DIZAJN SISTEMA

ORGANIZACIJA FUDBALSKOG TURNIRA

Verzija: 2.0
Autori: Slobodan Ignjatić, Milanko Petković, Miloš Janjić, Mirko Vidaković
Organizacija: 4MS Software
Datum: 22.09.2017

Sadržaj

1. Uvod

1.1 Namjena sistema

Namjena ovog sistema je da omogući lakšu organizaciju fudbalskog turnira. Povećaće efikasnost i pojednostaviti osnovne funkcije poput vođenja evidencije o ekipama, takmičarima i sudijama, rasporedu utakmica, odigranim utakmicama i njihovim rezultatima. Takođe će omogućiti korisnicima da učestvuju u forumskoj zajednici na kojoj će razmijenjivati svoja mišljenja, glasati na anketama i pristupati najnovijim vijestima. Biće omogućeno upravljanje članovima kao i akcijama članova te zajednice.

1.2 Projektni ciljevi

Organizacija fudbalskog turnira će da omogući vođenje evidencije o ekipama, takmičarima, sudijama i drugim angažovanim licima, kao i rasporedu utakmica, odigranim utakmicama, rezultatima.

1.3 Definicije, skraćenice

DBMS	Skraćenica od Database Management System. Softver namjenjen za upravljanje bazom podataka.
restore podataka	Postupak povratka podataka iz arhive.
requester	Objekat koji zahtjeva uslugu.
provider	Objekat koji pruža usluga.
compile time	Vrijeme koje kompajler utroši na izvršavanje operacija kompajliranja.
run time	Vrijeme izvršavanja programa.
design time	Vrijeme pisanja koda i dizajniranja grafičkog interfejsa.
event - driven	Kontrola toka koja se zasniva na upravljanje događajima.

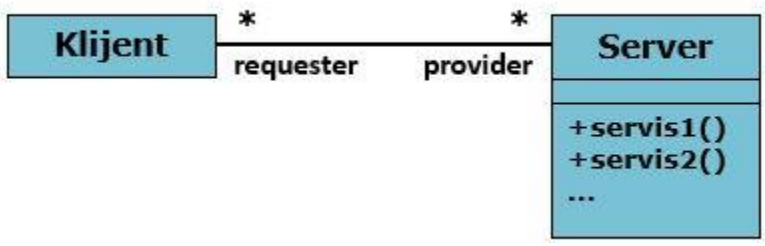
1.4 Referentni dokumenti

[1] Wikipedia, the free encyclopedia, *https://en.wikipedia.org*
[2] Eric Freeman, Elisabeth Robson, Bert Bates, Kathy Sierra - Head First Design Patterns, 2004

1.5 Kratak pregled dokumenta i predložene arhitekture

Ovaj dokument daje potpun opis dizajna sistema. U njemu je navedena arhitektura sistema (koja se zasniva na klijent-server arhitekturnom stilu), kao i svih njegovih podsistema. Prikazani su dijagrami komponenti i razmještaja, konceptualni model za ciljni sistem, statička i dinamička kontrola pristupa (lista mogućnosti za sve učesnike, odnosno, proxy šablon za pristup forumu). Takođe će biti pojašnjena kontrola toka (redoslijed izvršavanja akcija) u sistemu i prikazan dijagram graničnih stanja sistema (granični slučaj upotrebe).

2. Arhitektura postojećeg sistema



Slika 1 - Klijent/Server arhitekturni stil

Sistem će biti zasnovan na klijent/server arhitekturi, gdje jedan ili više servera obezbjeđuje servise klijentima.

Klijent/server arhitektura je razvijena kao:

- višenamjenska
- modularna infrastruktura
- zasnovana na slanju i primanju poruka

Ima za cilj:

- unapređene upotrebljivosti
- felskibilnost
- interoperabilnost
- skalabilnost

3. Predložena arhitektura

3.1 Kratak pregled arhitekture i funkcionalnosti podsistema

U klijent/server arhitekturi možemo uočiti tri klase komponenti: klijent, server i mreža.

