

*Jesenji semestar, 2023/24*

*PREDMET: SE321 OBEZBEĐENJE KVALITETA, TESTIRANJE I ODRŽAVANJE SOFTVERA*

Projektni zadatak – Finansijska aplikacija

Ime i prezime: **Jovan Pavkovic**

Broj indeksa: **4442**

Sadržaj

[1. Uvod 4](#_Toc94570297)

[2. Opis sistema za testiranje 4](#_Toc94570298)

[3. Zahtevi 4](#_Toc94570299)

[3.1 Funkcionalni zahtevi 4](#_Toc94570300)

[3.1.1 Pristup sistemu (Login) 4](#_Toc94570301)

[3.1.2 Evidencija troškova i prihoda 5](#_Toc94570302)

[3.1.3 Pretraživanje prometa 5](#_Toc94570303)

[3.1.4 Evidencija izvoda 5](#_Toc94570304)

[3.1.5 Generisanje izveštaja 5](#_Toc94570305)

[3.1.6 Upravljanje banki 6](#_Toc94570306)

[3.1.7 Upravljanje računa 6](#_Toc94570307)

[3.1.9 Upravljanje prometa 6](#_Toc94570308)

[3.2 Nefunkcionalni zahtevi 7](#_Toc94570309)

[3.2.1 Performanse 7](#_Toc94570310)

[3.2.2 Bezbednost 7](#_Toc94570311)

[3.2.3 Sigurnost 7](#_Toc94570312)

[3.2.4 Integritet 7](#_Toc94570313)

[3.2.5 Promenljivost 8](#_Toc94570314)

[3.2.6 Ponovna upotrebljivost 8](#_Toc94570315)

[4. Dijagram slučajeva korišćenja 8](#_Toc94570316)

[5. Funkcionalnosti koje će biti testirane 9](#_Toc94570317)

[6. Funkcionalnosti koje neće biti testirane 9](#_Toc94570318)

[7. Strategija testiranja 10](#_Toc94570319)

[7.1 Tipovi testiranja koji će se sprovesti i u kojoj meri 10](#_Toc94570320)

[7.2 Kriterijumi 10](#_Toc94570321)

[7.3 Tablica povezanosti slučaji testiranja 10](#_Toc94570322)

[7.4 Osoblje uključeno u testiranje 11](#_Toc94570323)

[7.5 Vremenski raspored testiranja 11](#_Toc94570324)

[8. Primena 12 Casper pravila 11](#_Toc94570325)

[8.1 Estimacija veličine izvornog koda 11](#_Toc94570326)

[8.2 Estimacija dokumentacije 11](#_Toc94570327)

[8.3 Estimacija odudaranja korisničkih zahteva 12](#_Toc94570328)

[8.4 Estimacija broja slučajeva testiranja 12](#_Toc94570329)

[8.5 Estimacija potencijalnog broja gešaka 12](#_Toc94570330)

[8.6 Estimacija efikasnosti otklanjanja greške 12](#_Toc94570331)

[8.7 Estimacija efikasnosti organizovanog otklanjanja grešaka 12](#_Toc94570332)

[8.8 Estimacija efikasnosti otklanjanja grešaka nakon puštanja softvera u rad 12](#_Toc94570333)

[8.9 Estimacija trajanja realizacije projekta 12](#_Toc94570334)

[8.10 Estimacija potrebnih ljudi za realizaciju projekta 12](#_Toc94570335)

[8.11 Estimacija ljudi potrebnih za održavanje softvera 12](#_Toc94570336)

[8.12 Estimacija ukupnih napora u realizaciji softverskog projekta 12](#_Toc94570337)

[9. Realizacija testiranja softverskog sistema 13](#_Toc94570338)

[9.1 Primena McCabe-ova siklomatske slozenosti 13](#_Toc94570339)

[9.1.2 Generisanje izveštaja 17](#_Toc94570340)

[9.2 Dokumentovanje koda 18](#_Toc94570341)

[10. Junit testiranje 18](#_Toc94570342)

[10.1 UserServiceImplTest 19](#_Toc94570343)

[10.2 CategoryServiceImplTest 20](#_Toc94570344)

[10.3 TrosakServiceImplTest 21](#_Toc94570345)

[11. Metode testiranja 22](#_Toc94570346)

[11.1 Metoda crne kutije – Black Box 22](#_Toc94570347)

[11.2 Metoda bele kutije – White Box 22](#_Toc94570348)

[11.2.1 Pokrivenost naredbi 23](#_Toc94570349)

[12. Integraciono testiranje 23](#_Toc94570350)

[13. Održavanje, dalji razvoj I kontinualni process poboljšavanja softvera 24](#_Toc94570351)

[13.1 Model estimacije troškova prilikom održavanja odabrane aplikacije 24](#_Toc94570352)

[13.2 Pristup za postizanje višekratne upotrebljivosti 25](#_Toc94570353)

[13.2.1 Primer rejuzabilnosti klasa primenom nasleđivanja: 25](#_Toc94570354)

[13.2.2 Decorator Design pattern: 27](#_Toc94570355)

[14. Zaključak 27](#_Toc94570356)

[15. Literatura 27](#_Toc94570357)

# Uvod

Testiranje softvera je neophodna funkcija kako bi se otkrile greške koje su se javile u svim fazama razvoja softvera koje se mogu ispraviti nakon otkrivanja. Testiranje obuhvata različite vrste testova koji se izvršavaju kako bi se softverski proizvod osigurao tako da on neće imati funkcionalne i nefunkcionalne nedostatke. Ukoliko program ne izvršava funkciju za koju je razvijen, postaje neefikasan, i on će biti zamenjen. Softversko testiranje danas predstavlja aktvinost koja obuhvata kompletan proces razvoja i održavanja i kao takva čini veoma važan deo cele konstrukcije softvera.

