriverManager - olakšava upravljanje potrebnim verzijama browser-ima
- TestNG - koristi se za pisanje i pokretanje testova
- Junit - koristi se za pisanje i pokretanje testova
- slf4j-simple - Jednostavna evidencija facade za Java-u pruža Java evidenciju API-ja pomoću jednostavnog uzorka facade

Automatsko testiranje i primena testova je izvršena na postojećem online webshop-u TrinitiShop. Nakon pokretanja testova, upotrebom IntelliJ moguće je praćenje izvršavanja koraka sa prikazom sistemskih poruka o uspešnosti testiranja, kao i brzina izvršavanja za svaki od navedenih koraka.

Maven

Maven je alat za automatizaciju preuzimanja paketa koji se prevashodno koriste za Java projekte. On se takođe može koristiti za izgradnju i upravljanje projektima. Informacije o potrebnim paketima se čuvaju unutar **pom.xml** fajla (Kôd 1). Pom.xml sadrži informacije o svim napisanim testovima i njihovom redosledu izvršavanja na osnovu prioriteta.

```
<?xmlversion="1.0" encoding="UTF-8"?>
<projectxmlns="http://maven.apache.org/POM/4.0.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">
<modelVersion>4.0.0</modelVersion>

<groupId>rs.ac.singidunum</groupId>
<artifactId>diplomski-singidunum</artifactId>
<version>1.0-SNAPSHOT</version>

<properties>
<project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
<project.reporting.outputEncoding>UTF-8</project.reporting.outputEncoding>
<maven.compiler.source>14</maven.compiler.source>
<maven.compiler.target>14</maven.compiler.target>
</properties>
<dependencies>
<!-- https://mvnrepository.com/artifact/org.seleniumhq.selenium/selenium-java -->
<dependency>
<groupId>org.seleniumhq.selenium</groupId>
<artifactId>selenium-java</artifactId>
<version>3.141.59</version>
</dependency>
<!-- https://mvnrepository.com/artifact/junit/junit -->
<dependency>
<groupId>junit</groupId>
<artifactId>junit</artifactId>
<version>4.13</version>
<scope>test</scope>
</dependency>
<!-- https://mvnrepository.com/artifact/org.testng/testng -->
<dependency>
<groupId>org.testng</groupId>
<artifactId>testng</artifactId>
<version>7.1.0</version>
<scope>test</scope>
</dependency>
```

```
<!-- https://mvnrepository.com/artifact/io.github.bonigarcia/webdrivermanager -->
<dependency>
<groupId>io.github.bonigarcia</groupId>
<artifactId>webdrivermanager</artifactId>
<version>4.0.0</version>
</dependency>
<!-- https://mvnrepository.com/artifact/org.slf4j/slf4j-simple -->
<dependency>
<groupId>org.slf4j</groupId>
<artifactId>slf4j-simple</artifactId>
<version>1.7.30</version>
<scope>test</scope>
</dependency>

</dependencies>
</project>
```