Server ima zadatak da optimalno upravlja zajedničkim resursima (najčešće podacima), bazom podataka kojoj pristupa više klijenata, kontolom pristupa i očuvanju integriteta podataka za sve aplikacije.

Klijentske aplikacije vrše upravljanje korisničkim interfejsom i izvršavaju dio logike aplikacije. Klijent unosi podatke, poziva server, server izvršava odgovarajući servis i vraća rezultat klijentu.

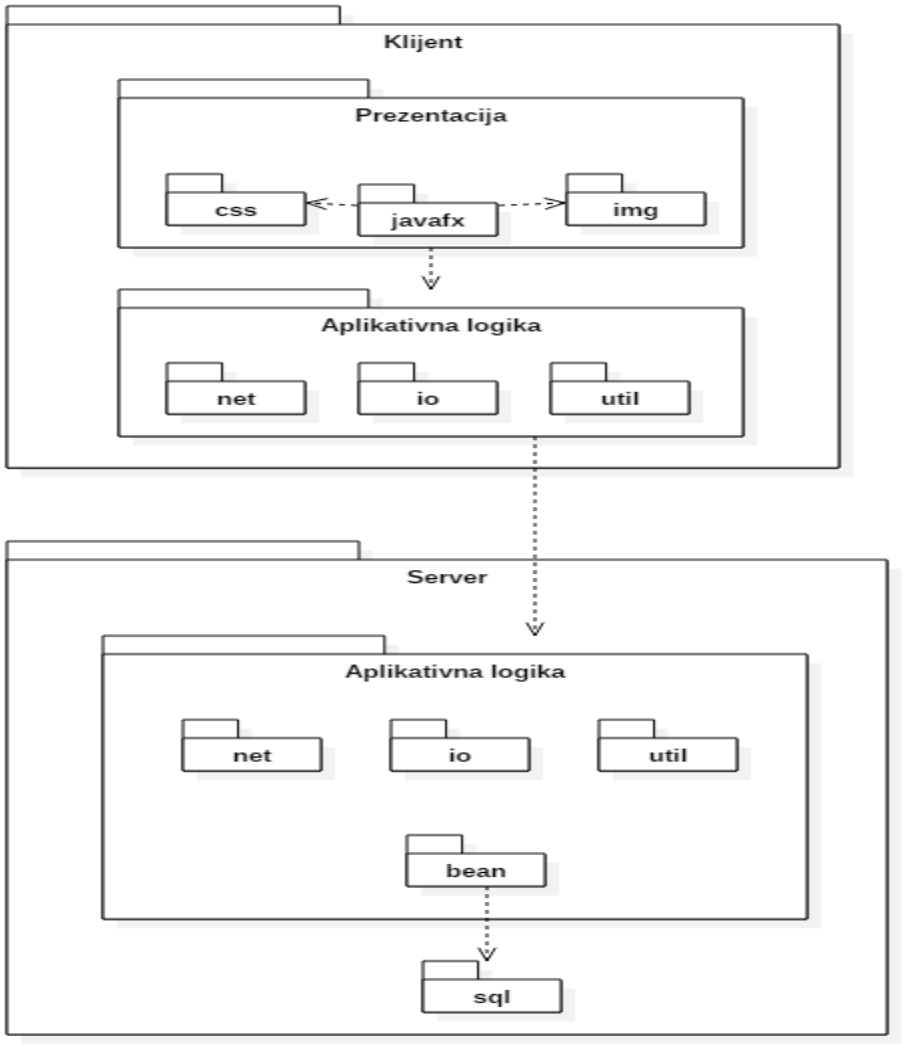
Računarska mreža omogućava prenos podataka između klijenta i servera.

