Analiza i modeliranje podvorbenog sustava s mogućnošću predbilježbe

Manolić, Kristian

SVEUČILIŠTE U ZAGREBU

FAKULTET PROMETNIH ZNANOSTI

Kristian Manolić

ANALIZA I MODELIRANJE PODVORBENOG SUSTAVA S MOGUĆNOŠĆU PREDBILJEŽBE

DIPLOMSKI RAD

ZAGREB, 2020.

SVEUČILIŠTE U ZAGREBU
FAKULTET PROMETNIH ZNANOSTI
POVJERENSTVO ZA DIPLOMSKI ISPIT

Zagreb, 16. ožujka 2020.

Zavod: **Zavod za informacijsko komunikacijski promet**
Predmet: **Analiza i modeliranje prometnih sustava**

DIPLOMSKI ZADATAK br. 5555

Pristupnik: **Kristian Manolić (0135236955)**
Studij: Promet
Smjer: Informacijsko-komunikacijski promet

Zadatak: **Analiza i modeliranje podvorbenog sustava s mogućnošću predbilježbe**

Opis zadatka:

Opisati značajke modeliranja procesa primjenom UML jezika te definirati zahtjeve podvorbenih sustava s mogućnošću predbilježbi. Identificirati objekte takvih sustava i opisati njihova ponašanja, inteakcije i unutarnja stanja. Koristeći različite poglede na sustav, modelirati interakcije između sudionika različitih slučajeva upotrebe podvorbenih sustava s mogućnošću predbilježbi. Na primjeru prikazati primjenu modela podvorbenih sustava s mogućnošću predbilježbi.

Mentor:

doc. dr. sc. Marko Matulin

Predsjednik povjerenstva za diplomski ispit:

Sveučilište u Zagrebu

Fakultet prometnih znanosti

DIPLOMSKI RAD

ANALIZA I MODELIRANJE PODVORBENOG SUSTAVA S MOGUĆNOŠĆU PREDBILJEŽBE

ANALYSIS AND MODELING OF THE QUEUING SYSTEM WITH RESERVATION CAPABILITIES

Mentor: doc.dr.sc. Marko Matulin

Student: Kristian Manolić, bacc.ing.traff.

JMBAG: 0135236955

Zagreb, rujan 2020.

SAŽETAK

Svrha istraživanja u ovom radu je analizirati i prikazati značajke podvorbenih sustava s mogućnošću predbilježbi te definirati glavne zahtjeve i procese unutar takvog sustava. Predbilježbama se nastoji potaknuti korisnika da dolazi u vrijeme kada poslužitelj ima manje posla, čime se nastoji skratiti vrijeme čekanja korisnika i povećati efikasnost iskorištenja kapaciteta repa. Svi modeli takvog sustava su opisani odgovarajućim UML (eng. *Unified Modeling Language*) modelima. Na temelju izrađenih modela opisanih ovim radom, izrađena je web aplikacija koja omogućuje korisnicima da se predbilježe za željeni termin i upravljaju svojim predbilježbama. Aplikacija je izrađena pomoću MVC (eng. *Model-View-Controller*) obrasca razvoja web aplikacija. Sama aplikacija je izrađena pomoću programskog alata Microsoft Visual Studio 2019, dok je za izradu baze podataka korišten Microsoft SQL Server Management Studio 2014.

KLJUČNE RIJEČI: podvorbeni sustav; UML; modeliranje; web aplikacija;

SUMMARY

The purpose of this thesis is to analyze and present the features of queuing system with reservation capabilities and define the main requirements and processes within such a system. Reservation is used to encourage users to come at a time when the server has less work, which results in shorten user waiting times and increase the efficiency of tail capacity utilization. All models of such a system are described by the corresponding UML (Unified Modeling Language) models. Based on the models described in this paper, a web application was created that allows users to register for the desired date and manage their resevations. The application was created using the MVC (Model-View-Controller) web application design model. The application itself was created using the Microsoft Visual Studio 2019 software tool, while Microsoft SQL Server Management Studio 2014 was used to create the databases.

