Đorđe Krstović
SPECIFIKACIJA I MODELOVANJE SOFTVERA

APLIKACIJA ZA ELEKTRONSKO GLASANJE: E-GLASAČ

Beograd, 2017

Univerzitet Singidunum
Tehnički fakultet

Aplikacija za elektronsko glasanje: eGlasač

- Projektni rad -

Predmet: **Specifikacija i modelovanje softvera**

Profesor:
prof. dr Angelina Njeguš

Student:
Đorđe Krstović

Beograd, 2017. Godine

Sadržaj

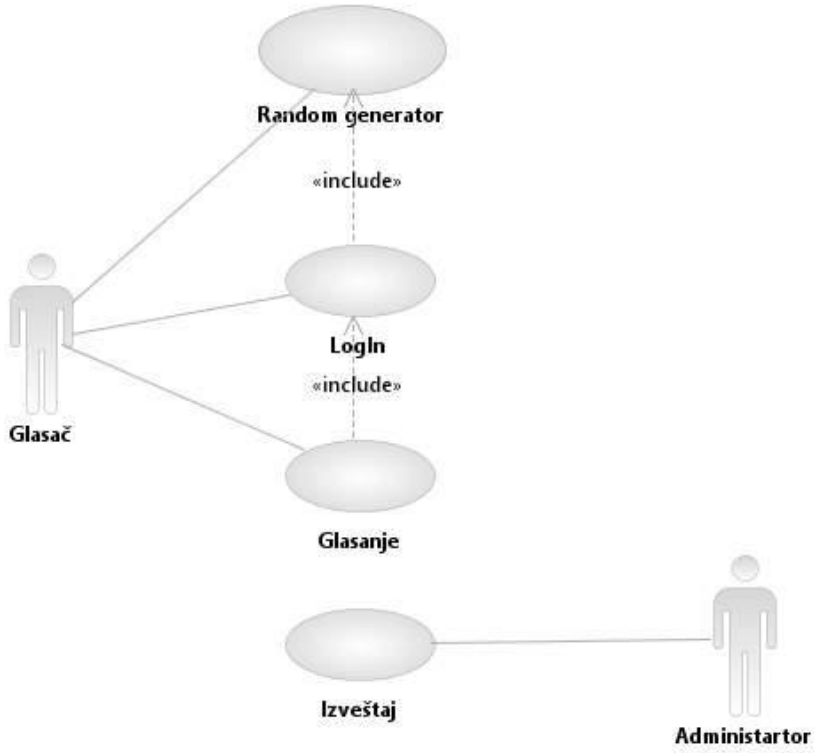
1. Uvod

Cilj ove aplikacije jeste da omogući elektronsko glasanje putem računara, brzu obradu podataka, tj. rezultata glasanja, da privuče što veći broj građana na glasanje, i da smanji vreme čekanja. U fokusu ovog projektnog rada biće modelovanje aplikacije eGlasač, prikupljanje zahteva i modelovanje sistema, analiza informacionog sistema, analiza slučajeva korišćenja i projektovanje informacionog sistema. Za proučavanje ovog projektnog rada potrebno je poznavanje UML jezika.

2. Prikupljanje zahteva i modelovanje sistema

2.1. Use case dijagram

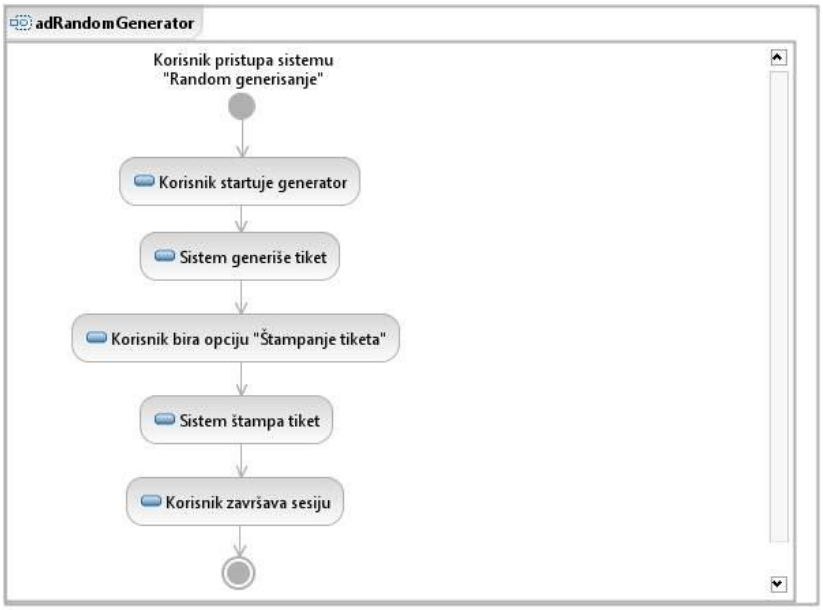
Use case dijagram aplikacije "eGlasač" predstavlja funkcionalnost i željeno ponašanje sistema. Prikazuje postojanje dva aktera, i četiri slučajeva korišćenja, njihove veze i zavisnosti. U ovom slučaju imamo aktera (Glasač, slika 2.1.1.) koji ima pristup slučajevima korišćenja (Random generator, LogIn, Glasanje), gde je use case "Random generator" obavezan kako bi se pristupilo "LogIn" use case-u, takođe "LogIn" je neophodan kako bi se pristupilo use case-u "Glasanje". Odnosno između ovih use case-ova postoji veza "include". S druge strane postoji još jedan akter (Administrator) koji ima pristup use case-u "Izveštaj". Korišćena veza između aktera i use case-a je asocijacija.