3.2 Dekompozicija sistema

Dvoslojna arhitektura se sastoji od tri komponente distribuirane u dva sloja – klijentskom i serverskom.

Te tri komponente su:

- korisnički interfejs – sesije, unos teksta, dijaloški prozori, prikaz na ekranu
- upravljanje procesima – generisanje, izvođenje i nadgledanje procesa i neophodnih resursa
- upravljanje podacima – servisi vezani za dijeljenje podataka i datoteka



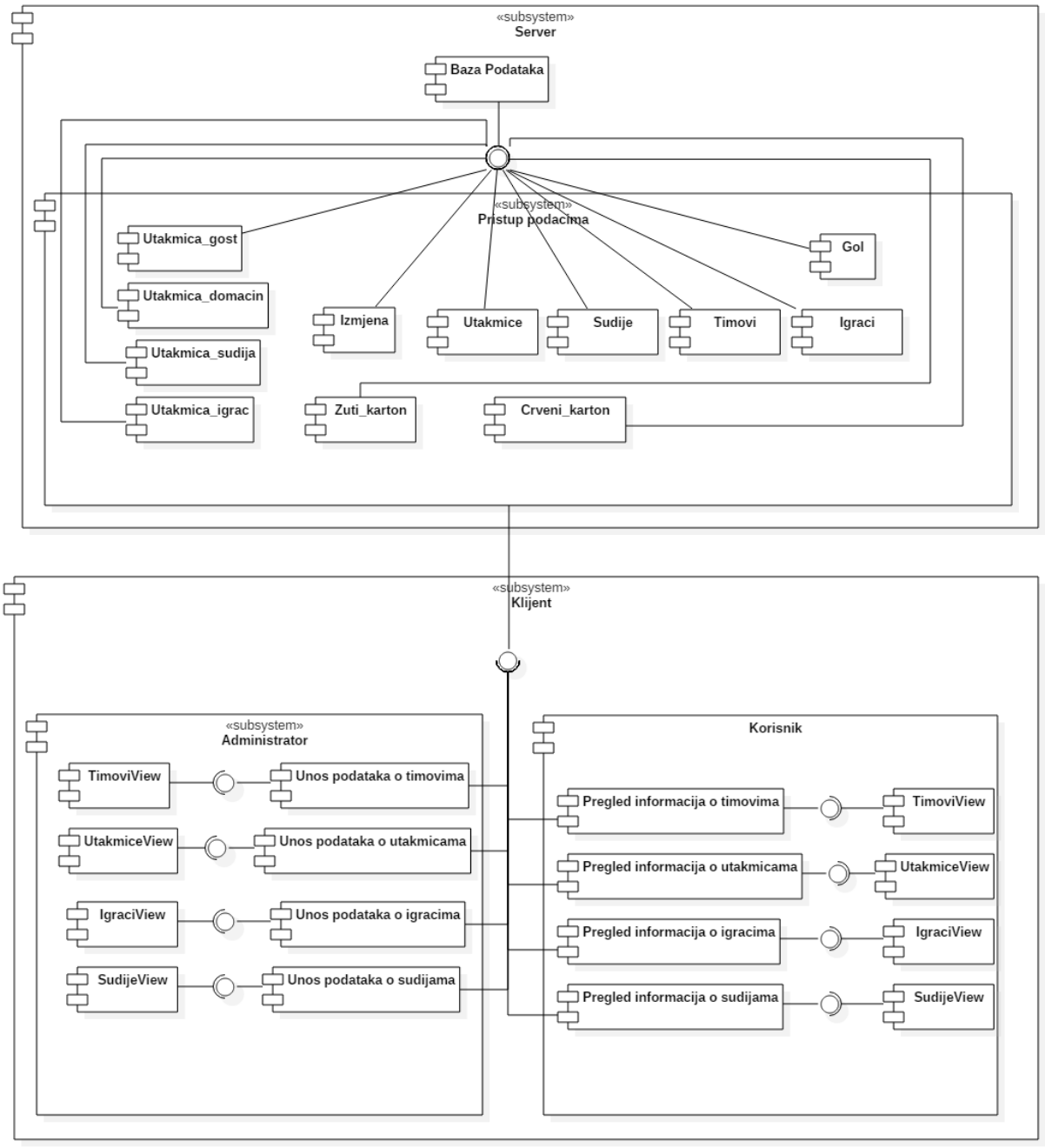
Slika 2 - Dekompozicija sistema (package diagram)