# Opis sistema za testiranje

Firma se bavi elektronskim učenjem kao i proizvodnjom softvera. Trenutni finansijski sistem koji se koristi je nepotpun i fali mu dosta funkcionalnosti kako bi sistem ispunjavao da bude komplentan finansijski sistem. Postojeći sistem za finansije je star, što dovodi do dosta depricated funkcionalnosti, tаkođe pisano je u zastareloj tehnologiji.

Zbog ovih razloga našafirma je razvila potpuno nov sistem koji omogućava direktoru da vrši finansije firme bez problema. Direktor ima rolu super admina što mu omogućava da upravlja korisnicima, evidentiranje troškova i prihoda, generisanje izveštaja, upravljanje bankama i računima. Sistem je dostupan online na serveru, ali njoj pristupa direktor sa njegovog računara koji se nalazi na prostoru firme. Razlika između super admina i drugih korisnika je u tome što super admin za razliku od ostalih, ima mogućnost upravljanja korisnicima.

# Zahtevi

## 3.1 Funkcionalni zahtevi

### 3.1.1 Pristup sistemu (Login)

REQ-1: Svaki korisnik mora da se preko svojih kredencijala uloguje na sistem.

REQ-2: Direktoru treba omogućiti pregled svih korisnika u sistemu.

REQ-3: Direktoru treba omogućiti unos podataka koji se traže za dodavanje novog korisnika.

Direktor unosi korisničko ime, lozinku i rolu.

REQ-4: Direktoru treba omogućiti izmenu podataka koji se traže pri izmeni korisnika.

Direktor unosi korisničko ime, lozinku i rolu.

REQ-5: Direktoru treba omogućiti deaktiviranje korisnika.

### 3.1.2 Evidencija troškova i prihoda

REQ-6: Direktoru treba omogućiti pregled svih troškova i prihoda u firmi.

REQ-7: Direktoru treba omogućiti unos podataka koji se traže za dodavanje novog prihoda i troška.

Direktor unosi naziv pihoda ili troška, cenu i likvidivnost.

REQ-8: Direktoru treba omogućiti izmenu podataka koji se traže pri izmeni prihoda ili troška.

Direktor unosi naziv pihoda ili troška, cenu i likvidivnost.

REQ-9: Direktoru treba omogućiti brisanje svih prihoda i troškova.

### 3.1.3 Pretraživanje prometa

REQ-10: Direktoru treba omogućiti pretraživanje svih prometa po određenim atributima.

Dirketor može da pretraži promet po uplatiocu/primaocu, broj racuna, datum izvoda, kategorije, napomeni, iznosu, stanju, broju računa.

### 3.1.4 Evidencija izvoda

REQ-12: Direktoru treba omogućiti prikaz svih izvoda.

REQ-13: Direktoru treba omogućiti unos podataka koji se traže za dodavanje novog izvoda. Direktor unosi broj računa, broj izvoda, datum izvoda.

REQ-14: Direktoru treba omogućiti izmenu svih podataka koji se traže pri izmeni izvoda ako on nije proknjižen. Direktor unosi broj računa, broj izvoda, datum izvoda.

REQ-15: Direktoru treba omogućiti brisanje svih izvoda iz sistema ako izvod nije proknjižen.

### 3.1.5 Generisanje izveštaja

REQ-16: Direktoru treba omogućiti pregled svih izveštaja koje je moguće generisati.

REQ-17: Direktoru treba omogućiti generisanje izveštaja koje izabere.

### 3.1.6 Upravljanje banki

REQ-18: Direktoru treba omogućiti pregled svih banki.

REQ-19: Direktoru treba omogućiti unos podataka koji se traže prilikom dodavanja nove banke. Direktor treba da unese naziv banke i adresu.

REQ-20: Direktoru treba omogućiti izmenu svih podataka koji se traže prilikom izmene banke. Direktor treba da unese naziv banke i adresu.

REQ-21: Direktoru treba omogućiti brisanje banki.

### 3.1.7 Upravljanje računa

REQ-22: Direktoru treba omogućiti prikaz svih računa u sistemu.

REQ-23: Direktoru treba omogućiti unos podataka koji se traže prilikom dodavanja novog računa. Direktor treba da unese broj računa, sumu, naziv računa, konto i stanje.

REQ-24: Direktoru treba omogućiti izmenu svih podataka koji se traže prilikom izmene računa. Direktor treba da unese broj računa, sumu, naziv računa, konto i stanje.

REQ-25: Direktoru treba omogućiti brisanje svih računa.

### 3.1.9 Upravljanje prometa

REQ-26: Direktoru treba omogućiti prikaz svih prometa u sistemu za izvod koji je izabrao.

REQ-27: Direktoru treba omogućiti unos podataka koji se traže prilikom dodavanja novog prometa. Direktor treba da unese broj računa, kategoriju, uplata ili plaćanje, konto, uplatilac primalac.

REQ-28: Direktoru treba omogućiti izmenu svih podataka koji se traže prilikom izmene prometa. Direktor treba da unese broj računa, sumu, naziv računa, konto i stanje.

REQ-29: Direktoru treba omogućiti brisanje svih prometa.

## 3.2 Nefunkcionalni zahtevi

### 3.2.1 Performanse

PER-1: Vreme potrebno da se korisniku vrati lista svih podataka neće trajati duže od 2.0 sekunde, ako je internet veza stabilna.

PER-2: Čekanje od klika za generisanje izveštaja do početka preuzimanja fajla ne sme da bude više od 5s, ako je internet veza stabilna I u zavisnosti od veličine fajla.

### 3.2.2 Bezbednost

SAF-1: Privatne informacije o korisnicima ne smeju biti prikazane javno, lozinke korisnika uvek moraju biti enkriptovane I ne smeju da se prikazuju.

