**SE321 OBEZBEĐENJE KVALITETA, TESTIRANJE I ODRŽAVANJE**

**SOFTVERA**



**Jesenji semester** **2021/22**

**PROJEKTNI ZADATAK**

PROJEKAT- Sistem lanca apoteka

Student: Uros Milovanovic 4191

Asistent: Sara Nikolic

Sadrzaj projekta

[Uvod 3](#_Toc93443435)

[Specifikacija softverskog sistema koji se testira (fokus je na zahtevima softverskog sistema) 4](#_Toc93443436)

[Kratak opis poslovnog problema koji se rešava softverskim sistemom i njegova veza sa okruženjem 4](#_Toc93443437)

[Funkcionalni zahtevi softverskog sistema 4](#_Toc93443438)

[Nefunkcionalni zahtevi softverskog sistema 4](#_Toc93443439)

[Dijagram slučajeva korišćenja softverskog sistema 5](#_Toc93443440)

[Funkcionalnosti softverskog sistema koje će biti testirane 6](#_Toc93443441)

[Funkcionalnosti softveskog sistema koje neće biti testirane 7](#_Toc93443442)

[Strategija testiranja 7](#_Toc93443443)

[Tipovi testiranja koji će se sprovesti i u kojoj meri 7](#_Toc93443444)

[Kriterijum 7](#_Toc93443445)

[Tablica povezanosti zahtevi-slučajevi testiranja 8](#_Toc93443446)

[Osoblje uključeno u testiranje 8](#_Toc93443447)

[Vremenski raspored testiranja 8](#_Toc93443448)

[Primena 12 Casper pravila 8](#_Toc93443449)

[Realizacija testiranja softverskog sistema 10](#_Toc93443450)

[Pregled koda 10](#_Toc93443451)

[Primena McCabe-ova siklomatske slozenosti 10](#_Toc93443452)

[*Test slučaj 1* 11](#_Toc93443453)

[*Test slučaj 2* 12](#_Toc93443454)

[*Test slučaj 3* 13](#_Toc93443455)

[JUnit testiranje 15](#_Toc93443456)

[Testiranje metodom crne kutije – Black Box 18](#_Toc93443457)

[Ekvivalentno parcelisanje 18](#_Toc93443458)

[Testiranje metodom bele kutije – White Box 19](#_Toc93443459)

[Pokrivenost petlji 20](#_Toc93443460)

[Integraciono testiranje 20](#_Toc93443461)

[ODRŽAVANJE, DALJI RAZVOJ I KONTINUALNI PROCES POBOLJŠANJA SOFTVERA 21](#_Toc93443462)

[Model estimacije troškova prilikom održavanja odabrane aplikacije 21](#_Toc93443463)

[Pristup za postizanje višekratne upotrebljivosti 22](#_Toc93443464)

[ZAKLJUČAK 23](#_Toc93443465)

[LITERATURA 23](#_Toc93443466)

# Uvod

Testiranje softvera je neophodno kako bi se otkrile greske nacinjene u svim fazama razvoja softvera, koje se nakon otkrivanja mogu ispraviti. Testiranje softvera obuhvata različite vrste testiranja kako bi se osiguralo da softverski proizvod neće imati funkcionalne i nefunkcionalne nedostatke, a sve u cilju smanjenja ukupnih troškova razvoja softvera, poboljšanja njegovog kvaliteta i udobnosti korišćenja. Ako program ne obavlja funkciju za koju je razvijen, postaje neprofitabilan i postoji šansa da će ga potencijalni korisnik zameniti konkurentnim softverom.

Softversko testiranje danas predstavlja aktivnost koja obuhvata kompletan proces razvoja i održavanja i kao takva čini veoma važan deo cele konstrukcije softvera. Testiranje softvera nije aktivnost koja počinje samo nakon kompletiranja faze kodiranja, već aktivnost koja se izvodi zbog evaluacije kvaliteta proizvoda i njegovog poboljšanja, putem identifikovanja defekata i problema.