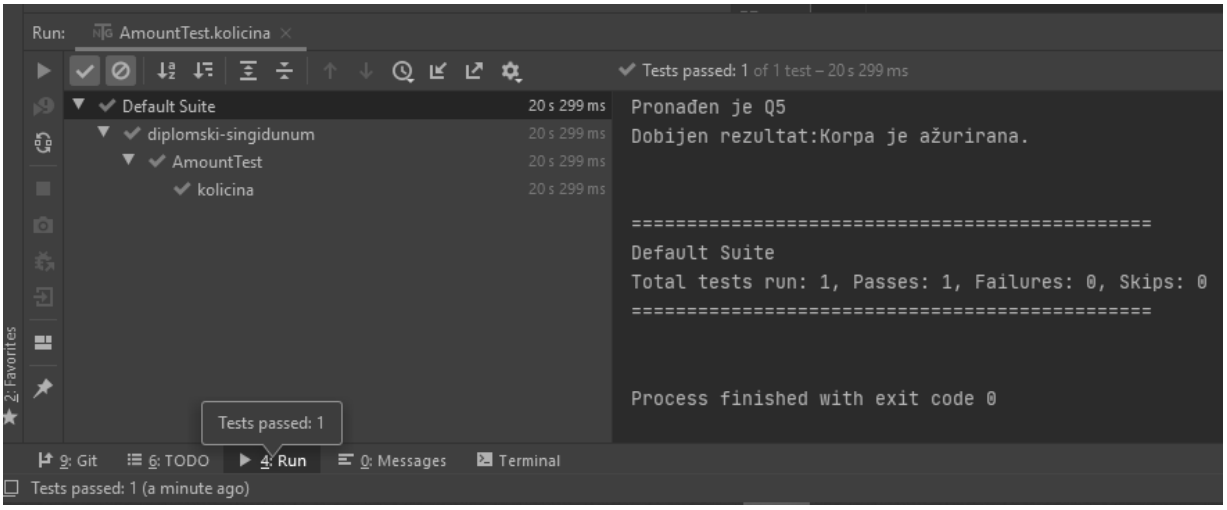
Kôd 1. *pom.xml*

```
import io.github.bonigarcia.wdm.config.DriverManagerType;
import io.github.bonigarcia.wdm.managers.ChromeDriverManager;
import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.chrome.ChromeDriver;
import org.openqa.selenium.support.ui.ExpectedConditions;
import org.openqa.selenium.support.ui.Select;
import org.openqa.selenium.support.ui.WebDriverWait;
import org.testng.Assert;
import org.testng.annotations.Test;

import java.util.List;

publicclassAmountTest {
@Test
publicvoidkolicina() {
      ChromeDriverManager.getInstance(DriverManagerType.CHROME).setup();
WebDriverchrome = newChromeDriver();
String url = "https://www.trinitishop.pdpro.media";
chrome.get(url);

// Define elements
WebElement automobili = chrome.findElement(By.xpath("//*[@id=\"primary-menu\"]/ul/li[2]/a"));
automobili.click();

// Take each and every tag which have an id attribute inside the list
List<WebElement>automobili_title = chrome.findElements(By.tagName("h2"));

// Count elements
System.out.println("Total tagswith is: " + automobili_title.size());

// Print all id values one by one
for(inti =0;i<automobili_title.size();i++){
System.out.println("Tagvalue "+ i +" is: " + automobili_title.get(i).getText());
}

for(WebElement proizvod: automobili_title) {
```

```
System.out.println(proizvod.getText().contains("AUDI Q5"));
if(proizvod.getText().contains("AUDI Q5")){
System.out.println("Pronađen je Q5");
proizvod.click();
break;
}
      }

// Define elements
WebElement stakla = chrome.findElement(By.className("wcpa_check"));
Selectfelne = newSelect(chrome.findElement(By.name("select-1565126249735")));
WebElementkorpa_button = chrome.findElement(By.name("add-to-cart"));

// Enter data
stakla.click();
felne.selectByVisibleText("22");
korpa_button.click();
WebElementpregled_korpe = chrome.findElement(By.xpath("//*[@id=\"main\"]/div[1]/div/a"));
pregled_korpe.click();

//Define element
WebElementkolicina_promena =
chrome.findElement(By.name("cart[4a84ff67af39d7da7d86cc3705b0de29][qty]"));

//Enter data
kolicina_promena.clear();
kolicina_promena.sendKeys("2");

//Define element
WebElementazuriranje_korpe = chrome.findElement(By.name("update_cart"));

//Enter data
azuriranje_korpe.click();

WebDriverWaitwait = newWebDriverWait(chrome, 10);
wait.until(ExpectedConditions.visibilityOfElementLocated(By.xpath("//*[@id=\"post-
6\"]/div[2]/div/div[1]/div")));

// Assert
WebElement rezultat = chrome.findElement(By.xpath("//*[@id=\"post-6\"]/div[2]/div/div[1]/div"));

String ocekivani_rezultat = "Korpa je ažurirana.";

System.out.println("Dobijen rezultat:" +rezultat.getText());
Assert.assertEquals(rezultat.getText(),ocekivani_rezultat);
}
}
```