SAF-2: Ukoliko je korisnik neaktivan više od 10 minuta, sistem će ga automatski izlogovati I naterati da se uloguje ponovo.

### 3.2.3 Sigurnost

SEC-1: Sistem će zaključati korisnički nalog nakon pet uzastopna neuspešna pokušaja prijave

u roku od 5 minuta.

SEC-2: Korisnik sa deaktiviranim nalogom neće smeti da koristi taj nalog, što znači da neće moći da pristupi sistemu.

SEC-3: Korisnici će svoje lozinke dobijati isključivo od strane studentske službe.

### 3.2.4 Integritet

INT-1: Sistem vrši sigurnosne kopije podataka prilikom generisanja svakog izveštaja.

INT-2: Sistem se štiti od neovlašćenog dodavanja, brisanja ili modifikacije podataka.

INT-3: Nakon izvršenja sigurnosne kopije podataka, sistem će potvrditi kopiju rezervne kopije

u odnosu na original i prijaviti sve razlike.

INT-4: Svako čuvanje novog stanja će raditi duplu proveru, što će povećati sigurnost pri čuvanju podatka.

### 3.2.5 Promenljivost

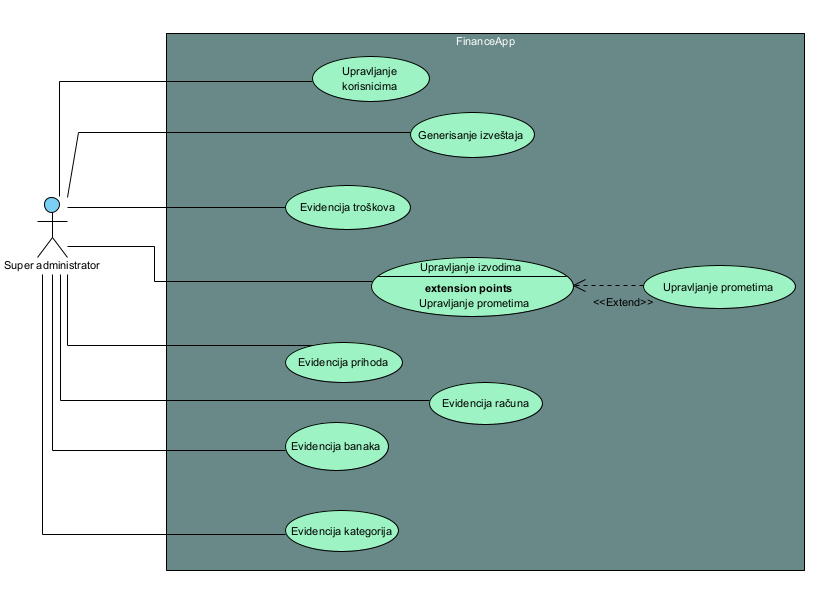
MOD-1: Provera validnosti biće znatno umanjena zbog toga što će za svaki deo koda biznis logike biti pisan odgovarajući test.

### 3.2.6 Ponovna upotrebljivost

REU-1: Aplikacija će koristiti komponente kao što su tabele, paginacije, status barove, itd. koje mogu da se ponovo koriste u aplikaciji.

REU-2: Kod će biti pisan uz praćenje OOp pristupa razvoja što će olakšati reuzibilnost.

# Dijagram slučajeva korišćenja



Slika 1. Dijagram slučajeva korišćenja

# Funkcionalnosti koje će biti testirane

1. Baza podataka sa svim uredjajima I njihovim podacima
2. Funkcionalnost banke
3. Funkcionalnost izvoda
4. Funkcionalnost prometa
5. Funkcionalnost korisnika
6. Funkcionalnost troškova
7. Funkcionalnost prihoda
8. Funkcionalnost kategorija
9. Integracija sa izveštajima

|  |  |  |  |
| --- | --- | --- | --- |
| Funkcionalnost | Izdanje 1.0 | Izdanje 1.1 | Izdanje 1.2 |
| FE-1 | Potpuno implementirano | / | / |
| FE-2 | Nije implementirano | Potpuno implementirano | / |
| FE-3 | Nije implementirano | Potpuno implementirano | / |
| FE-4 | Nije implementirano | Potpuno implementirano | / |
| FE-5 | Nije implementirano | Potpuno implementirano | / |
| FE-6 | Nije implementirano | Potpuno implementirano | / |
| FE-7 | Nije implementirano | Potpuno implementirano | / |
| FE-8 | Nije implementirano | Potpuno implementirano | / |
| FE-9 | Nije implementirano | Nije implementirano | Potpuno implementirano |