KEY WORD: queuing system; UML; modeling; web application;

Sadržaj

1. Uvod

Sustavi naručivanja korisnika neke usluge, tj. podvorbeni sustavi s mogućnošću predbilježbi, su sustavi koji omogućavaju upravljanje vremenom i trajanjem pružanja neke usluge korisnicima. Predbilježbama se nastoji potaknuti korisnika da dolazi u vrijeme kada poslužitelj ima manje posla, čime se nastoji skratiti vrijeme čekanja korisnika i povećati efikasnost iskorištenja kapaciteta repa. Kako bi se olakšao proces analize i modeliranja navedenih sustava, sustav se dijeli u više lako shvatljivih modela, što omogućuje lakšu komunikaciju, razradu i implementaciju rješenja. Za prikaz takvih malih dijelova sustava se koristi jezik za ujedinjeno modeliranje UML (eng. *Unified Modeling Language*).

Svrha ovog istraživanja je analizirati i prikazati značajke podvorbenih sustava s mogućnošću predbilježbi te definirati glavne zahtjeve i procese unutar takvog sustava. Svi zahtjevi i procesi unutar sustava su prikazani pomoću jezika za ujedinjeno modeliranje (UML). Na temelju izrađenih modela sustava prikazanih pomoću UML-a izrađena je web aplikacija.

Diplomski rad se sastoji od devet funkcionalno povezanih teza:

1. Uvod
2. Značajke modeliranja procesa primjenom UML-a
3. Definiranje zahtijeva na sustava s mogućnošću predbilježbi
4. Modeliranje svojstva, ponašanja i stanja objekta
5. Modeliranje toka podataka i toka upravljanja primjenom dijagrama aktivnosti
6. Modeliranje interakcije između sudionika
7. Modeliranje implementacije sustava
8. Web aplikacija podvorbenog sustava s mogućnošću predbilježbe
9. Zaključak.

U drugom poglavlju rada su objašnjeni glavni pojmovi u procesu modeliranja sustava te su objašnjeni pogledi na promatrani sustav i faze razvoja sustava.

U trećem poglavlju je objašnjena glavna problematika sustava s mogućnošću predbilježbe. Pomoću dijagrama su prikazani glavni zahtjevi na jedan takav sustav.

U četvrtom poglavlju obrađena je tematika dijagrama klasa i dijagrama stanja i prikazana su sva svojstva, ponašanja i stanja objekta u podvorbenim sustavim s mogućnošću predbilježbe.

Modeliranje toka podataka i toka upravljanja primjenom dijagrama aktivnosti je naziv petog poglavlja ovog rada. U njemu su prikazane sve aktivnosti koje su potrebne za uspješno obavljanje rada sustava.

U šestom poglavlju prikazani i objašnjeni su dijagrami međudjelovanja i dijagrami suradnje koji opisuju cijeli predbilježbe termina.

Modeliranje implementacije sustava je sedmo poglavlje ovog rada. U njemu su opisani dijagrami komponenata i dijagrami rasporeda te su pomoću dijagrama komponenata prikazane sve fizičke komponente sustava i njihova logička povezanost.

U osmom poglavlju je prikazana izrađena sama web aplikacija na temelju dobivenih generičkih modela sustava.

U zadnjem poglavlju autor daje svoj konačni komentar na cijelu obrađenu tematiku rada.

2. Značajke modeliranja procesa primjenom UML-a

Model je pojednostavljeni prikaz nekog stvarnog objekta ili sustava. Naglašava samo bitne značajke nekog objekta ili sustava, dok se ostale značajke pojednostavljuju ili se u potpunosti zanemaruju. Prilikom izgradnje modela objekt ili sustav se promatra s više stajališta, pružajući detaljan uvid u sustav u različitim trenucima. Modeli programskih sustava izrađuju se u jeziku za modeliranje, poput UML-a, [1].