Slika 2.1.1. Use-case dijagram (eGlasač, 2017)

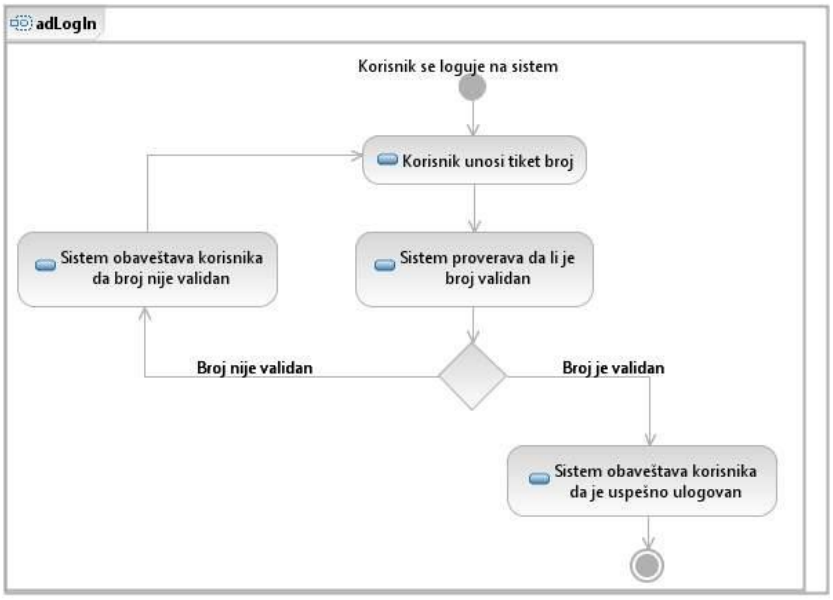
2.2. Dijagram aktivnosti

Dijagram aktivnosti prikazuje šta sistem treba da uradi, tok aktivnosti i predstavlja algoritam za izvršavanje određenog use case-a. “Random generator” (Slika 2.2.1) predstavlja tok izvršavanja od startovanja generatora do završetka sesije.



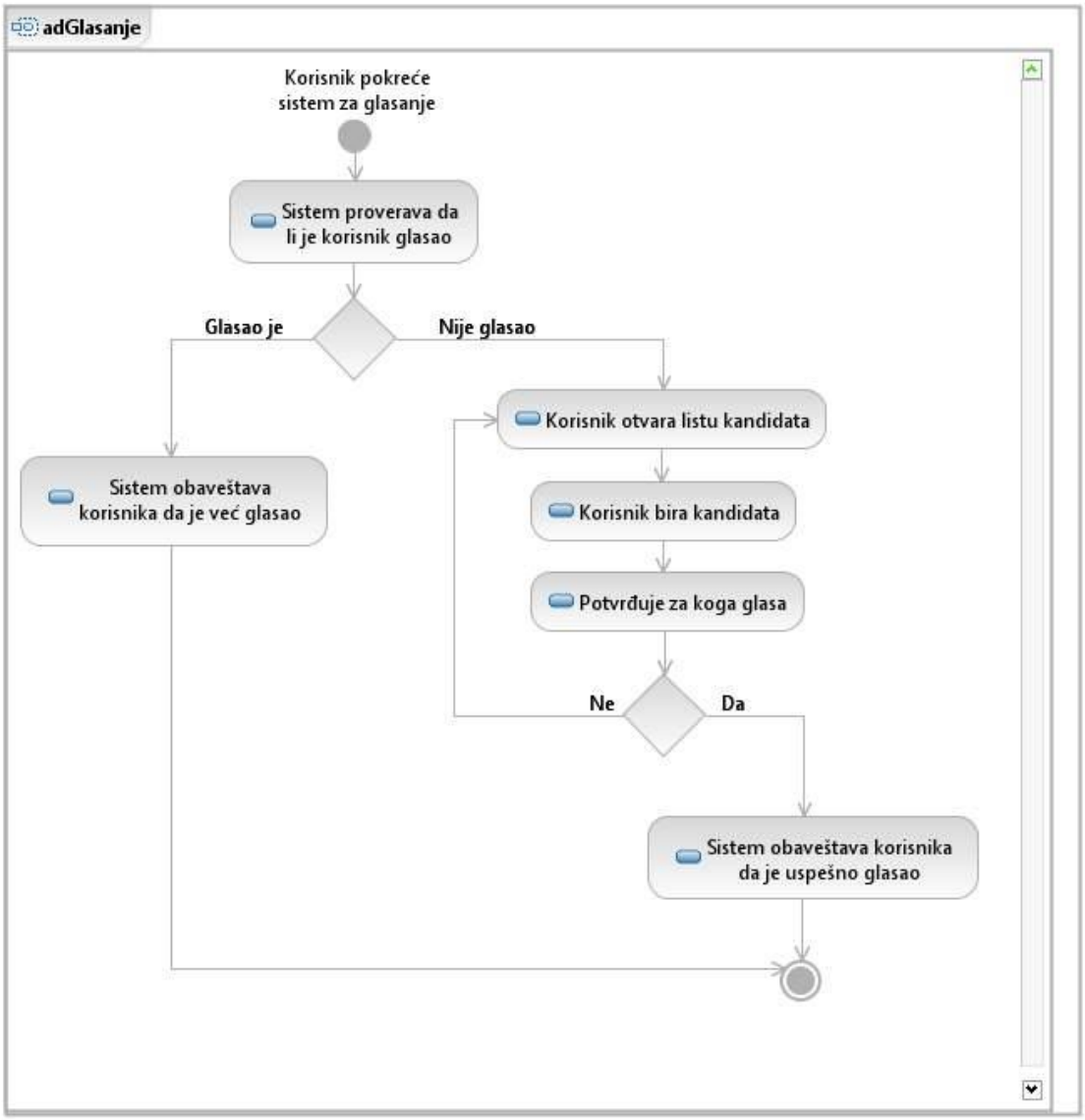
Slika 2.2.1. Dijagram aktivnosti “Random generator” (eGlasač, 2017)

