Univerzitet u Beogradu
Matematički fakultet

Marko Trajkov

Projekat i implementacija sistema za upravljanje problemima u razvoju softvera

master rad

Beograd
2012

Autor: Marko Trajkov
Naslov: Projekat i implementacija sistema za upravljanje problemima u razvoju softvera

Mentor: dr Saša Malkov, Univerzitet u Beogradu - Matematički fakultet
Članovi komisije: dr Vladimir Filipović, Univerzitet u Beogradu - Matematički fakultet
dr Filip Marić, Univerzitet u Beogradu - Matematički fakultet

Datum odbrane: 28.09.2012

Sadržaj

1. Uvod

U mnogim softverskim projektima sistemi za upravljanje problemima u razvoju softvera igraju centralnu ulogu: oni omogućavaju da korisnici komuniciraju sa programerima i obaveštavaju ih o problemima u radu njihovog softvera. Sa druge strane, programeri mogu da čuvaju informacije o nerazrešenim problemima i zahtevaju više informacija od korisnika. Danas su sistemi za upravljanje problemima u razvoju softvera postali nezamenjiv alat za praćenje velikih projekata. Bez ovih sistema, praćenje velike količine problema je veoma otežano ili skoro nemoguće. Zbog toga se danas može naći veći broj ovakvih sistema. Oni se razlikuju po kvalitetu, bezbednosti i po funkcijama koje nude korisniku.

Sistem za upravljanje problemima se može posmatrati kao deo mnogo šireg sistema za praćenje projekata, ili kao samostalan sistem. Oba pristupa imaju i svoje prednosti i svoje nedostatke, a u ovom radu sistem za upravljanje problemima biće posmatran kao jedan zaseban sistem.

1.1 Metodologija

Prilikom projektovanja i izrade ovog rada, korišćeni su neki elementi metodologije objektno orijentisane analize i dizajna (engl. *OOAD – Object Oriented Analysis and Design*). Sistem se posmatra kao skup objekata koji međusobno interaguju i razvijan je prolaskom kroz četiri faze.

Definisanje problema se ogleda kroz analiziranje problema i definisanja granica sistema. U ovoj fazi treba da se opiše problem, da se navede u kojoj formi treba da bude rešenje i koje uslove treba da zadovolji.

Objektno orijentisana analiza primenjuje tehnike za anliziranje zahteva sistema, fokusirajući se na to šta sistem radi. Polazeći od domena problema, gradi se konceptualni model koji opisuje traženi proizvod. On je obično predstavljen pomoću dijagrama slučajeva upotrebe. Konceptualni model doprinosi dekompoziciji sistema, jer izdvaja koncepte iz realnog sveta domena problema, identifikujući njihove atribute i veze.

Objektno orijentisan dizajn koristi elemente modela koji je konstruisan u analizi i od njega pravi rešenje koje je bliže implementaciji. Izdvajaju se klase, odnosno skupovi objekata sa istim svojstvima i ponašanjem, i modeliraju se dijagramom klasa. Na osnovu uočenih entiteta i njihovih odnosa, pravi se struktura baze podataka predstavljena modelom entiteta i odnosa. Dizajn je fokusiran na to kako sistem radi a rezultat je detaljniji opis implementacije sistema.

Implementacija je poslednja faza koja se sastoji u implementiranju konačnog rešenja. Ona može sadržati posebne detalje o optimizaciji sistema, kao i detalje vezane za testiranje ispravnosti rešenja. Za to se mogu koristiti posebni alati, a više detalja o tome se može naći u odeljku 5.

Na kraju se može izdvojiti još jedna faza, a to je održavanje sistema. U ovom radu neće biti puno pažnje posvećeno održavanju sistema, jer je fokus na konstrukciji rešenja.

1.1.1 Slučajevi upotrebe

Slučaj upotrebe (engl. *use case*) opisuje interakciju između učesnika i sistema, koja se sastoji od niza koraka koje sprovodi učesnik, kako bi sproveo odgovarajuću akciju u sistemu. Slučajevi upotrebe su korisni jer pomažu programerima da lakše uvide koje funkcije softvera je potrebno implementirati kao i to kako one treba da izgledaju. Oni se dokumentuju odgovarajućim opisom pojedinačnih slučajeva upotrebe, a ilustruju dijagramima slučajeva upotrebe. Dijagrami slučajeva upotrebe se sastoje od *slučaja upotrebe* (prikazan elipsom), koji je povezan sa *učesnikom*, kao i *odnosom* između učesnika i slučaja upotrebe (slika 1).

Slika 1.

Pored toga, dijagramima se mogu prikazivati i odnosi između slučajeva upotrebe, kao i odnosi između učesnika. Odnosi između slučajeva upotrebe mogu biti: *odnos uključivanja*, kojim se ukazuje da jedan slučaj upotrebe predstavlja deo ili korak drugog složenog slučaja upotrebe i *odnos proširivanja*, kojim se navodi da jedan slučaj upotrebe može da proširi i ispolji celokupno ponašanje drugog slučaja upotrebe. Odnos između učesnika može biti predstavljen *odnosom generalizacije*. Isti odnos može biti primenjen i na slučajeve upotrebe. Odnos generalizacije ukazuje da jedan slučaj upotrebe (učesnik) može biti samo specijalan slučaj drugog, opšteg slučaja upotrebe (učesnika).

Opis jednog slučaja upotrebe obuhvata:

- Kratak opis – opis elementarnih činjenica potrebnih za razumevanje slučaja upotrebe.
- Aktere – koji akteri učestvuju u datom slučaju upotrebe.
- Preduslove – uslovi koji moraju biti ispunjeni pre izvršavanja slučaja upotrebe.
- Postuslove – uslovi koji moraju biti ispunjeni da bi se slučaj upotrebe smatrao uspešno izvršenim.
- Glavni tok – tok koji predstavlja uobičajan tok događaja. Sastoji se od niza koraka koje obavlja akter kako bi se izvršio navedeni slučaj upotrebe.
- Alternativne tokove – ako nešto može da krene drugačije od glavnog toka, navodi se ovde.

1.1.2 Dijagrami entiteta i odnosa

Model entiteta i odnosa predstavlja apstraktnu reprezentaciju podataka. Ovaj model se najčešće koristi za modeliranje relacione baze podataka a dijagrami koji se koriste u ovom modelu se nazivaju *dijagrami entiteta i odnosa*. Na ovim dijagramima se predstavljaju *entiteti* i *odnosi* između tih entiteta. Entitet predstavlja neki objekat u sistemu, dok odnos

predstavlja vezu između dva entiteta. Svaki odnos ima svoju kardinalnost. Postoje tri osnovna tipa kardinalnosti odnosa između dva entiteta i oni su prikazani na slici 2.

	jedan prema jedan - jedan entitet X može biti pridružen najviše jednom entitetu Y. Jedan entitet Y može biti pridružen najviše jednom entitetu X.
	jedan prema više - jedan entitet X može biti pridružen većem broju entiteta Y. Jedan entitet Y može biti pridružen najviše jednom entitetu X.
	više prema više - jedan entitet X može biti pridružen većem broju entiteta Y. Jedan entitet Y može biti pridružen većem broju entiteta X.

Slika 2. Prikaz tri osnovna tipa kardinalnosti

2. Analiza problema i postojećih rešenja

2.1 Definisanje problema

Sistem za upravljanje problemima u razvoju softvera dozvoljava kako pojedincima tako i grupi programera, da efikasno prate probleme u svom radu. Imajući kompletne informacije u izveštaju, programer može brže i lakše rešiti nastali problem. Stoga, izbor dobrog sistema za upravljanje problemima pomaže da se skrati vreme razvoja, da se poveća produktivnost, da se poveća zadovoljstvo korisnika i unapredi komunikacija između samih programera.

Danas moderni projekti postaju sve kompleksniji i teži za praćenje. Kako projekat prolazi kroz mnoge faze (alfa, beta, kandidat za objavljivanje i kroz mnoge druge verzije), praćenje velike količine problema u njima postaje otežano. Zbog toga je važno pratiti i beležiti probleme koji se javljaju tokom životnog ciklusa takvog projekta.

Verzije postojećih projekata povećavaće se tokom vremena. Kompleksnost projekata nastaviće da raste. Isti problemi nastaviće da se pojavljuju sa svakom generacijom programera. Stoga je važno da se informacije o prethodnim problemima čuvaju kako bi se poboljšao budući kvalitet softvera.

Sistem za upravljanje problemima treba da omogućava:

- Napredno upravljanje problemima
- Pregledanje dnevnika izmena i detalja za svaki problem
- Elementarne operacije upravljanja projektima
- Pregledanje izveštaja za projekte
- Pregledanje statističkih detalja vezanih za projekte
- Upravljanje korisnicima i njihovim ulogama u projektima
- Rešenje zasnovano na veb tehnologijama

U ovom radu se pod pojmom *problem* podrazumeva malo širi kontekst i on obuhvata sve ono što može da krene naopako, o čemu treba diskutovati ili neki neuobičajen tok izvođenja posla. Prema tome, problem obuhvata sledeće tipove kartica u sistemu: bag, rizik, unapređenje, pitanje, modifikacija. Ovi primeri tipova problema imaju samo opisnu ulogu. Oni se mogu menjati i prilagođavati potrebama sistema.

2.2 Pregled postojećih rešenja

Na tržištu danas ima nekoliko gotovih rešenja. Neka od njih postoje već duže vreme, tokom koga su popravljani i unapređivani. Ovde će biti navedena neka od najvažnijih rešenja, kao i osnovne informacije o njima.

2.2.1 *Bugzilla*

Bugzilla je nastala 1998 godine. Prvobitno je korišćena u pravljenju *Netscape Communicator*-a kao sistem za upravljanje problemima u softveru. Posle toga je postala veoma popularan sistem za upravljanje problemima i danas se koristi u brojnim projektima.

Posle instaliranja, podešavanja i privikavanja, pokazuje se kao veoma dobar sistem za upravljanje problemima. Njen interfejs se nije preterano menjao pa su se njeni korisnici uglavnom navikli na njega. To je verovatno i jedan od razloga što je još uvek popularna. Jedni od njenih glavih atributa su jednostavnost i brzina, tako da su pozivi ka bazi podataka svedeni na minimum, kao i generisanje bogatog *HTML*-a. Probleme može postavljati svaki korisnik i biće dodeljeni nekom korisniku. *Bugzilla* ima veoma napredan sistem za izveštaje, moguće je i pravljenje raznih vrsta dijagrama kao i slanje rezultata određene pretrage, pojedincu ili grupi.

2.2.2 *Mantis*

Mantis je besplatan sistem za upravljanje problemima u softveru, nastao 2000 godine. Pisan je u *PHP*-u i ima podršku za različite sisteme za upravljanje bazama podataka kako i za različite operativne sisteme. Glavna mana mu je njegov interfejs koji ne zadovoljava moderne standarde. Sa druge strane lak je za navigaciju, čak i za neiskusne korisnike. Nema neke naprednije funkcije (npr. dijagrame), mada se oni mogu dobiti korišćenjem neke dodatne biblioteke npr. *GraphWiz* ili *JpGraph*. Ukratko, na celom sistemu treba još dosta poraditi pre svega na novim funkcijama i dizajnu.

2.2.3 *BugTracker.NET*

BugTracker.NET je besplatan sistem za upravljanje problemima u softveru. Zasnovan je na vebu, a pisan je korišćenjem *ASP.NET*, *C#* i *Microsoft SQL Servera*. Ima veoma intuitivan interfejs i moguće je uz pomoć alata za hvatanje sadržaja ekrana, dodati grešku uz samo par klikova. Moguća je i njegova integracija sa sistemom *Subversion*.

2.2.4 *Flyspray*

Flyspray je sistem za upravljanje problemima u softveru napisan u *PHP*-u. On je besplatan, objavljen pod licencom *GNU GPL* (*General Public License*) 2003. godine. Pošto je otvorenog koda, svi mogu da ga modifikuju i prilagođavaju svojim potrebama. Grafički interfejs nije komplikovan, a od dodatnih funkcija ima podršku za više baza podataka, više operativnih sistema, podršku za više projekata, obaveštenja o promenama (elektronska pošta, kanale informisanja tj. *RSS*, *Jabber*), dijagrame i drugo.

2.2.5 *Redmine*

Redmine je besplatan sistem za upravljanje projektima i problemima koji je nastao 2006 godine. Na dizajn *Redmine*-a je značajno uticao *Trac*, softverski paket sa sličnim mogućnostima. Pisan je korišćenjem *Ruby on Rails* razvojnog okvira (engl. *framework*). *Redmine* je sistem otvorenog kada i nalazi se pod licencom *GNU GPL (General Public License)*. Fleksibilan je, i mogu se definisati sopstveni statusi i tipovi. Korisnik može imati različite uloge na svakom projektu. Korisnički interfejs je veoma jednostavan, intuitivan i lak za navigaciju.

2.2.6 *Bugzero*

Bugzero je sistem za upravljanje problemima u softveru zasnovan na vebu. Ima podršku za više različitih baza podataka, kao i podršku za više operativnih sistema. Ima intuitivan korisnički interfejs ali mu nedostaju neke naprednije funkcije. Glavna mana mu je to što je komercijalan, pa se mogu se naći i bolji sistemi za upravljanje problemima koji su besplatni.

2.3 Pregled osnovnih funkcija postojećih rešenja

Može se uočiti da prethodno navedena rešenje imaju uglavnom sličan skup funkcija. To su uglavnom funkcije koje su se tokom vremena pokazale kao korisne, pa stoga nalaze svoje mesto u današnjim sistemima. Neke od tih funkcija su prikazane u tabeli 2.

	Bugzilla	***Mantis***	***BugTracker .NET***	***Flyspray***	***Redmine***	***Bugzero***
Pretraga	da	da	da	da	da	da
Obaveštenja (elektronska pošta, kanali informisanja tj. *RSS*)	da	da	da	da	da	da
Izveštaji	da	ne	da	da	da	da
Dijagrami	da	ne	da	ne	da	ne
Podrška za više platformi	da	da	ne	da	da	da
Podrška za vise sistema za upravljanje bazama podataka	da	da	ne	da	da	da
Podrška za umetke (engl. *Plugin*)	da	da	ne	da	da	ne
Podrška za neki sistem za kontrolu verzija	da	da	da	ne	da	ne

Tabela 2: Osnovne mogućnosti. Tabela predstavlja pregled dostupnosti osnovnih elemenata u nekim današnjim sistemima.

2.4 Kako unaprediti postojeća rešenja

Mnoga od navedenih rešenja postoje već duži vremenski period. Tokom vremena njihovi nedostaci su uglavnom otklonjeni ali i pored toga postoje stvari koje se mogu unaprediti. Pokazuje se da su u današnjim sistemima, u opisu problema, izostavljeni neki parametri, koji bi mogli da pomognu programeru da lakše uoči uzrok problema i time ga brže i lakše otkloni. Neki od tih parametara su: praćenje steka, koraci pri reprodukciji, uočeno ponašanje, očekivano ponašanje, test primeri, slika ekrana, zavisnosti, dnevnik greške, primer izvornog koda. Ovi parametri se danas obično unose u polje koje je namenjeno za opis problema. Ovi parametri, po mnogim istraživanjima [2, 4], visoko su kotirani po nivou korisnosti za programere, a sa druge strane su najčešće izostavljani. Izdvajanje tih parametara u posebna polja bi moglo da ima svoje prednosti: korisnicima koji unose probleme bilo bi posebno signalizirano da je potrebno uneti date informacije, čime bi se

povećala verovatnoća da te informacije budu unesene. Dalje, bila bi omogućena pretraga i kategorizacija problema po tim poljima.

Just, Premraj i *Zimmermann* navode još par preporuka koje bi mogle da poboljšaju trenutne sisteme za upravljanje problemima [2]. Neke od tih preporuka su:
Preporuka #1: Omogućiti različite korisničke interfejse za korisnike sa različitim iskustvom (početnik, ekspert). Prikazivati naznake korisnicima koji prijavljuju greške, koje informacije bi trebalo da obezbede i kako da ih prikupe.
Preporuka #2: Integrisati reputaciju u korisničke profile, kako bi označili iskusne korisnike. Hooimeijer i Weimer mere reputaciju korisnika kao procenat prijavljenih problema koji su ispravljeni [5].

Još jedna zgodna karakteristika mogla bi biti i zapis o tome *kako je problem rešen*. Način na koji je problem rešen ne omogućava da se direktno problem ispravi jer se on unosi po uspešnom ispravljanju problema, ali može biti koristan kasnije kada se pojavi sličan problem u istom ili nekom drugom projektu. Spisak ovih parametara je dat u tabeli 3, zajedno sa informacijom da li je on prisutan u konkretnom sistemu za upravljanje problemima.

	Bugzilla	***Mantis***	***BugTracker .NET***	***Flyspray***	***Redmine***	***Bugzero***
Praćenje steka	ne	ne	ne	ne	ne	ne
Koraci pri reprodukciji	ne	da	ne	ne	ne	ne
Uočeno ponašanje	ne	ne	ne	ne	ne	ne
Očekivano ponašanje	ne	ne	ne	ne	ne	ne
Test primeri	ne	ne	ne	ne	ne	da
Slika ekrana	ne	ne	da	ne	ne	ne
Zavisnosti	ne	ne	da	ne	ne	da
Kako je problem rešen	ne	ne	ne	ne	ne	ne
Dnevnik greške (*log*)	ne	ne	ne	ne	ne	ne
Primer izvornog koda	ne	ne	ne	ne	ne	ne

Tabela 3: Relevantni parametri koji omogućavaju lakše ispravljanje problema. Tabela predstavlja pregled dostupnosti relevantnih parametara u nekim današnjim sistemima.

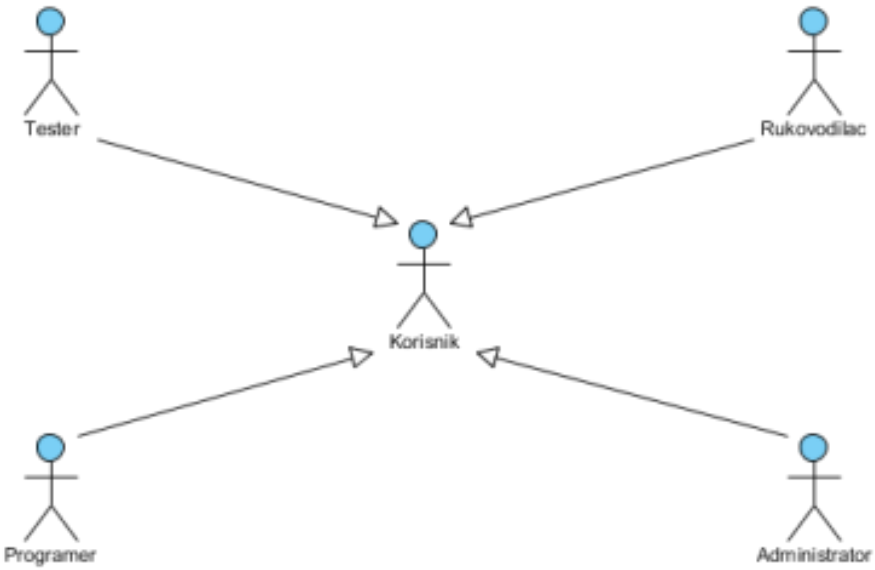
Kao što se može uočiti, većina navedenih parametara u današnjim sistemima nije prisutna. Ovo može predstavljati nedostatak samog sistema za upravljanje problemima, jer upravo neke relevantne informacije mogu biti slučajno izostavljene. Unapređenje postojećih sistema se može ogledati upravo kroz uključivanje navedenih informacija u sistem. Jedan takav sistem, sa tim dodatnim parametrima, biće predstavljen u ovom radu.

