

SVEUČILIŠTE JURJA DOBRILE U PULI
FAKULTET INFORMATIKE

Kristijan Cetina

**Testiranje klijentskih komponenti web aplikacija pomoću alata Playwright i
Cypress**

DIPLOMSKI RAD

Pula, 2024.

SVEUČILIŠTE JURJA DOBRILE U PULI
FAKULTET INFORMATIKE

Kristijan Cetina

**Testiranje klijentskih komponenti web aplikacija pomoću alata Playwright i
Cypress**

DIPLOMSKI RAD

JMBAG:	2424011721, izvanredni student
Studijski smjer:	Informatika
Kolegij:	Diplomski rad
Znanstveno područje:	Društvene znanosti
Znanstveno polje:	Informacijske i komunikacijske znanosti
Znanstvena grana:	Informacijski sustavi i informatologija
Mentor:	dr.sc. Nikola Tanković

Pula, travanj, 2024. godine

Zahvala

Zahvaljujem svojem mentoru na izdvojenom vremenu i podršci, kako na izradi ovog rada, tako i tijekom cijelog studiranja na Fakultetu informatike.

Zahvaljujem se i svojim timskim kolegama, s kojima sam od samog početka sudjelovao na svim timskim zadacima i njihovoj pomoći pri individualnom radu.

Zahvaljujem se i svim ostalim profesorima i djelatnicima na nesebičnoj potpori kada je god to bilo potrebno.

Naposljetku veliko hvala mojoj obitelji na potpori i razumijevanju tijekom mojeg ponovnog studiranja.

Cool quote

Izjava o samostalnosti izrade završnog rada

Izjavljujem da sam završni rad na temu *Testiranje klijentskih komponenti web aplikacija pomoću alata Playwright i Cypress* samostalno izradio uz pomoć mentora, koristeći navedenu stručnu literaturu i znanje stečeno tijekom studiranja. Završni rad pisan je u duhu hrvatskoga jezika.

Student: Kristijan Cetina

Sažetak

Sažetak HR

Ključne riječi

Playwright, JavaScript, open-source,

Sommario

Sommario IT

Parole chiave:

Playwright, JavaScript, open-source,

Abstract

Abstract EN

Keywords:

Playwright, JavaScript, open-source,

Popis oznaka i kratica

Oznaka	Opis	Jedinica
t	vrijeme (sekunda)	s
θ	temperatura (Celzijev stupanj)	$^{\circ}C$
ν	brzina	m/s
s	udaljenost u metrima	m
f	frekvencija	Hz
C	kapacitet kondenzatora	F
	Veličina memorije	MB, 1MB = 1048576 bajtova

Kratica	Opis
GPS	Global Positioning System - Sustav globalnog pozicioniranja
SD	Secure Digital - format memorijske kartice
μ SD, microSD	mikro Secure Digital - kartica manjih fizičkih dimenzija
PWM	Pulse Width Modulation - Pulsno-širinska modulacija
IDE	Integrated Development Environment - Integrirano razvojno okruženje
GND	Točka nultog potencijala
SW	Software
HW	Hardware
NMEA	National Marine Electronics Association
UTC	Coordinated Universal Time - Standardno vrijeme
.csv	comma-separated values - vrijednosti odvojene zarezom
.md	Markdown datoteka

Korišteni strani pojmovi

Pojam	Opis
Product owner	osoba zadužena za određivanje prioriteta zahtjeva

Sadržaj

Sažetak	IV
Ključne riječi	IV
Sommario	IV
Parole chiave:	IV
Abstract	IV
Keywords:	IV
Popis oznaka i kratica	V
0 Uvod i opis zadatka	1
0.1 Opis i definicija problema	1
0.2 Cilj i svrha rada	1
0.3 Hipoteza rada	1
0.4 Metode rada	1
0.5 Struktura rada	2
1 Uvod u testiranje programskog rješenje i osiguranje kvalitete	3
1.1 Proces testiranja	3
1.2 Ciljevi testiranja	4
1.3 Uloga testera u timu	5
2 Playwright	6
2.1 Opis i pregled paketa	6
2.2 Instalacija	6
2.3 Osnovni test	6
2.4 Kontinuirana integracija - CI	6
3 Testiranje nakon nadogradnje na novu verziju	7
3.1 Postojeći način testiranja	7
3.2 Automatsko testiranje procesa nadogradnje	8
4 Zaključak	10
Literatura	11
Popis slika	11
A Programski kod ...	12

Poglavlje 0

Uvod i opis zadatka

Tema ovog rada proizašla je iz autorove želje za proučavanjem tematike te kao gorljivim poklonikom metode učenja kroz praktičan rad i primjenu stečenog znanja i iskustva na rješavanje realnog problema.