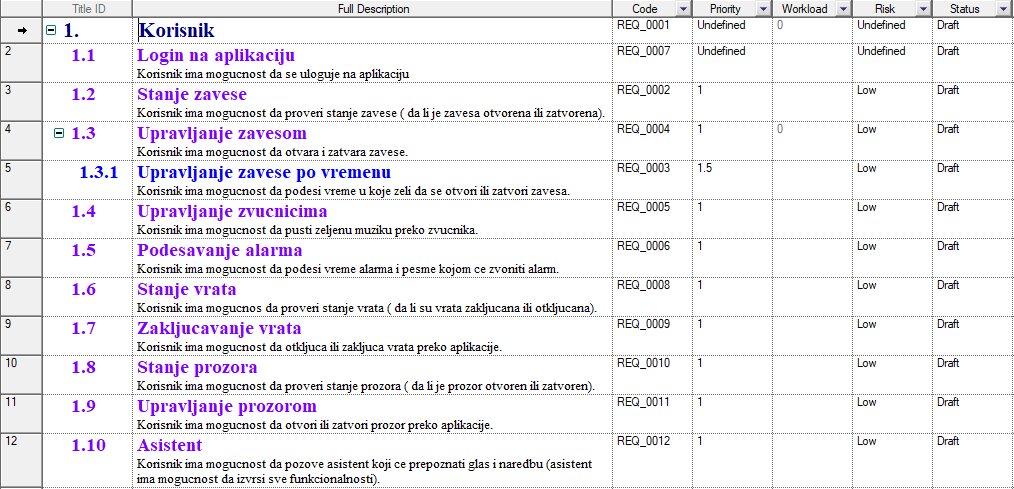
# Specifikacija softverskog sistema koji se testira (fokus je na zahtevima softverskog sistema)

## Kratak opis poslovnog problema koji se rešava softverskim sistemom i njegova veza sa okruženjem

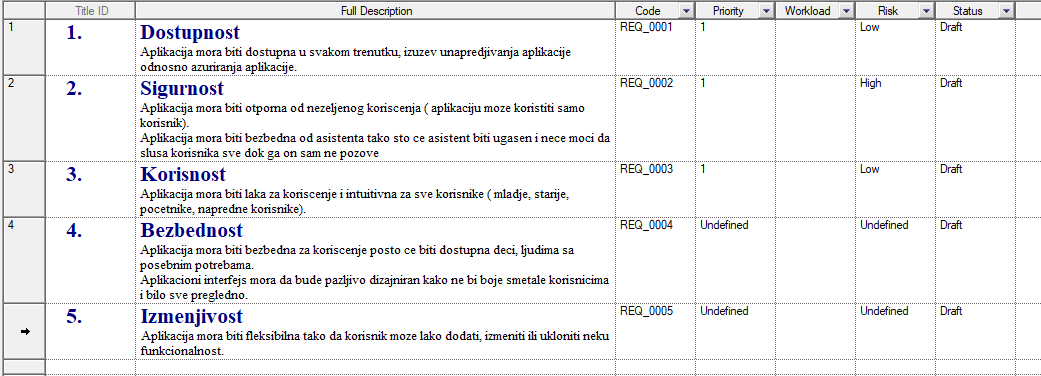
Ovo je nova era, era digitalizacije gde tehnologija radi za tebe. Kako tehnologija napreduje tako se doslo do ideje da se napravi rezim pametna kuca. Ova inovativnost je nastala prilikom prikupljanja informacija od ljudi sa invaliditetom koji imaju problem sa kretanjem I obavljanjem drugih fizickih aktivnosti, a nemaju pomoc staratelja ili kucne pomocnice. Takodje, ova ideja je inspirisala ljude sa velikim statusom da svoje kuce unaprede tehnologijom koja bi ona radila odredjene stvari umesto njih. Te je pametna kuca postala veoma aktuelna u drustvu kako zbog posebnih potreba, tako I zbog licnog uzitka.

Softver koji je razvijen od strane nas, ima funkcionalnosti koje ce korisniku kroz par klika ili izdavanje naredbe pomocnom asistentu olaksati upravljanje cele kuce (upravljanje prozorima, upravljanje vratima, upravljanje zavesama, podesavanje alarma, pustanje muzike I jos mnoge druge stvari). Softver je namenjen za siri opseg korisnika sto znaci da je toliko prost za korisceje da mogu da ga koriste deca, stariji, osobe sa invaliditetom, osobe sa vecim I manjim znanjem tehnologije I racunara.