Dijagram aktivnosti “LogIn” (Slika 2.2.2) predstavlja algoritam za logovanje na sistem aplikacije “eGlasač”.



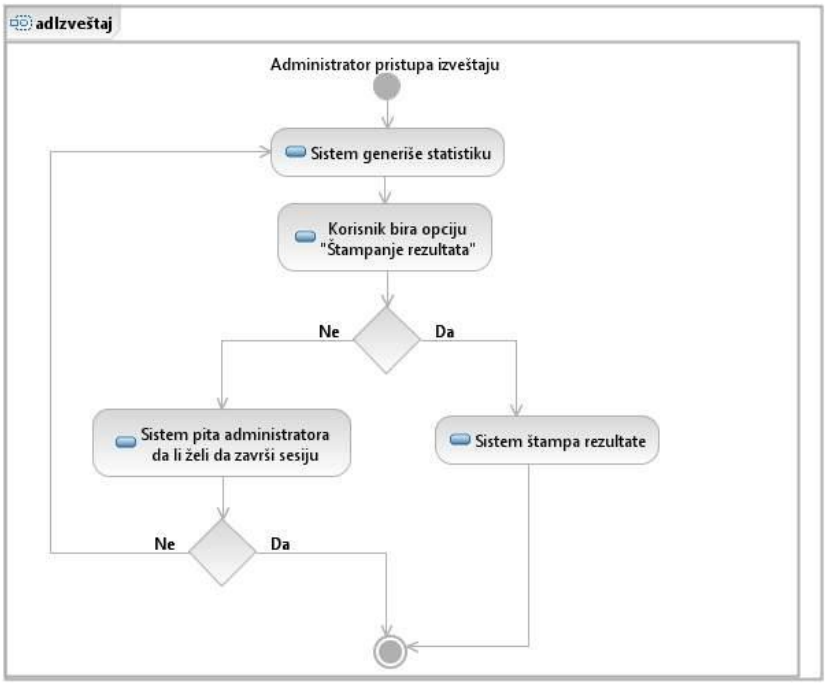
Slika 2.2.2. Dijagram aktivnosti “LogIn”, (eGlasač. 2017)

Dijagram aktivnosti “Glasanje” (Slika 2.2.3) predstavlja logiku nakon logovanja na sistem za glasanje, odnosno algoritam pri proveri da li je glasač već glasao i odabiru kandidata za kog korisnik aplikacije glasa.



Slika 2.2.3. Dijagram aktivnosti “Glasanje” (eGlasač, 2017)

Dijagram aktivnosti “Izveštaj” (Slika 2.2.4.) je dijagram aktivnosti use case-a “Izveštaj” i predstavlja algoritam interakacije administrator-sistem.

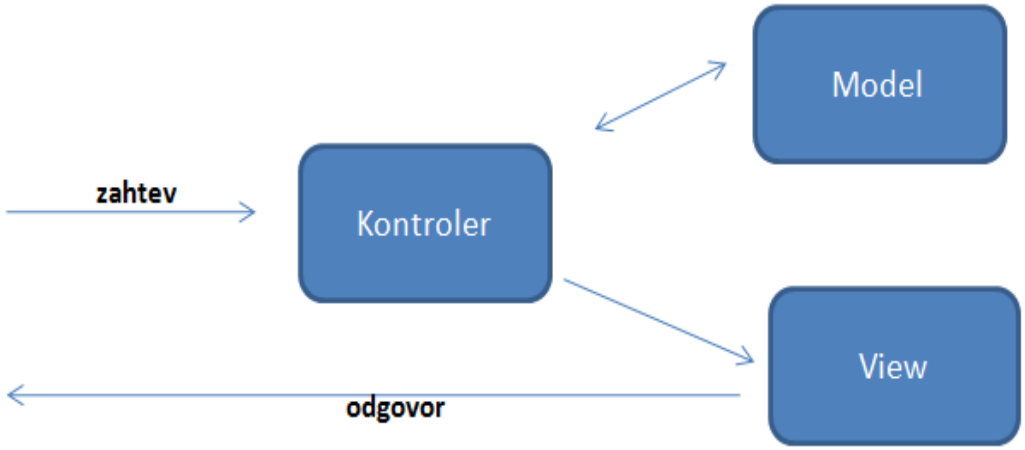


Slika 2.2.4. Dijagram aktivnosti “Izveštaj” (eGlasač, 2017)