3. Definisanje zahteva

3.1 Učesnici u sistemu

Učesnici u ovom sistemu su podeljeni po ulogama. Svaka uloga se dodeljuje za konkretni projekat tako da korisnik može imati različite uloge za različite projekte.

Razlikuju se 4 osnovne vrste učesnika za svaki projekat: *tester*, *programer*, *rukovodilac* i *administrator* (videti sliku 3). Može postojati više različitih vrsta učesnika koji imaju iste uloge. To znači da se mogu dodavati nove vrste učesnika koji bi delili iste uloge. Na primer, može se dodati vrsta učesnika *reporter*, koji bi imao iste privilegije kao *tester* jer ima iste uloge. Time se postiže da *reporter* i *tester* imaju iste uloge, a da su u stvari dve različite vrste učesnika. Tako se npr. mogu razdvajati senior programeri od junior programera. Ta podešavanja su ostavljena administratoru sistema a ovde ćemo se pozabaviti različitim ulogama učesnika.



Slika 3. Uloge

Tester
Ovu ulogu imaju svi korisnici koji su se prijavili na sistem, i žele da prijave primećeni problem ili nedostatak. Oni su obično neiskusni u prijavljivanju problema, pa su informacije koje oni obezbeđuju obično nižeg kvaliteta. To svakako ne znači da ih treba zanemariti, jer njihove informacije ponekad mogu biti dragocene. Zbog toga im treba omogućiti pristup sistemu i prijavljivanje problema, ali naravno sa ograničenim privilegijama. Tester ima osnovne privilegije koje uključuju: dodavanje novog problema, pretragu i pregled postojećih problema.

Programer
Programeri su osobe koji se bave razvojem datog softvera i koji su dužni da date probleme otklanjaju. Programer se vezuje za određeni projekat, tako da on može biti programer u jednom projektu, a u drugom projektu imati sasvim drugu ulogu. Njihove privilegije

predstavljaju nadskup privilegija testera, u smislu da i oni mogu prijavljivati probleme, menjati ih i slično. Problem koji je prijavljen biće dodeljen nekom programeru i on će raditi na njegovom otklanjanju.

Rukovodilac
Rukovodilac se vezuje za određeni projekat i on upravlja tim projektom. On ima privilegije da dodaje korisnike u projekat, menja i definiše projekat i druge stvari.

Administrator
Administratori su na vrhu ove lestvice. Oni mogu da obavljaju sve uloge koje mogu i rukovodioci i programeri samo što u praksi to verovatno neće raditi. Administrator je dužan da nadgleda sistem za upravljanje problemima, da udalji nekog korisnika sistema ako primeti nedolično ponašanje, u smislu suspendovanja njegovog korisničkog računa na neki vremenski period.

3.2 Objekti u sistemu

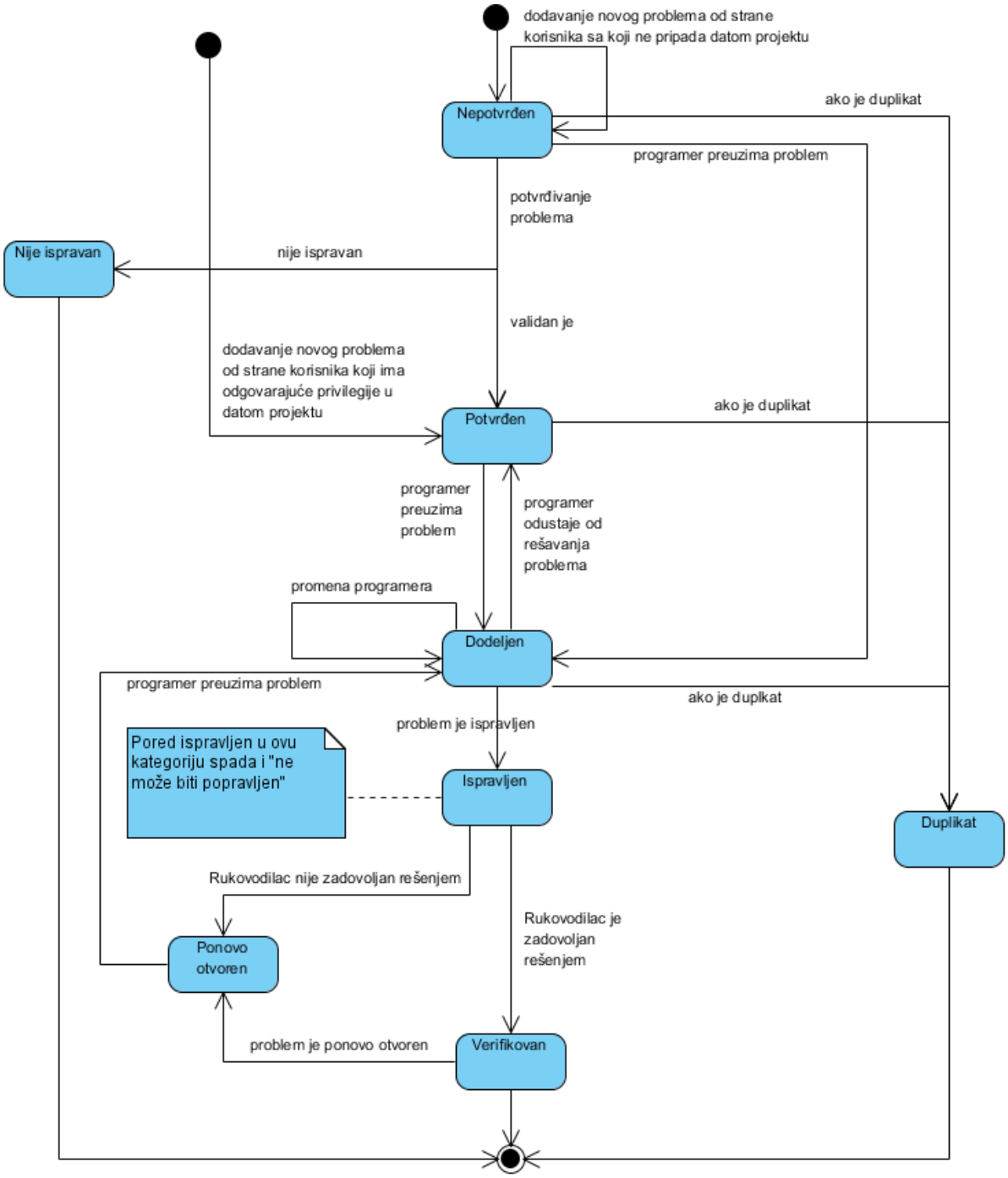
Glavni objekti u sistemu za upravljanje problemima u razvoju softvera jesu problem i projekat.

Problem
Problem je sve ono što može biti dodato u sistem kao jedinstvena celina. On može biti različitog tipa: bag, rizik, unapređenje, pitanje, modifikacija. Ovo nisu fiksirane vrednosti i mogu se menjati kao i dodavati nove. Problemi se mogu prijavljivati od strane svih korisnika sistema, a za njihovo rešavanje zaduženi su programeri.

U sistemu za upravljanje problemima problem prolazi kroz određena stanja:

- nepotvrđen – kada korisnik bez viših privilegija (u ovom slučaju Tester) doda novi problem, problem će biti označen kao *nepotvrđen.*
- potvrđen – u ovo stanje se može doći na dva načina. Jedan od njih je kada korisnik sa odgovarajućim privilegijama doda novi problem, a drugi je kada korisnik sa odgovarajućim privilegijama prebaci nepotvrđen problem u ovo stanje.
- nije ispravan – ako novi problem koji je dodat ne predstavlja ispravno unet problem, onda se on prebacuje u ovo stanje. Ovo stanje označava da uneti problem ne treba razmatrati, niti posvetiti vreme njegovom rešavanju.
- duplikat – ponekad se može javiti kopija ili problem veoma sličan nekom već postojećem problemu. Takav problem se označava kao duplikat.
- dodeljen – kada programer želi da preuzme odgovornost za rešavanje nekog problema, on ga dodeljuje sebi. Ovo se radi kako ne bi dva programera istovremeno radila na istom problemu ne znajući jedan za drugog. Ako se programer predomisli, on može odustati od rešavanja problema i time problem ponovo prelazi u stanje *potvrđen*.
- ispravljen – kada programer ispravi preuzeti problem, on ga prebacuje u stanje *ispravljen.*
- ne može biti popravljen – ako programer ne može da popravi problem, on ga prebacuje u stanje *ne može biti popravljen*

- verifikovan – ako je rukovodilac projekta zadovoljan rešenjem, on može iz stanja *ispravljen* ili *ne može biti popravljen*, prebaciti problem u stanje *verifikovan*. Ovo je završno stanje.
- ponovo otvoren – iz nekog razloga problem može postati ponovo aktivan. Tada se problem prebacuje u stanje *ponovo otvoren*.



Slika 4: Dijagram stanja problema

Dijagram stanja problema je prikazan na slici 4. Na dijagramu su prikazani i korisnici koji mogu dovesti problem u dato stanje. Ako nije naveden korisnik onda se podrazumeva da to

može uraditi svaki korisnik kome je dodeljena ta privilegija. Dijagram je opšteg oblika i problem može biti bilo kog tipa.

Projekat
Svi problemi pripadaju nekom projektu. Projekat predstavlja grupu problema koji se odnose na isti proizvod (softver). Projekat pravi administrator sistema i tom prilikom dodeljuje ga osobi koja je zadužena za taj projekat. Sve dalje operacije nad projektom su u nadležnosti rukovodioca projekta. Projekat ima svoje članove, tj. korisnike koji imaju određenu ulogu na tom projektu.

3.3 Slučajevi upotrebe

Na globalnom planu postoje pet osnovne grupe slučajeva upotrebe:

- *Pregledanje i dodavanje problema* odnosi se na osnovnu ulogu u sistemu, koju mogu da imaju svi učesnici.
- *Upravljanje problemima u okviru projekta* se odnosi na rad sa problemima u okviru nekog projekta. To obuhvata preuzimanje odgovornosti za problem, njegove izmene i slične stvari koje se mogu raditi sa njima.
- *Upravljanje projektom* je malo višeg nivoa i trebalo bi da obuhvati deo poslova nešto šireg sistema za upravljanje projektima. Pre svega bi to bilo upravljanje korisnicima projekta i definisanje određenih parametara vezanih za taj projekat.
- *Upravljanje korisničkim nalogom* odnosi se na poslove poput pravljenja naloga, prijavljivanja na sistem i menjanja ličnih podataka.
- *Administriranje sistema* sadrži poslovi koje obično obavlja administrator i to su uglavnom neke tehničke stvari. U nastavku teksta će detaljnije biti prikazana svaka od ovih grupa.

3.3.1 Pregledanje i dodavanje problema

Ova grupa slučajeva upotrebe se odnosi na osnovne akcije koje su dostupne svim korisnicima sistema: *Pregledanje informacija o postojećem problemu*, *Prijavljivanje novog problema* i *Pretraživanje problema* (slika 5).

Slučaj upotrebe: Pregledanje informacija o postojećem problemu

Kratak opis: Detaljan pregled informacija o postojećem problemu.
Akteri:

- Korisnik (Tester, Programer, Rukovodilac, Administrator)

Preduslovi:

- Korisnik je prijavljen na sistem.

Postuslovi:

- Korisnik se informisao o detaljima određenog problema.

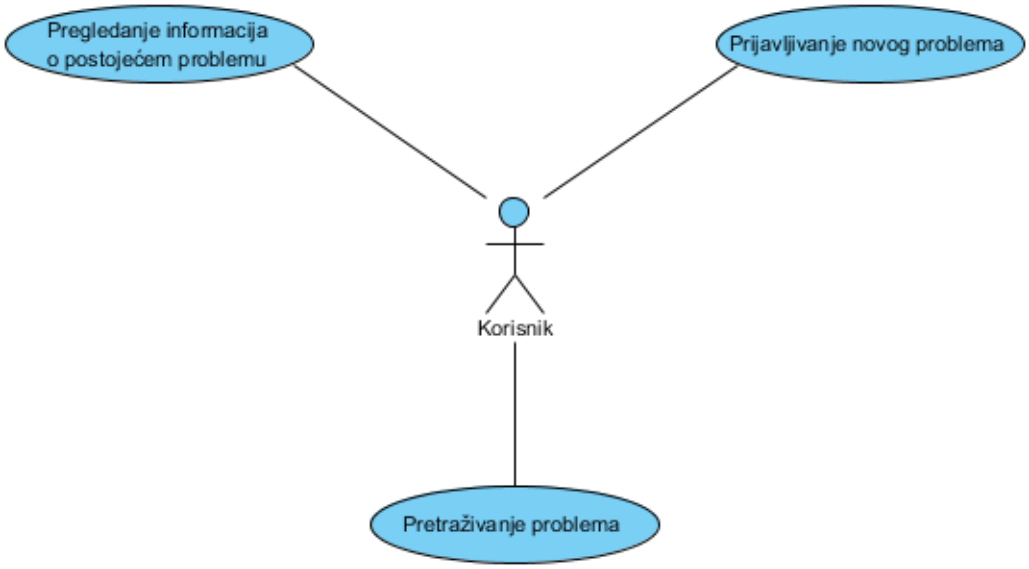
Glavni tok:

1. Korisnik bira opciju za prikaz projekata iz glavnog menija.

2. Iz prikazane liste projekta korisnik bira projekat čije probleme želi da pregleda.
3. Iz prikazane liste problema[1] korisnik bira problem koji želi da pregleda.
4. Korisnik pregleda sadržaj problema.
5. Korisnik završava pregled sadržaja problema.

Alternativni tokovi:

Nema alterantivnih tokova.



Slika 5. Dijagram slučajeva upotrebe – Pregledanje i dodavanje problema

Slučaj upotrebe: Prijavljivanje novog problema

Kratak opis: Prijavljivanje novog problem i unošenje odgovarajućih informacije o njemu. Postoje dva načina za dodavanje novog problema: običan i preko čarobnjaka. Oni su obuhvaćeni jednim slučajem upotrebe jer obavljaju isti posao.

Akteri:

- Korisnik (Tester, Programer, Rukovodilac, Administrator)

Preduslovi:

- Korisnik je prijavljen na sistem i nalazi se na stranici sa listom problema.

Postuslovi:

- Problem je uspešno prijavljen.

Glavni tok:

1. Korisnik bira opciju za dodavanje novog problema.
2. Korisnik unosi informacije o problemu: naslov, opis, tip problema, ozbiljnost problema, prioritet problema, status problema, verzija projekta, koraci pri reprodukciji, uočeno ponašanje, očekivano ponašanje, test primere, izvorni kod, dnevnik greške, informacije sa steka (engl. *stack trace*), očekivani broj potrebnih časova rada.

   2.1. Korisnik bira polje koje želi da popuni.

[1] Izvorni kod koji prikazuje kako se generiše lista problema se nalazi u dodacima (Odeljak 7.3).

2.2. Korisnik menja dato polje unošenjem nove vrednosti ili odabirom stavke iz padajućeg menija.
2.3. Korisnik prelazi na korak 2.1 ili ukoliko nema više polja koja želi da popuni prelazi na korak 3.

3. Korisnik bira opciju za čuvanje podataka koja će novi problem sačuvati u bazi.

Alternativni tokovi:

1.1. Korisnik bira opciju za dodavanje novog problema preko čarobnjaka.
1.1.1. Korisnik unosi informaciju u polje koja mu se traži.
1.1.2. Korisnik prelazi na sledeću stranicu odabirom opcije za prelazak na sledeću stranicu.
1.1.3. Korisnik prelazi na korak 1.2 ukoliko nema više polja koja treba da popuni.
1.2. Korisnik završava sa unošenjem novog problema odabirom opcije za čuvanje podataka.
3.1 Korisnik odustaje od prijavljivanja novog problema.

Slučaj upotrebe: Pretraživanje problema

Kratak opis: Nalaženje traženog problema tako što se unesu ključne reči ili kriterijum pretrage.

Akteri:

- Korisnik (Tester, Programer, Rukovodilac, Administrator)

Preduslovi:

- Korisnik je prijavljen na sistem.

Postuslovi:

- Korisnik je dobio rezultate pretrage.

Opis:

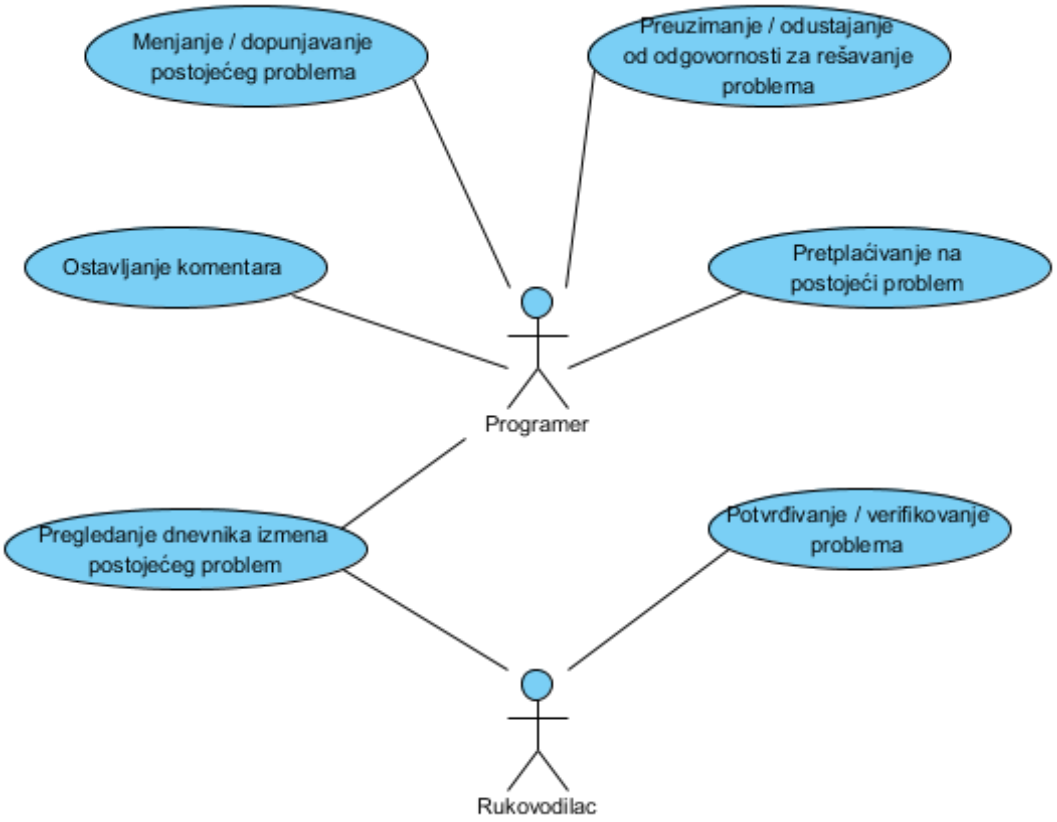
1. Korisnik bira opciju za pretraživanje problema.
2. Korisnik unosi ključne reči po kojima se vrši pretraživanje problema.
3. Korisnik može odabrati naprednu pretragu i onda dobija opciju za kriterijum pretraživanja. Kriterijumi se biraju iz ponuđene liste. Kriterijum se može definisati po:
   - Projektu kome traženi problem pripada.
   - Tipu problema
   - Ozbiljnosti problema
   - Prioritetu problema
   - Statusu problema
   - Korisniku koji je prijavio problem
4. Korisnik potvrđuje unos.
5. Korisnik dobija rezultat.