Kôd 2. *AmountTestCase.java*

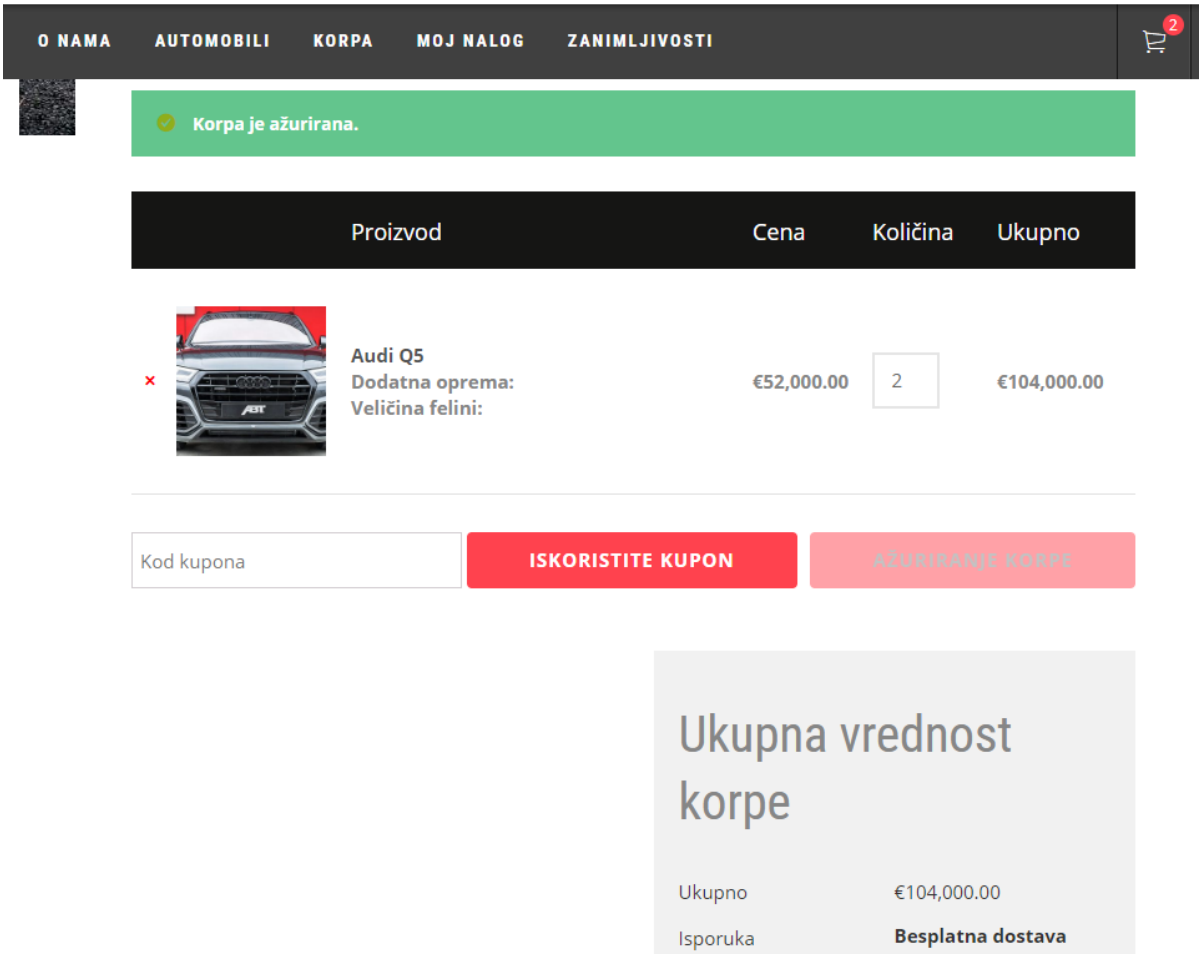
Slika 2. prikazuje brzinu izvršavanja, broj pokrenutih testova, uspešnost izvršavanja testa, sistemske poruke, pronalazak elemenata Q5 prilikom pokretanja Chrome browser-a upotrebom IntelliJ IDE-a. Prilikom izvršavanja Kôd 1. upotrebom xpath-a traži se ID klasa (primary-menu) i izvršava se klik. Nakon čega se uzimaju svi elementi koji imaju h2 atribute i smeštaju se unutar liste. Upotrebom funkcije size() dobija se ukupan broj pronađenih ID elemenata koji imaju h2 atribut.



Slika 2. Izvršavanje AmountTest kôda

Izvor: sopstvena izrada

Slika 3. prikazuje proces automatizovanog testiranja primenom gore navedenog koda (Kôd 2). Na vizuelnom prikazu moguće je videti uspešnost pronalaska vozila Audi Q5 i promenu količine unutar korpe kupca. U slučaju da navedeni model ne postoji ili da nije moguće izvršiti promenu količine usled nedostatka proizvoda/vozila, doći će do pucanja testa.



Slika 3. Pronalazak elemenata i promena količine

Izvor: sopstvena izrada

Timovi koji rade na brzoj isporuci softvera vrhunskog kvaliteta u svom radu, koriste kontinuiranu integraciju CI (engl. CI - Continuous Integration) za pokretanje svih testova, dobijajući odgovarajuće rezultate testiranja. CI je postala sastavni deo softverskog ekosistema zajedno sa primarnim alatom Jenkins. Jenkis je najzanimljivi alat kada je u pitanju kontinuirani razvoj i integracija softvera, igrajući ključnu ulogu u životnom ciklusu razvoja softvera. Primena CI alata kao što je Jenkins, omogućava programerima automatizaciju projekata, kreiranje izveštaja i vođenje računa o ispravnosti napisanog koda; time kreirajući dodatno vreme za testiranjem i osiguravanjem kvaliteta napisanog softverskog koda.

Automatsko testiranje pomaže timu u automatizaciji testova koji su deo CI razvojne linije/razvojne cevi (engl. Development Pipeline), postižući kontinuiranu isporuku različitih funkcionalnih verzija. Primenom automatskog testiranja moguće je koristiti različite tipove testova: funkcionalne, regresione, integracione i jedinstvene testove. Svi navedeni tipovi mogu se pokretati upotrebom odgovarajućeg CI alata, kao što je Jenkins ili neki drugi.