3. Analiza infromacionog sistema

3.1. Arhitektura sistema

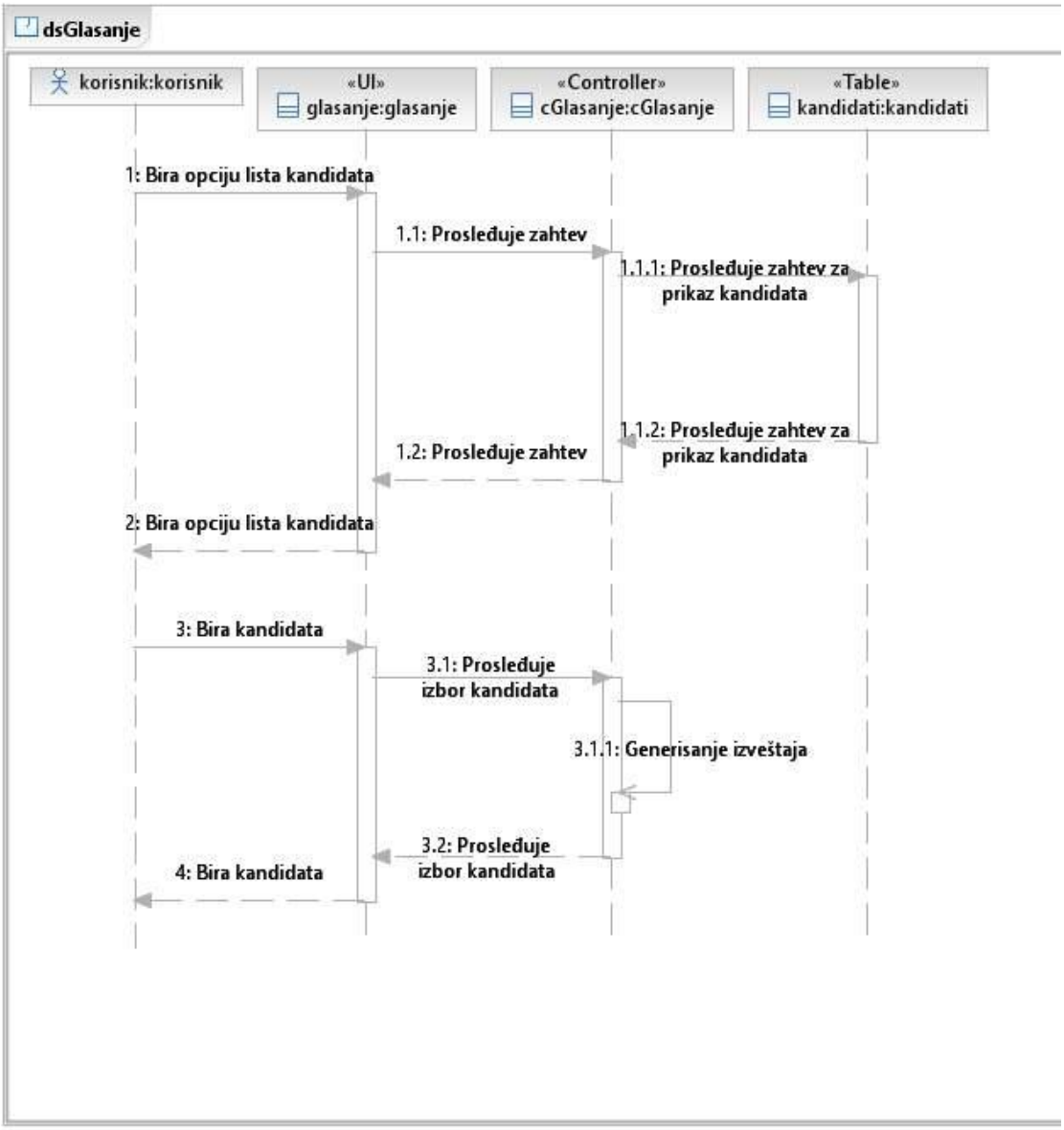
Arhitektura sistema primenjena za modelovanje aplikacije “eGlasač” je MVC arhitektura (Slika 3.1.1.). MVC predstavlja softverski patern Model-view-controller koji odvaja prikaz infromacija od interakcije korisnika sa tim informacijama. Model se sastoji od podataka aplikacije, poslovnih pravila, logika i funkcija. Ova arhitektura deli sistem u tri dela: Model, view, controller. Gde view obezbeđuje korisniku GUI preko koga korisnik unosi informacije i bira određene opcije. Controller prihvata zahtev od korisnika i poziva određenu operaciju u model-u, i ukoliko model promeni stanje obaveštava view o tome. Model predstavlja stanje sistema koje mogu promenit operacije model-a.