0.1 Opis i definicija problema

gdfg

0.2 Cilj i svrha rada

yeryy

0.3 Hipoteza rada

Hipoteza ovog rada je da promjenom primjerenih metoda testiranja programskog proizvoda može se značajno smanjati količina grešaka (*bugova*, *engl. bugs*) u finalnom proizvodu koji se isporučuje krajnjem korisniku te ostvariti uštede u resursima za njihovo ispravljanje.

0.4 Metode rada

Tijekom izrade ovoga rada korištene su različite znanstveno-istraživačke metode od kojih je svaka najprikladnija postavljenom izazovu, a one su:

- Istraživačka metoda - za stjecanje uvida u zadane okvire zadatka
- Metoda logičke analize i sinteze - za prikupljanje podataka iz literature
- Deskriptivna metoda - za izradu uvodnog i završnog dijela projektnog zadatka
- Eksperimentalna metoda - u potrazi za optimalnim rješenjima za zadani dio problema

0.5 Struktura rada

Struktura ovoga rada podjeljena je u logičke cjeline. Nakon uvoda i objašnjavanja rada, u poglavlju

Kompletan Git repozitorij ovog rada javno je dostupan na <https://github.com/KristijanCetina/jsTesting>

Uvod u testiranje programskog rješenje i osiguranje kvalitete

```
graph LR; BR[Business requirements] --> SR[System requirements]; SR --> TR[Technical requirements]; TR --> CS[Component specification]; CS --> C[Coding]; C --> UT[Unit testing]; UT --> IT[Integration testing]; IT --> ST[System testing]; ST --> AT[Acceptance testing]; AT --> BR; ST --> SR; IT --> TR; UT --> CS;
```

Legend:

- Thin grey arrow: Verification & validation
- Thick black arrow: Development & integration

1.1 Proses testiranja

Tokom samog razvijanja nove funkcionalnosti u softwareskom proizvodu često se izvršava ručno testiranje kako bi se utvrdilo da li proces razvoja ide u zadanom smjenu i u konačnici da se radi ono što je dogovoreno bilo interno s timom ili čak i sa klijentom.

Nakon što se završi sam razvoj nove funkcionalnosti onda se radi i završno funkcionalno testiranje i izrada automatskih testova koji će se u budućnosti izvršavati automatski u zadanom intervalu kako bi se osiguralo da uvođenje nove funkcionalnosti ne uvode nove pogreške na već ispravnim funkcionalnostima. Poznato kao i regresijsko testiranje.

Bitno je testirati realne scenarije koji se očekuju da moraju zadovoljiti, kao i one scenarije od kojih se očekuje da nesmiju proći test. To se radi u svrhu potvrde da test zaista radi ono što je namijenjen, a ne da imamo propust u samom testu koji uvijek vraća pozitivan rezultat ili da testirana funkcija nema implementiranu validaciju ulaznih parametara koji onda mogu izazvati nepoželjno ponašanje programa.

1.2 Ciljevi testiranja

Ciljevi testiranja moraju zadovoljavati nekoliko kriterija, a to su:

- Specifičnost
- Mjerljivost
- Ostvarljiv
- Realističan
- Vremenski ograničen

Neki od ciljeva testiranja su: [1]

Verifikacija i validacija

Cilja testiranja nije samo pronaći pogreške u kodu ili dizajnu. Cilj je verificirati da software zaista radi ono što je namijenjen i kako je zamišljen. Jedan od rezultata testiranja je i izvještaj (*test report*).