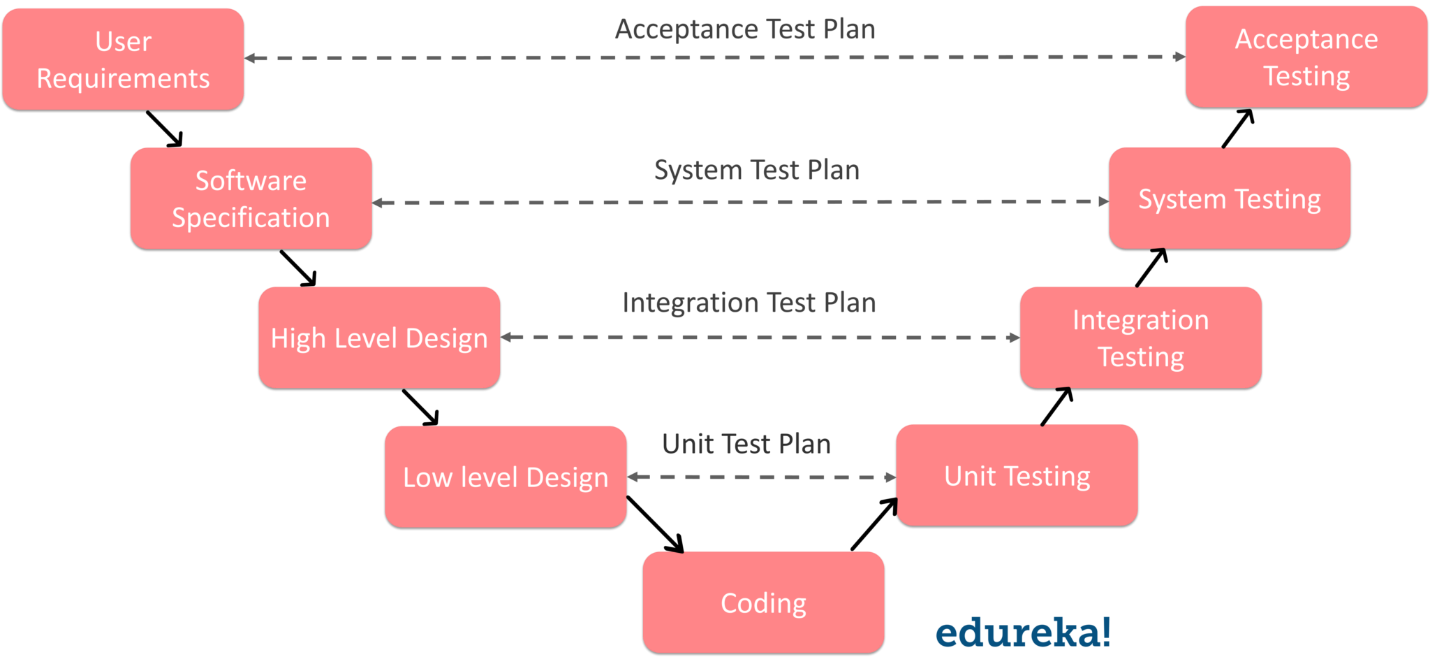
# Funkcionalnosti koje neće biti testirane

Funkcionalnosti koje su ostavljene za sledeće izdanje će iterativno biti testirane u sledećem ciklusu.

# Strategija testiranja

## Tipovi testiranja koji će se sprovesti i u kojoj meri

V-model predstavlja modifikaciju modela vodopada tako da ona demnostrira odnos aktivnosti testiranja sa svim fazama analize i dizajna. Naziv dobija prema V izgledu, pri čemu kodiranje predstavlja ishodište V-modela, gde su analiza i dizajn na levoj, a testiranje i održavanje na desnoj strani.



Slika 2.V-model

## Kriterijumi

* Broj grešaka kategorije 1=0
* Broj grešaka kategorije 3=0
* Broj grešaka kategorije 2 manje od 5 po modulu;
* Broj uspešno obavljenih testova mora biti 99 %;
* Svi planirani testovi moraju biti izvršeni

## Tablica povezanosti slučaji testiranja

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Broj zahteva** | **Test 1-login** | **Test 2-Banke** | **Test 3-Izvodi** | **Test 4-Prometi** | **Test 5-Prihodi** | **Test 6- Troškovi** | **Test 7-Kategorije** | **Test 8-Izveštaji** | **Test 9-Baza podataka** |
| **REQ\_1** | **X** |  |  |  |  |  |  |  | **X** |
| **REQ\_2** | **X** | **X** |  |  |  |  |  |  | **X** |
| **REQ\_3** | **X** |  | **X** |  |  |  |  |  | **X** |
| **REQ-4** | **X** |  | **X** | **X** |  |  |  |  | **X** |
| **REQ-5** | **X** |  |  |  | **X** |  |  |  | **X** |
| **REQ-6** | **X** |  |  |  |  | **X** |  |  | **X** |
| **REQ-7** | **X** |  |  |  |  |  | **X** |  | **X** |
| **REQ-8** | **X** |  |  |  |  |  |  | **X** | **X** |

## Osoblje uključeno u testiranje

|  |  |  |
| --- | --- | --- |
| Uloga | Korisnik | Testira |
| Tester 1 | / | * Upravljanje korisnicima * Evidencija troškova * Evidencija prihoda |
| Tester 2 | / | * Evidencija kategorija * Evidencija banaka * Evidencija računa |
| Tester 3 | / | * Upravljanje izvodima * Upravljanje prometima |
| Tester 4 | / | * Generisanje izveštaja |

## Vremenski raspored testiranja

|  |  |  |
| --- | --- | --- |
| Aktivnost | Planirani rok | Krajni rok |
| Poručivanje sistema | 07.01.2024. | 14.01.2024. |
| Pisanje test slučaja | 26.01.2024. | 31.01.2024. |
| Izvršavanje test slučajeva | 05.02.2024. | 30.02.2024. |
| Isporuka | 01.03.2024. | 03.03.2024. |

# Primena 12 Casper pravila

## Estimacija veličine izvornog koda

AFP = 5\*4 + 2\*5 + 3\*4 + 4\*10 + 2\*7 = 96

L= 96 x 53= 5088 LOC

## Estimacija dokumentacije

Dokumentacija= FP 1.15 stranica

Dokumentacija= 1601.15 = 342 stranice

## Estimacija odudaranja korisničkih zahteva

Troškovi odudaranja od korisničkih zahteva će biti na mesečnom nivou u proseku 3% vrednosti od celokupno planiranog budžeta.

## Estimacija broja slučajeva testiranja

FP(broj podešenih funkcionalnih tačaka)=160

Broj test slučajeva= FP1.2

Broj test slučajeva=1601.2= 441 test slučajeva

## Estimacija potencijalnog broja gešaka

Potencijalni broj grešaka=FP1.25

Potencijalni broj grešaka= 1601.25= 569 mogućih grešaka

## Estimacija efikasnosti otklanjanja greške

U svakoj fazi testiranja broj pronađenih I otklonjenih grešaka je u proseku 20.