Slika 3.1.1. MVC Arhitektura sistema

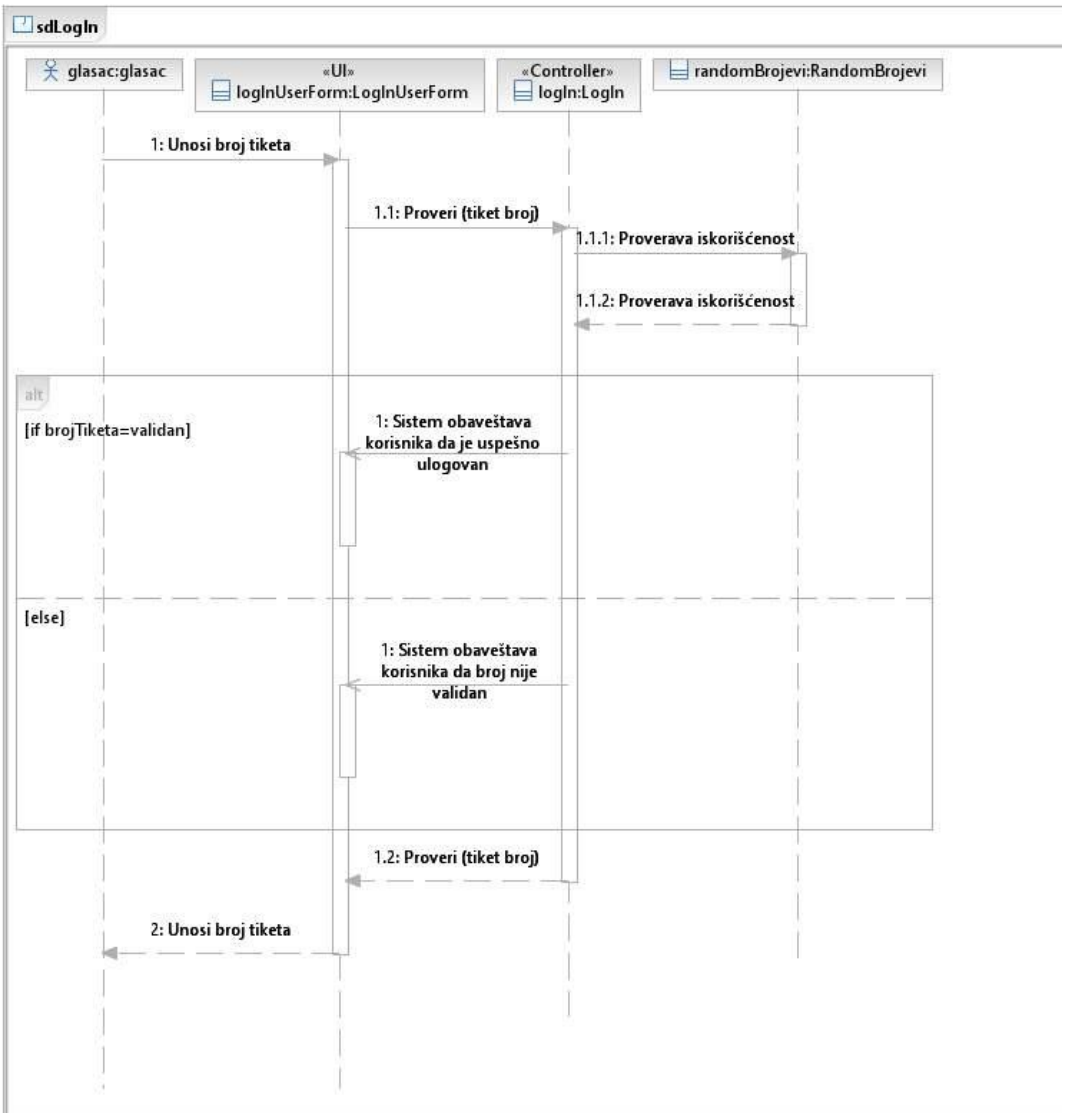
3.2. Analiza slučajeva korišćenja

Dijagram sekvenci “Glasanje” (Slika 3.2.1) opisuje interakcije između objekata uređenih hronološkim redom. Pokazuje objekte koji učestvuju u interakciji i poruke koje šalju objekti. Objašnjava nam kako i na koji način interaguju objekti, i koje poruke i zahteve prosleđuju. U prvom delu dijagrama prvo korisnik zahteva listu kandidata, UI prosleđuje zahtev, a zatim controller prosleđuje zahtev za prikaz kandidata. U drugom delu dijagrama korisnik aplikacije bira kandidata sa liste koju mu je prosledio sistem, zatim UI prosleđuje izbor kandidata, a nakon toga controller generiše izveštaj.



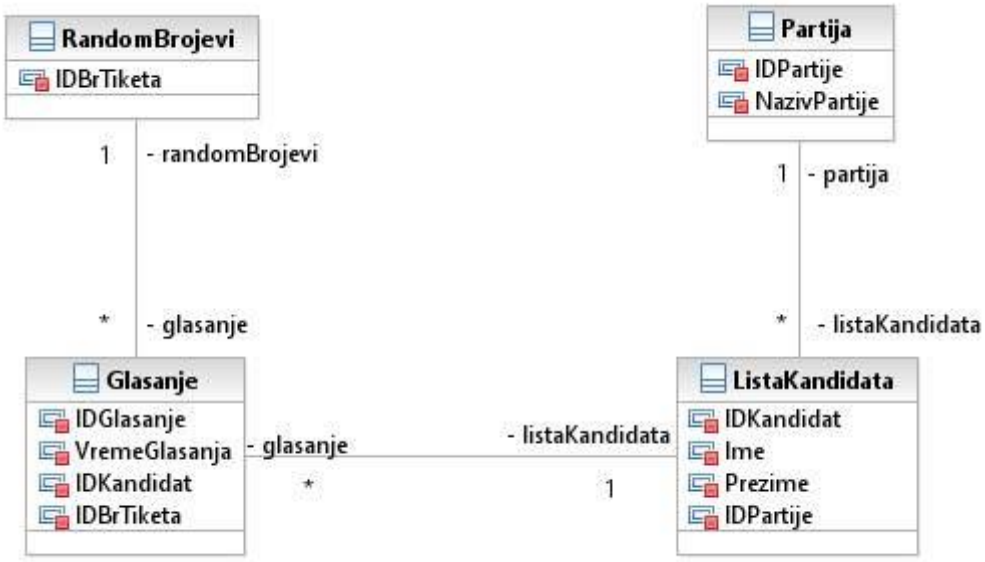
Slika 3.2.1. Dijagram sekvenci “Glasanje” (eGlasač, 2017)