Alternativni tokovi:

Nema problema koji zadovoljavaju kriterijum pretrage. U tom slučaju prikazuje se odgovarajuća poruka.

3.3.2 Upravljanje problemom u okviru projekta

Ova grupa slučajeva upotrebe se odnosi na sve aktivnosti koje se tiču upravljanja problemima*: Menjanje ili dopunjavanje postojećeg problema, Ostavljanje komentara, Preuzimanje ili odustajanje od odgovornosti za rešavanje problema, Pretplaćivanje na postojeći problem, Potvrđivanje ili verifikovanje problema* i *Pregledanje dnevnika izmena postojećeg problema* (slika 6).



Slika 6. Dijagram slučajeva upotrebe – Upravljanje problemom u okviru projekta

Slučaj upotrebe: Ostavljanje komentara

Kratak opis: Programer može da ostavi komentar za problem ili da odgovori na postojeći komentar za isti problem.

Akteri:

- Programer

Preduslovi:

- Programer se nalazi na stranici sa informacijama o problemu za koji želi da ostavi komentar.

Postuslovi:

- Komentar je unet i sačuvan.

Opis:

1. Programer bira opciju za ostavljanje komentara.
2. Sistem prikazuje formular korisniku za unos komentara.

3. Programer unosi komentar.
4. Programer izdaje zahtev za čuvanje komentara.
5. Komentar se beleži u sistemu i programer dobija potvrdu da je komentar sačuvan.

Alternativni tokovi:

4.1 Korisnik odustaje od unosa komentara.

Slučaj upotrebe: Pregledanje dnevnika izmena postojećeg problem

Kratak opis: Pregledanje dnevnika izmena postojećeg problema omogućava uvid u sve izmene problema od njegovog prijavljivanja u sistemu.

Akteri:

- Korisnik (Programer, Rukovodilac projekta)

Preduslovi:

- Korisnik je prijavljen na sistem i nalazi se na stranici sa listom problema.

Postuslovi:

- Korisnik se informisao o detaljima izmena određenog problema.

Glavni tok:

1. Korisnik bira problem čiji dnevnik izmena želi da vidi.
2. Korisnik bira opciju za pregled dnevnika izmena.
3. Korisnik pregleda dnevnik izmena.

Alternativni tokovi:

Nema alternativnih tokova.

Slučaj upotrebe: Pretplaćivanje na postojeći problem

Kratak opis: Programer se pretplaćuje na izmene i komentare za određeni problem. Pretplata je moguća u vidu određenih kanala informisanja (engl. *RSS*) ili elektronske pošte.

Akteri:

- Programer

Preduslovi:

- Programer je prijavljen na sistem i nalazi se na stranici sa listom problema.

Postuslovi:

- Programer je pretplaćen na sadržaj.

Glavni tok:

1. Programer bira problem na koji želi da se pretplati.
2. Iz padajuće liste programer bira vrstu preplate.
3. Programer potvrđuje izbor.
4. Programer dobija potvrdu o preplati.

Alternativni tokovi:

3.1 Programer odustaje od zahteva.

Slučaj upotrebe: Menjanje ili dopunjavanje postojećeg problema

Kratak opis: Tokom životnog ciklusa informacije o problemu mogu da se menjaju i dopunjavaju.

Akteri:

- Programer

Preduslovi:

- Programer je prijavljen na sistem i nalazi se na stranici sa listom problema.

Postuslovi:

- Izmenjen sadržaj problema je sačuvan.

Glavni tok:

1. Programer bira problem koji želi da izmeni ili dopuni.
2. Programer menja sadržaj problema.
   2.1 Programer bira polje koje želi da izmeni.
   2.2 Programer menja dato polje unošenjem nove vrednosti ili odabirom stavke iz padajućeg menija.
   2.3 Programer prelazi na korak 2.1 ili je odlučio da završi sa izmenama.
3. Programer bira opciju za čuvanje podataka koja će izmenjen problem sačuvati u bazi.

Alternativni tokovi:

3.1 Programer odustaje od izmene sadržaja problema.

Slučaj upotrebe: Potvrđivanje ili verifikovanje problema

Kratak opis: Potvrđivanje problema označava da je problem ispravno prijavljen i da je spreman za rešavanje. Verifikovanje problema označava da je rukovodilac zadovoljan rešenjem problema.

Akteri:

- Rukovodilac projekta

Preduslovi:

- Rukovodilac je prijavljen na sistem i nalazi se na stranici sa listom problema.

Postuslovi:

- Problem je potvrđen ili verifikovan.

Glavni tok:

1. Rukovodilac bira problem koji želi da potvrdi ili verifikuje čime mu se prikazuje stranica sa detaljima problema.
2. Rukovodilac pregleda problem.
   2.1 Rukovodilac bira opciju za potvrđivanje problema.
   2.2 Rukovodilac bira opciju za verifikovanje problema
3. Rukovodilac dobija potvrdu o uspešno obavljenoj operaciji.

Alternativni tokovi:

Nema alternativnih tokova.

Slučaj upotrebe: Preuzimanje ili odustajanje od odgovornosti za rešavanje problema

Kratak opis: Programer preuzima odgovornost za rešavanje problema. Ukoliko odustaje od preuzete odgovornosti problem postaje slobodan za preuzimanje drugim programerima.
Akteri:

- Programer

Preduslovi:

- Programer je prijavljen na sistem i nalazi se na stranici sa listom problema.

Postuslovi:

- Programer je preuzeo odgovornost za rešavanje problema ili je odustao od toga.

Glavni tok:

1. Programer bira problem za koji želi da preuzme odgovornost ili da odustane od nje.
2. Programer bira opciju za preuzimanje ili odustajanje od odgovornosti za rešavanje problema.
3. Programer dobija obaveštenje da je operacija uspešno završena.

Alternativni tokovi:

3.1 Programer dobija informaciju o grešci.

3.3.3 Upravljanje projektom

Pored upravljanja problemima, sistem treba da omogućava osnovne operacije sa projektima. Ova grupa slučajeva upotrebe se odnosi na sve one aktivnosti koje se tiču rada sa projektima: *Dodeljivanje uloge korisniku*, *Brisanje uloge korisniku*, *Ažuriranje uloge korisniku*, *Menjanje ili dopunjavanje postojećeg projekta*, *Pregledanje informacija o postojećem projektu* i *Pregledanje dnevnika izmena za postojeći projekat* (slika 7).

Slučaj upotrebe: Pregledanje informacija o postojećem projektu

Kratak opis: Pregledanje informacija o postojećem projektu omogućava korisniku da se bolje upozna sa konkretnim projektom.
Akteri:

- Rukovodilac, Programer, Administrator

Preduslovi:

- Korisnik je prijavljen na sistem

Postuslovi:

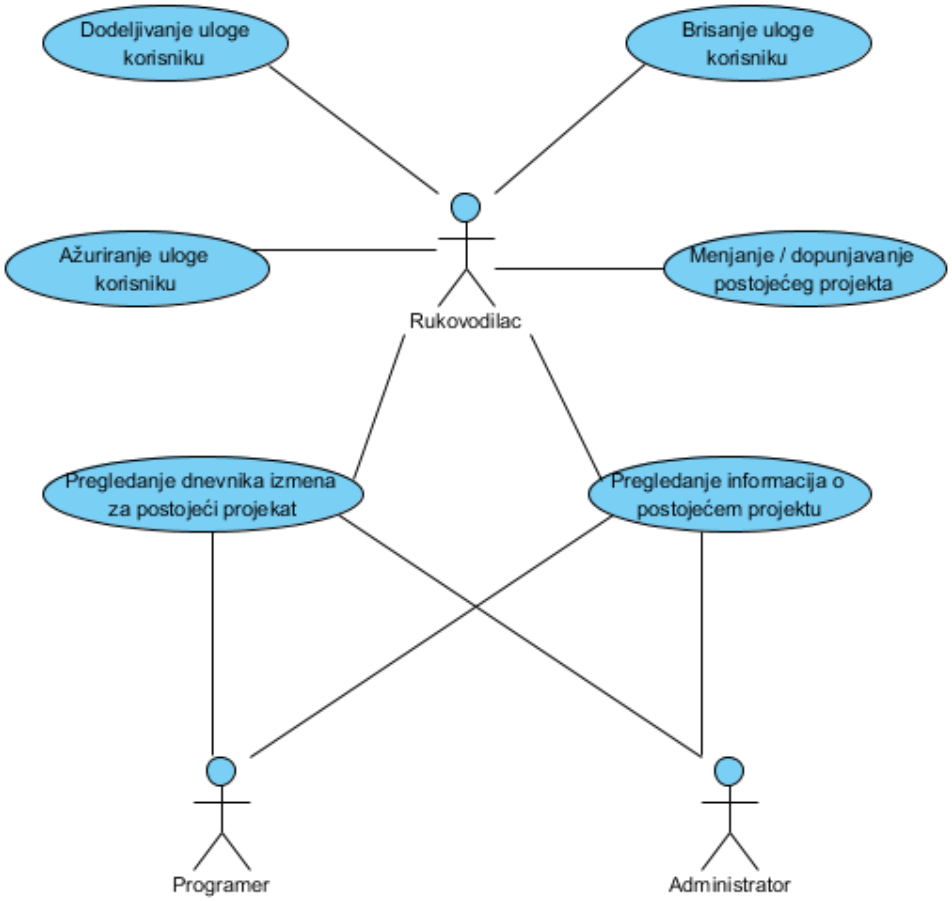
- Korisnik se informisao o detaljima projekta.

Glavni tok:

1. Korisnik bira opciju za pregled svih projekata u sistemu.
2. Korisnik bira projekat iz ponuđene liste svih projekata.
3. Korisnik bira bita opciju za pregledanje detaljnih informacija o projektu.
4. Korisnik pregleda informacije o projektu.
5. Korisnk završava sa pregledom informacija.

Alternativni tokovi:

U svakom koraku korisnik može da odustane od pregleda informacija o projektu.



Slika 7. Dijagram slučajeva upotrebe – Upravljanje projektom

Slučaj upotrebe: Dodeljivanje uloge korisniku

Kratak opis: Svakom korisniku može biti dodeljena neka uloga.
Akteri:

- Rukovodilac

Preduslovi:

- Rukovodilac je prijavljen na sistem i nalazi se na stranici za pregledanje informacija o projektu.

Postuslovi:

- Korisniku je dodeljena uloga.

Glavni tok:

1. Rukovodilac bira opciju za uređivanje korisnika.
2. Rukovodilac bira opciju za dodavanje uloge korisniku.
3. Rukovodilac bira korisnika iz ponuđene liste svih korisnika.
4. Rukovodilac bira projekat iz ponuđene liste projekata.
5. Rukovodilac bira privilegije koje želi da dodeli korisniku. Te privilegije uključuju:
   - dodeljivanje članstva postojećem projektu
   - mogućnost ostavljanja komentara na sve probleme projekta

- mogućnost menjanja informacija svih problema projekta

6. Rukovodilac potvrđuje izmene.
7. Korisnik dobija informaciju elektronskom poštom o dodeljenoj ulozi.

Alternativni tokovi:

U svakom koraku rukovodilac može da odustane od izabrane akcije.

Slučaj upotrebe: Brisanje uloge korisniku

Kratak opis: Svaka dodeljena uloga se može obrisati.

Akteri:

- Rukovodilac

Preduslovi:

- Rukovodilac je prijavljen na sistem.

Postuslovi:

- Uloga je obrisana.

Glavni tok:

1. Rukovodilac bira opciju za uređivanje korisnika.
2. Rukovodilac bira korisnika iz ponuđene liste svih korisnika.
3. Rukovodilac bira projekat iz ponuđene liste projekata koje on rukovodi.
4. Rukovodilac bira opciju za brisanje uloge korisniku.
5. Rukovodilac potvrđuje izmene.
6. Rukovodilac dobija informaciju elektronskom poštom da mu je data uloga obrisana.

Alternativni tokovi:

U svakom koraku rukovodilac može da odustane od izabrane akcije.

Slučaj upotrebe: Ažuriranje uloge korisniku

Kratak opis: Svaka dodeljena uloga korisniku se može promeniti ukoliko korisnik treba da dobije nižu ili višu ulogu.

Akteri:

- Rukovodilac

Preduslovi:

- Rukovodilac je prijavljen na sistem.

Postuslovi:

- Korisniku je ažurirana uloga.

Glavni tok:

1. Rukovodilac bira opciju za uređivanje korisnika.
2. Rukovodilac bira projekat iz ponuđene liste projekata koje on rukovodi.
3. Rukovodilac bira korisnika iz ponuđene liste korisnika.
4. Rukovodilac bira novu ulogu korisnika iz ponuđene liste.
5. Rukovodilac potvrđuje izmene.
6. Korisnik dobija informaciju elektronskom poštom da mu je uloga izmenjena.

Alternativni tokovi:

U svakom od koraka rukovodilac može da odustane od izabrane akcije.

Slučaj upotrebe: Menjanje ili dopunjavanje postojećeg projekta

Kratak opis: Tokom životnog ciklusa projekta, informacije o projektu se mogu izmeniti ili dopuniti.

Akteri:

- Rukovodilac

Preduslovi:

- Rukovodilac je prijavljen na sistem.

Postuslovi:

- Izmenjen sadržaj projekta je sačuvan.

Glavni tok:

1. Rukovodilac bira opciju za pregled svih projekata u sistemu.
2. Iz prikazane liste rukovodilac bira projekat koji želi da izmeni ili dopuni.
3. Rukovodilac menja sadržaj projekta.
   3.1 Rukovodilac bira polje koje želi da izmeni ili dopuni.
   3.2 Rukovodilac menja dato polje unošenjem nove vrednosti.
   3.3 Rukovodilac prelazi na korak 3.1 ili je odlučio da završi sa izmenama.
4. Rukovodilac bira opciju za čuvanje podataka koja će izmenjen projekat sačuvati u bazi.

Alternativni tokovi:

2.1 Rukovodilac nema odgovarajuće privilegije za izbrani projekat. U tom slučaju mu se pokazuje odgovarajuća poruka o grešci.

3.3.1 Rukovodilac je završio sa izmenama i prelazi na korak 4.

Slučaj upotrebe: Pregledanje dnevnika izmena za postojeći projekat

Kratak opis: Sistem omogućava pregledanje dnevnika izmena za postojeći projekat.

Akteri:

- Rukovodilac, Programer, Administrator

Preduslovi:

- Korisnik je prijavljen na sistem.

Postuslovi:

- Korisnik se informisao o detaljima izmena određenog projekta.

Glavni tok:

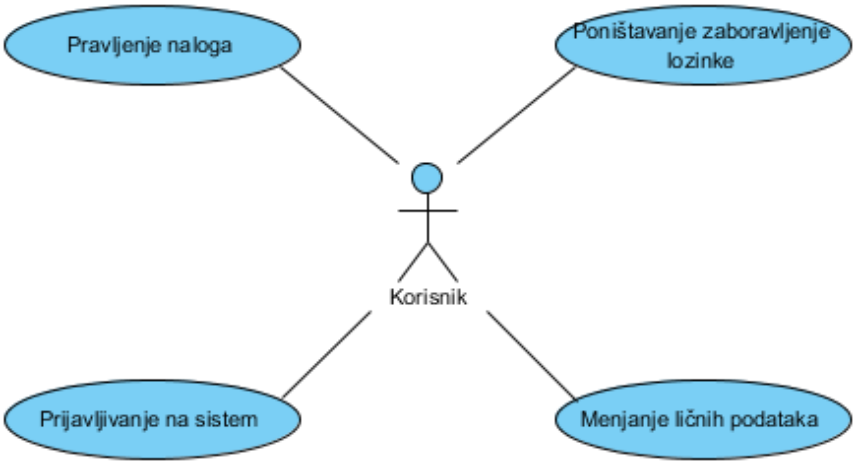
1. Korisnik bira opciju za pregled svih projekata u sistemu.
2. Iz prikazane liste korisnik bira projekat čiji dnevnik izmena želi da vidi.
3. Korisnik bira opciju za pregledanje dnevnika izmena.
4. Korisnik pregleda dnevnik izmena.

Alternativni tokovi:

Nema alternativnih tokova.

3.3.4 Upravljanje korisničkim nalogom

Ova grupa slučajeva upotrebe se odnosi na sve one aktivnosti koje se odnose na upravljanje korisničkim nalogom: *Pravljenje naloga*, *Poništavanje zaboravljene lozinke*, *Prijavljivanje na sistem* i *Menjanje ličnih podataka* (slika 8). Ovi slučajevi upotrebe se odnose na sve korisnike sistema.



Slika 8. Dijagram slučajeva upotrebe – Upravljanje korisničkim računom

Slučaj upotrebe: Pravljenje naloga

Kratak opis: Svaki korisnik mora posedovati nalog kako bi koristio sistem.
Akteri:

- Korisnik (Tester, Programer, Rukovodilac, Administrator)

Preduslovi:

- Nema.

Postuslovi:

- Korisnik je uspešno napravio nalog.

Glavni tok:

1. Korisnik podnosi zahtev za pravljenje naloga.
2. Korisnik dobija formular za unos osnovnih informacija.
3. Korisnik unosi tražene informacije.
4. Korisnik dobija elektronsku poštu sa linkom za aktiviranje naloga, kako bi potvrdio da poseduje unetu adresu elektronske pošte.
5. Klikom na dobijeni link korisnik uspešno završava proces pravljenja naloga.

Alternativni tokovi:

U svakom od koraka, korisnik može da odustane od pravljenja naloga.

Slučaj upotrebe: Poništavanje zaboravljene lozinke

Kratak opis: U slučaju da zaboravi lozinku, korisnik može poništiti staru i uneti novu lozinku.

Akteri:

- Korisnik (Tester, Programer, Rukovodilac, Administrator)

Preduslovi:

- Korisnik poseduje već napravljen nalog.

Postuslovi:

- Korisnik je uspešno promenio lozinku.

Glavni tok:

1. Korisnik podnosi zahtev za promenu lozinke.
2. Korisnik dobija elektronsku poštu u kojoj je link za poništavanje lozinke.
3. Korisnik klikom na dobijeni link dolazi na stranicu za promenu lozinke.
4. Korisnik unosi novu lozinku.
5. Korisnik potvrđuje unos i nova lozinka postaje aktivna.

Alternativni tokovi:

U slučaju greške u bilo kom koraku, korisnik ponavlja isti proces.

Slučaj upotrebe: Prijavljivanje na sistem

Kratak opis: Korisnici moraju biti prijavljeni na sistem kako bi mogli da ga koriste.

Akteri:

- Korisnik (Tester, Programer, Rukovodilac, Administrator)

Preduslovi:

- Korisnik poseduje već napravljen nalog.

Postuslovi:

- Korisnik je prijavljen na sistem ili je dobio poruku o neuspešnom prijavljivanju.

Glavni tok:

1. Korisnik podnosi zahtev za prijavljivanje na sistem.
2. Korisnik dobija formular za unos korisničkog imena i lozinke.
3. Korisnik unosi korisničko ime i lozinku.
4. Korisnik dobija informaciju da je uspešno prijavljen.
5. Korisniku se prikazuje početna stranica.

Alternativni tokovi:

4.1 Greška tokom prijavljivanja. Korisnik nije ispravno uneo podatke.

4.1.1 Unosi se informacija o neuspešnom prijavljivanju u dnevnik prijavljivanja.

Više o bezbednosnim aspektima prilikom prijavljivanja na sistem se može pogledati u dodacima (odeljak 8.4).