## Estimacija efikasnosti organizovanog otklanjanja grešaka

Inspekcija dizajna će pronaći I otkloniti u proseku 80% grešaka u sistemu, ovaj pristup će biti skuplji od pretnodnog u koraku 6, ali će biti efikasniji.

## Estimacija efikasnosti otklanjanja grešaka nakon puštanja softvera u rad

Programeri zaduženi za održavanje softvera mogu ispraviti određen broj grešaka koji zavisi od TMM i CMM nivoa.

## Estimacija trajanja realizacije projekta

FP(broj podešenih funkcionalnih tačaka)=160 DM=FP0.4 [KM] kalendarskih meseciDM= FP0.4=1600.4= 7.5 [KM] kalendarskih meseci

## 8.10 Estimacija potrebnih ljudi za realizaciju projekta

FP(broj podešenih funkcionalnih tačaka)=160

Prosečna produktivnost projektanata=150

Broj projektanata= FP/Prosečna produktivnost projektanata Broj projektanata=160 /= 2 projektanta

## 8.11 Estimacija ljudi potrebnih za održavanje softvera

FP(broj podešenih funkcionalnih tačaka)=160

Broj ljudi za održavanje= FP/ prosečna efikasnost održavanja

Broj ljudi za održavanje=FP/ 750= 160 /750 = 1 čovek

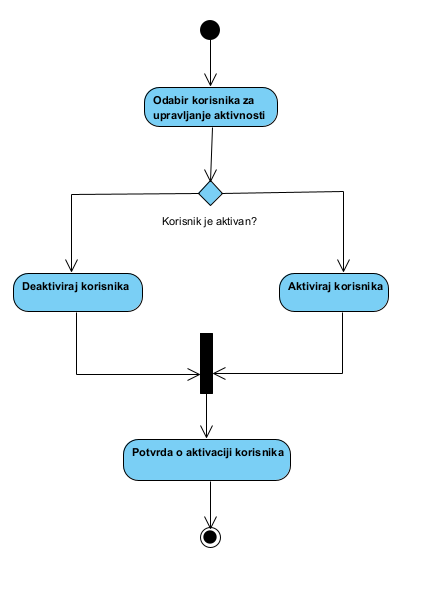
## 8.12 Estimacija ukupnih napora u realizaciji softverskog projekta

Ukupni Napor= Ukupno vreme \* broj ljudi= 7.5\*2 = 15 čovek-meseci

# Realizacija testiranja softverskog sistema

## Primena McCabe-ova siklomatske slozenosti

#### Upravljanje korisnicima (aktivacija/deaktivacija korisnika)



**v(G) = E - N + 2P**

• E - broj grana na grafu programa,

• N - broj čvorova na grafu,

• P - broj nepovezanih komponenti ili se može posmatrati kao broj izlaza iz programa

**v(G) = 6 - 5 + 2 = 3 test slučajeva**

#### *9.1.1.1 Test slučaj 1*

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Naslov | Provera upravljanje korisnicima | Rev 1 | Autor | Jovan Pavkovic | Datum | 16.01.2024. |
| Cilj | Provera ispravnosti sistema za upravljanje korisnika | | Reference |  | | |
| Test uslovi |  | | | Neophodno vreme za izvršenje test slucaja | | 3 min |

|  |  |
| --- | --- |
| Opis postavke za testiranje | |
|  | Postavka aplikacije je takva da korisnik dobije opciju za izvrsavanje u odnosu na stanje korisnika |
|  | Aplikacija je pokrenuta i to od super administratora koji bira opciju za upravljanje korisnika |
|  | Super administrator koristi računar za pristup sajtu i web stranici |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Definicija testa | | | Izvršenje testa | |
| Uslovi | Ulazni podaci | Očekivani rezultati | Aktuelni rezultati | Broj problema |
| Super administrator je pristupio  I odabrao opciju upravljanje korisnicima | / | Nakon odabira sistem otvara odredjene opcije | Sistem je odreagovao očekivano i odradio naredbu | 0 |

|  |  |
| --- | --- |
| Opis postuslova | |
|  | Korisniku se prikazuju odredjene opcije za upravljanje korisnika |

#### *9.1.1.2 Test slučaj 2*

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Naslov | Provera upravljanje aktivnosti korisnika | Rev 1 | Autor | Jovan Pavkovic | Datum | 16.01.2024. |
| Cilj | Provera ispravnosti sistema za upravljanje korisnika | | Reference |  | | |
| Test uslovi |  | | | Neophodno vreme za izvršenje test slucaja | | 3 min |

|  |  |
| --- | --- |
| Opis postavke za testiranje | |
|  | Postavka aplikacije je takva da super administrator dobije opciju za izvršavanje u odnosu na stanje korisnika |
|  | Aplikacija je pokrenuta i to od super administratora koji bira opciju za upravljanje korisnika |
|  | Super administrator koristi računar za pristup sajtu i web stranici |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Definicija testa | | | Izvršenje testa | |
| Uslovi | Ulazni podaci | Očekivani rezultati | Aktuelni rezultati | Broj problema |
| Korisnik je pristupio stranici za prikaz korisnika, gde mu se prikazuju korisnici sa njihovim stanjima | / | Nakon odabira zeljene opcija izvrsava se naredba | Sistem je odreagovao očekivano i odradio naredbu | 0 |

|  |  |
| --- | --- |
| Opis postuslova | |
|  | Korisniku je prikazana poruka da je uspesno izvrsio promenu stanja aktivnosti |