Dijagram sekvenci “LogIn” (Slika 3.2.2.) opisuje takođe interakcije između objekata uređenih hronološkim redom. U ovom slučaju dijagram sekvenci prikazuje na koji način objekti interaguju pri logovanju korisnika na sistem. Na sledećem dijagramu sekvenci vidimo kako glasač prosleđuje broj tiketa u UI, zatim UI zahteva proveru tog broja, a nakon toga controller proverava iskorišćenost. Nakon toga imamo alternativni fragment sa uslovima ukoliko je broj validan, odnosno nije validan. U zavisnosti od toga sistem obaveštava korisnika da je uspešno ulogovan ili nije.



Slika 3.2.2. Dijagram sekvenci “LogIn” (eGlasač, 2017)

Dijagram klasa (Slika 3.2.3.) opisuje strukturu sistema, objašnjavajući klase unutar njega, atribute, i odnose, odnosno veze. U sledećem slučaju imamo vezu koja se zove asocijacija, svaka klasa ima svoje atribute, i one koji su nasleđeni. Svaka klasa kojoj je veza više prema jedan (* - 1) nasleđuje atribut iz klase koja ima na svom izlazu 1.

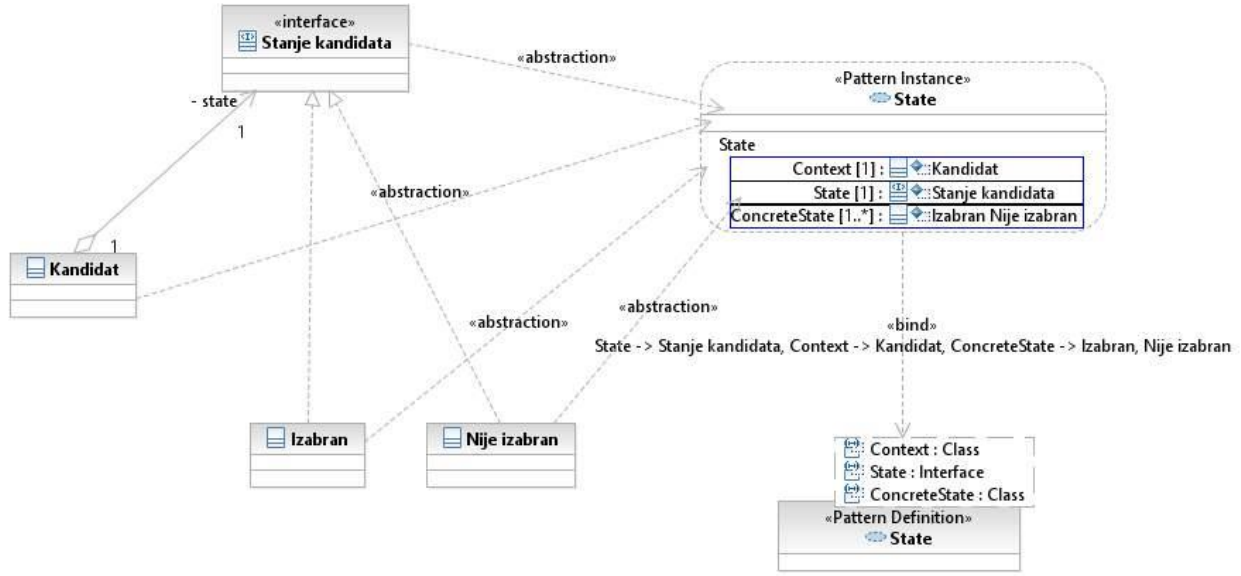


Slika 3.2.3. Dijagram klasa (eGlasač, 2017)