Jezik za ujedinjeno modeliranje, skraćeno UML je alat, pomagalo koje je prvenstveno namijenjeno programskoj podršci za vizualizaciju, opis, izgradnju i dokumentaciju programskog sustava. UML nastoji pružiti dobro definirane koncepte modeliranja programskih sustava, upotrebom raznih pravila, radi bolje komunikacije i izgradnje modela. Glavni cilj je stvoriti model razumljiv svim članovima tima, a da pritom budu jasno definirane funkcionalnosti i arhitektura sustava, [2].

UML dolazi iz objektno orijentirane pozadine, odnosno može se smatrati nasljednikom objektno orijentiranog dizajna i analize. Objekti su entiteti iz stvarnog svijeta, a osnovni pojmovi poput apstrakcije, enkapsulacija, nasljeđivanja i polimorfizma se mogu predstaviti pomoću UML-a. Objekti i klase zajedno čine statičku strukturu sustava, koja uključuje i međusobne odnose između različitih objekta u sustavu. Klase pružaju definicije za format podataka i procedura za određenu vrstu objekta ili klase. Može se reći da je objekt instanca klase, odnosno da objekt posjeduje sva svojstva i metode koje su opisane klasom tog objekta, [3].

S obzirom da UML dolazi iz objektno orijentirane pozadine nužno je najprije poznavati osnovne koncepte objektno orijentiranog pristupa, a to su apstrakcija, enkapsulacija, nasljeđivanje i polimorfizam. Apstrakcija je pojednostavljenje objekta iz stvarnog svijeta, gdje se promatraju samo bitne značajke za opis objekta koje su potrebne za pravilno modeliranje i izradu sustava. Enkapsulacija je prekrivanje podataka na način kojim se podaci i metode organiziraju u određenu strukturu kako bi sakrili način na koji su implementirani u sustav, [4]. Nasljeđivanje omogućuje određenoj klasi da naslijedi svojstva, metode, podatke i događaje iz neke druge klase. Time se dopušta određenoj klasa da posvoji člana druge klase. Klasa koja nasljeđuje člana neke druge klase naziva se izvedenom klasom, a klasa od koje se nasljeđuju članovi osnovnom klasom. Stvaranjem odnosa između različitih klasa stvara se hijerarhija klasa, [5]. Polimorfizam je svojstvo objekta da promijeni svoj oblik, odnosno tip podataka.

Svaki objekt izvedene klase se može predstaviti kao objekt osnovne klase na određenim mjestima kao što su npr. parametri u metodi. U tom slučaju deklarirani tip podataka više nije isti kao oni koji se pozivaju prilikom izvođenja određene metode, [6].

Modeliranje je jedna od najbolje prihvaćenih i dokazanih inženjerskih tehnika. Model arhitekture sustava pomaže u prezentiranju konačnog proizvoda korisniku. Modeliranje je značajnije za velike sustave, jer se takvi sustavi ne mogu shvatiti u potpunosti. Kako bi se postigao bolji uvid u sustav, modelari se usredotočuju na samo jedan aspekt/pogled sustava. Time je omogućeno modelaru da radi na većoj razini apstrakcije, što u konačnici rezultira boljim modelom sustava, [7].

Na temelju modela se donose planovi o implementaciji, uključujući predviđene troškove i utrošene resurse. Donošenje preciznih predikcija i procjena je skoro pa nemoguće ako je model loše napravljen. Svaki dobar model mora biti: točan, razumljiv, konzistentan i promjenjiv. Prilikom modeliranja modelar mora obratiti pažnju na potrebe, funkcionalnosti i strukturu sustava. Sve potrebe i funkcionalnosti sustava zapravo predstavljaju zahtjeve na promatrani sustav. Modeli često nisu direktno vidljivi od strane modelara i korisnika, ali se mogu izraziti putem dijagrama, teksta ili kombinacijom teksta i dijagrama, [2].