#### *9.1.1.3 Test slučaj 3*

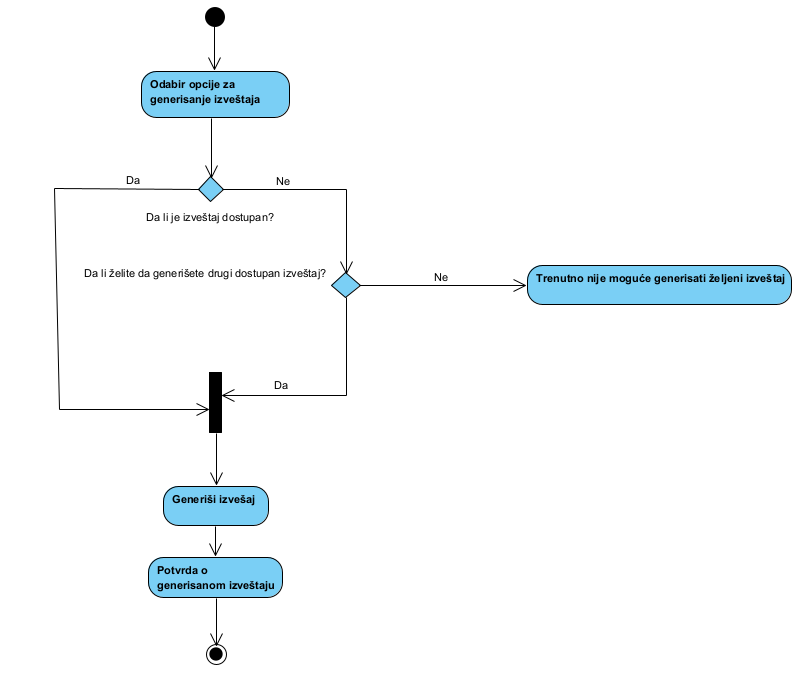
|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Naslov | Provera upravljanje korisnika | Rev 1 | Autor | Jovan Pavkovic | Datum | 16.01.2024. |
| Cilj | Provera ispravnosti sistema za upravljanje korisnicima | | Reference |  | | |
| Test uslovi |  | | | Neophodno vreme za izvršenje test slucaja | | 3 min |

|  |  |
| --- | --- |
| Opis postavke za testiranje | |
|  | Postavka aplikacije je takva da super administrator dobije opciju za izvršavanje u odnosu na stanje korisnika |
|  | Aplikacija je pokrenuta i to od super administratora koji bira opciju za upravljanje korisnika |
|  | Super administrator koristi računar za pristup sajtu i web stranici |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Definicija testa | | | Izvršenje testa | |
| Uslovi | Ulazni podaci | Očekivani rezultati | Aktuelni rezultati | Broj problema |
| Korisnik je pristupio stranici za prikaz korisnika, gde mu se prikazuju korisnici sa njihovim stanjima | / | Nakon odabira zeljene opcija izvrsava se naredba | Sistem je odreagovao očekivano i odradio naredbu | 0 |

|  |  |
| --- | --- |
| Opis postuslova | |
|  | Korisniku je prikazana poruka da je uspesno izvrsio promenu stanja aktivnosti |

## Generisanje izveštaja



**v(G) = E - N + 2P**

• E - broj grana na grafu programa,

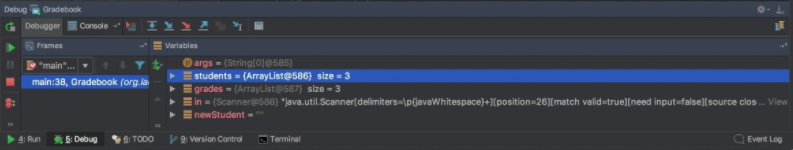
• N - broj čvorova na grafu,

• P - broj nepovezanih komponenti ili se može posmatrati kao broj izlaza iz programa

**v(G) = 8 - 6 + 2\*2 = 6 test slučajeva**

## Dokumentovanje koda

Kod čini ¼ faze u razvoju samog softvera, pored ostalih faza procesa razvoja softvera. On je podjednako važan kao i ostale faze, međutim najviše vidljivih problema koji mogu da se jave jeste u kodu. Jedna od velikih praksi koje se sada rade, jeste pregled, refaktorovanje i testiranje koda. Programeri koriste razne alate pri testiranju njihovog koda “Debagger”. Ti alati su kreirani radi otklanjanje grešaka u programu. Svi radni okviri za kucanje koda sadrže najmodernije Debaggere koji pomažu programerika u razvoju softvera.



# Junit testiranje

Java programski jezik u sebi ima ugrađen radni okvir (framework) pod nazivom Junit. On služi za testiranje pojedinjenih softverskih modula ili jedinica gde se ispituju klase, metode, programi ili podprogrami. To testiranje se izvršava izolovano od ostalog dela sistema pri čemu su te jedinice pisane od strane tima ili jednog programera

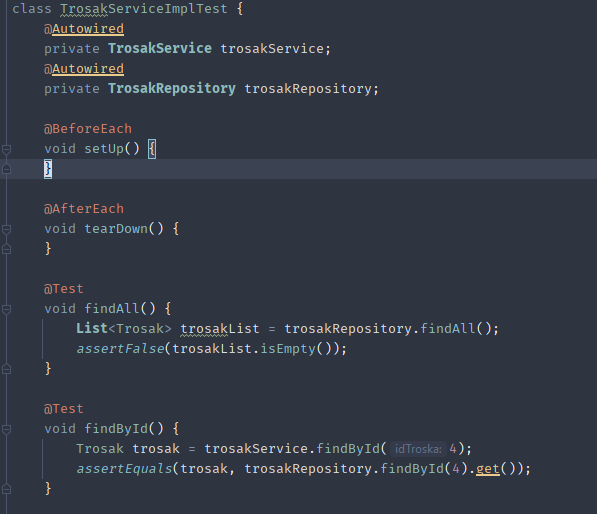
Prednosti Junita su te što je jednostavan za korišćenje, testovi se pišu lako bez ikakvog napora i lako se razumeju. Testiranje je ogranizovano po klasama, paketima ili modulima. Junit je integrisan uglavno sa svim modernim razvojnim okruženjima.