7. ZAKLJUČAK

Globalne promene u onlajn poslovanju i stepenu prihvaćenosti su pokazatelj da se možda nikada nećemo vratiti na tradicionalni način prodaje i potpuno otvaranje društva za socijalnom interakcijom. Kvalitet web stranica mora biti na izuzetno visokom nivou. Osiguravanje kvaliteta podrazumeva veoma dobru tehničku podršku, ljubaznost prema korisniku, sa odgovarajućom brzinom i efikasnošću web stranice koju korisnik koristi. Postoje tehničke greške koje mogu predstavljati ogromne probleme i uticati na kupovinu, registraciju i preuzimanje na mreži. Različite veličine ekrana i verzije pretraživača, mogu dovesti do problema prilikom izvršavanja animacija ili drugih grafičkih elemenata, koje se ne prikazuju i ne funkcionišu na određenim pretraživačima. Podjednako je potrebno osigurati da se web stranica brzo učita. Softver igra važnu ulogu za klijenta, zbog čega je stalna komunikacija veoma važna, kako bi se ispunili skoro svi njegovi zahtevi i došlo do traženog kraja rezultata u vidu primenljivog softvera. Zato inženjeri sa solidnim iskustvom u radu na razvoju softvera moraju postaviti principe tzv. agilnih metodologija razvoja softvera, koje odgovaraju specifičnostima projekata. Glavni principi definisanih metodologija se navode u agilni manifest sa deklaracijom zavisnosti u vidu upotrebe iterativnog i inkrementalnog pristupa, fleksibilnosti rada, vrste verzija (daily, nightly, beta build), vreme isporuke proizvoda/modula u delovima sa uvođenjem promena, itd. Radi uspešne realizacije web aplikacija, neophodno je razumevanje osnova programskih jezika i problema koji se mogu javiti pri primeni. Testiranje softvera i kontrola kvaliteta su procesi kojima se poboljšava kvalitet aplikacija. Testiranje se vrši u svakoj fazi razvoja softvera, počevši od specifikacija zahteva, dizajna, kôdiranja, do trenutka kada krajnji korisnici prihvate aplikaciju i nakon toga. Zbog toga je neophodno napraviti adekvatan sistem testiranja koji bi omogućio razvojnom timu pravovremeno uočavanje eventualnih problema i njihovo rešavanje. Pandemija je brzo promenila naše ponašanje prema internet kanalima, a verovatno će doći i do trajnih promena i nakon nje. Kontrola kvaliteta softvera će mnogim kompanijama biti konstantan izazov da prežive u kratkom roku procesa prilagođavanja. Kriza takođe predstavlja priliku za odvažne kompanije, da ambiciozno i pravovremeno ulažu u svoje internet poslovanje kako bi postali tržišni lideri.

CITIRANA DELA

Dakic, P., Kocić, S., Paspalj, D., & Popović, M. (2016). Importance of responsive web design for education of students using faculty website. *Sinteza.*

Director's Guide to Effective Ecommerce Testing. (n.d.). Retrieved from https://www.scnsoft.com/blog/ecommerce-testing-guide

IEEE Standard for Software Test Documentation. (n.d.). (IEEE) Retrieved from https://ieeexplore.ieee.org/document/7106441

IPCC. (2006). *Quality assurance and Quality control*. Preuzeto sa https://www.reddcompass.org/mgd-content-v2/dita-html/en/s2_3_4.html

ISO. (2012, 02). *ISO/IEC 15504-5:2012 Information technology — Process assessment — Part 5: An exemplar software life cycle process assessment model*. Retrieved from ISO: https://www.iso.org/standard/60555.html

ISO/IEC 12207:2008 - Systems and software engineering — Software life cycle processes. (n.d.). Retrieved from https://www.iso.org/standard/43447.html

Nyrop, M., Nathan, A., Lindquist, M. B., & Karlsen, J. T. (2020). *COVID-19 will permanently change e-commerce in Denmark*. Preuzeto sa Deloitte: https://www2.deloitte.com/content/dam/Deloitte/dk/Documents/strategy/e-commerce-covid-19-onepage.pdf

Paulk, M. C., Weber, C. V., Garcia, S. M., Chrissis, M. B., & Bush, M. (1993). *Key Practices of the Capability Maturity ModelSM, Version 1.1* (Vol. February 1993). Pittsburgh, Pennsylvania: Carnegie Mellon University. doi:https://pdfs.semanticscholar.org/6d36/a04c74bf128d96cf8d03ece19cb7189f1bea.pdf

SoftwareTestingHelp. (2020, 11 13). *Sample Test Case Template With Test Case Examples*. Retrieved from Software Testing Help: https://www.softwaretestinghelp.com/test-case-template-examples/