U praksi često nije moguće opisati sustav kroz samo jedan model, iz tog razloga se prilikom analize i dizajna sustava koristi više modela. Kako bi što bolje identificirali zahtjeve na sustav potrebno je koristi što više pogleda ili gledišta na sustav. Postoji pet glavnih pogleda ili gledišta na sustav, a to su:

1) Pogled slučaja uporabe – ovim pogledom se nastoji dati uvid u funkcionalnosti koje sustav treba zadovoljavati. Funkcionalnosti na sustav su definirane od strane aktera u sustavu, odnosno s njihovog gledišta na sustav. Akteri u sustavu mogu biti korisnici sustava ili neki drugi sustav. Ovo je najvažniji pogled na sustav jer se na temelju njega baziraju svi drugi pogledi na sustav, najčešće se iskazuje pomoću dijagrama slučaja uporabe, a rjeđe pomoću dijagrama aktivnosti.

2) Logički pogled – njime se opisuje na koji način sustav pruža svoje funkcionalnosti, odnosno na koji način su ostvarene. Logički pogled daje uvid u statičku strukturu sustava (objekti, klase i međusobni odnosi između klasa/objekta) i dinamičku suradnju koja nastaje kada dva objekta u sustavu razmjenjuju poruke kako bi ostvarili određenu funkcionalnost sustava. Statička struktura se prikazuje dijagramom klasa i objekta, a dinamička struktura se prikazuje dijagramom stanja i dijagramom aktivnosti.

3) Pogled ostvarenja – usmjeren je na elemente kojima se sustav ostvaruje te opisuje njihovu strukturu i međusobnu povezanost. U ovom pogledu se pružaju dodatne informacije o programskim kodovima, bazama podataka i svoj popratnoj administrativnoj dokumentaciji.

4) Pogled procesa – usmjeren je na podjelu sustava u više procesa, koji se izvršavaju istovremeno te imaju sposobnost komuniciranja jedni s drugim. Svrha ovog pogleda je povećanje efikasnosti korištenja resursa sustava, istovremeno izvršavanje više procesa i bolje baratanje asinkronim zahtjevima postavljenim na sustav, [2].

5) Pogled rasporeda – daje uvid u prostorni smještaj razvijene programske i sklopovske podrške koja čini sustav, [7].

UML dijagrami imaju široku primjenu u modelima životnog ciklusa programskog sustava. Jedan od takvih modela je ujedinjeni proces ili skraćeno UP (eng. *Unified Process*). To je radni tijek razvoja aplikacija koji pruža disciplinarni pristup dodjele radnih zaduženja unutar razvojne organizacije. Sastoji se od šest najboljih praksa u razvoju sustava kao što su: razvijanje iteracijom, upravljanje zahtjevima, primjena arhitekture komponenata, vizualno modeliranje, provjera kvalitete sustava i upravljanje promjenama, [8].

Model ujedinjenog procesa se sastoji od četiri faze razvoja: početna faza, faza razrade, faza izrade i faza prijelaza. U početnoj fazi određuje se poslovna svrha sustava, identificiranje glavnih aktera u sustavu i njihova međusobna interakcija. Faza razrade analizira potencijalne probleme u sustavu, uspostavlja čvrstu arhitekturu sustava, plan projekta i eliminira elemente koji predstavljaju rizik u radu sustava. U fazi izrade se sve prestale komponente i značajke implementiraju u proizvod ili sustav. Svaka značajka i komponenta se temeljito testira prije implementacije. U fazi prijelaza proizvod ili sustav se pruža krajnjem korisniku na korištenje, potom se otklanjaju potencijalne poteškoće i greške u sustavu koje nisu bile uočene ranije, [9].

Trenuci prelaska iz faza se nazivaju prekretnice. U tim trenucima projektni menadžer odnosno poslovođa donosi odluke o daljnjem nastavljanju ili otkazivanju projekta. Odluka poslovođe se može temeljiti na preostalom opsegu posla, financijskim troškovima, preostalim financijskim sredstvima, predviđeni plan rada ili nekim od drugih čimbenika.

