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 zadatcima 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 ***ldkjalsdkjalskdjlskdjl*** 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)	$°C$
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
μSD, 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

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

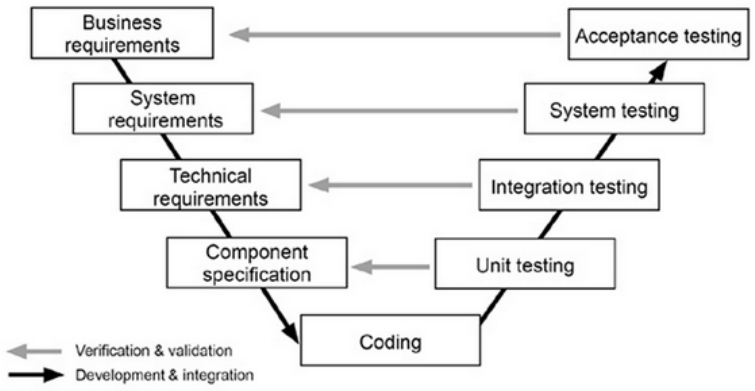
Struktura ovoga rada podjeljena je u logičke cjeline. Nakon uvoda i objašnjavanja rada, u poglavlju

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

Poglavlje 1

Uvod u testiranje programskog rješenje i osiguranje kvalitete

Testianje softwareskog proizvoda se provodi u cilju osiguravanja kvalitete samog proizvoda. Svaka faza razvoja ima svoje specifične zahtjeve i načine testiranja, a njihov pregled je dan na slici 1.1. Pojedine funkcije u kodu se pokrivaju unit testovima koji se brinu da pojedine komponente rade ono što su namjenjene na najnižoj razini. To su testovi koji se u pravilu izvršavaju vrlo često prilikom pisanja koda kako bi se osiguralo da promjena neke funkcije ili komponente nije poremetila njen rezultat.



Slika 1.1: Proces testiranja u Scrum metodologiji rada

1.1 Proces testiranja

Svaki proces testiranja započinje sa izradom plana testiranja unutar kojeg se definira šte se testira na koji način te koji su uvjeti da se zadani test smatra

uspješnim (*engl. Acceptance critera*).

Tokom samog razvijanja nove funkcionalnosti u softwareskom prozvodu 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 namjenjen 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 prihvačanje 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

Cypress

Cypress se reklamira kao moderan alat za testiranje fronend aplikacija nove generacije [3]. Često se uspoređuje s drugim popularnim alatom - Seleniumom. Međutim, Cypress i Selenium se u mnogičemu razlikuju, a glavne razlike se tiču samog pristupa testiranju i samim time drukčije arhitekture što omogućava Cypressu kompetitivne prednosti. Više o tome u nastavku.

Cypress omogućava pisanje i izvršavanje raznih tipova testova kao što su:

- testovi na krajnjim točkama (*end-to-end test, e2e test*)
- Integracijski testovi
- Unit testovi

U kratko, Cypress može testirati sve što se izvršava u browseru.

2.1 Opis i pregled paketa

Glavne značajke Cypress alata su:

Putovanje kroz vrijeme (*Time Travel*)

Cypress sorema snimke stanja kako prolzi kroz testove kako bi kasnije mogli pregledati što se točno događalo tokom izvršavanja. To nam omogućava da točno vidimo zašto neki test nije prošao i u kojem je stanju aplikacija bila u tom trenutku.

2.2 Instalacija

2.3 Osnovni test

2.4 Kontinuirana integracija - CI

Poglavlje 3

playwright

Nešto više o Playwrightu

3.1 Opis i pregled paketa

3.2 Instalacija

3.3 Osnovni test

3.4 Kontinuirana integracija - CI

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] Cypress, "Cypress documentation page." `https://docs.cypress.io/`. (3.4.2022.).

[4] Microsoft Inc., "Playwright homepage." `https://playwright.dev/`. (3.4.2022.).

[5] 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.

[6] T. Linz, *Testing in scrum: A guide for software quality assurance in the agile world*. Rocky Nook, Inc., 2014.

[7] R. Löffler, B. Güldali, and S. Geisen, "Towards model-based acceptance testing for scrum," *Softwaretechnik-Trends, GI*, 2010.

Popis slika

Dodatak A

Programski kod ...