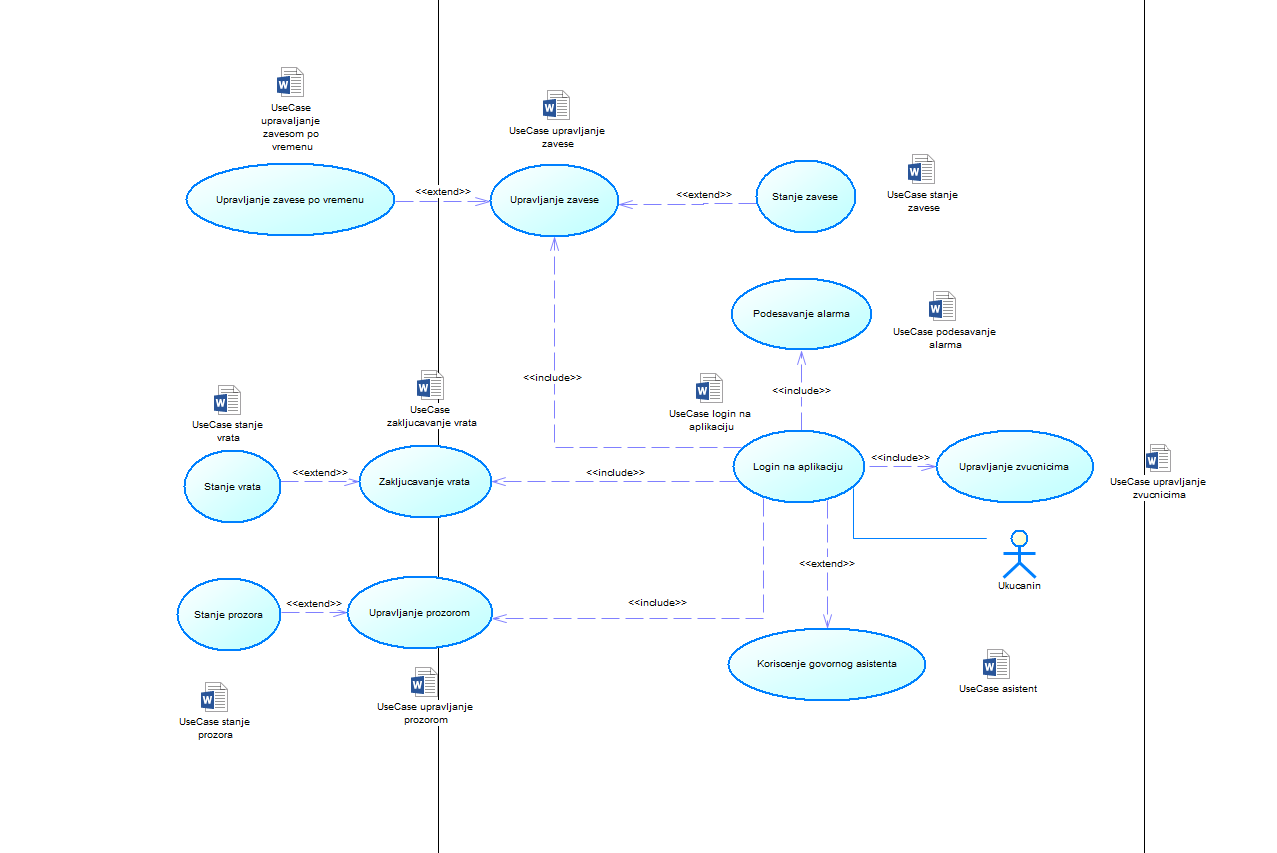
## Funkcionalni zahtevi softverskog sistema



## Nefunkcionalni zahtevi softverskog sistema



## Dijagram slučajeva korišćenja softverskog sistema



## Funkcionalnosti softverskog sistema koje će biti testirane

1. Baza podataka sa svim uredjajima I njihovim podacima
2. Funkcionalnost upravljanjem prozorima
3. Funkcionalnost upravljanjem zavesama
4. Funkcionalnost upravljanjem vratima
5. Funkcionalnost upravljanjem zvucnicima
6. Podesavanje alarma (vreme alarma I melodija)
7. Funkcionalnost aplikacija na fizickom uredjaju
8. Upravljanje pomocnim asistentom
9. Integracija sa eksternim sistemom Youtube

|  |  |  |  |
| --- | --- | --- | --- |
| 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 | Nije implementirano | Potpuno implementirano |
| FE-7 | Nije implementirano | Potpuno implementirano | / |
| FE-8 | Nije implementirano | Nije implementirano | Potpuno implementirano |
| FE-9 | Nije implementirano | Potpuno implementirano | / |

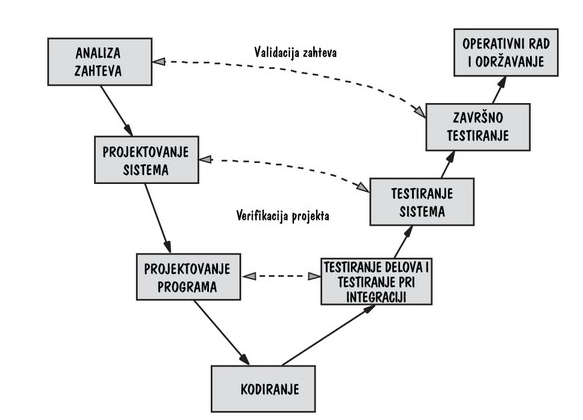
## Funkcionalnosti softveskog sistema koje neće biti testirane

Ostale funkcije koje su ostavljene za sledece izdanje ce biti testirane u sedecem ciklusu razvoja I testiranja.