4. Projektovanje informacionog sistema

4.1. Dizajn patern

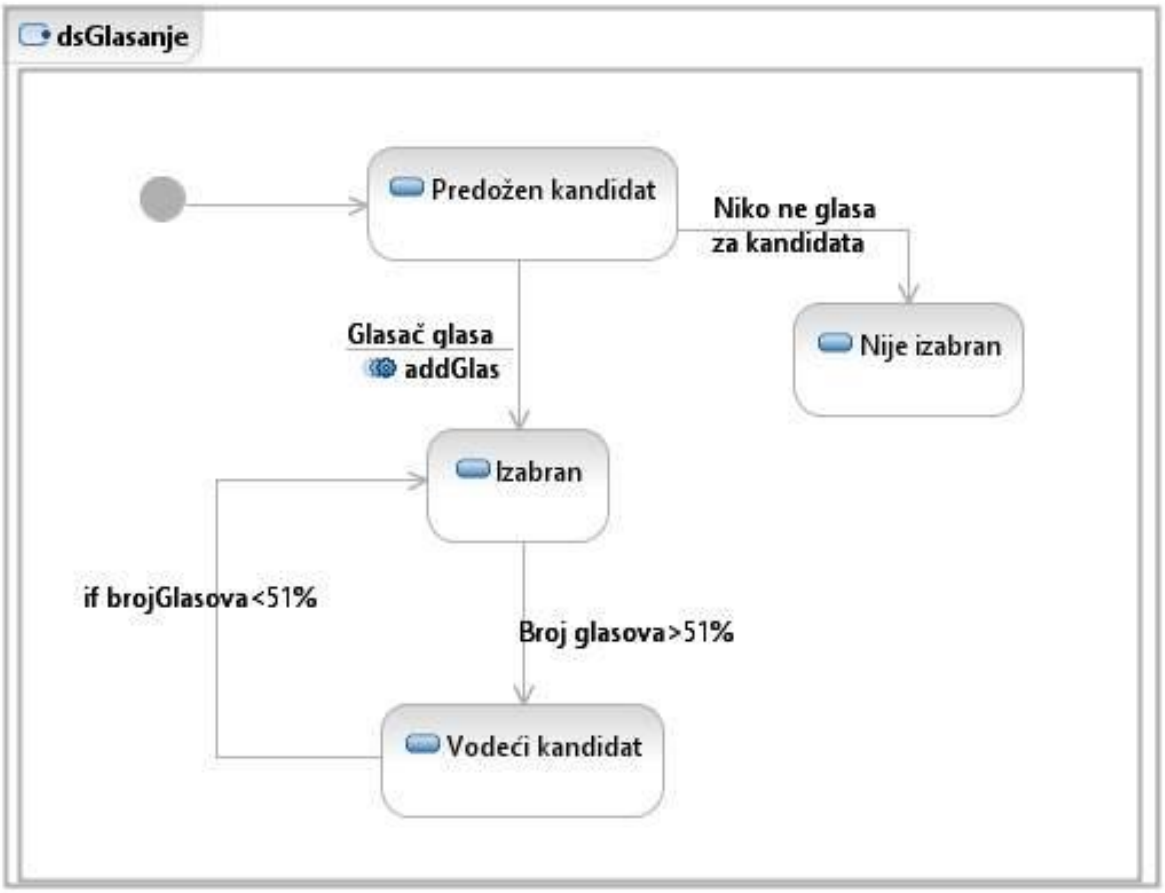
Dizajn patern (Slika 4.1.1.) predstavlja osnovni inženjerski priručnik za dizajniranje softverskih sistema. Konkretno ovaj dizajn patern prikazuje stanja koja može imati kandidat. Prikazana su stanja koja može imati kandidat (izabran/nije izabran), interfejs (stanje kandidata), i kandidat koji predstavlja kontekst.



Slika 4
.1.1. Dizajn patern “Stanje kandidata” (eGlasač, 2017)

4.2. Dijagram stanja

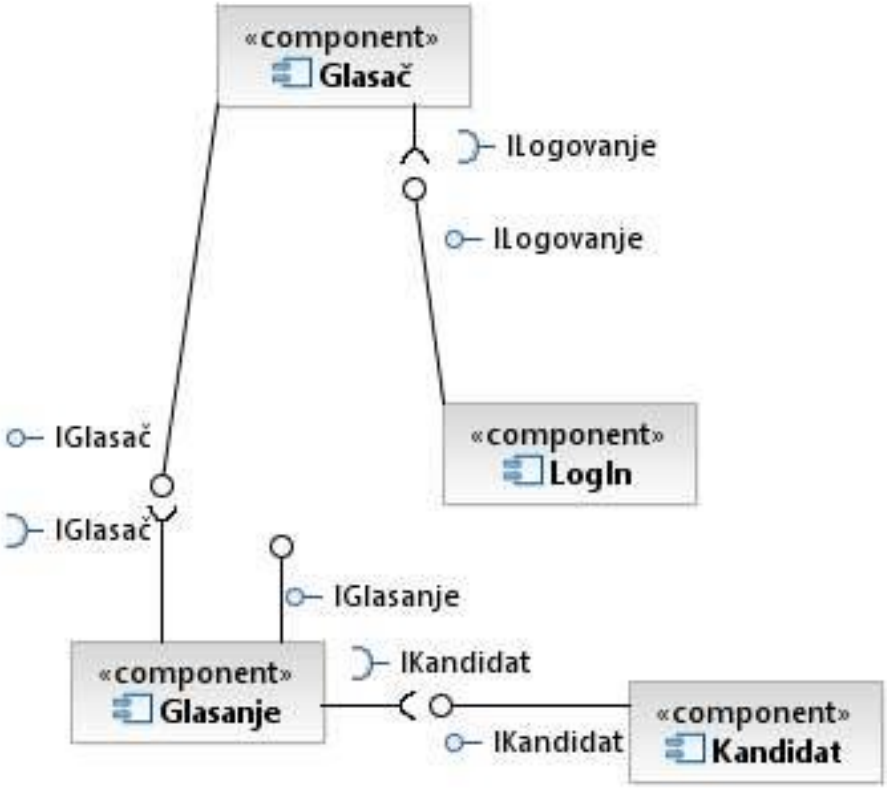
Dijagram stanja (Slika 4.2.1.) prikazuje kako objekat iz jednog stanja prelazi u neko drugo stanje, i pravila koja upravljaju tom promenom. Na sledećoj slici prikazana su stanja kandidata. Kandidat može biti predložen, nakon toga može biti izabran, odnosno ne mora biti izabran. Nakon biranja kandidata proverava se da li broj glasova iznosi više od 51% ili manje. U zavisnosti od toga sistem odlučuje da li kandidat dobija stanje "Vodeći kandidat".