Slučaj upotrebe: Menjanje ličnih podataka

Opis: Svi korisnici sistema imaju mogućnost izmene ličnih podataka.

Akteri:

- Korisnik (Tester, Programer, Rukovodilac, Administrator)

Preduslovi:

- Korisnik je prijavljen na sistem.

Postuslovi:

- Lični podaci korisnika su izmenjeni i sačuvani.

Glavni tok:

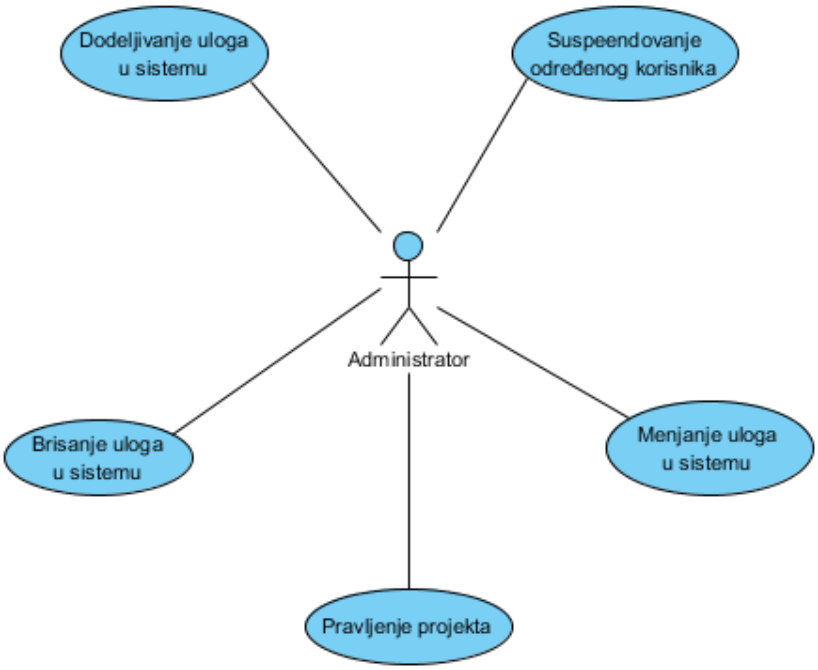
1. Korisnik bira opciju za izmenu ličnih podataka.
2. Korisnik menja podatke.
   2.1 Korisnik bira podatak koji želi da izmeni.
   2.2 Korisnik menja podatak unosom nove vrednosti.
   2.3 Korisnik se vraća na korak 2.1 ili je odlučio da završi sa izmenama.
3. Korisnik potvrđuje izmenu.
4. Korisnik dobija informaciju o uspešnoj izmeni.

Alternativni tokovi:

2.2.1 Korisnik je završio izmene i prelazi na korak 3.

3.3.5 Administriranje sistema

Ova grupa slučajeva upotrebe odnosi se na sve one aktivnosti koje se ne tiču rada sa problemima i projektima. već se pre svega odnosi na neku vrstu administriranja sistema: *Dodeljivanje uloga u sistemu*, *Menjanje uloga u sistemu*, *Brisanje uloga u sistemu*, *Suspendovanje određenog korisnika* i *Pravljenje projekata*. Pregled ovih aktivnosti se može videti na slici 9.



Slika 9. Dijagram slučajeva upotrebe – Administriranje sistema

Slučaj upotrebe: Dodeljivanje uloga u sistemu

Opis: Pravljenjem naloga korisnik dobija ulogu testera. Više privilegije mu se dodeljuju od strane odgovarajuće nadležne osobe.

Akteri:

- Administrator

Preduslovi:

- Administrator je prijavljen na sistem.

Postuslovi:

- Korisniku je dodeljena nova uloga.

Glavni tok:

1. Administrator bira opciju za uređivanje uloga u sistemu.
2. Administrator bira opciju za dodavanje nove uloge.
3. Iz ponuđene liste korisnika, administrator bira korisnika kome želi da dodeli ulogu.
4. Iz ponuđene liste uloga, administrator bira odgovarajuću ulogu.
5. Iz ponuđene liste projekata, administrator bira projekat.
6. Administrator potvrđuje izmene.
7. Korisnik dobija informaciju o tome elektronskom poštom.

Alternativni tokovi:

U svakom koraku, administrator može da odustane od započetog procesa.

Slučaj upotrebe: Menjanje uloga u sistemu

Opis: Dodeljene uloge se mogu menjati.

Akteri:

- Administrator

Preduslovi:

- Administrator je prijavljen na sistem.

Postuslovi:

- Korisniku je izmenjena uloga.

Glavni tok:

1. Administrator bira opciju za uređivanje uloga u sistemu.
2. Administrator bira opciju za menjanje uloga u sistemu.
3. Iz ponuđene liste korisnika, administrator bira korisnika kome želi da izmeni ulogu.
4. Iz ponuđene liste uloga, administrator bira ulogu koju želi da izmeni.
5. Administrator potvrđuje izmene.
6. Korisnik dobija informaciju o tome elektronskom poštom.

Alternativni tokovi:

U svakom koraku, administrator može da odustane od započetog procesa.

Slučaj upotrebe: Brisanje uloga u sistemu

Opis: Svaka dodeljena uloga se može obrisati.

Akteri:

- Administrator

Preduslovi:

- Administrator je prijavljen na sistem.

Postuslovi:

- Korisniku je obrisana uloga.

Glavni tok:

1. Administrator bira opciju za uređivanje uloga u sistemu.
2. Administrator bira opciju za brisanje uloge.
3. Iz ponuđene liste, administrator bira korisnika.
4. Iz ponuđene liste, administrator bira ulogu koju želi da obriše.
5. Administrator potvrđuje izmene.
6. Korisnik dobija informaciju o tome elektronskom poštom.

Alternativni tokovi:

U svakom koraku, administrator može da odustane od započetog procesa.

Slučaj upotrebe: Suspendovanje određenog korisnika

Opis: Korisnik može biti suspendovan na određeno vreme ako ne koristi sistem na propisan način.

Akteri:

- Administrator

Preduslovi:

- Administrator je prijavljen na sistem.

Postuslovi:

- Korisnik je suspendovan na određeno vreme.

Glavni tok:

1. Administrator bira opciju za uređivanje uloga u sistemu
2. Administrator bira opciju za suspendovanje korisnika.
3. Iz ponuđene liste, administrator bira korisnika koga želi da suspenduje.
4. Administrator bira vremenski period na koji želi da ga suspenduje.
5. Administrator potvrđuje izmene.
6. Korisnik koji je suspendovan dobija informaciju o tome elektronskom poštom.

Alternativni tokovi:

U svakom koraku, administrator može da odustane od datog procesa.

Slučaj upotrebe: Pravljenje novog projekta

Kratak opis: Pravljenje novog projekta i unošenje odgovarajućih informacija o njemu.

Akteri:

- Administrator

Preduslovi:

- Administrator je prijavljen na sistem.

Postuslovi:

- Projekat je uspešno napravljen.

Glavni tok:

1. Administrator bira opciju za pregled projekata.
2. Administrator bira opciju za pravljenje novog projekta.
3. Administrator unosi informacije o projektu.
   3.1 Administrator bira polje koje želi da popuni.

3.2 Administrator menja izabrano polje unošenjem nove vrednosti ili odabirom stavke iz padajućeg menija.

3.3 Administrator prelazi na korak 3.1 ili je odlučio da završi sa izmenama.

4. Administrator bira opciju za čuvanje podataka koja će novi projekat sačuvati u bazi.

Alternativni tokovi:

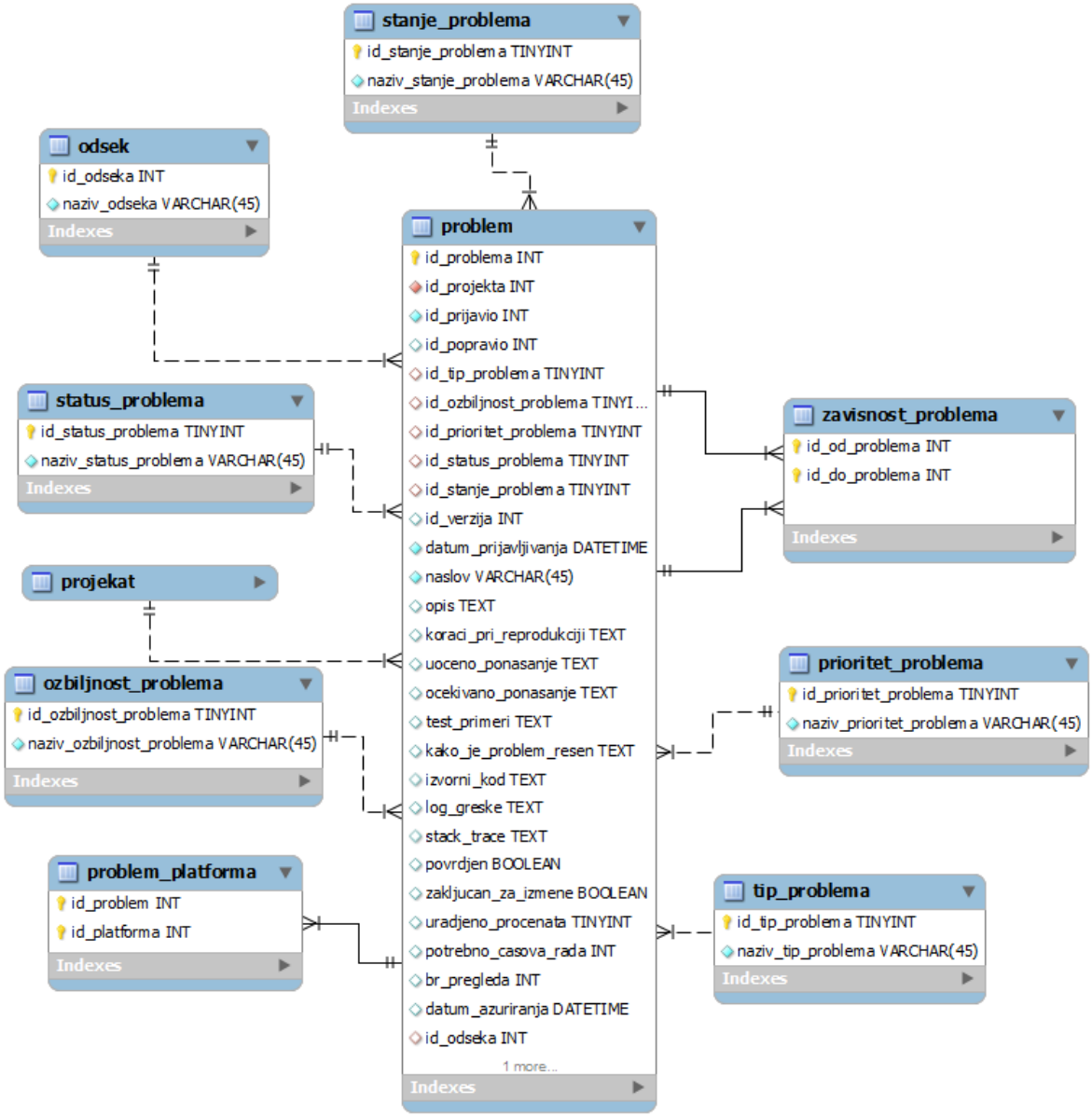
3.3.1 Administrator je završio sa unosom i prelazi na korak 4.

4.1 Administrator odustaje od pravljenja novog projekta.

4. Projektovanje

4.1 Projektovanje baze podataka

Analizom slučajeva upotrebe, napravljen je *prošireni model entiteta i odnosa* (engl. *Extended Entity Relationship model*, *EER*). Model će biti prikazan iz delova koji će kasnije biti sklopljeni u celinu.

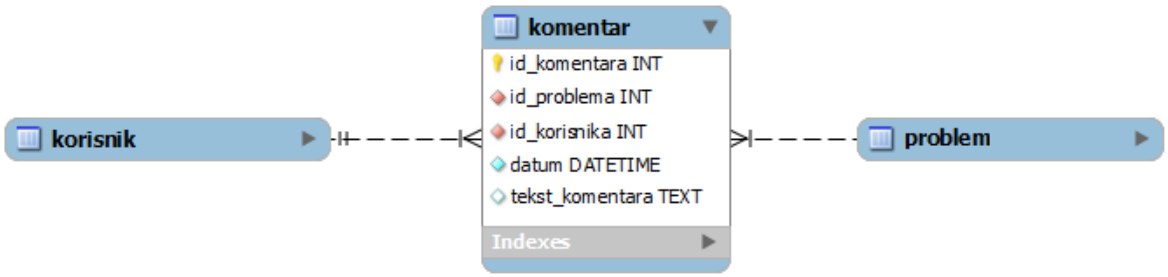


Slika 10. Entitet *problem* sa pomoćnim entitetima

Glavni entitet u sistemu je *problem*. Svaki problem može imati svoj status, ozbiljnost, tip prioritet i stanje. Oni nisu fiksirani i mogu se menjati i prilagođavati konkretnim potrebama. Zbog toga su oni izdvojeni u posebne entitete: *status_problema*, *ozbiljnost_problema*, *tip_problema*, *prioritet_problema* i *stanje_problema*. Problem može da zavisi od nekog drugog problema. Međusobne zavisnosti problema beleže se u entitetu *zavisnost_problema*. Problem se može javljati na više platformi, pa se informacija o tome čuva u entitetu

problem_platforma. Informacije o vezi entiteta *problem* sa entitetom *korisnik* će biti izložene zasebno u nastavku odeljka. Opisana struktura entiteta može se videti na slici 10.

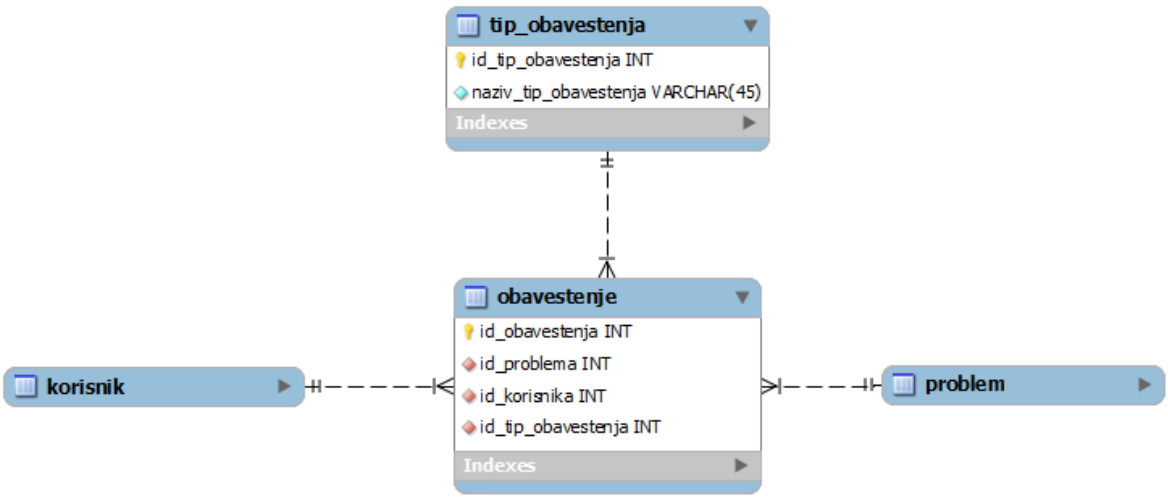
U sistemu je moguće ostaviti komentar na svaki problem. Informacije o komentarima se čuvaju u posebnom entitetu *komentar*. Ovaj entitet je prikazan na slici 11.



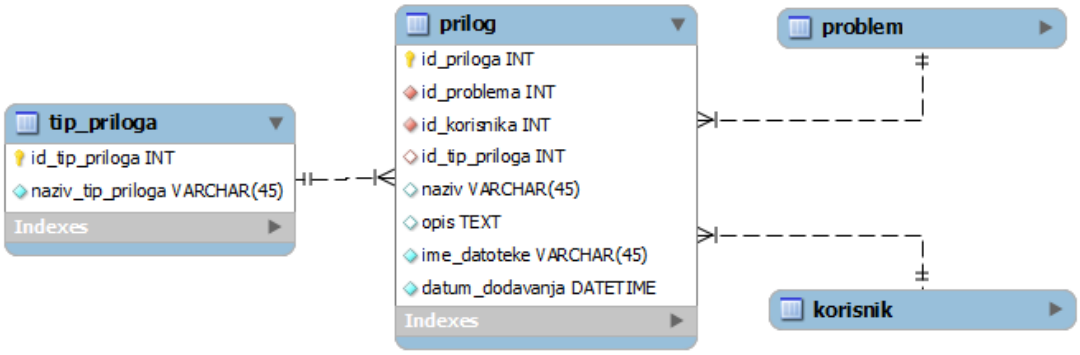
Slika 11. Entitet *komentar*

Korisnik se može pretplatiti na obaveštenja za svaki pojedinačan problem. Kada neki drugi korisnik sistema promeni neku informaciju o problemu ili doda komentar, korisnik koji se pretplatio na taj problem dobiće odgovarajuće obaveštenje. Informacije o obaveštenjima se čuvaju u zasebnom entitetu *obavestenja*, a različiti tipovi (vrste) obaveštenja se čuvaju u posebnom entitetu *tip_obavestenja*. Razlog izdvajanja tipa obaveštenja u poseban entitet je da bi se omogućila fleksibilnost prilikom izmene i dodavanja novih tipova obaveštenja. Opisanu strukturu entiteta je moguće videti na slici 12.

Za svaki problem je moguće ostaviti neki prilog (engl. *Attachement*). Informacije o prilozima se čuvaju u entitetu *prilog* a različiti tipovi priloga u zasebnom entitetu *tip_priloga*. Slično kao i za obaveštenja, izdvajanje tipa priloga u poseban entitet je urađeno zbog fleksibilnosti, tj. lakše izmene i dopune novim tipovima priloga. Opisanu strukturu entiteta je moguće videti na slici 13.