3.2 HW/SW mapiranje

UML dijagrami za HW/SW mapiranje:

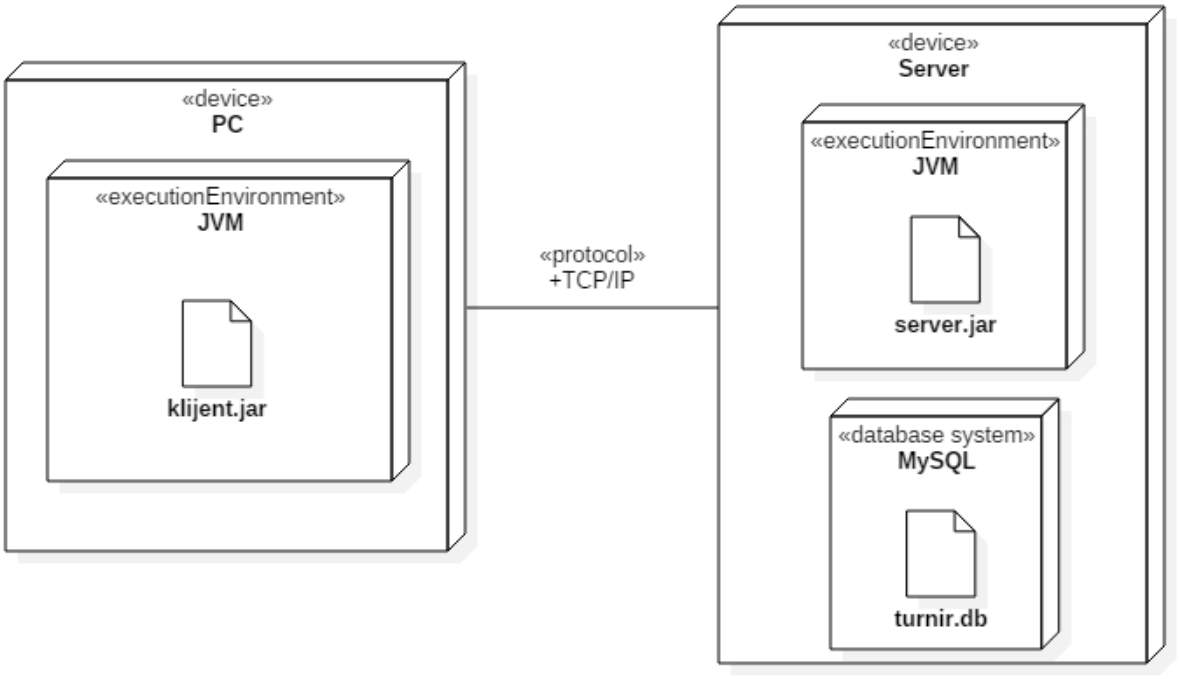
- dijagram komponenata (component diagram)
- dijagram razmještaja (deployment diagram)

Dijagram komponenti se koristi za modelovanje zavisnosti između komponenata u sistemu (design time, compile time i run time).



Slika 3 - Dijagram komponenti (component diagram)

Dijagram razmještaja se koristi za modelovanje rasporeda/razmještaja komponenata u eksploataciji (run time).