Slika 4.2.1. Dijagram stanja "Stanja kandidata" (eGlasač, 2017)

4.3. Dijagram komponenti

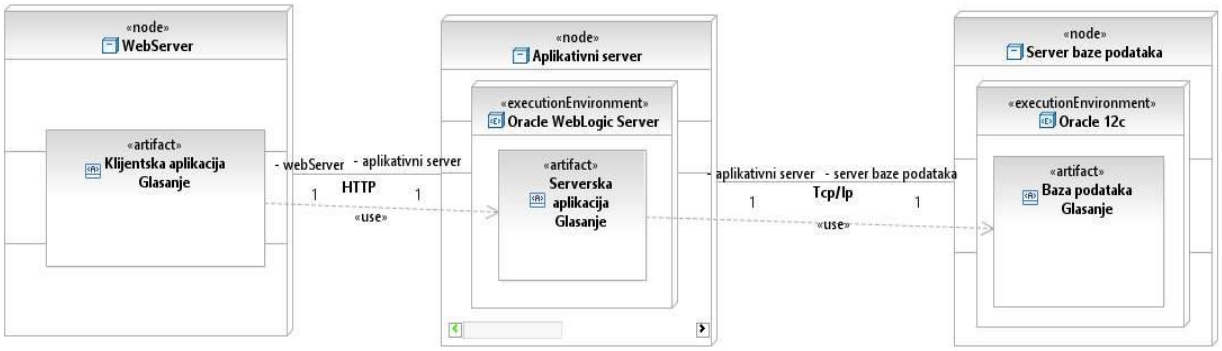
Dijagram komponenti (Slika 4.3.1.) modeluje strukturu softvera ukazujući i na zavisnosti između softverskih komponenti. Prikazuje nam sta komponenta nudi, i šta zahteva. U sledećem primeru vidi se da komponenta Glasač nudi IGlasač, a zahteva ILogovanje, komponenta LogIn u ovom slučaju samo nudi interfejs ILogovanje, komponenta Glasanje nudi interfejs IGlasanje, a zahteva IGlasač i IKandidat, komponenta Kandidat nudi interfejs IKandidat.



Slika 4.3.1. Dijagram komponenti (eGlasač, 2017)

4.4. Dijagram raspoređivanja

Dijagram raspoređivanja (Slika 4.4.1) prikazuje kroz Node objekte WebServer, aplikativni server, i server baze podataka. Zatim u prvom Node-u (WebServer) nalazi se artifact Klijentska aplikacija glasanje, koja je povezana sa artifactom Serverska aplikacija glasanje vezom "usage", njihova dva node-a su povezana vezom asocijacije (HTTP). Zatim artifact Serverska aplikacija glasanje je povezana vezom "usage" sa artifactom Baze podataka glasanje, njihovi node-ovi su povezani takođe vezom asocijacije (Tcp/Ip).



Slika 4.4.1. Dijagram raspoređivanja (eGlasač, 2017)

5. Zaključak

Potrebno je kreirati preostale dijagrame koji su neophodni i napisati detaljne kodove za rad aplikacije, povezati sve segmente i osposobiti aplikaciju za rad sa serverima (Web server, Aplikativni server, Server baze podataka). Za funkcionalni rad aplikacije potrebno je kreirati bazu podataka u kojoj će se nalaziti kandidati, partije i rezultati glasanja. Neophodno je kreirati što jednostavniji GUI kako bi se korisnicima omogućila jednostavna, laka i brza interakcija. Aplikacija koristi MVC arhitekturu, zbog korisničkog interfejsa koji nudi korisniku, pomoću kog on unosi podatke i poziva određene operacije koje treba da se izvrše. Trenutno stanje aplikacije je u alfa fazi, zbog toga je potrebno odraditi sve što je navedeno i unaprediti aplikaciju do beta faze kako bi se prosledila korisnicima na testiranje.

6. Literatura

1. Njeguš, A. (2015) Poslovni informacioni sistemi. Univerzitet Singidunum. Beograd.
2. Njeguš, A. (2017) Informacioni sistemi. Prezentacije sa predavanja. Univerzitet Singidunum. Beograd.
3. MVC, Wikipedia. Dostupno na https://sr.wikipedia.org/wiki/MVC_arhitektura [pregledano 12.11.2017.]

7. POPIS AKRONIMA

Akronim	Naziv na engleskom	Značenje
UML	Unified Modeling Language	Objedinjeni jezik za modelovanje
MVC	Model-view-controller	Arhitektura sistema
GUI	Graphical User Interface	Grafički korisnički interfejs
UI	User Interface	Korisnički interfejs