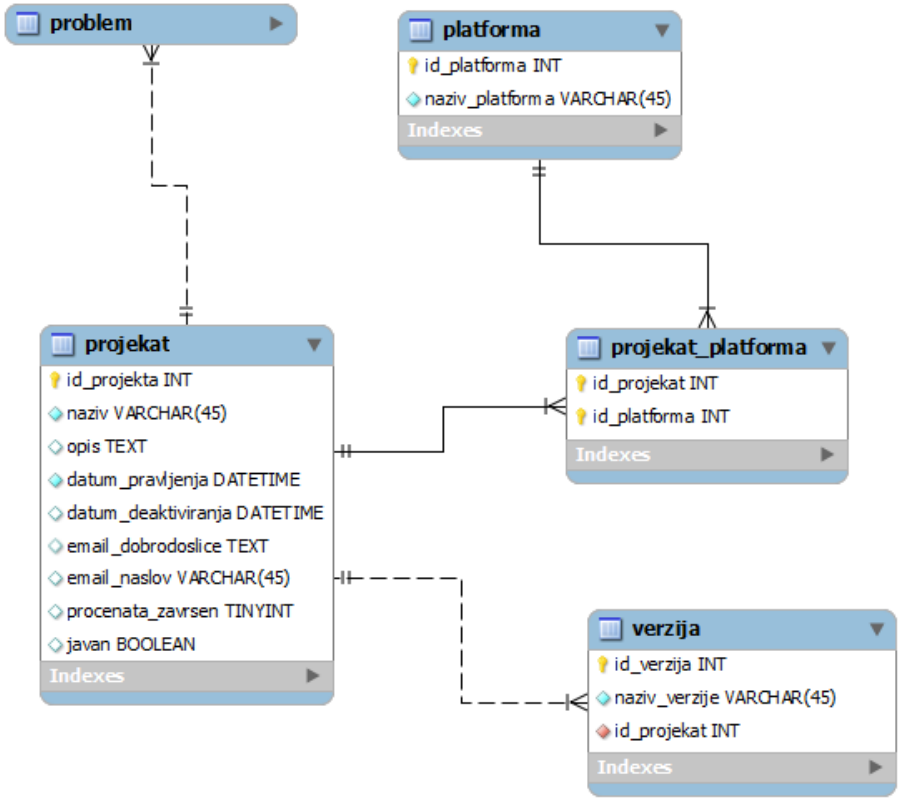


Slika 12. Entitet *obavestenje*



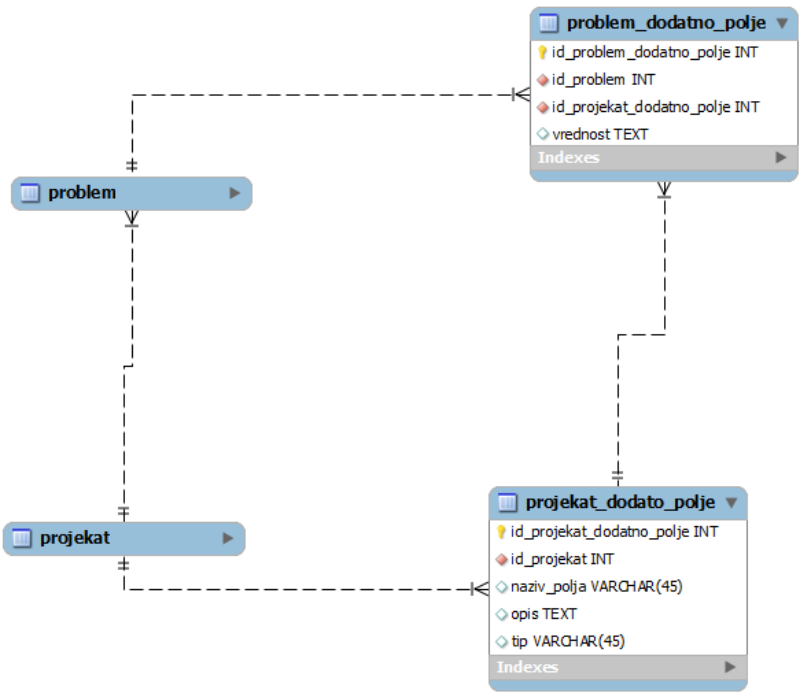
Slika 13. Entitet *prilog*

Problemi pripadaju nekom projektu. Projekat se može realizovati na više platformi pa je zbog toga potrebna informacija o mogućim platformama. Informacije o različitim platformama se čuvaju u entitetu *platforma*, a informacija o tome za koje platforme se projekat razvija u asocijativnom entitetu *projekat_platforma*. Projekat može imati više verzija pa je potrebno čuvati i razne verzije projekata. Ta informacija se čuva u entitetu *verzija*. Na slici 14 je prikazan opisani skup entiteta.



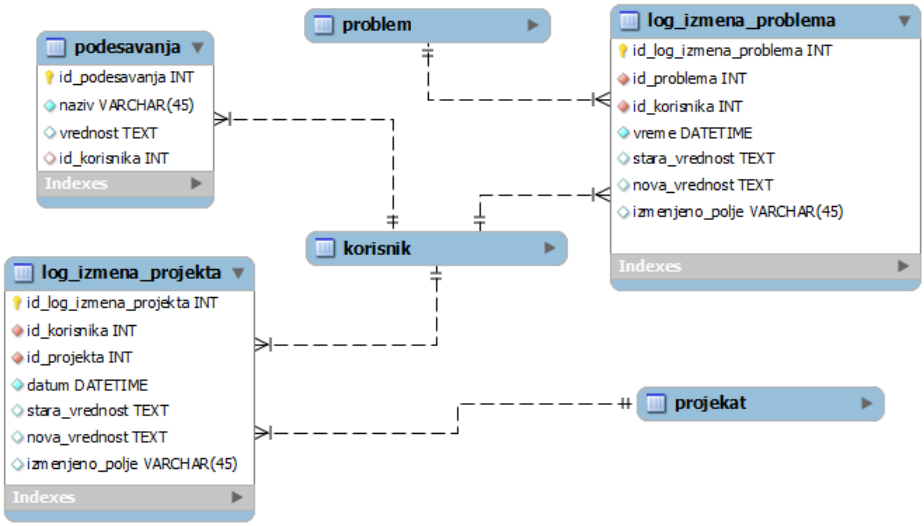
Slika 14. Entitet *projekat* sa pomoćnim entitetima

Sistem sadrži mogućnost dodavanja novih polja. Na nivou baze podataka, ovaj problem se rešava uvođenjem novog entiteta *projekat_dodatno_polje* i on sadrži informaciju o tome koji projekat sadrži koja dodatna polja. Za čuvanje vrednosti dodatnih polja, uvodi se još jedan entitet *problem_dodatno_polje*. Opisani skup entiteta je prikazan na slici 15.



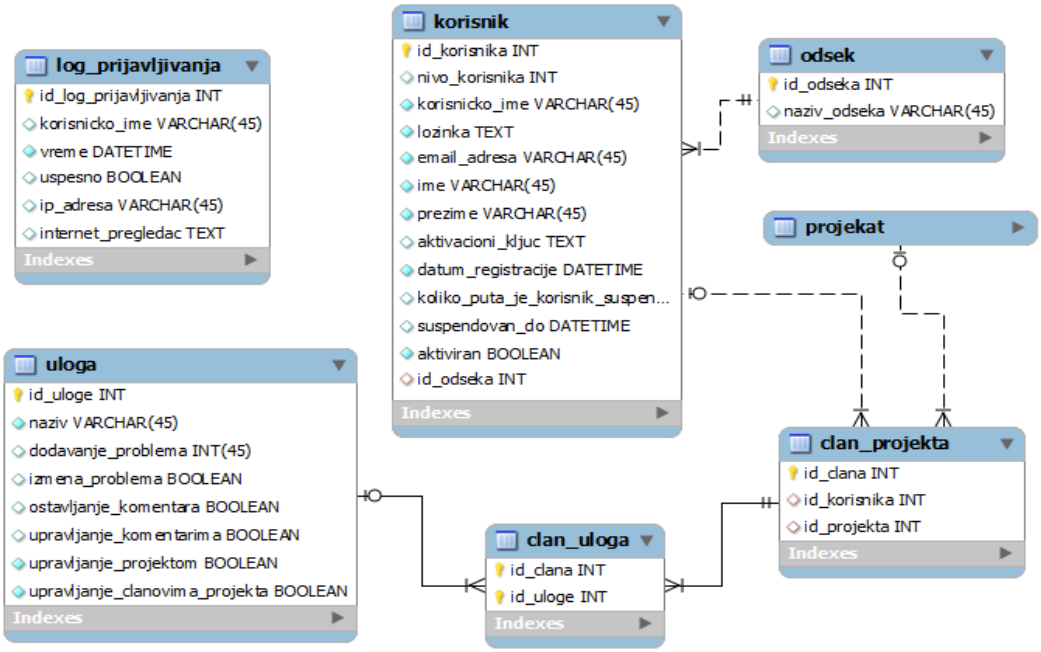
Slika 15. Entitet *dodatno_polje* sa pomoćnim entitetima

U sistemu se vode informacije o svim izmenama problema i projekta. Ove informacije se čuvaju u entitetima *log_izmena_problema* i *log_izmena_projekta*. Pored toga, ovoj grupi biće priključen još jedan nezavisan entitet *podesavanja*. Njegova uloga je da čuva razna podešavanja vezana za ceo sistem. Opisanu strukturu entiteta je moguće videti na slici 16.

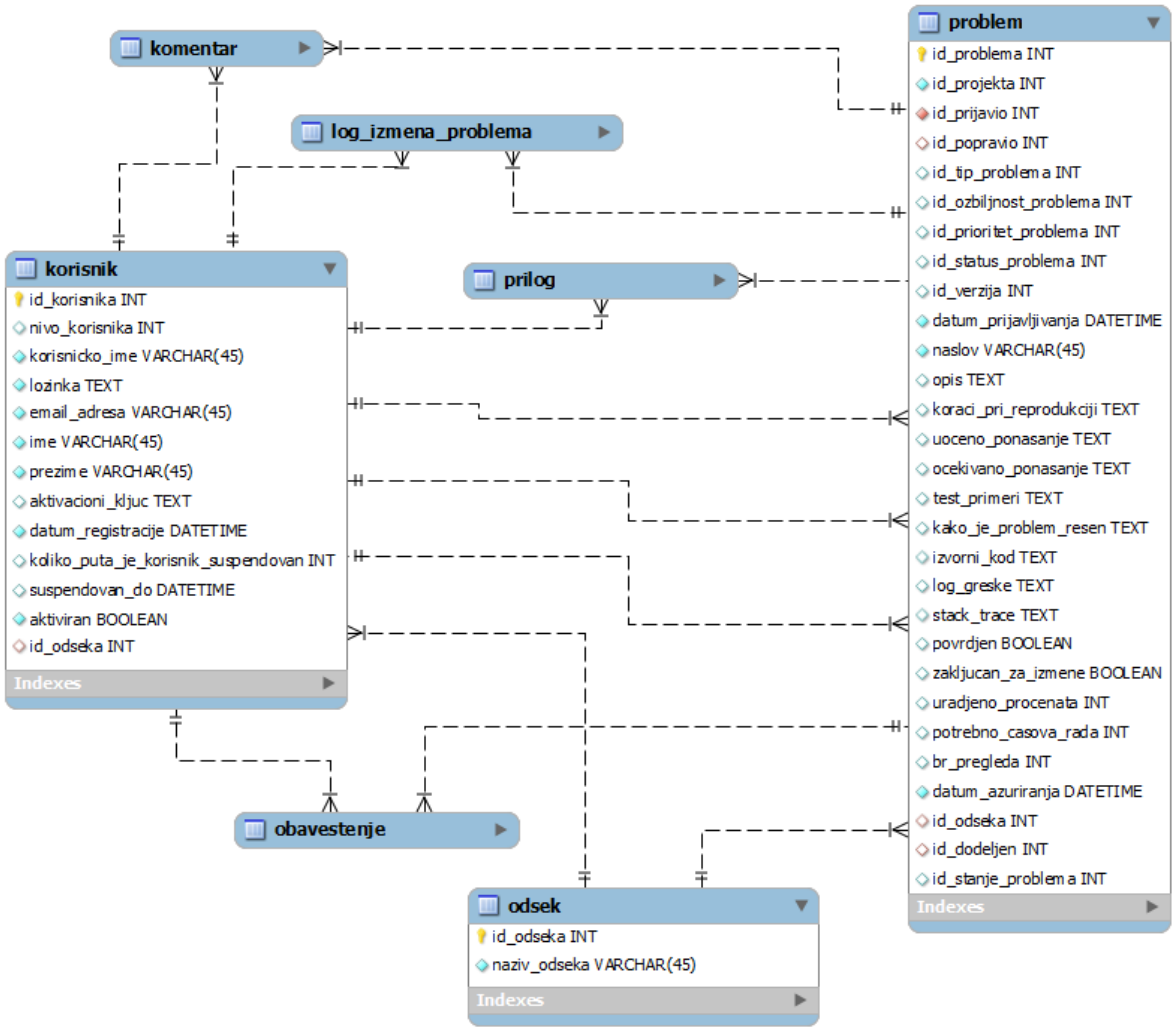


Slika 16. Entiteti *log_izmena_problema, log_izmena_projekta* i *podesavanja*

Korisnici u sistemu su predstavljeni entitetom *korisnik*. Različite uloge korisnika se čuvaju u entitetu *uloga*. Korisnik može biti član nekog projekta. Informacije o tome čuvaju se u entitetu *clan_projekta*. Svaki korisnik dobija svoje uloge u konkretnom projektu. Konkretne uloge članova projekta se beleže u asocijativnom entitetu *clan_uloga*. Pored toga, beleže se i sva prijavljivanja korisnika na sistem za šta je namenjen entitet *log_prijavljivanja*. Opisana struktura entiteta može se videti na slici 17



Slika 17 Entiteti *korisnik* i *uloge*



Slika 18 Veza entiteta *korisnik* i *problem*

Veze između entiteta *korisnik* i *problem* su izdvojene u poseban dijagram prikazan na slici 18. Za svaki problem se vodi informacija o tome ko ga je prijavio. Ta informacija se beleži kao strani ključ na tabelu *korisnik*. Informacija o tome kom korisniku je dodeljen koji problem se beleži kao strani ključ na tabelu *korisnik*.

Sada se ovi delovi mogu sklopiti u jednu celinu. Pre nego što bude prikazan celokupan dijagram biće ukratko opisan svaki entitet koji se u njemu pojavljuje.

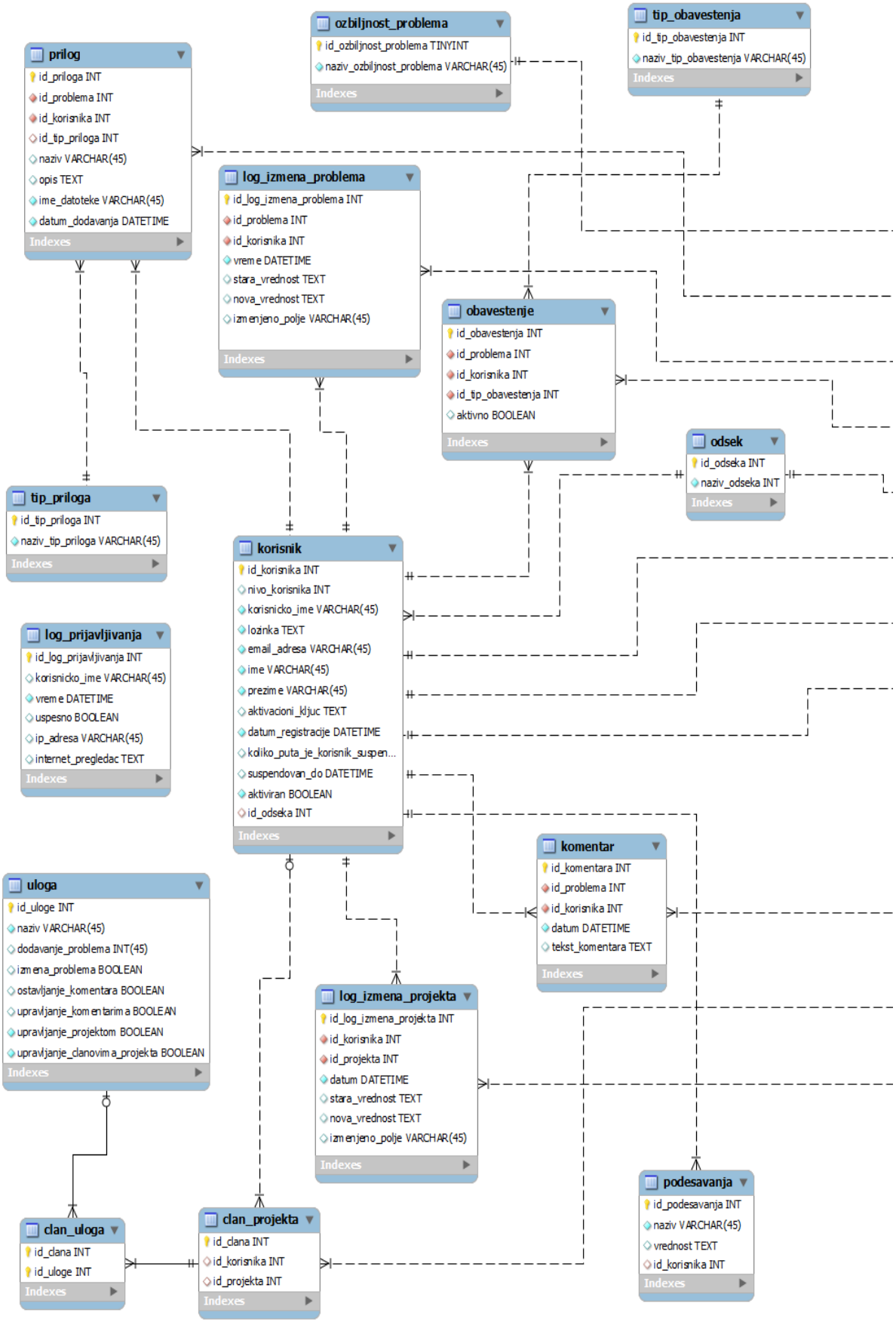
1. **problem** – entitet koji sadrži informacije o svim problemima. Više informacija o problemu može se naći u odeljku 3.2.
2. **korisnik** – sadrži informacije o svim korisnicima sistema. Trenutno postoje četiri različita tipa korisnika (tester, programer, rukovodilac ili administrator). Više o korisnicima može se naći u odeljku 3.1.
3. **projekat** – sadrži informacije o svim projektima u sistemu. U okviru projekta se prijavljuju problemi.
4. **komentar** – za svaki problem je moguće ostavljati komentare. Komentare ostavljaju korisnici sistema koji žele da naznače neke dodatne informacije o problemu ili žele da diskutuju o njemu.
5. **prilog** – za svaki problem je moguće priložiti neki prilog. Prilog može biti različitog tipa (slika, dijagram, dokument, ...).
6. **uloga** – ovaj entitet sadrži sve uloge koje korisnik sistema može imati, kao i privilegije koje su dodeljene tim ulogama. Uloge se mogu menjati i dopunjavati.
7. **tip_problema** – ovaj entitet sadrži informacije o svim tipovima problema koji se mogu javljati na sistemu (*bag*, *rizik*, *unapređenje*, *pitanje* i *modifikacija*). Postojeći tipovi problema se mogu menjati kao i dodavati novi.
8. **ozbiljnost_problema** – (engl. *Severity*) predstavlja entitet koji označava koliko je problem ozbiljan. Ponuđene vrednosti su: *kritična*, *visoka*, *srednja*, *niska* i *bez vrednosti*, ali se one mogu menjati kao i dodavati nove.
9. **prioritet_problema** – (engl. *Priority*) predstavlja entitet koji označava prioritet problema. Ponuđene vrednosti su: *kritičan*, *visok*, *srednji*, *nizak* i *bez vrednosti*, ali se one mogu menjati kao i dodavati nove.
10. **status_problema** – (engl. *Status*) predstavlja entitet koji sadrži moguće statuse jednog problema. Ponuđene vrednosti su: *nepotvrđen*, *potvrđen*, *nije validan*, *duplikat*, *dodeljen*, *ispravljen*, *ne može biti popravljen*, *verifikovan* i *ponovo otvoren*. Ponuđene vrednosti se mogu menjati kao i dodavati nove.
11. **stanje_problema** – (engl. *State*) predstavlja entitet koji sadrži sva moguća stanja problema. Ponuđene vrednosti su: *otvoren*, *zatvoren* i *na čekanju*, ali se one mogu menjati po potrebi kao i dodavati nove.
12. **tip_priloga** – ovaj entitet sadrži informacije o svim tipovima priloga koji se mogu priložiti uz problem (slika, video, dijagram, zvuk ili dokument). Postojeće vrednosti se mogu menjati ili dodavati nove.
13. **log_izmena_problema** – u ovom entitetu se beleže sve izmene koje su napravljene na problemu, kao i korisnik koji ih je napravio.
14. **log_prijavljivanja** – u ovom entitetu se beleže sva prijavljivanja na sistem (uspešna i neuspešna). Na ovaj način se vodi evidencija o eventualnim pokušajima upada na sistem.
15. **log_izmena_projekta** – u ovom entitetu se beleže sve izmene koje su napravljene u projektu kao i korisnik koji ih je napravio.
16. **dodatno_polje** – pored već standardnih polja, za svaki projekat se mogu dodati još neka dodatna polja. Ona mogu biti specifična samo za taj projekat.