## UserServiceImplTest

## CategoryServiceImplTest



## TrosakServiceImplTest



# Metode testiranja

## 11.1 Metoda crne kutije – Black Box

Metoda crna kutije je metod softverskog testiranja u kojem unutrašnja struktura sistema koji se testira nije poznata testeru, tester tu klasu vidi kao crnu kutiju tj. ne zna kako je softver implementiran, nema uvid u kod. Black box testiranje predstavlja šta klasa radi sa gledališta korisnika te klase, ne posmatra se klasa iza interjesa klase.

Dozvoljeni format datuma je: dd-mm-yyyy gde dd pretstavlja dan, mm pretstavlja koji je mesec I yyyy pretstavlja koja je godina.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Naslov | Validnost datuma izvoda nekog računa | Rev 1 | Autor | Jovan Pavkovic | Datum | 16.01.2024. |
| Cilj | Provera ispravnosti sistema za validaciju pri unosu datuma od i datuma do kod napredne pretrage. | | Reference |  | | |
| Test uslovi |  | | | Neophodno vreme za izvršenje test slucaja | | 5 min |

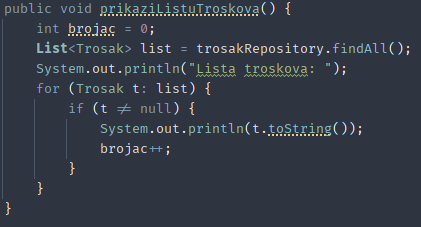
|  |  |
| --- | --- |
| Opis postavke za testiranje | |
|  | Sistem prihvata samo predefinisani odnosno dozvoljeni format broja koji je u sledećem formatu: dd-MM-yyyy |
|  | Aplikacija je pokrenuta |
|  | Korisnik je prisutan na stranici za naprednu pretragu gde unosi datume od i datume do po validnom formatu. |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Definicija testa | | | Izvršenje testa | |
| Uslovi | Ulazni podaci | Očekivani rezultati | Aktuelni rezultati | Broj problema |
| Korisnik je uneo podatke u polja za pretragu izvoda po datumu | / | Vraća se lista svih izvoda sa datumom izvoda u opsegu datuma od i datuma do po kome korisnik pretražuje | Sistem je vratio listu svih izvoda i prometa | 0 |

## 11.2 Metoda bele kutije – White Box

Metoda bele kutije je metod softverskog testiranja u kojem su testeri prisutni prilikom implementacije, što znači da imaju uvid u kod i znaju kako je softver implementiran.

### 11.2.1 Pokrivenost naredbi



Iz datog primera vidimo da je pokrivenost naredbe u metodi takva da je petlja pretraživala listu troškova sve dok nije null.

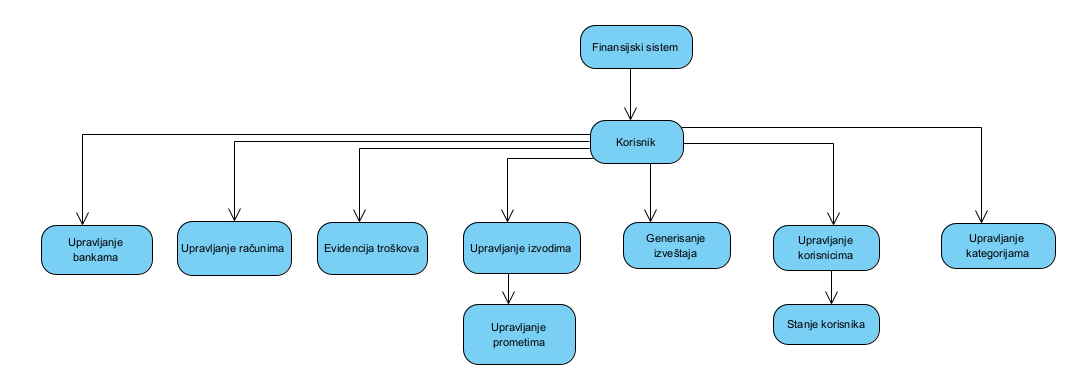
# 12. Integraciono testiranje

Integraciono testiranje se radi posle jediničnog testiranja (“Unit testing”).

Postoje dva osnovna prilaza da bi se izvršilo integraciono testiranje:

1. Top down tj. Odozgodo-nadole
2. Bottom up tj. Odozdo-nagore

Za naš primer mi ćemo koristiti tehniku odozgo-nadole tj. Top down tehniku.



# 13. Održavanje, dalji razvoj I kontinualni process poboljšavanja softvera

## 13.1 Model estimacije troškova prilikom održavanja odabrane aplikacije

***Vođa projekta*** - Radi na upravljanju svih aktivnosti na projektu, dobijaće izveštaje od strane softverskih inženjera i administratora sistema.

Predviđeno radno vreme na projektu je 80 radnih sati. Cena rada po satu iznosi 40 evra.

80 \* 40 = 3200 evra

***Administrator sistema*** – Podešavanje softvera i hardvera u serverskom delu sistema, unos novih korisnika, promena stanja svih korisnika, itd..

Predviđeno radno vreme na projektu je 200 radna sata. Cena rada po satu iznosi 30 evra.

200 \* 30 = 6000 evra

***Softverski inženjer*** ***1***– Biće zadužen za grafički korisnički interfejs, na kreiranju dokumentacije i na planiranju modula za integraciju za dostavu kao i front endu aplikacije.

Predviđeno radno vreme na projektu je 350 radnih sati. Cena rada po satu iznosi 50 evra.

350 \* 50 = 17500 evra

***Softverski inženjer 2*** – Biće zadužen za vezu starog sistema i novog koji se razvija, na kreiranju modela baze podataka i na planiranju modula e-recept, ako i back enda i baze podataka sa kojom backend komunicira

Predviđeno radno vreme na projektu je 350 radnih sati. Cena rada po satu iznosi 150 evra.