Slika 4 - Dijagram razmještaja (deployment diagram)

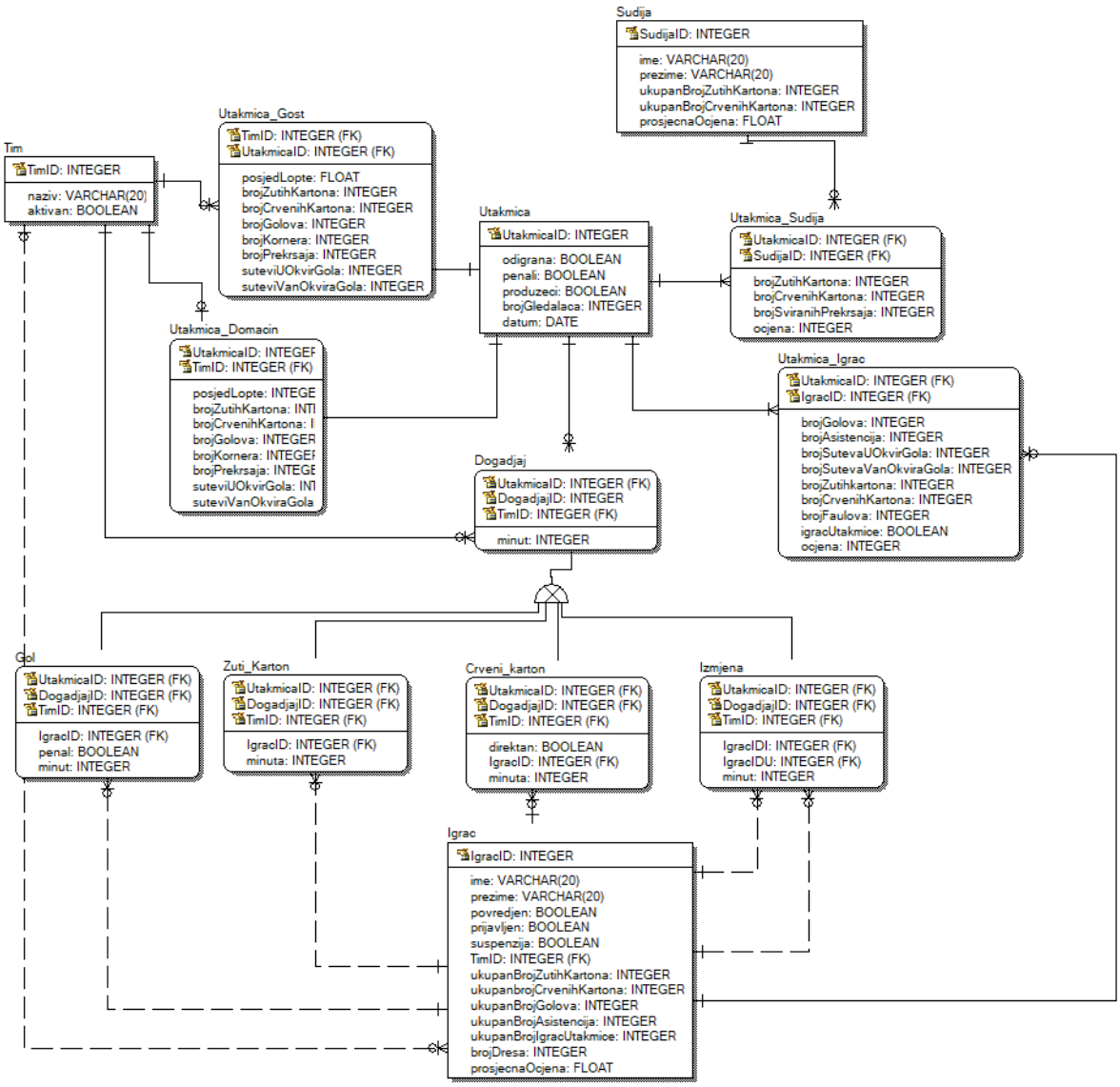
3.4 Perzistentni sloj

Perzistentni (trajni) objekti - objekti čiji je životni vijek duži od jednog izvršavanja aplikacije i čije stanje mora da se sačuva između dva izvršavnja aplikacije.