Prioretiziranje pokrivenosti

U idealnom svijetu sa neograničenim resursima svaki dio koda i funkcionalnosti bi bio pokriven testovima, ali nažalost to nije moguće. Zato je bitno pravilno odrediti što je prioritet te što će se pokriti testovima. U pravilu su to one funkcionalnosti i značajke koje nisu vidljive na prvi pogled čim se otvori program jer su takvi problemi lako uočljivi svakome. Isto tako, beskonačni testovi uzimaju mnogo dragocjenog vremena pa je i u tom pogledu bitno odrediti što se treba testirati.

Sljedivost

Dokumentiranje testova kada se nešto i kako testiralo je bitno kako bi se u slučaju pojave problema moglo odrediti kada je i koja promjena uzrokovala neželjeno ponašanje proizvoda. To je posebno bitno u određenim kategorijama softwarea kao npr. u financijskom poslovanju gdje se može tražiti dodatna odgovornost samog proizvoda.

1.3 Uloga testera u timu

Glavna uloga testera, kao i samog procesa testiranja, je dodatna sigurnosna mreža koja je samo zadnja karika u lancu procesa testiranja i osiguranja kvalitete proizvoda. Dok su programeri (*developeri*, *engl. developers*) zaduženi za pisanje unit testova, testeri su zaduženi za pisanje i izvršavanje funkcionalnih testova. Testeri također pomažu product owneru sa izradom kriterija za uspješno prihvatanje rezultata testova [2]. Bitno je napomenuti kako su i developeri i testeri dio istog tima te kao tim imaju zajednički cilj - isporuka najkvalitetnijeg proizvoda moguće unutar zadanih parametara.

Poglavlje 2

Playwright

Playwright je open-source biblioteka za automatizaciju testiranja web preglednika i web skrapanja koju je razvio Microsoft. Omogućuje automatizaciju testiranja web aplikacija na Chromiumu, Firefoxu i WebKit-u s jednim API-jem.

Prednosti Playwrighta:

- Jednostavan za korištenje: Playwright ima intuitivan API koji je sličan JQuery-ju i Cypress-u.
- Brz i pouzdan: Playwright je optimiziran za brzinu i pouzdanost, što ga čini idealnim za testiranje web aplikacija u produkciji.
- Svestran: Playwright se može koristiti za testiranje različitih tipova web aplikacija, uključujući jednostavne web stranice, jednostruke web aplikacije (SPA) i višestruke web aplikacije (MPA).
- Podržava više jezika: Playwright se može koristiti s raznim jezicima programiranja, uključujući JavaScript, TypeScript, Python, Java i C#.

Playwright se može koristiti za:

- Automatizaciju UI testova: Playwright se može koristiti za pisanje automatiziranih UI testova koji provjeravaju funkcionalnost web aplikacija.
- Web skrapanje: Playwright se može koristiti za prikupljanje podataka sa web stranica.
- Generiranje screenshot-ova i videozapisa: Playwright se može koristiti za generiranje screenshot-ova i videozapisa web stranica.

2.1 Opis i pregled paketa

2.2 Instalacija

2.3 Osnovni test

2.4 Kontinuirana integracija - CI

Poglavlje 3

Testiranje nakon nadogradnje na novu verziju

Jedna od čestih aktivnosti svakog razvojnog tima je nadogradnja programa na novu verziju. U poduzeću je praksa da se pušta u produkcijski rad nova verzija programa jednom mjesečno za glavnog klijenta. Kako klijent ima postrojenja na 6 kontinenta uz dodatna postrojenja koja se nalaze u međunarodnim vodama svih svjetskih oceana proces nadogradnje je podijeljen po regijama. To su Americas (Sjeverna i Južna Amerika), EU (Europa i Bliski istok + Afrika) te AP (Azija i Pacifik). Svaka od regije ima više od 15 postrojenja koja koriste zasebne servere što znači da se nadogradnja vrši na više od 15 lokacija po regiji. Već iz ovoga je izvjesno kako je to puno potencijalnih problema i zastoja koja se mogu pojaviti te treba osigurati da je proces nadogradnje čim je više optimalan bez nepotrebnih zastoja i prekida u radu.