# Strategija testiranja

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

V-model predstavlja modifikaciju modela vodopada, koja demonstrira odnos aktivnosti testiranja sa fazama analize i dizajna. Naziv je dobio prema V izgledu, pri čemu kodiranje predstavlja ishodište V-modela, gde su analiza i dizajn na levoj, a testiranje i održavanje na desnoj strani V izgleda.



### Kriterijum

* 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 zahtevi-slučajevi testiranja

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Broj zahteva** | **Test 1-login** | **Test 2-Zavesa** | **Test 3-Zvucnik** | **Test 4-Alarm** | **Test 5-Baza** |
| **REQ\_1** | **X** |  |  |  | **X** |
| **REQ\_2** | **X** | **X** |  | **X** | **X** |
| **REQ\_3** | **X** |  | **X** | **X** | **X** |

## Osoblje uključeno u testiranje

|  |  |  |
| --- | --- | --- |
| Uloga | Korisnik | Testira |
| Tester 1 | Milos Milosevic | * Upravljanje prozorom * Upravljanje vratima * Upravljanje zavesom |
| Tester 2 | Milica Tasic | * Podesavanje alarma * Upravljanje zvucnicima |
| Tester 3 | Nikola Pesic | * Upravljanje pomocnim asistentom |

## Vremenski raspored testiranja

|  |  |  |
| --- | --- | --- |
| Aktivnost | Planirani rok | Krajni rok |
| Porucivanje sistema | 05.01.2022. | 15.01.2022. |
| Pisanje test slucaja | 16.01.2022. | 26.01.2022. |
| Izvrsavanje test slucajeva | 30.01.2022. | 30.02.2022. |
| Isporuka | 01.03.2022. | 03.03.2022. |

## Primena 12 Casper pravila

1. **Estimacija veličine izvornog koda**

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

L= 96 x 53= 5088 LOC

1. **Estimacija dokumentacije**

Dokumentacija= FP 1.15 stranica

Dokumentacija= 1601.15 = 342 stranice

1. **Estimacija odudaranja korisničkih zahteva**

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

1. **Estimacija broja slučajeva testiranja**

FP(broj podešenih funkcionalnih tačaka)=160

Broj test slučajeva= FP1.2

Broj test slučajeva=1601.2= 441 test slučajeva

1. **Estimacija potencijalnog broja grešaka**

Potencijalni broj grešaka=FP1.25

Potencijalni broj grešaka= 1601.25= 569 mogućih grešaka

1. **Estimacija efikasnosti otklanjanja greške**

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

1. **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.

1. **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.

1. **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

**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

**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

**12. Estimacija ukupnih napora u realizaciji softverskog projekta**

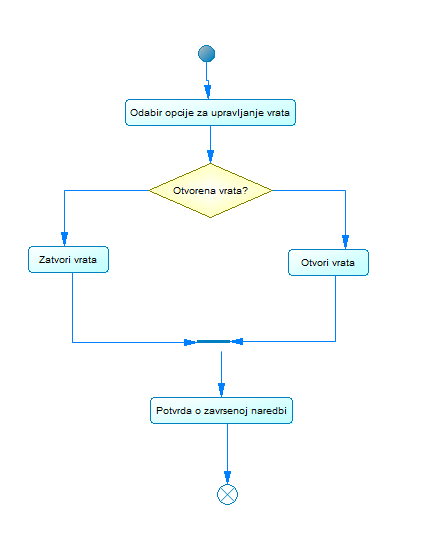
Ukupni Napor= Ukupno vreme \* broj ljudi= 7.5\*2 = 15 čovek-meseci

# Realizacija testiranja softverskog sistema

## Pregled koda

Pregled koda je u zadnje vreme postalo sve cesca praksa u razvojnim timovima. To je faza posla gde se predaje kod na proveru pre spajanja grana ili pustanja koda u produkciju. Najpoznatiji pregled koda je GitHub gde programeni postavljaju svoje projekte I svoje kodove sa opisom svakog komita I dobijaju predlog za izmenu koda gde mogu da private I odbiju promenu.