17. **projekat_dodatno_polje** – ovaj entitet sadrži informaciju o tome koje dodatno polje pripada kom projektu kao i informaciju o tom dodatnom polju (naziv, opis i tip polja).
18. **problem_dodatno_polje** – ovaj agregirani entitet sadrži odnos između entiteta *problem* i *projekat_dodatno_polje*. On čuva vrednost dodatnog polja za konkretni problem.
19. **platforma** – za svaki problem se može označiti na kojoj platformi se javlja. Ovaj entitet sadrži informacije o svim mogućim platformama za određeni projekat.
20. **verzija** – za svaki problem se može naznačiti i u kojoj verziji projekta se javlja. U ovom entitetu se beleže sve moguće verzije datog projekta.
21. **tip_obavestenja** – sadrži sve moguće tipova obaveštenja. Korisnik može da se pretplati preko elektronske pošte na komentare, na izmene problema ili na oba.
22. **projekat_platforma** – kako je kardinalnost odnosa između entiteta *projekat* i *platforma* „više prema više“, dodaje se asocijativni entitet *projekat_platforma* koji opisuje koji projekat se radi za koje platforme.
23. **clan_projekta** – ovaj entitet sadrži informacije o članovima projekta. Član projekta je korisnik koji ima neku ulogu na nekom projektu.
24. **obavestenje** – ovaj entitet sadrži informacije o korisničkim pretplatama na različite tipove obaveštenja. Korisnik se pretplaćuje na obaveštenja za svaki problem posebno.
25. **zavisnost_problema** – ovaj asocijativni entitet opisuje međusobnu zavisnost problema, tj. koji problem zavisi od kog problema.
26. **odsek** – ovaj entitet sadrži informacije o svim odsecima koji se nalaze u kompaniji koja koristi ovaj sistem. Odsek (engl. *Department*) predstavlja jedno odeljenje ili grupu ljudi koji obavljaju srodnu vrstu posla. Npr. razvoj, rukovodstvo, prodaja, podrška, administracija.
27. **problem_platforma** – pošto se jedan problem može javljati na više platformi, informacije o tome se čuvaju u entitetu *problem_platforma*.
28. **clan_uloga – u** ovom entitetu se čuvaju informacije o ulogama svakog člana. Svaki član može imati više različitih uloga.
29. **podesavanja –** u ovom entitetu čuvaju se podešavanja o samom sistemu i podešavanja za same korisnike.

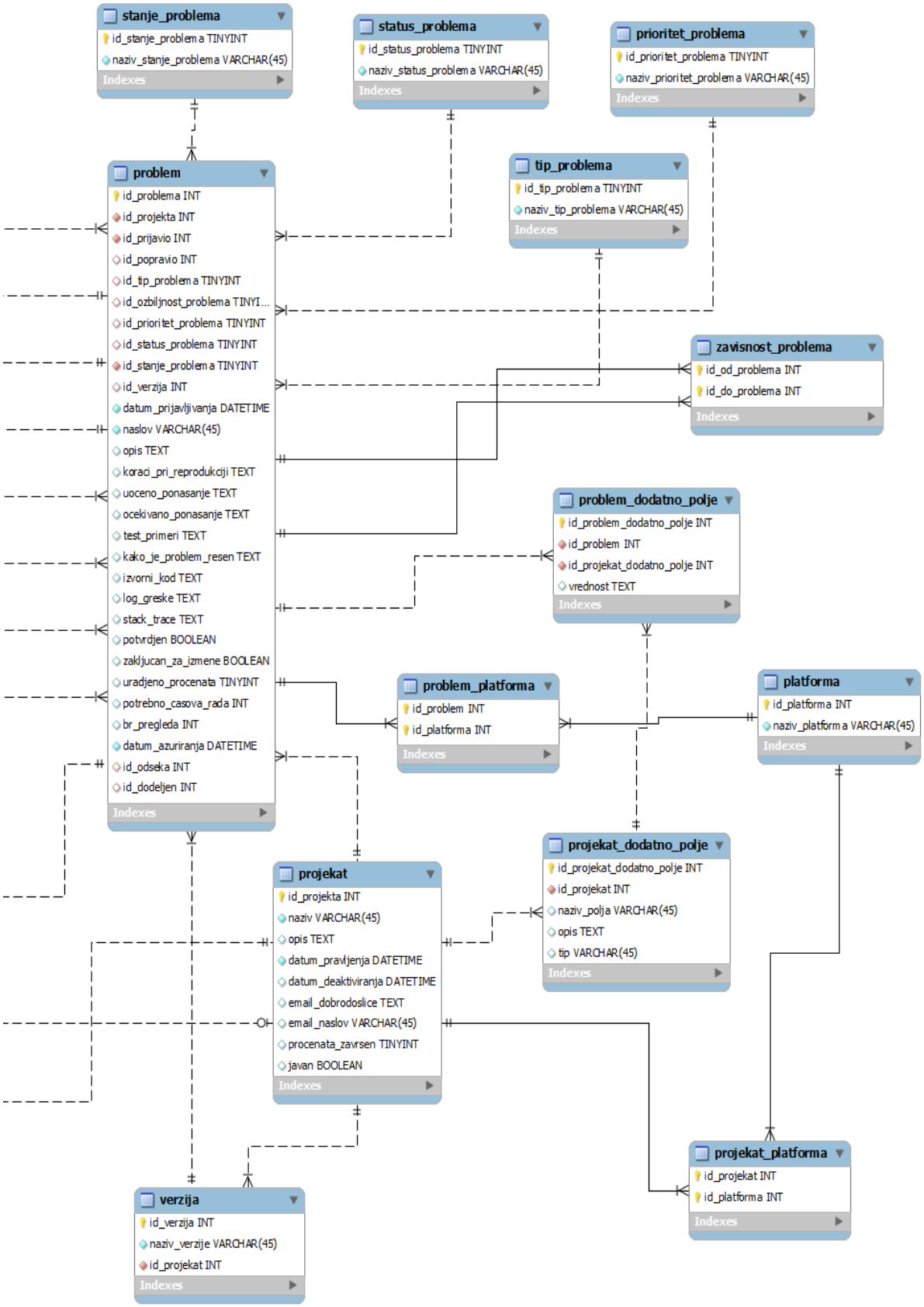
Nešto kasnije, u odeljku 8.1, biće detaljnije opisan svaki entitet. Celokupna shema baze podataka je prikazana je na slikama 19 i 20.

4.2 Korisnički interfejs

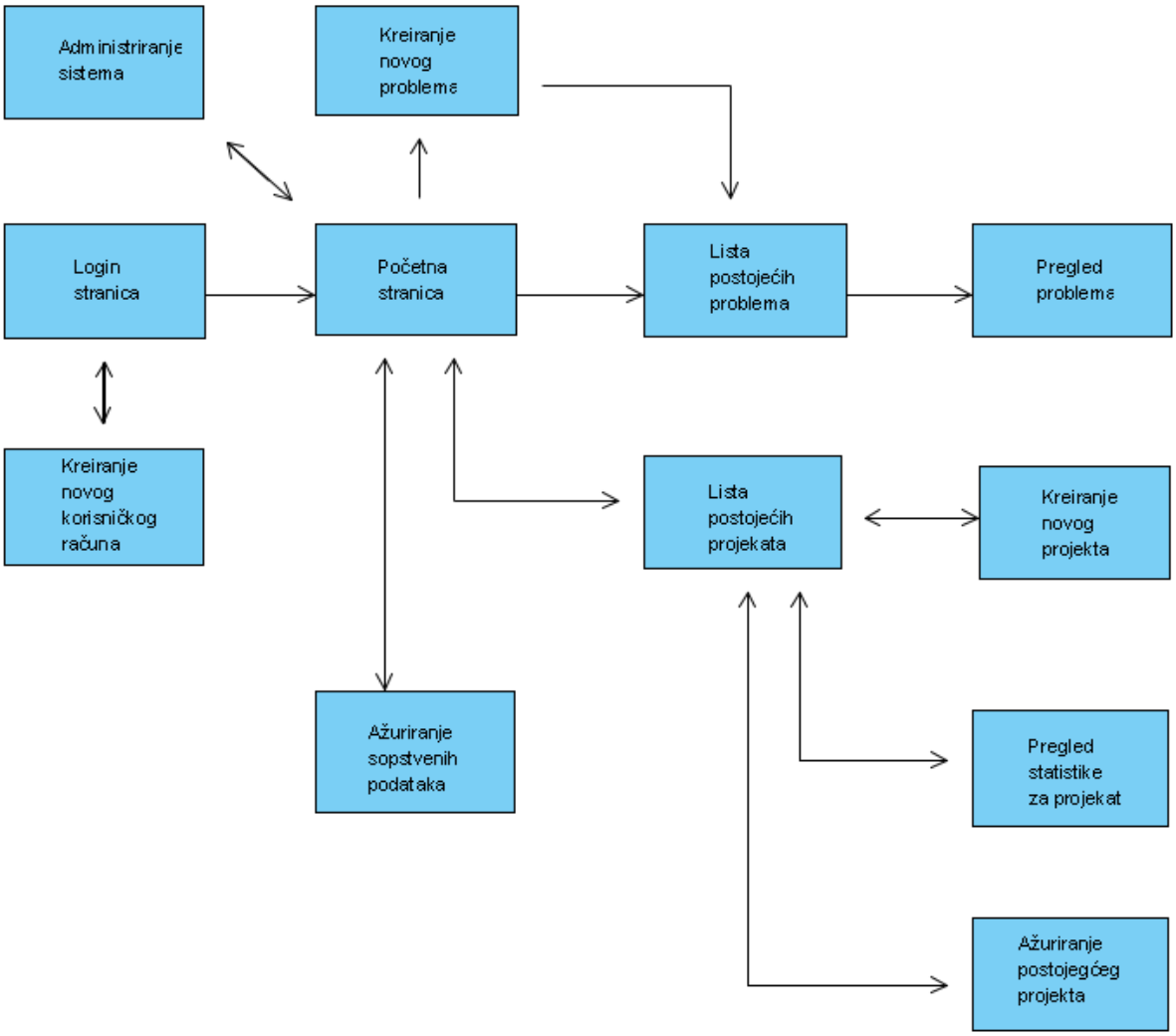
Korisnički interfejs je dizajniran sa ciljem da bude što jednostavniji i intuitivniji. Povezanost stranica koje čine korisnički interfejs sistema za upravljanje problemima u razvoju softvera prikazana je dijagramom zavisnosti stranica, koji se može videti na slici 21. Svaka stranica je prikazana jednim pravougaonikom, a strelica označava vezu između dve stranice, tj. da se sa jedne stranice može preći na drugu. Primer sa slikama korisničkog interfejsa se nalazi u dodacima (odeljak 8.2) .



Slika 19: Shema baze podataka, deo 1



Slika 20: Shema baze podataka, deo 2



Slika 21: Dijagram zavisnosti stranica

5. Implementacija

5.1 Razvojno okruženje

Za razvoj i testiranje sistema za upravljanje problemima u razvoju softvera korišćen je veb server *XAMPP*. *XAMPP* je akronim koji označava sledeće:

X – kao oznaka za podršku za više platformi
A – *Apache HTTP Server*
M – *MySQL*
P – *PHP*
P – *Perl*

Apache HTTP server je danas široko korišćen i podržan je na skoro svim platformama. *Apache HTTP* server je softver otvorenog koda i nalazi se pod licencom fondacije *Apache*.

Za bazu podataka se koristi *MySQL* koji je danas široko korišćen u aplikacijama zasnovanim na vebu. *MySQL* je zasnovan na relacionom modelu koji je predstavio *E.F.Codd*.

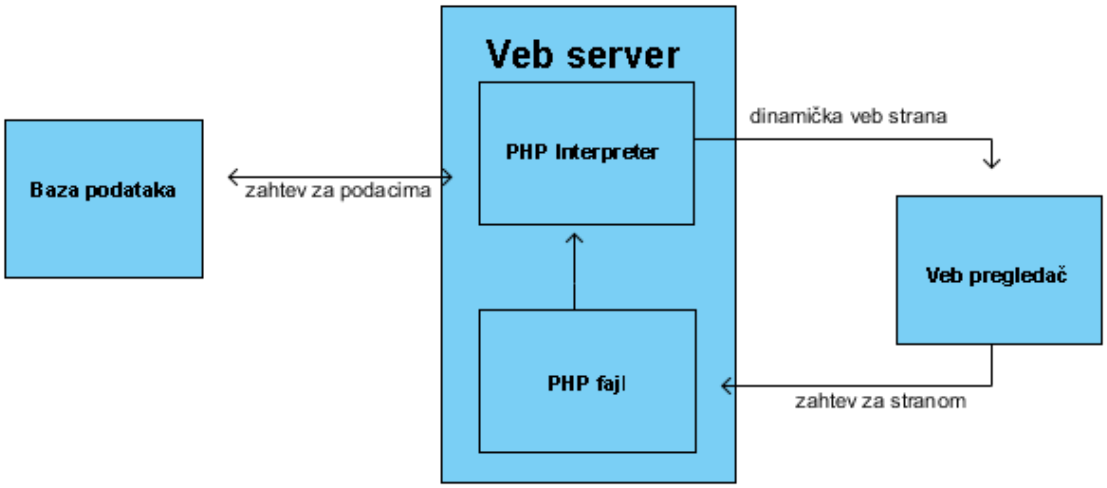
XAMPP je namenjen veb dizajnerima i veb programerima, kao razvojni alat za testiranje njihovih aplikacija zasnovanih na vebu bez potrebe za postavljanjem na veb server. Za testiranje aplikacije korišćeni su veb pregledači *Mozilla Firefox* i *Google Chrome*, a za debagovanje alat *Firebug* [12].

5.2 Programski jezik

Sistem za upravljanje problemima u softveru implementiran je u programskom jeziku *PHP*. Dizajniran je za razvoj aplikacija zasnovanih na vebu sa dinamičkim sadržajem. *PHP* je besplatan, objavljen pod licencom PHP koja je nezavisna i nije deo licence *GNU GPL* (*General Public License*). Originalno, pod akronimom *PHP* je stajalo „*Personal Home Pages*“, dok se danas tumači kao *„PHP: Hypertext Preprocessor”*.

Kada korisnik pošalje zahtev preko veb pregledača za neku php stranicu, veb server prepoznaje ekstenziju i poziva odgovarajući interpreter (u ovom slučaju *PHP* interpreter). PHP interpreter parsira fajl, ako on sadrži neki upit, onda se taj upit prosleđuje bazi podataka. Baza podataka obrađuje upit i vraća tražene informacije PHP interpreteru. PHP interpreter formatira podatke u obliku *HTML* stranice, koja se šalje klijent da bi se prikazala u njegovom veb pregledaču. Prethodno opisani scenario se može videti na slici 22.

Za *PHP* postoje razni razvojni okviri (engl *framework*) koji su napisani sa ciljem da olakšaju razvoj *PHP* aplikacija. U ovom radu korišćen je *CodeIgniter* [10]. *CodeIgniter* je razvojni okvir otvorenog koda i služi za razvoj dinamičkih *PHP* aplikacija, zasnovanih na vebu. On olakšava i ubrzava razvoj *PHP* aplikacija, organizujući izvorni kod u obliku arhitekture „Model Pogled Kontroler“ (engl. *Model View Controller - MVC*). Pored toga, obezbeđuje i biblioteku najčešće korišćenih klasa i funkcija. Primer izvornog koda koji ilustruje upotrebu arhitekture *MVC* u *CodeIgniter*-u se nalazi u dodacima (odeljak 8.3)



Slika 22: Prikaz rada PHP interpretera

Pored *PHP*-a, za generisanje strana na serverskoj strani, korišćen je i *JavaScript* za obradu podataka na klijentskoj strani. *JavaScript* je dinamički, slabo tipiziran jezik, koji podržava imperativnu programsku paradigmu. Primarna upotreba *JavaScript*-a je za pisanje funkcija koje služe za upravljanje *Document Object Model*-om (*DOM*). Pošto se *JavaScript* izvršava lokalno na korisnikovom veb pregledaču, veb pregledač može brzo da odgovara na korisnikove komande, bez potrebe za slanjem nepotrebnih upita serveru. Time se značajno ubrzava rad aplikacija. Za *JavaScript* je korišćena biblioteka *JQuery* [11]. *JQuery* je otvorenog koda, dizajniran sa ciljem da olaška i unapredi manipulisanje *DOM* objektima.

Za prikaz stranica korišćeni su *HTML* i *CSS* (*Cascading Style Sheets*). *CSS* je primarno dizajniran da razdvoji sadržaj dokumenta od njegovog načina prikazivanja, omogućavajući time znatno pojednostavljenje *HTML* koda, a pored toga i mnoga unapređenja u dizajnu samih veb stranica.

6. Diskusija

Cilj ovog rada je da ukaže na prostor za unapređenja postojećih sistema za upravljanje problemima u razvoju softvera. Neka od njih, kao što su: osnovni i napredni korisnički interfejs, rangiranje korisnika (reputacija korisnika) i obezbeđivanje relevantnih informacija o problemu, opisana su u odeljku 2.4. Pored prikazanih unapređenja, u nekim naučnim radovima su izložene još neke preporuke. Mnoge od njih nije lako praktično primeniti ali ipak predstavljaju dragocene smernice koje bi trebalo pratiti prilikom daljih unapređenja ovog sistema.

Preporuka 1: Obezbediti dodatne alate za korisnike, koji bi im olakšali prikupljanje i pripremu potrebnih informacija [4].
Preporuka 2: Pronaći dobrovoljce koji bi prevodili izveštaje o problemima na razne jezike. Sam sistem bi trebalo da obezbedi podršku za više jezika [4].
Preporuka 3: Nije dovoljno samo rešiti postojeći problem, potrebno je i nagraditi korisnika koji je prijavio problem [4].
Preporuka 4: Obezbediti alat za pretraživanje izveštaja o problemima koji je dovoljno dobar, a sa druge strane i veoma jednostavan za upotrebu [4].
Preporuka 5: Ohrabriti korisnike da prilikom opisivanja problema unose dodatne informacije o njemu, kako za nove probleme tako i za postojeće. Obezbediti alat koji bi omogućio spajanje problema [4].

Ovo su neke od smernica koje će biti razmatrane u sledećim verzijama ovog sistema. Neki od narednih koraka u daljem razvoju ovog sistema su: ubacivanje podrške za ponovni pregled koda (engl. *Code Review)*, podršku i integraciju sa sistemima za reviziju izvornog koda (*SVN, Git* i *Mercurial*) kao i podršku za razne vrste dijagrama i izveštaja.

7. Zaključak

U radu je opisan sistem za upravljanje problemima u razvoju softvera. Ovaj sistem predstavlja aplikaciju zasnovanu na vebu koja omogućava unošenje novog problema, njegovo praćenje i menjanje. Time se olakšava upravljanje ogromnom količinom problema, koji mogu da se jave u današnjim projektima.