3.1 Postojeći način testiranja

Do uvođenja automatskih testova standardna procedura se sastojala od toga da AM regiju detaljno testira 3-4 testera (inženjera za kontrolu kvalitete - QA) koji bi svaki od njih provjerio 3 do 4 postrojenja (site). U prosjeku, za provjeriti jedno postrojenje je trebalo 20 do 30 minuta kako bi se osiguralo da su sve promjene aplicirane ispravno te da nisu izazvale neželjene nuspojave i prestanak rada postojećih funkcionalnosti. Naravno, zbog velikog broja funkcionalnosti te nedostatka vremena da se svaka od njih ručno provjeri bili smo vrlo izbirljivi što će se testirati s obzirom na kritičnost neke funkcionalnosti. Cijeli taj proces bi trajao oko 1:30 do 2 sata po članu test tima, a sve ukupno bi se trošak procijenio na 6-7 radnih čovjek-sati. Ako to pomnožimo s cijenom rada lako se dolazi do ukupnog troška testiranja nakon nadogradnje.

Nakon što se verificiralo da je novi paket programa ispravan, te ako klijenti nisu prijavili ozbiljne probleme koji ih sprječavaju u radu putem sustava za prijavu kvara (*showstopper*), moglo se pristupiti nadogradnji sljedeće regije Asia Pacific.

Tada bi se ponovila opet ista procedura, uz eventualne izmjene da bi bio 1 član testinog tima manje kako bi se smanjili prekovremeni sati jer nije bilo potrebe za provjerom svih funkcionalnosti, već je bilo prihvatljivo da se provjeri i reducirani set za koji bi trebalo maksimalno 20 minuta po postrojenju što bi ukupno iznosilo oko 5 čovjek-sati po regiji. Iako je to manji trošak, i dalje je značajni trošak. Čim više što se je termin za radove na AP regiji djelomično izvan redovnog radnog vremena poduzeća tako da treba uračanti i trošak prekovremenog rada. To je posebno izraženo za EU

regiju kod koje pak termin za radove je u potpunosti izvan redovnog radnog vremena pa je u tom slučaju sav rad je prekovremeni rad.

Uz sve navedeno dodatan problem ručnog testiranja je točnost i preciznost istog. Isto tako je normalno za očekivati kako će različiti tester i odraditi posao na različitoj razini kvalitete jer ipak nisu strojevi. Zato smo odlučili taj posao prepustiti strojevima. Barem u mjeri koliko je to moguće.

Iz gore navedenog je razvidno kako je bilo potrebno poboljšati proces testiranja nadogradnje primarno iz razloga povećanja kvalitete posla.

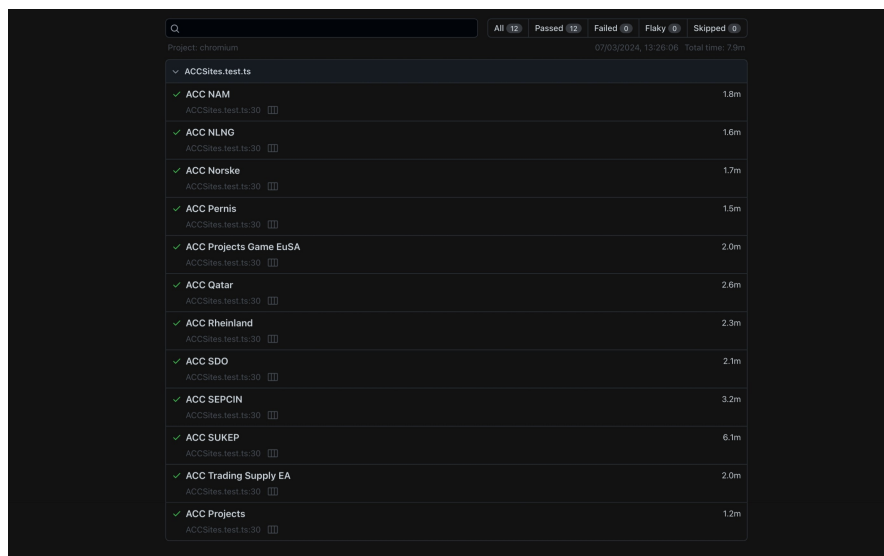
3.2 Automatsko testiranje procesa nadogradnje

Glavni cilj prelaska na automatski način testiranja je bio povećati preciznost testova i osigurati da se ne izostavi niti jedan korak koji se provjeravao pri ručnom testiranju.