Manipulacija perzistentim objektima je realizovana pomoću baze podataka (više konkurentnih procesa koji čitaju i/ili upisuju podatke).

Osnovne karakteristike:

- portabilnost, integritet podataka, sigurnost
- high-level I/O



Slika 5 - Konceptualni model sistema

3.5 Kontrola prava pristupa i sigurnost

Kontrola prava pristupa ostvarena je na dva načina: **statički** i **dinamički**. Statička kontrola pristupa realizovana je pomoću liste mogućnosti (učesnik može da izvrši operaciju nad nekim objektom ako lista njegovih mogućnosti sadrži odgovarajući par <klasa, operacija>).

Korisnik	
Anketa	glasaj(odabir:int)
Igrac	getIme():String
Igrac	getPrezime():String
Igrac	getBrojGolova():int
Igrac	getBrojAsistencija():int
Igrac	getBrojZutihKartona():int
Igrac	getBrojCrvenihKartona():int
Igrac	getBrojIgracUtakmice():int
Igrac	getProscjecnaOcjena():float
Sudija	getBrojZutihKartona():int
Sudija	getBrojCrvenihKartona():int
Sudija	getBrojPenala():int
Tim	getNaziv():String
Tim	getAktivan():boolean
Utakmica	getDatum:Date
Utakmica	getBrojGledalaca():int
IgracUtakmica	getBrojGolova():int
IgracUtakmica	getBrojAsistencija():int
IgracUtakmica	isZutiKarton():boolean
IgracUtakmica	isCrveniKarton():boolean
IgracUtakmica	getBrojIgracUtakmice():int
IgracUtakmica	getOcjena():int
IgracUtakmica	isjIgracUtakmice():boolean
IgracUtakmica	getSuteviUOkvirGola():int
IgracUtakmica	getSuteviVanOkviraGola():int
DomacinUtakmica	getBrojZutihKartona():int
DomacinUtakmica	getBrojCrvenihKartona():int
DomacinUtakmica	getBrojGolova ():int
DomacinUtakmica	getPosjedLopte ():float
DomacinUtakmica	getBrojKornera ():int
DomacinUtakmica	getBrojPrekrsaja ():int
DomacinUtakmica	getSuteviUOkvirGola():int
DomacinUtakmica	getSuteviVanOkviraGola():int
GostUtakmica	getBrojZutihKartona():int
GostUtakmica	getBrojCrvenihKartona():int
GostUtakmica	getBrojGolova ():int
GostUtakmica	getPosjedLopte ():float
GostUtakmica	getBrojKornera ():int
GostUtakmica	getBrojPrekrsaja ():int
GostUtakmica	getSuteviUOkvirGola():int
GostUtakmica	getSuteviVanOkviraGola():int
SudijaUtakmica	getBrojZutihKartona():int
SudijaUtakmica	getBrojCrvenihKartona():int
SudijaUtakmica	getBrojPrekrsaja():int

Tabela 1 - Lista mogućnosti (Korisnik)

Administrator	
Igrac	dodajIgraca():boolean
Igrac	izmijeniIgraca():boolean
Igrac	obrisiIgraca():boolean
Tim	dodajTim: boolean
Tim	izmijeniTim: boolean
Tim	obrisiTim: boolean
Utakmica	dodajUtakmicu():boolean
Utakmica	izmijeniUtakmicu():boolean
Utakmica	obrisiUtakmicu():boolean
Sudija	dodajSudiju():boolean
Sudija	izmijeniSudiju():boolean
Sudija	obrisiSudiju():boolean