## Primena McCabe-ova siklomatske slozenosti



**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**

## Test slučaj 1

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Naslov | Provera upravljanje vratima | Rev 1 | Autor | Uros Milovanovic | Datum | 15.01.2022. |
| Cilj | Provera ispravnosti sistema za upravljanje vratima | | 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 vrata |
|  | Aplikacija je pokrenuta i to od korisnika koji bira opciju za upravljanje vrata |
|  | Korisnik koristi tablet za odabir naredbe |

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

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

## Test slučaj 2

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Naslov | Provera upravljanje vratima | Rev 1 | Autor | Uros Milovanovic | Datum | 15.01.2022. |
| Cilj | Provera ispravnosti sistema za upravljanje vratima | | 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 vrata |
|  | Aplikacija je pokrenuta i to od korisnika koji bira opciju za upravljanje vrata |
|  | Korisnik koristi tablet za odabir naredbe |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Definicija testa | | | Izvršenje testa | |
| Uslovi | Ulazni podaci | Očekivani rezultati | Aktuelni rezultati | Broj problema |
| Korisnik je pristupio uredjaju za upravljanje kuce I izabrao opciju za otvaranje vrata | / | 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 zeljenu naredbu |

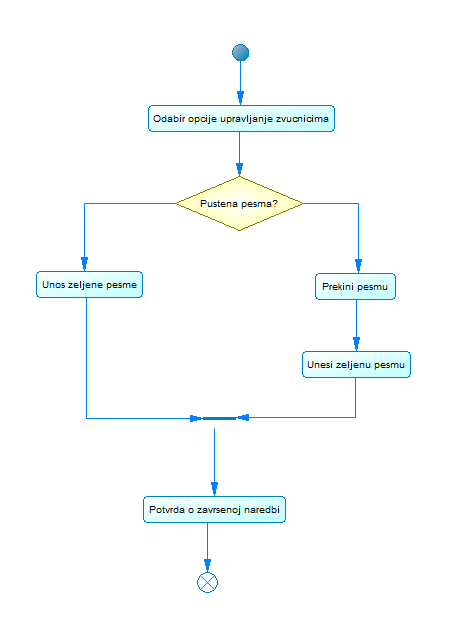
## Test slučaj 3

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Naslov | Provera upravljanje vratima | Rev 1 | Autor | Uros Milovanovic | Datum | 15.01.2022. |
| Cilj | Provera ispravnosti sistema za upravljanje vratima | | 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 vrata |
|  | Aplikacija je pokrenuta i to od korisnika koji bira opciju za upravljanje vrata |
|  | Korisnik koristi tablet za odabir naredbe |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Definicija testa | | | Izvršenje testa | |
| Uslovi | Ulazni podaci | Očekivani rezultati | Aktuelni rezultati | Broj problema |
| Korisnik je pristupio uredjaju za upravljanje kuce I izabrao opciju za zatvaranje vrata | / | 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 zeljenu naredbu |



**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) = 7 – 5 + 4 = 6 test slučajeva**

## JUnit testiranje

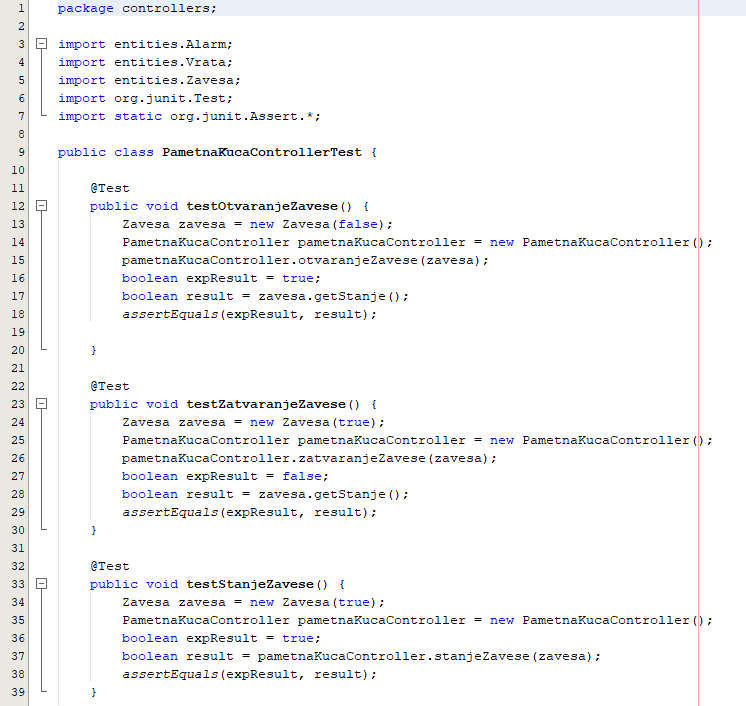
Testiranje modula proverava da li pojedinacne funkcije i komponente ispravno funkcionisu sa svim ocekivanim tipovima ulaza, u skladu sa dizajnom komponente. Jedinicno testiranje treba vrsiti u kontrolisanom okruzenju gde rezultati metoda mogu biti precizno proucen i biti im odredjena validnost.