Drugi cilj je bio smanjiti vrijeme potrebno za testiranje sa prethodnih 30 minuta na ispod 5 minuta.

Dodatan cilj je bio omogućiti daljni rast broja klijenata i instalacija koja se mogu nadograditi istovremeno, a bez da se mora uložiti značajno više vremena. Idealno smo htjeli postići $\mathcal{O}(\log n)$ rast potrošenog vremena za svakog novog klijenta.

Nakon što je provedena nadogradnja s automatskim testiranjem potvrđeno je da su glavni ciljevi ostvareni te da će se nastaviti s implementacijom primjenjene tehnologije i za ostale potrebe. Konkretno, vrijeme testa po postrojenju je iznosilo od 2-4 minuta, ovisno o performansama servera i količini podataka koje klijent ima za razne izvještaje. Na slici 3.1 je prikazan izvještaj koji Playwright generira nakon završenog testiranja.



Q		All (12)	Passed (12)	Failed (0)	Flaky (0)	Skipped (0)
Project: chromium 07/03/2024, 13:26:00, Total time: 7.9m						
ACCStres.test.ts						
✓	ACC NAM					1.8m
	ACCStres.test.ts:30					
✓	ACC NLNG					1.6m
	ACCStres.test.ts:30					
✓	ACC Norske					1.7m
	ACCStres.test.ts:30					
✓	ACC Pernis					1.5m
	ACCStres.test.ts:30					
✓	ACC Projects Game EuSA					2.0m
	ACCStres.test.ts:30					
✓	ACC Qatar					2.6m
	ACCStres.test.ts:30					
✓	ACC Rheinland					2.3m
	ACCStres.test.ts:30					
✓	ACC SDO					2.1m
	ACCStres.test.ts:30					
✓	ACC SEPCIN					3.2m
	ACCStres.test.ts:30					
✓	ACC SUKEP					6.1m
	ACCStres.test.ts:30					
✓	ACC Trading Supply EA					2.0m
	ACCStres.test.ts:30					
✓	ACC Projects					1.2m
	ACCStres.test.ts:30					

Slika 3.1: Izvještaj nakon završenog automatskog testiranja

Niti jedan test nije pokazivao znakove problema, ako ih zaista nije bilo (*false-negative* testovi). Svi uočeni problemi nakon isporuke nove verzije su bili riješeni u rekordnom roku, a tome je pridonjelo vrlo kratko vrijeme do otkrivanja problema.

Kao dodatan benefit je primjećeno da se sada tester i mogu više fokusirati na specifične rubne slučajeve koje je potrebno verificirati te na taj način dodatno

smanjiti broj prijavljenih nedostataka od strane korisnika te posljedično povisiti kvalitetu isporučenog proizvoda.

Poglavlje 4

Zaključak

U ovom radu prikazan je postupak izrade

Literatura

- [1] S. Quadri and S. U. Farooq, “Software testing—goals, principles, and limitations,” *International Journal of Computer Applications*, vol. 6, no. 9, p. 1, 2010.
- [2] A. Mundra, S. Misra, and C. A. Dhawale, “Practical scrum-scrum team: Way to produce successful and quality software,” in *2013 13th International Conference on Computational Science and Its Applications*, pp. 119–123, IEEE, 2013.
- [3] Microsoft Inc., “Playwright homepage.” <https://playwright.dev/>. (3.4.2022.).
- [4] B. J. Sauser, R. R. Reilly, and A. J. Shenhar, “Why projects fail? how contingency theory can provide new insights—a comparative analysis of nasa’s mars climate orbiter loss,” *International Journal of Project Management*, vol. 27, no. 7, pp. 665–679, 2009.
- [5] T. Linz, *Testing in scrum: A guide for software quality assurance in the agile world*. Rocky Nook, Inc., 2014.
- [6] R. Löffler, B. Güldali, and S. Geisen, “Towards model-based acceptance testing for scrum,” *Softwaretechnik-Trends, GI*, 2010.

Popis slika

1.1	Proces testiranja u Scrum metodologiji rada	3
3.1	Izvještaj nakon završenog automatskog tesiranja	8

Dodatak A

Programski kod ...