Osim ranije navedenih faza razvoja sustava, ujedinjeni proces prolazi i kroz šest radnih tijekova. Radni tijek se sastoji od niza povezanih skupa aktivnosti koje izvode suradnici na

projektu. Opisuje se odgovarajućim UML modelima za pridruženi radni tijek u pojedinim fazama razvoja, [7]. Sustav u svom životnom ciklusu prolazi kroz sve faze razvoja i pripadajućih radnih tokova više puta, a u svakoj iteraciji sustava može se naglasiti drugačiji radni tijek. Prema [10], šest glavnih radnih tijekova su:

1. Radni tijek poslovnog modeliranja – u njemu se određuje vizija organizacije u kojoj će sustav biti implementiran i na temelju te vizije odrediti odgovarajuće procese, uloge i zaduženja, [10].

2. Radni tijek prikupljanja zahtijeva – definira koje funkcionalnosti sustav mora zadovoljavati, s obzirom na definirane korisničke zahtjeve. Model slučaja uporabe se definira u početnoj fazi sustava, gdje se zahtjevi na sustav gledaju na visokoj osnovi, [7].

3. Radni tijek analize i dizajna – glavna svrha je pokazati na koji način sustav ostvaruje pojedinu funkciju koja je definirana u modelu slučajeva uporabe. To rezultira modelom analize koji pokazuje koje su glavne komponente sustava i na koji način komponente međusobno komuniciraju.

4. Radni tijek izrade – glavna svrha radnog tijeka izrade je odrediti komponente koje se implementiraju u sustav i odabrati odgovarajuću arhitekturu komponenata. Ujedinjeni proces opisuje način na koji se određene komponente mogu ponovno iskoristiti ili implementirati, u svrhu lakšeg održavanja sustava.

5. Radni tijek provjere sustava – opisuje na koji način će se izvršiti provjera izgrađenog sustava, [7]. Provjera sustava se odvija u svim fazama ujedinjenog procesa kako bi se što ranije pronašle poteškoće i greške u sustavu, [10].

6. Radni tijek implementacije – glavna zadaća je osigurati isporuku izgrađenog sustava ili aplikacije korisnicima.

Osim šest glavnih tokova [10] definira i tri potporna radna tijeka, a to su:

1. Radni tijek okoline – opisuje potrebne aktivnosti koje su potrebne za prilagodbu ujedinjenog procesa organizaciji. To uključuje sve procese i potporne alate koji su potrebni za uspostavu odgovarajuće okoline razvoja sustava.

2. Radni tijek upravljanja konfiguracijom i promjenama – usredotočuje se na dobru komunikaciju unutar tima. Zbog potencijalno velikog broja osoblja koji rade na istom projektu može doći do izgubljenih izmjena na sustavu, iz tog razloga potrebno je naglasiti svaku izmjenu sustava ili aplikacije. Radni tijek upravljanja konfiguracijom i promjenama promatra tri specifična područja: upravljanje konfiguracijom, upravljanje zahtjevima za promjenom i upravljanje statusom i promjenama. Upravljanje konfiguracijom je odgovorno za sustavno strukturiranje proizvoda odnosno aplikacije. Upravljanje zahtjevima za promjenom brine se za pravilno obilježavanje i pospremanje svake verzije sustava, a za upravljanje i praćenje zahtjeva brine se CRM (eng. *Customer relationship management*). Upravljanje statusom i promjenama se brine o dobroj dokumentaciji prijašnjih verzija sustava i praćenju razvoja sustava, [10].

3. Radni tijek upravljanja projektom – Osim samog planiranja razvoja sustava, kao što su planiranje iteracija i faza procesa razvoja sustav, projektni upravitelj ili poslovođa ne smije zaboraviti na upravljanje osobljem, dobavljačima i resursima. Treba voditi brigu o konstantom usavršavanju zaposlenika, [10].