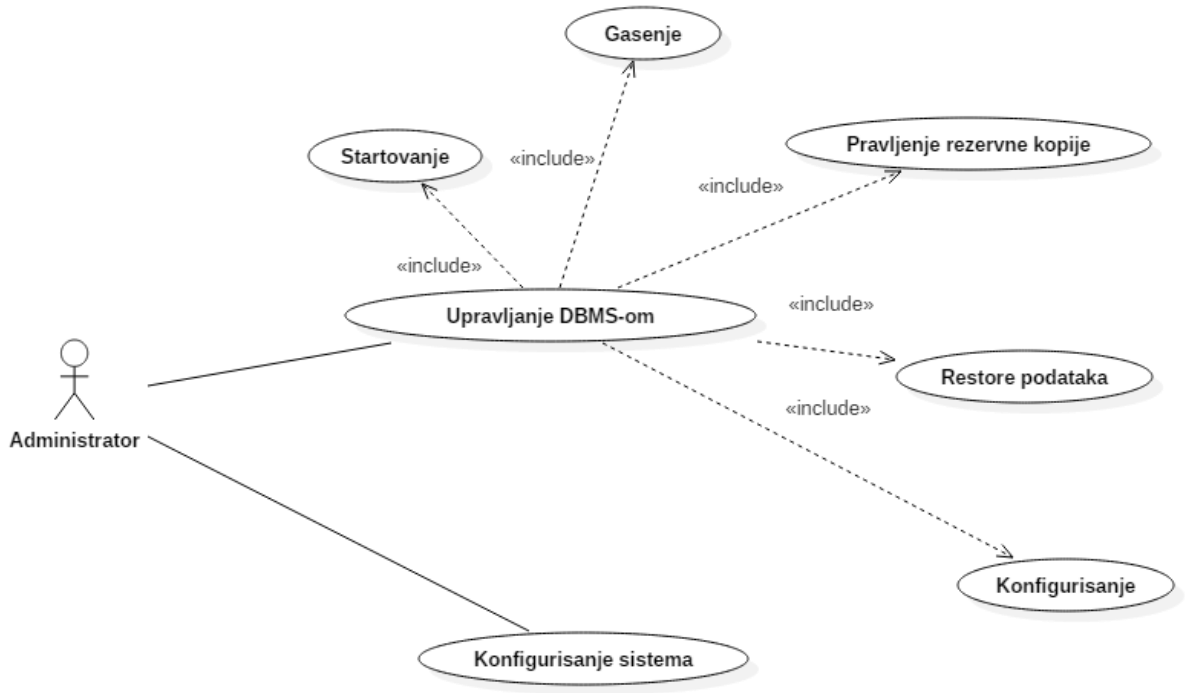
Tabela 2 - Lista mogućnosti (Administrator)

3.6 Kontrola toka

U ovome sistemu kontrola toka se ostvaruje pomoću mehanizma za upravljanje događajima (event-driven). Event-driven arhitektura pomoću centralne jedinice prihvata sve podatke, a zatim ih prosljeđuje odvojenim modulima koji rukovode određenom vrstom događaja. Ova arhitektura je pogodna za sisteme sa korisničkim interfejsom, za asinhrone sisteme sa asinhronim tokom podataka i sisteme gdje jedan blok komunicira sa malim brojem drugih blokova.

3.7 Granična stanja sistema

Granična stanja sistema se modeluju graničnim slučajevima upotrebe (boundary use cases). Alternativno se koristi i termin administrativni slučajevi upotrebe, jer je korespondentni učesnik najčešće sistem administrator.



Slika 6 - Granični slučajevi upotrebe

Granična stanja sistema obuhvataju akcije koje se izvršavaju veoma rijetko. Te akcije se odnose na upravljanje bazom podataka koje uključuje startovanje baze podataka koja se nalazi u sklopu servera, kao i njeno gašenje pri konfigurisanju servera, pravljenje rezervne kopije (nakon određenog vremenskog perioda) radi očuvanja integriteta podataka i omogućavanja *restore* podataka.