Java programski jezik u sebi ima ugradjeno testiranje pod nazivom Junit. Junit je framework za testiranje u Javi I on se veoma lako generise. U ovom slucaju testiramo sve klase koje su generisane od strane naseg modela koji smo napravili u PowerDesigneru.

JUnit testovi koje smo uradili su za sledece funkcionalnosti sistema:

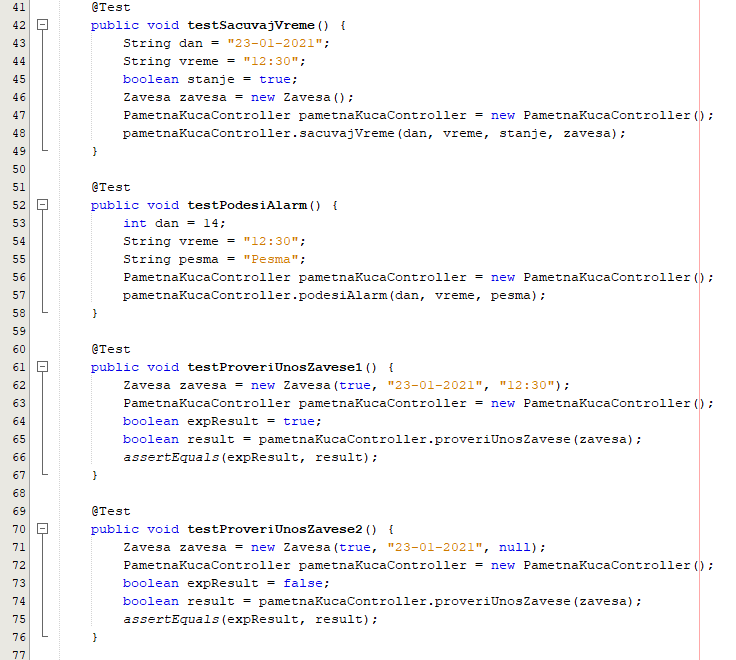
* Otvaranje zavese
* Zatvaranje zavese
* Stanje zavese
* Sacuvaj vreme
* Podesi alarm
* Proveri unos zavese1
* Proveri unos zavese2
* Obrisi alarm1
* Obrisi alarm2
* Stanje vrata

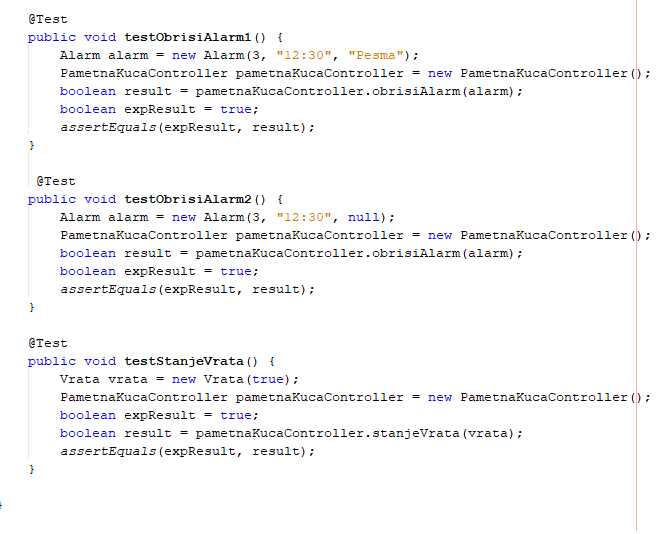
U nasem slucaju prilikom testiranja svi testovi su prosli bez greske.



Test za otvaranje zavese prvo izvlacimo vrednost false sto znaci da je zavese zatvorena zatim pozivamo opciju otvaranje zavese pomocu koje se menja vrednost zavese I otvara zavesa. Zatvaranje zavese funkcionise na istom principu.

Stanje zavese proveravamo tako sto postavimo ocekivani rezultat I pozovemo funkciju stanje zavese I ukoliko se poklapaju rezultati test ce proci uspesno.





## Testiranje metodom crne kutije – Black Box

### Ekvivalentno parcelisanje

Tehnika podele na klase ekvivalencije polazi od ideje da se ulazni podaci mogu razvrstati u reprezentativne klase tako da se za sve pripadnike jedne klase program ponaša na sličan način. Te reprezentativne klase su nazvane klasama ekvivalencije. U ovom slučaju, klase ekvivalencije su međusobno disjunktne i pokrivaju ceo prostor vrednosti ulaza.