Cilj ovog rada je da, proučavajući postojeća rešenja problema, ponudi predlog aplikacije koja bi umanjila nedostatke koji se javljaju kod aktuelnih sistema za upravljanje problemima u razvoju softvera. Kroz manje izmene i dopune postojećih rešenja, ovaj predlog rešenja predstavlja izvesno unapređenje postojećih sistema za upravljanje problemima.

Sistem je projektovan i implementiran primenom savremenih metoda za dizajn i razvoj aplikacija zasnovanih na vebu. Za čuvanje podataka korišćen je relacioni sistem za upravljanje bazom podataka *MySQL*. Za dinamičko generisanje veb sadržaja, korišćen je jezik *PHP* koji generiše *HTML* kod za prikaz rezultata na klijentskoj strani. Kao dopuna toga, korišćen je *JavaScript* za obradu *DOM*-a i *CSS* za lepše formatiranje i prikaz stranica.

Tokom projektovanja ovog sistema težilo se ka ispunjavanju preporuka naučnih radova koji se bave ovom temom. Neke od preporuka koje je teško ostvariti u ovakvom obimu rada su ignorisane dok one realnije su sprovedene u delo. Naravno da postoji još dosta mesta za unapređenja koja će biti uzeta u razmatranje tokom nadogradnje ovog sistema.

Izradom ovog master rada proširio sam svoje znanje iz navedenih programskih jezika. Takođe sam proširio svoje znanje iz projektovanja i dizajna softverskih rešenja kao i projektovanja baza podataka. Bolje sam upoznao i izučio već široko korišćene sisteme za upravljanje problemima, a svoje znanje u ovoj oblasti usavršiću tokom vremena.

8. Dodaci

8.1 Shema baze podataka

Shema baze podataka je dobijena prevođenjem opisanih entiteta, na dijagramu entiteta i odnosa, u tabele. U nastavku je prikazan detaljan opis tako dobijenih tabela relacione baze podataka sistema za upravljanje problemima u razvoju softvera.

Tabela uloga

Naziv kolone	Tip podataka	PK/SK	NULL vrednost	Opis
id_uloge	INT	PK	ne	Identifikator uloge.
naziv	VARCHAR		ne	Naziv uloge.
dodavanje_problema	BOOLEAN		ne	Označava da li korisnik sa tom ulogom ima privilegiju da dodaje nove probleme. Ima vrednost *false* ako korisnik nema tu privilegiju, *true* ako je ima.
izmena_problema	BOOLEAN		ne	Označava da li korisnik sa tom ulogom ima privilegiju da menja probleme. Ima vrednost *false* ako korisnik nema tu privilgiju, *true* ako je ima.
ostavljanje_komentara	BOOLEAN		ne	Označava da li korisnik ima privilegiju da ostavlja komentare u okviru datog projekta. Ima vrednost *false* ako korisnik nema tu privilegiju, *true* ako je ima.
upravljanje_komentarima	BOOLEAN		ne	Označava da li korisnik ima privilegiju da upravlja komentarima u projektu. Ima vrednost *false* ako korisnik nema tu privilegiju, *true* ako je ima.
upravljanje_projektom	BOOLEAN		ne	Označava da li korisnik ima privilegiju da upravlja projektima. Ima vrednost *false* ako korisnik nema tu privilegiju, *true* ako ima.
upravljanje_clanovima_projekta	BOOLEAN		ne	Označava da li korisnik ima privilegiju da dodaje i brise korisnike u datom projektu. Ima vrednost *false* ako korisnik nema tu privilegiju, *true* ako je ima.

Tabela 4: Pregled kolona tabele uloga

Tabela korisnik

Naziv kolone	Tip podataka	PK/SK	NULL vrednost	Opis
id_korisnika	INT	PK	ne	Identifikator korisnika.
nivo_korisnika	INT		ne	Označava koju inicijalnu ulogu ima korisnik. Npr. 1 – administrator, 2 – rukovodilac, 3 – programer, 4 – tester, 0 – bez dodeljene inicijalne uloge.
korisnicko_ime	VARCHAR		ne	Korisničko ime koje se koristi za prijavljivanje na sistem. Mora biti jedinstveno u sistemu.
lozinka	TEXT		ne	Lozinka koja se vezuje za korisničko ime. Pomoću kombinacije korisničkog imena i lozinke, vrši se prijavljivanje na sistem.
email_adresa	VARCHAR		ne	Adresa elektronske pošte korisnika. Može služiti za poništavanje zaboravljene lozinke i za primanje obaveštenja na pretplaćeni problem.
ime	VARCHAR		ne	Ime korisnika.
prezime	VARCHAR		ne	Prezime korisnika.
aktivacioni_kljuc	TEXT			Ključ dodeljen korisniku prilikom registracije.
datum_registracije	DATETIME		ne	Datum i vreme kada je korisnik napravio svoj nalog.
koliko_puta_je_suspendovan	INT			Brojač koji označava koliko puta je neki korisnik suspendovan. Nula inicijalno.
suspendovan_do	DATETIME			Datum do kog je korisnik suspendovan. Ako je NULL, korisnik nije suspendovan.
aktiviran	BOOLEAN			Polje koje označava da li je korisnik potvrdio poslatu elektronsku poštu.
id_odseka	INT			Identifikator odseka, strani ključ na tabelu odsek.

Tabela 5: Pregled kolona tabele korisnik

Tabela projekat

Naziv kolone	Tip podataka	PK/SK	NULL vrednost	Opis
id_projekta	INT	PK	ne	Identifikator projekta.
naziv	VARCHAR		ne	Naziv projekta.
opis	TEXT			Opis projekta
datum_pravljenja	DATETIME			Datum pravljenja projekta.
datum_deaktiviranja	DATETIME			Projekat može postati deaktiviran. Ovo polje označava datim deaktiviranja. Ako je

				NULL, onda je projekat aktivan.
procenata_zavrsen	TINYINT			Polje koje označava koliko procenata projekata je završeno.
javan	BOOLEAN			Polje koje označava da li je projekat vidljiv korisnicima koji nisu njegovi članovi. Ako je *true* onda je projekat vidljiv, ako je *false* onda nije.

Tabela 6: Pregled kolona tabele projekat

Tabela platforma

Naziv kolone	Tip podataka	PK/SK	NULL vrednost	Opis
id_platforma	INT	PK	ne	Identifikator platforme.
naziv_platforma	VARCHAR		ne	Naziv platforme.

Tabela 7: Pregled kolona tabele platforma

Tabela status_problema

Naziv kolone	Tip podataka	PK/SK	NULL vrednost	Opis
id_status_problema	TINYINT	PK	ne	Identifikator statusa problema.
naziv_status_problema	VARCHAR		ne	Naziv statusa problema.

Tabela 8: Pregled kolona tabele status_problema

Tabela prioritet_problema

Naziv kolone	Tip podataka	PK/SK	NULL vrednost	Opis
id_prioritet_problema	TINYINT	PK	ne	Identifikator prioriteta problema.
naziv_prioritet_problema	VARCHAR		ne	Naziv prioriteta problema.

Tabela 9: Pregled kolona tabele prioritet_problema

Tabela ozbiljnost_problema

Naziv kolone	Tip podataka	PK/SK	NULL vrednost	Opis
id_ozbiljnost_problema	TINYINT	PK	ne	Identifikator ozbiljnosti problema.
naziv_ozbiljnost_problema	VARCHAR		ne	Naziv ozbiljnosti problema.

Tabela 10: Pregled kolona tabele ozbiljnost_problema

Tabela verzija

Naziv kolone	Tip podataka	PK/SK	NULL vrednost	Opis
id_verzija	INT	PK	ne	Identifikator verzije.
naziv_verzije	VARCHAR		ne	Naziv verzije.
id_projekat	INT	SK	ne	Identifikator projekta, strani ključ na tabelu *projekat.*

Tabela 11: Pregled kolona tabele verzija

Tabela tip_problema

Naziv kolone	Tip podataka	PK/SK	NULL vrednost	Opis
id_tip_problema	TINYINT	PK	ne	Identifikator tipa problema.
naziv_tip_problema	VARCHAR		ne	Naziv tipa problema.

Tabela 12: Pregled kolona tabele tip_problema

Tabela problem

Naziv kolone	Tip podataka	PK/SK	NULL vrednost	Opis
id_problema	INT	PK	ne	Identifikator problema.
id_projekta	INT	SK	ne	Identifikator projekta, strani ključ na tabelu *projekat.*
id_prijavio	INT	SK	ne	Identifikator korisnika koji je prijavio problem, strani ključ na tabelu *korisnik.*
id_tip_problema	TINYINT	SK		Identifikator tipa problema, strani ključ na tabelu *tip_problema.*
id_ozbiljnost_problema	TINYINT	SK		Identifikator ozbiljnosti problema, strani ključ na tabelu *ozbiljnost_problema.*
id_prioritet_problema	TINYINT	SK		Identifikator prioriteta problema, strani ključ na tabelu *prioritet_problema*.
id_status_problema	TINYINT	SK		Identifikator statusa problema, strani ključ na tabelu *status_problema*.
id_verzija	INT	SK		Identifikator verzije projekta, strani ključ na tabelu *verzija*.
datum_prijavljivanja	DATETIME			Datum prijavljivanja problema.
naslov	VARCHAR		ne	Naziv problema.
opis	TEXT			Opis problema.
koraci_pri_reprodukciji	TEXT			Koraci pri reprodukciji problema (engl. *Steps to reproduce*) tj. uputstvo za ponovno prouzrokovanje problema.
uoceno_ponasanje	TEXT			Kako izgleda izvršavanje aplikacije sa uočenim problemom.
ocekivano_ponasanje	TEXT			Kako bi trebalo da izgleda

				izvršavanje aplikacije bez problema.
test_primeri	TEXT			Test primeri na kojima se vidi opisani problem.
kako_je_problem_reseen	TEXT			Način na koji je problem rešen.
izvorni_kod	TEXT			Izvorni kod samog problema, ili deo koda u kome postoji problem.
log_greske	TEXT			Dnevnik greške koji generiše softver prilikom javljanja greške.
stack_trace	TEXT			Deo steka koji daje informacije o problemu.
potvrdjen	BOOLEAN			Svaki uneseni problem mora biti potvrđen od strane nekog korisnika sa višim privilegijama. Ovo polje označava da li je problem potvrđen.
zakljucan_za_izmene	BOOLEAN			Ako se desi da neki korisnik ili više njih nekontrolisano menjaju problem, on se može zaključati od strane rukovodioca projekta.
uradjeno_procenata	TINYINT			Koji deo (procenat) problema je ispravljen.
potrebno_casova_rada	INT			Koliko vremena je potrebno da se ispravi problem. Merna jedinica je jedan sat.
br_pregleda	INT			Koliko ukupno puta je konkretni problem pregledan.
id_odseka	INT			Identifikator odseka, strani ključ na tabelu *odsek.*
id_popravio	INT			Identifikator korisnika koji je popravio problem, strani ključ na tabelu *korisnik*.
id_prijavio	INT			Identifikator korisnika koji je prijavio problem, strani ključ na tabelu *korisnik*.
id_dodeljen	INT			Identifikator korisnika kome je dodeljen problem.
id_stanje_problema	TINYINT			Identifikator stanja problema, strani ključ na tabelu *stanje_problema.*

Tabela 13: Pregled kolona tabele problem

Tabela zavisnost_problema

Naziv kolone	Tip podataka	PK/SK	NULL vrednost	Opis
id_zavisnosti	INT	PK	ne	Identifikator zavisnosti
id_od_problema	INT	SK	ne	Identifikator problema koji zavisi od nekog problema, strani ključ na tabelu *problem.*
id_do_problema	INT	SK	ne	Identifikator problema od koga problem zavisi, strani ključ na tabelu *problem.*

Tabela 14: Pregled kolona tabele zavisnost_problema

Tabela stanje_problema

Naziv kolone	Tip podataka	PK/SK	NULL vrednost	Opis
id_stanje_problema	TINYINT	PK	ne	Identifikator stanja problema.
naziv_stanje_problema	TEXT			Naziv stanja.

Tabela 15: Pregled kolona tabele stanje_problema

Tabela tip_priloga

Naziv kolone	Tip podataka	PK/SK	NULL vrednost	Opis
id_tip_priloga	INT	PK	ne	Identifikator tipa priloga.
naziv_tip_priloga	VARCHAR		ne	Naziv tipa priloga.

Tabela 16: Pregled kolona tabele tip_priloga

Tabela prilog

Naziv kolone	Tip podataka	PK/SK	NULL vrednost	Opis
id_priloga	INT	PK	ne	Identifikator priloga.
id_problema	INT	SK	ne	Identifikator problema na koji je ostavljen prilog, strani ključ na tabelu *problem.*
id_korisnika	INT	SK	ne	Identifikator korisnika koji je ostavio prilog, strani ključ na tabelu *korisnik.*
id_tip_priloga	INT	SK		Identifikator tipa priloga, strani ključ na tabelu *tip_priloga.*
naziv	VARCHAR			Naziv priloga.
opis	TEXT			Opis priloga.
ime_datoteke	VARCHAR		ne	Ime datoteke.
datum_dodavanja	DATETIME		ne	Datum i vreme dodavanja priloga.

Tabela 17: Pregled kolona tabele prilog

Tabela komentar

Naziv kolone	Tip podataka	PK/SK	NULL vrednost	Opis
id_komentara	INT	PK	ne	Identifikator komentara.
id_problema	INT	SK	ne	Identifikator problema na koji je ostavljen komentar, strani ključ na tabelu *problem.*
id_korisnika	INT	SK	ne	Identifikator korisnika koji je ostavio komentar, strani ključ na tabelu *korisnik*
datum	DATETIME		ne	Vreme ostavljanja komentara.
tekst_komentara	TEXT			Tekst komentara.

Tabela 18: Pregled kolona tabele komentar

Tabela clan_projekta

Naziv kolone	Tip podataka	PK/SK	NULL vrednost	Opis
id	INT	PK	ne	Identifikator članstva.
id_projekta	INT	SK	ne	Identifikator projekta čiji je korisnik član, strani ključ na tabelu *projekat.*
id_korisnika	INT	SK	ne	Identifikator korisnika, strani ključ na tabelu *korisnik.*

Tabela 19: Pregled kolona tabele clan_projekta

Tabela projekat_platforma

Naziv kolone	Tip podataka	PK/SK	NULL vrednost	Opis
id_projekat	INT	PK	ne	Identifikator projekta, strani ključ na tabelu *projekat.*
id_platforma	INT	PK	ne	Identifikator operativnog sistema, strani ključ na tabelu *platforma.*

Tabela 20: Pregled kolona tabele projekat_platforma

Tabela tip_obavestenja

Naziv kolone	Tip podataka	PK/SK	NULL vrednost	Opis
id_tip_obavestenja	INT	PK	ne	Identifikator tipa obaveštenja.
naziv_tip_obavestenja	VARCHAR		ne	Naziv tipa obaveštenja.

Tabela 21: Pregled kolona tabele tip_obavestenja

Tabela obaveštenje

Naziv kolone	Tip podataka	PK/SK	NULL vrednost	Opis
id_obavestenja	INT	PK	ne	Identifikator obaveštenja.
id_problema	INT	SK	ne	Identifikator problema za koji se vrši notifikacija, strani ključ na tabelu *problem.*
id_korisnika	INT	SK	ne	Identifikator korisnika koji se pretplatio na obaveštenje, strani ključ na tabelu *korisnik.*
tip_obavestenja	INT	SK	ne	Identifikator tipa obaveštenja, strani ključ na tabelu *tip_obavestenja.*
aktivno	BOOLEAN		ne	Identifikator koji govori da li je obaveštenje aktivno ili nije. *true* ako je aktivno, *false* ako nije.

Tabela 22: Pregled kolona tabele obaveštenje

Tabela log_izmena_problema

Naziv kolone	Tip podataka	PK/SK	NULL vrednost	Opis
id_log_izmena_problema	INT	PK	ne	Identifikator jednog unosa u tabelu log_izmena_problema.
id_problema	INT	SK	ne	Identifikator problema na kom je izvršena izmena, strani ključ na tabelu *problem.*
id_korisnika	INT	SK	ne	Identifikator korisnika koji je izvršio izmenu, strani ključ na tabelu *korisnik.*
vreme	DATETIME		ne	Datum i vreme izmene.
stara_vrednost	TEXT			Stara vrednost polja.
nova_vrednost	TEXT			Nova vrednost polja.
izmenjeno_polje	VARCHAR			Naziv polja na kom je izvršena izmena.

Tabela 23: Pregled kolona tabele log_izmena_problema

Tabela log_prijavljivanja

Naziv kolone	Tip podataka	PK/SK	NULL vrednost	Opis
id_log_prijavljivanja	INT	PK	ne	Identifikator jednog unosa u dnevniku prijavljivanja.
korisnicko_ime	INT	SK		Korisničko ime sa kojim je posetilac pokušao da se prijavi na sistem.
vreme	DATETIME		ne	Vremenski trenutak u kome je izvršen pokušaj prijavljivanja.
uspesno	BOOLEAN			Polje koje označava da li je prijavljivanje uspešno ili ne. *true* ako je uspešno, *false* ako nije.
ip_adresa	VARCHAR			IP adresa korisnika prilikom pokušaja prijavljivanja na sistem.
internet_pregledac	VARCHAR			Polje u kome se beleži informacijo o internet pregledaču korisnika.

Tabela 24: Pregled kolona tabele log_prijavljivanja

Tabela log_izmena_projekta

Naziv kolone	Tip podataka	PK/SK	NULL vrednost	Opis
id_log_izmena_projekta	INT	PK	ne	Identifikator jednog unosa u tabelu izmena projekta.
id_korisnika	INT	SK	ne	Identifikator korisnika koji je izvršio izmenu u projektu, strani ključ na tabelu *korisnik.*
id_projekta	INT	SK	ne	Identifikator projekta na kome je izvršena izmena, strani ključ na

				tabelu *projekat.*
vreme	DATETIME		ne	Vremenski trenutak u kome je izvršena izmena.
stara_vrednost	TEXT			Stara vrednost polja.
nova_vrednost	TEXT			Nova vrednost polja.
izmenjeno_polje	VARCHAR(45)			Naziv polja na kome je izvršena izmena.

Tabela 25: Pregled kolona tabele log_izmena_projekta

Tabela problem_dodatno_polje

Naziv kolone	Tip podataka	PK/SK	NULL vrednost	Opis
id_problem_dodatno_polje	INT	PK	ne	Identifikator tabele problem_dodatno_polje.
id_problem	INT	SK	ne	Identifikator problema, strani ključ na tabelu *problem.*
id_projekat_dodatno_polje	INT	SK	ne	Strani ključ na tabelu projekat_dodatno_polje.
vrednost	TEXT			Vrednost dodatnog polja.

Tabela 26: Pregled kolona tabele projekat_dodatno_polje

Tabela projekat_dodatno_polje

Naziv kolone	Tip podataka	PK/SK	NULL vrednost	Opis
id_projekat_dodatno_polje	INT	PK	ne	Identifikator tabele projekat_dodatno_polje.
id_projekta	INT	SK	ne	Identifikator projekta, strani ključ na tabelu *projekat.*
naziv_polja	VARCHAR		ne	Naziv dodatnog polja.
opis	TEXT		ne	Opis dodatnog polja.
tip	VARCHAR	SK	ne	Tip dodatnog polja.