350 \* 150 = 52500 evra

***Tester*** – Angažovanje prilikom testiranja sistema, njegovih osnovih funkcija i novih modula. Testiraće i bazu podataka sa velikim brojem unetih lekova kao i grafički interfejs sistema.

Predviđeno radno vreme na projektu je 40 radnih sati. Cena rada po satu iznosi 20 evra.

40 \* 20 = 800 evra

Ostali resursi:

• PC-Server – 28000 evra

• PC 1 – 1200 evra

• PC 2 – 1200 evra

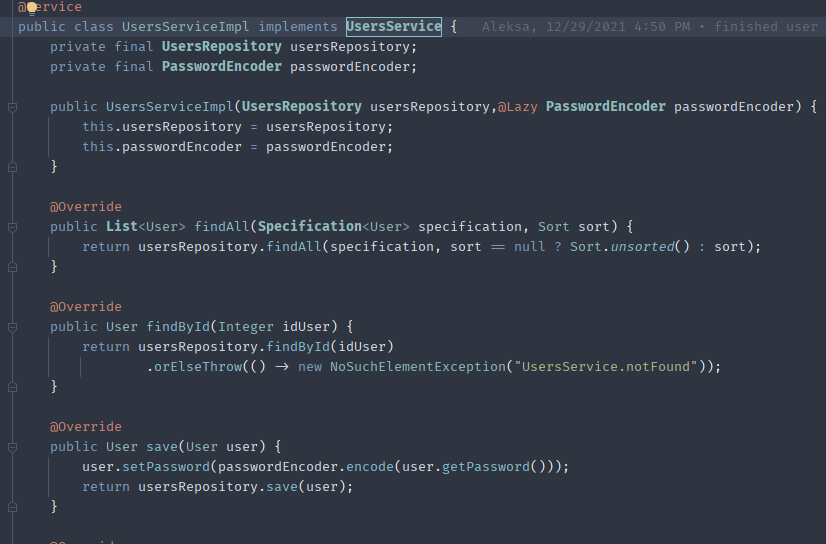
• PC 3 – 1200 evra

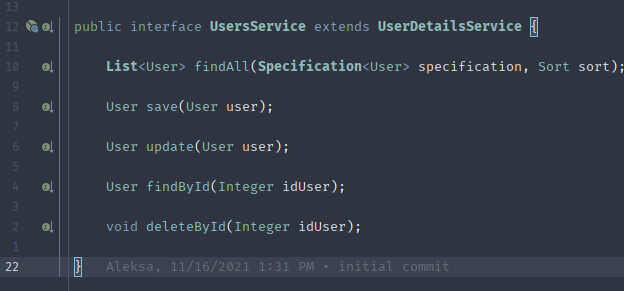
• PC 4 – 1200 evra

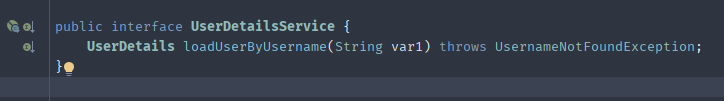
## 13.2 Pristup za postizanje višekratne upotrebljivosti

### 13.2.1 Primer rejuzabilnosti klasa primenom nasleđivanja:

Ponovna upotrebljivost (reusibility) je dobar način primene koda za lakše čitanje, organizaciju kao i za lakše pronalaženje greški i bugova. Kod koji može da se ponovo koristi u drugim klasama može da se postigne na više načina, a jedan od njih je nasleđivanje. Nasleđivanje je kada neka “Child” klasa nasleđuje svoju “Parent” klasu. To znači da će ti isti atributi I metode biti nasleđene u toj klasi, I moći će da ih koriste na njihovu korist. Klase koje mogu da se nasleđuju su abstract klase ili obične klase.

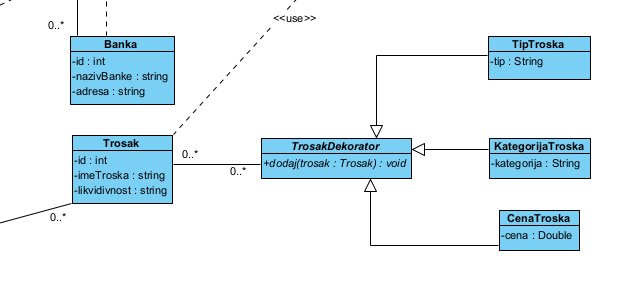






Generalizacija je tehnika koja se koristi da bi se na jednostavan način rešavale složene situacije. Podklase nasleđuju atribute i metode svojih klasa-roditelja (nad klasa) te se na taj način pojednostavljuje opis klasa sistema, jer su česti slučajevi kada više klasa imaju iste atribute i metode (operacije).

### 13.2.2 Decorator Design pattern:



Decorator šablon omogućava dodavanje nove funkcionalnoti kod postojećeg objekta bez menjanja njegove strukture. Ovaj šablon kreira Decorator klasu koji je omotoač originalne klase i obezbeđuje dodatnu funkcionalnost bez ikakvih promena klase troška. Decorator šablon obezbeđuje fleksibilnu alternativu korišćenju potklasa za proširenje funkcionalnosti.

# 14. Zaključak

Cilj svakog inženjera i organizacije je da imaju završen, funkcionalni projekat bez grešaka. Kako projekat zadovoljava zahteve korisnika kao i njegova očekivana tako će i njegovo zadovoljenje biti bolje. Zbog toga je potrebno obezbediti sistem visokog kvaliteta koji će garantovati njegovu pouzdanost. Testiranje softvera nudi dokaz da softver ispunjava sve poslovne zahteve klijenta.

# 15. Literatura

1. LAMS Nastavni materijali predmeta SE321- Obezbeđenje kvaliteta, testiranje i održavanje