Testiranje se obavlja samo za jednu reprezentativnu vrednost ulaza iz svake klase ekvivalencije, zato što se smatra da je to jednako delotvorno kao i testiranje bilo kojom drugom vrednošću iz iste 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 unetog datuma | Rev 1 | Autor | Uros Milovanovic | Datum | 03.01.2022. |
| Cilj | Provera ispravnosti sistema za validaciju pri unosu datuma | | Reference |  | | |
| Test uslovi |  | | | Neophodno vreme za izvršenje test slucaja | | 5 min |

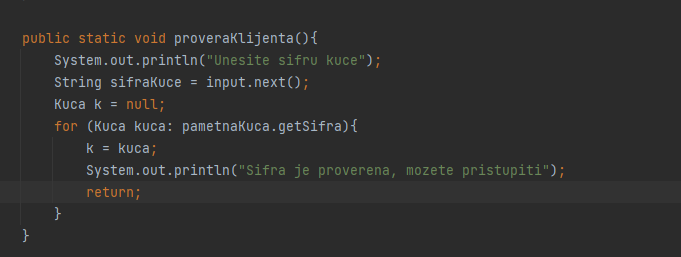
|  |  |
| --- | --- |
| Opis postavke za testiranje | |
|  | Sistem prihvata samo predefinisani odnosno dozvoljeni format broja koji je u sledećem formatu: dd-mm-yyyy |
|  | Aplikacija je pokrenuta i to od strane vlasnika odnosno korisnika aplikacije |
|  | Korisnik koristi tablet za unos podataka |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Definicija testa | | | Izvršenje testa | |
| Uslovi | Ulazni podaci | Očekivani rezultati | Aktuelni rezultati | Broj problema |
| Korisnik je izvršio unos datuma | Unet je datum 03-01-2022 koji zadovoljava formu | Ukoliko se uneti broj ne podudara sa definisanim formatom od sistema se očekuje da ispiše grešku i ponudi odnosno omogući ponovni unos | Sistem je odreagovao očekivano i zbog dobrog formata to jest unosa izvrsio naredbu | 0 |

## Testiranje metodom bele kutije – White Box

White box testiranje ili strukturno testiranje obuhvata ispitivanje interne strukture programa ili sistema. Testni podaci se dobijaju ispitivanjem logike programa ili sistema, bez brige o zahtevima koje treba da zadovolji. Tester poznaje internu strukutru i logiku programa, isto kao što mehaničar zna unutrašnji mehanizam automobila.

## Pokrivenost petlji



Iz datog primera vidimo da je pokrivenost petlje u metodi takva da je petlja pretraživala listu pametne kuce sve dok nije naišla na podudaranje unite sifre sa postojećom sifrom koje nasa pametna kuca ima sacuvane u bazi, te da nije prošla kroz celu listu odnosno izvršila sve iteracije.