Tabela 27: Pregled kolona tabele projekat_dodatno_polje

Tabela odsek

Naziv kolone	Tip podataka	PK/SK	NULL vrednost	Opis
id_odseka	INT	PK	ne	Identifikator odseka.
naziv_odseka	TEXT		ne	Naziv odseka.

Tabela 28: Pregled kolona tabele odsek

Tabela clan_uloga

Naziv kolone	Tip podataka	PK/SK	NULL vrednost	Opis
id_clana	INT	PK/SK	ne	Identifikator člana, strani ključ na tabelu *clan_projekta.*
id_uloge	INT	PK/SK	ne	Identifikator uloge, strani ključ na tabelu *uloga.*

Tabela 29: Pregled kolona tabele clan_uloga

Tabela problem_platforma

Naziv kolone	Tip podataka	PK/SK	NULL vrednost	Opis
id_problem	INT	PK/SK	ne	Identifikator problema, strani ključ na tabelu *problem.*
id_platforma	INT	PK/SK	ne	Identifikator operativnog sistema, strani ključ na tabelu *platforma.*

Tabela 30: Pregled kolona tabele problem_platforma

Tabela podesavanja

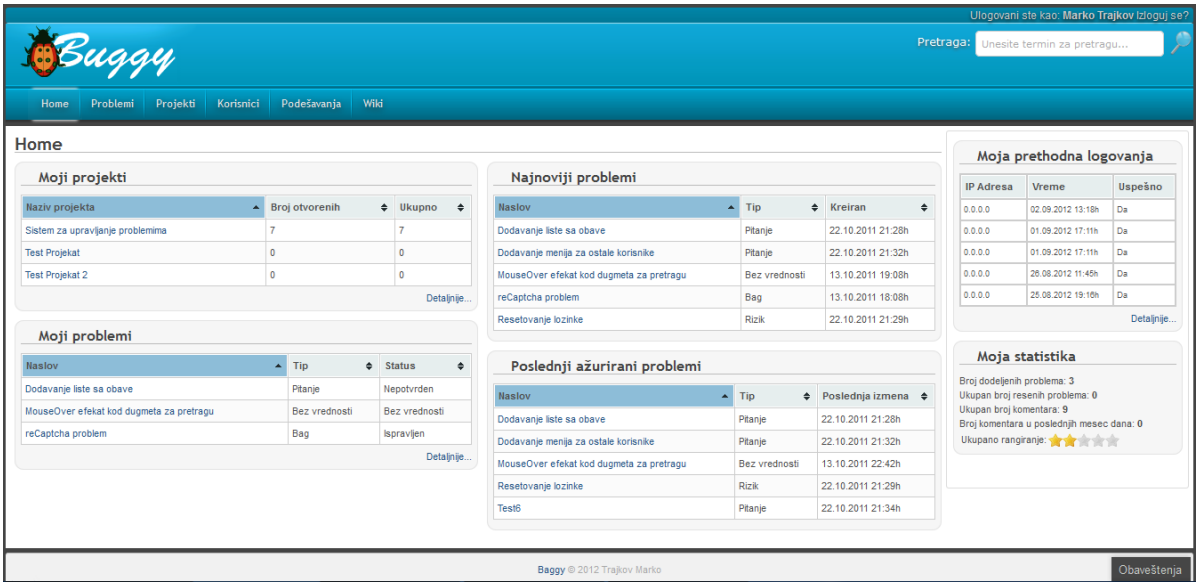
Naziv kolone	Tip podataka	PK/SK	NULL vrednost	Opis
id	INT	PK	ne	Identifikator podešavanja.
id_korisnika	INT	SK		Identifikator korisnika. Ovaj indikator sadrži informaciju o tome na kog korisnika se podešavanje odnosi. Ako je vrednost *NULL*, podešavanje se odnosi na sve korisnike.
naziv	VARCHAR			Naziv podešavanja
vrednost	VARCHAR			Vrednost podešavanja.

Tabela 31: Pregled kolona tabele podešavanja

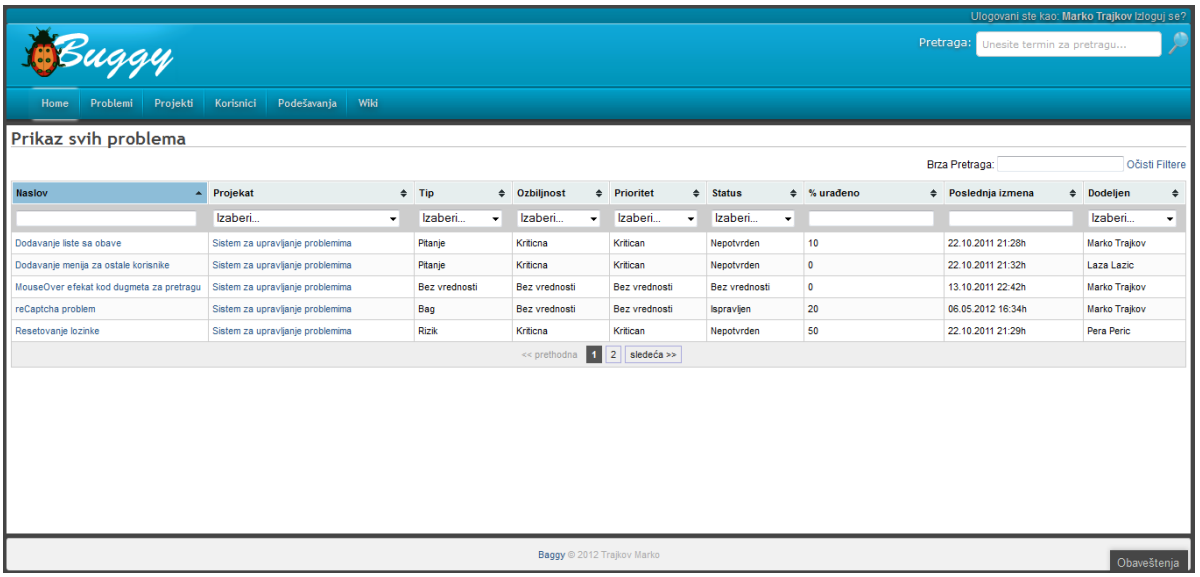
8.2 Elementi korisničkog interfejsa

Korisnik započinje svoj rad unošenjem *URL* adrese u odgovarajuće polje u svom veb pregledaču. U ovom slučaju adresa test verzije projekta nalazi se na sledećoj adresi: *http://buggy.uphero.com/www/CodeIgniter/* Učitavanjem aplikacije otvara se stranica za prijavljivanje na sistem (slika 23). Posle unošenja ispravne kombinacije korisničkog imena i lozinke korisnik će biti prebačen na početnu stranicu (slika 24). Sa početne stranice, ili sa bilo koje druge stranice, korisnik može da odabere željenu akciju i bude prebačen na neku drugu stranicu. Odabirom projekta, korisniku će biti prikazana stranica sa listom problema koji su postavljeni za taj projekat. Na slici 25. prikazana je stranica sa listom problema za izabrani projekat.

Slika 23: Stranica za prijavljivanje na sistem



Slika 24: Početna stranica



Slika 25: Stranica sa listom problema izabranog projekta

8.3 Uzorci koda

U ovom delu je opisan način na koji sistem radi kroz objašnjenje načina implementacije jednog slučaja upotrebe: „Prikaz liste problema izabranog projekta“. Cela aplikacija je organizovana u arhitekturi *MVC* (engl. *Model View Controller*) u kojoj se naglašava razdvajanje poslovne logike (engl *Controller*) od logike upravljanja podacima (engl. *Model*) i logike prikaza sadržaja (engl. *View*). U nastavku odeljka, biće prikazan po jedan deo svake od tri komponente.

Odabirom odgovarajuće stavke iz menija ili usmeravanjem internet pregledača na adresu: http://www.mojdomen.com/index.php/issues/showIssues/1/5, biće otvorena stranica sa problemima (slika 25). U *URL* (engl. *Uniform Resource Locator*) adresi, deo „*issues*“ označava koji će kontroler biti pozvan, a deo „showIssues“ označava koji će metod u okviru tog kontrolera biti izvršen. Ostali parametri, koji se nalaze u *URL* adresi, predstavljaju argumente koji će biti prosleđeni metodu. Izvorni kod kontrolera „*issues*“ prikazan je u nastavku.

```
<?php
if ( ! defined(platforma BASEPATH')){
     exit('No direct script access allowed');
}

class Issues extends CI_Controller
{

    public function __construct()
    {
        parent::__construct();
        checkAccess();
    }

    public function index()
    {
        $data = array();
        $this->load->view('common/notfound', $data);
    }

    public function showIssues($projectId, $limit=10, $page=1)
    {
        $this->load->model('issue');
        $this->load->model('project');

        $data['projectName'] = $this->project->getProjectName($projectId);
        $data['data']        = $this->issue->getIssues($projectId, $limit,
                                                       ($page-1) * $limit);
        $data['count']       = $this->issue->getIssuesCount($projectId);
        $data['projectId']   = $projectId;
        $data['page']        = "issues/showIssues";

        $this->load->view('template/default', $data);
    }

    public function issueDetails($issueId)
    {
        $this->load->view("common/underConstruction");
    }

}
```

Izvršavanjem metoda „*showIssues*“, pozivaju se odgovarajući metodi u modelima „*issue*“ i „*project*“. Izvorni kod modela „*issue*“ prikazan je u nastavku.

```php
<?php

//Issue Model

class Issue extends CI_Model {

    public function __construct() {
        parent::__construct();
    }

    public function getIssues($projectId, $limit=10, $offset=0)
    {
        $db = Bugy_DB('problem_db');

        $projectId = $db->escape($projectId);

        $data = array();

        $query = "SELECT id_problema,
                         naslov,
                         tip_problema.naziv_tip_problema,
                         ozbiljnost_problema.naziv_ozbiljnost_problema,
                         prioritet_problema.naziv_prioritet_problema,
                         status_problema.naziv_status_problema,
                         uradjeno_procenata,
                         datum_azuriranja
                    FROM problem, tip_problema, ozbiljnost_problema,
                         prioritet_problema, status_problema
                   WHERE id_projekta = $projectId
                     AND tip_problema.id_tip_problema = problem.id_tip_problema
                     AND ozbiljnost_problema.id_ozbiljnost_problema =
                         problem.id_ozbiljnost_problema
                     AND prioritet_problema.id_prioritet_problema =
                         problem.id_prioritet_problema
                     AND status_problema.id_status_problema =
                         problem.id_status_problema
                   LIMIT $limit OFFSET $offset;";

        $res = $db->query($query);
        if ($res->num_rows() == 0)
        {
            return false;
        }

        $i=0;
        while($row = $res->_fetch_assoc())
        {
            $data[$i]["id_problema"]         = $row["id_problema"];
            $data[$i]["naslov"]              = $row["naslov"];

            $data[$i]["tip_problema"]        = $row["naziv_tip_problema"];
            $data[$i]["ozbiljnost_problema"] = $row["naziv_ozbiljnost_problema"];
            $data[$i]["prioritet_problema"]  = $row["naziv_prioritet_problema"];
            $data[$i]["status_problema"]     = $row["naziv_status_problema"];

            $data[$i]["uradjeno_procenata"]  = $row["uradjeno_procenata"];
            $data[$i]["datum_azuriranja"]    = $row["datum_azuriranja"];

            $assagnee = $this->getAssignee($row["id_problema"]);
            $data[$i]["dodeljen"] = $assagnee["data"];
            $i++;
        }

        return $data;
    }

    public function getIssuesCount($projectId)
    {
        $db = Bugy_DB('problem_db');

        $projectId = $db->escape($projectId);

        $data = array();

        $query = "SELECT id_problema,
                         naslov,
                         tip_problema.naziv_tip_problema,
```

```
                        ozbiljnost_problema.naziv_ozbiljnost_problema,
                        prioritet_problema.naziv_prioritet_problema,
                        status_problema.naziv_status_problema,
                        uradjeno_procenata,
                        datum_azuriranja
                  FROM problem, tip_problema, ozbiljnost_problema,
                        prioritet_problema, status_problema
                 WHERE id_projekta = $projectId
                   AND tip_problema.id_tip_problema = problem.id_tip_problema
                   AND ozbiljnost_problema.id_ozbiljnost_problema =
                        problem.id_ozbiljnost_problema
                   AND prioritet_problema.id_prioritet_problema =
                        problem.id_prioritet_problema
                   AND status_problema.id_status_problema =
                        problem.id_status_problema;";

        $res = $db->query($query);
        return $res->num_rows();
    }

    public function getAssignee($issueId)
    {
        $db = Buggy_DB(BUGGY_DB);

        $issueId = $db->escape($issueId);
        $data = NULL;

        $query = "SELECT korisnik.id_korisnika,
                        korisnik.ime,
                        korisnik.prezime
                  FROM problem, korisnik
                 WHERE problem.id_dodeljen = korisnik.id_korisnika
                   AND problem.id_problema=$issueId;";

        $res = $db->query($query);
        if ($res->num_rows() == 0)
        {
            return array("num_rows" => $res->num_rows(), "data" => $data);
        }

        while($row = $res->_fetch_assoc())
        {
            $data .= $row["ime"] . " " . $row["prezime"] . "<br />";
        }

        //remove last <br />
        $data = substr($data, 0, (strlen($data)-6));

        return array("num_rows" => $res->num_rows(), "data" => $data);
    }
}
```

Pozivanjem metoda modela, kontroler dobija podatke. Te podatke prosleđuje pogledu (engl. *View*). Pogled je zadužen da primljene podatke prikaze korisniku na odgovarajući način. Izvorni kod pogleda za prikaz podataka o problemima prikazan je u nastavku teksta.

```
<script>
$(document).ready(function()
{
    $("#issues").tablesorter({sortList:[[0,0]], widgets: ['zebra']});
});
</script>

<h2>Projekat: <?=$projectName?></h2>

<img width="12" height="12" src="<?=IMG_PATH?>add.png">
<a href="#">Dodaj novi</a>

<table id="issues" class="tablesorter" border="0" cellpadding="0" cellspacing="1">
    <thead>
        <tr>
            <th class="header headerSortDown">Naslov</th>
            <th class="header">Tip</th>
```

```
            <th class="header">Ozbiljnost</th>
            <th class="header">Prioritet</th>
            <th class="header">Status</th>
            <th class="header">% urađeno</th>
            <th class="header">Poslednja izmena</th>
            <th class="header">Dodeljen</th>
        </tr>
    </thead>
    <tbody>
        <?if($data!==false):?>
            <?foreach($data as $key => $value):?>
                <tr>
                    <td>
                        <a href="<?=WWW_PATH?>issues/issueDetails/<?=
                                                    $value["id_problema"];?>">
                            <?=$value["naslov"]?>
                        </a>
                    </td>
                    <td><?=$value["tip_problema"]?></td>
                    <td><?=$value["ozbiljnost_problema"]?></td>
                    <td><?=$value["prioritet_problema"]?></td>
                    <td><?=$value["status_problema"]?></td>
                    <td><?=$value["uradjeno_procenata"]?></td>
                    <td><?=$value["datum_azuriranja"]?></td>
                    <td><?=$value["dodeljen"]?></td>
                </tr>
            <?endforeach;?>
        <?else:?>
            <td colspan="8">
                 <div style="text-align: center;">
                       Nema rezultata za ovaj upit.
                 </div>
            </td>
        <?endif;?>
    </tbody>

    <tfoot>
        <tr>
        <td colspan="9">
            <?php
                $n = $this->uri->total_segments();
                $pageno = (int)$this->uri->segment(5);
                $perpage = (int)$this->uri->segment(4);

                if($pageno==false)
                    $pageno = 1;
                echo getPaginationString($pageno, $count, $perpage, "",
                      WWW_PATH . "issues/showIssues/$projectId/$perpage/", "");
            ?>
        </td
        </tr>
    </tfoot>
</table>
```

8.4 Sigurnost (prijavljivanje na sistem)

HTTPS (*HTTP* preko *SSL*-a) je najčešće korišćen mehanizam kojim se obezbeđuje privatnost podataka na internetu. Nažalost, *SSL* je za mnoge manje primene previše složen pa se stoga ne isplati koristiti ga u manjim aplikacijama. U takvim situacijama je potreban neki jednostavniji metod. Jedan takav metod je metod potvrđivanja autentičnosti zasnovan na izazovu i odgovoru (engl. *Challenge-Response Authentication Mechanism*)

Najveći nedostatak metoda koji ne koriste *SSL* je taj što se sav sadržaj, pa samim tim i lozinka, šalju kao običan ne šifrovan tekst, koji se lako može uhvatiti od strane napadača. Ali ako lozinka nikad ne napusti klijenta kao običan tekst, već kao šifrovan tekst, onda je on napadaču beskoristan.

Za tu svrhu se može koristiti jednosmerna funkcija za otisak (engl. *hash*) kao što je *MD5* ili *SHA1*. One se koriste da bi šifrovali lozinku u njen otisak, koji se zatim šalje umesto lozinke. Iako bi ovaj način obezbedio sigurnost lozinke, on je ranjiv na neke napade (npr, „*replay attack*") gde napadač može iskoristiti ukradeni (engl. *sniffed*) otisak i iskoristiti ga za prijavljivanje na sistem.

Ovaj napad se može izbeći pomoću deljene „javne tajne". Prvo server na slučajan način generiše izazov (engl. *challenge*) i šalje ga klijentu. Klijent onda izračunava vrednost otiska tog stringa kombinovanog sa lozinkom. Dobijena vrednost otiska se šalje serveru koji može da je potvrdi, jer zna sve vrednosti. Ako se vrednosti otisaka poklapaju, klijentu se dozvoljava pristup sistemu. Ista niska koju generiše server (*izazov*) se ne koristi ponovo, tako da ga napadač ne može iskoristi za napad. Izračunavanje *MD5* otiska se vrši pomoću Paj-ove *MD5* biblioteke za *JavaScript*[7].

8.5 Pravljenje korisničkog računa

Svako može postati korisnik sistema pravljenjem naloga. Da bi se sprečili lažni korisnički računi, prilikom unosa ličnih podataka korisnik mora da unese i vrednost sa slike (engl. *Captcha*). Na ovaj način se sprečavaju automatizovane aplikacije (engl. *bots* ili *web robots*) koji generišu lažne korisničke račune. Za ovu svrhu je korišćen *Google reCaptcha* sistem [8], koji se danas koristi u gotovo svim aplikacijama zasnovanim na vebu. Primer korisničkog interfejsa za pravljenje korisničkog računa se može videti na slici 26.

Slika 26. Stranica za pravljenje korisničkog računa

9. Reference

[1] T. Zimmermann, R. Premraj, *Improving Bug Tracking Systems*, Maj 2009.
[2] N. Bettenburg, S. Just, *What makes a good bug report?*, Novembar 2008, strane 308–318.
[3] S. Breu, J. Sillito, *What developers ask about your bug*?, Decembar 2008.
[4] S. Just, R. Premraj, *Towards the Next Generation of Bug Tracking Systems*, September 2008.
[5] P. Hooimeijer, W. Weimer, *Modeling bug report quality*, ASE, 2007, strane 34–43.
[6] Visual Paradigm for UML User's Guide, `http://www.visual-paradigm.com`
[7] Paul Johnston, *JavaScript MD5 biblioteka*, `http://pajhome.org.uk/crypt/md5/`
[8] reCaptcha, `http://www.google.com/recaptcha`
[9] World Wide Web Consortium, `http://www.w3schools.com/`
[10] CodeIgniter, `http://www.codeigniter.com`
[11] JQuery, `http://www.jquery.com`
[12] Firebug, `http://getfirebug.com`