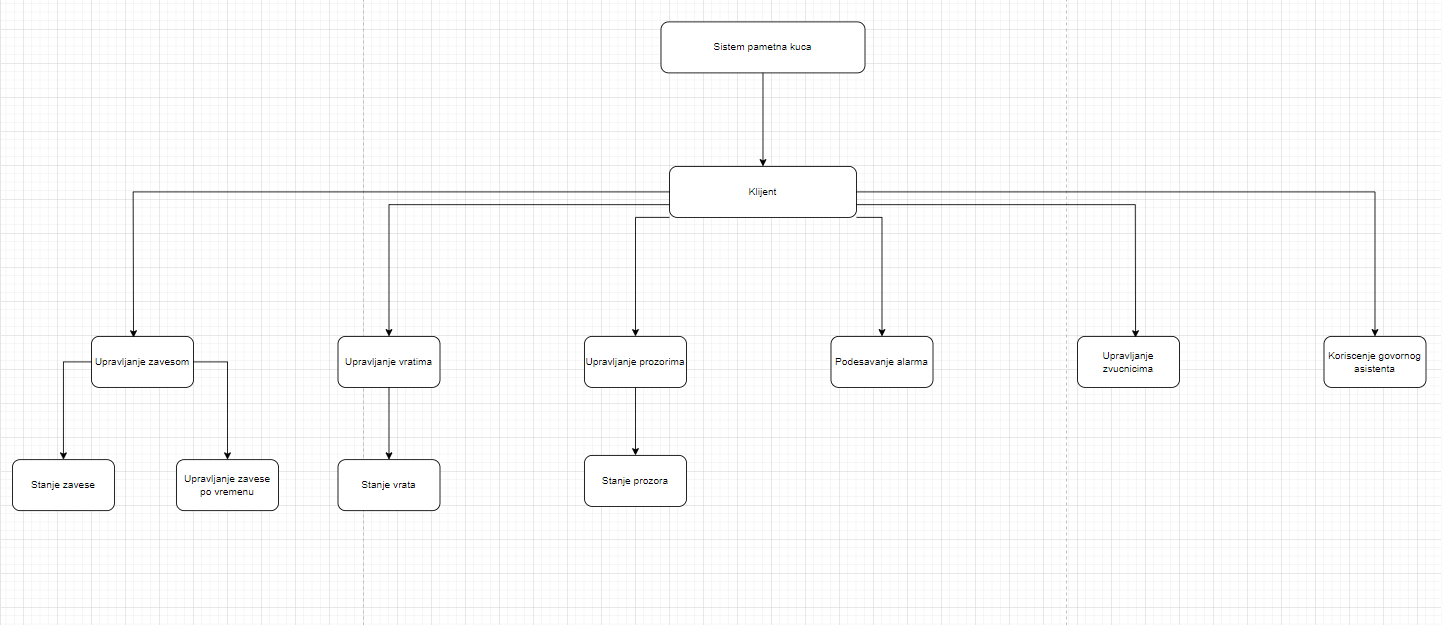
## Integraciono testiranje

Integraciono testiranje se radi posle jediničnog testiranja (“Unit testing”). Nakon što su sve individualne jedinice (metode) napravljene i testirane, počinjemo sa kombinovanjem ovih jedinično-iztestiranih modula i tako radimo integraciono testiranje. Integraciono testiranje se obavlja u toku razvoja softvera a ne na kraju razvoja softvera. Koristi se mogu dobiti od ovakve pažljive integacije su lakša dijagnoza defekata, manje defekata, manje vremena do prvog operativnog proizvoda, kraće ukupno vreme razvoja proizvoda, veće šanse da se projekat uradi uspešno i na samom kraju pouzdanije procene rokova i naravno bolji kvalitet programa.

Postupak integracionog testiranja:

1. individualni moduli se prvo testiraju u izolaciji od ostalih modula odnosno jedinično testiraju
2. integrisati tj. kombinovati jedinično-iztestirane module, jedan po jedan, u složenije celine
3. testirati sve dok se ne integrišu svi moduli, i to uključujući kombinovano ponašanje modula
4. testirati svaki ovaj složeni modul kao jednu jedinicu
5. glavni cilj da se testiraju interfejsi između jedinica/modula i da se izvrši validacija da li su zahtevi implementirani korektno

Putanje integracionog testiranja sistema koristeći top-down tehniku



# ODRŽAVANJE, DALJI RAZVOJ I KONTINUALNI PROCES POBOLJŠANJA SOFTVERA

## Model estimacije troškova prilikom održavanja odabrane aplikacije

***Vođa projekta*** - Radiće 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, lokacija, QR codova.

Predviđeno radno vreme na projektu je 200 radna sata. Cena rada po satu iznosi 22 evra. 200 \* 22 = 4400 evra

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

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

Predviđeno radno vreme na projektu je 350 radnih sati. Cena rada po satu iznosi 50 evra. 350 \* 50 = 17500 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 15 evra. 40 \* 15 = 600 evra

Ostali resursi:

• PC-Server – 18000 evra

• PC 1 – 900 evra

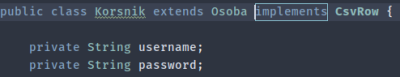
• PC 2 – 900 evra

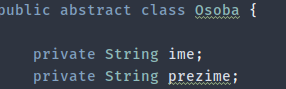
• PC 3 – 900 evra

• PC 4 – 900 evra

## Pristup za postizanje višekratne upotrebljivosti

Primer rejuzabilnosti klasa primenom nasleđivanja:





Generalizacija je tehnika koja se koristi da bi se na jednostavan način rešavale složene situacije. Uopštene klase sadrže atribute i operacije (metode) koje imaju druge klase, koje su onda podklase uopštene klase. 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).

Primer primene design pattern-a

Decorator šablon dozvoljava dodavanje nove funkcionalnosti postojećeg objekta bez menjanja njegove strukture.Ovaj šablon kreira Decorator klasu omotač originalne klase i obezbeđuje dodatnu funkcionalnost bez promena potpisa metoda klase

# ZAKLJUČAK

Centar svakog poslovanja je zadovoljan korisnik. Zato je potrebno obezbediti u samom startu proizvod visokog kvaliteta koji će garantovati njegovu vrednost i pouzdanost. Kreiranje softvera se može posmatrati i kao poslovni dogovor. Softver treba da iznese klijentske potrebe bez greške. Testiranje softvera nudi dokaz za softver ispunjava poslovne zahteve klijenta.

# LITERATURA

1. LAMS Nastavni materijali predmeta SE32- Obezbeđenje kvaliteta, testiranje i održavanje
2. https://dtc.rs/sr/testiranje-softvera
3. Softvera - Univerzitet Metropolitan Beograd