Kroz sljedeća poglavlja bit će prikazani, opisani i objašnjeni UML dijagrami na primjeru podvorbenog sustava s mogućnošću predbilježbe. Dobiveni generički UML modeli će se primijeniti u 8. poglavlju ovog rada na specifičnom primjeru predbilježbe termina na liječnički pregled od strane korisnika, za koji je izrađena web aplikacija.

Izrađena web aplikacija prati navedene faze razvoje modela i radne tijekove. U početnoj fazi razvoja sustava su identificirani glavni akteri unutar sustava, objekti i njihova stanja. U ovoj fazi su napravljeni dijagrami slučaja uporabe, klasa i stanja, koji su prikazani u trećem i četvrtom poglavlju ovog rada. Ovom fazom je obuhvaćen radni tijek poslovnog modeliranja i prikupljanja zahtijeva. U fazi razrade su detaljno analizirani problemi sa kojima se sustav suočava. U ovoj fazi su razrađeni dijagrami aktivnosti, međudjelovanja i suradnje, navedeni dijagrami sustava su prikazani u petom i šestom poglavlju ovog rada. Ovom fazom je obuhvaćen radni tijek analize i dizajna. U fazi izrade izrađen je dijagram komponenata koji pokazuje koje fizičke komponente su uključene u radu sustava te je izrađena sama aplikacija sustava. Dijagram komponenata i aplikacija su prikazani u sedmom i osmom poglavlju ovog rada. Time je obuhvaćen radni tijek izrade sustava. Faza prijelaza, radni tijekovi provjere sustava i implementacije te svi potporni radni tijekovi su van opsega ovog rada, zbog

potencijalno velikog broja korisnika i potrebitih resursa koji su potrebni za provedbu navedenih radnih tijekova i faze razvoja, za navedeni specifični primjer.

3. Definiranje zahtijeva na sustava s mogućnošću predbilježbi

Kvaliteta usluge je jedna od najbitnijih stavki svakog poduzeća koje se bavi pružanjem usluga. Ako korisnik nije zadovoljan svojim iskustvom pružanja usluge u većini slučajeva više neće koristiti usluge tog poduzeća. Dva glavna parametra koji utječu na kvalitetu pružanja usluge u podvorbenim sustavima su vrijeme provedeno u redu čekanja i vrijeme trajanja posluživanja korisnika.

Upotrebom predbilježbi u podvorbenim sustavima se nastoji utjecati na proces dolazaka korisnika pred sustav. Korisnici bi u tom slučaju trebali dolaziti pred sustav u vremenskim jedinicama koje su jednake vremenu trajanja posluživanja jednog korisnika. Time se nastoji vrijeme čekanja korisnika u redu svesti na nulu. Predbilježbe se koriste kada se ne može utjecati na proces posluživanja korisnika. Razlog tomu može biti: nedovoljno resursa sustava, gleda se samo na smanjenje troškova, nebriga za korisnike, itd. Zbog nemogućnosti utjecanja na proces posluživanja korisnici su prisiljeni čekati u redu na posluživanje, što posljedično uzrokuje nezadovoljstvo korisnika.

Problem je u tome što korisnici često ne mogu doći u vrijeme kada je poslužitelj slobodan ili je vremenski rok u kojem poslužitelj pruža uslugu kratak. Većina korisnika je u mogućnosti doći u isto doba dana, odnosno prije ili poslije svog radnog vremena, što posljedično uzrokuje dolazak velikog broja korisnika u kratkom vremenskom intervalu, a poslužitelj zbog nebrige ili manjka resursa ne može posluživati korisnike prikladnom brzinom posluživanja. Navedeni problem se najčešće pokušava riješiti primjenom predbilježbi.

Za pomoć pri uklanjanju navedenih problema u ovakvom tipu podvorbenih sustava u 8. poglavlju je predložena web aplikacija na specifičnom primjeru sustava predbilježbe termina na liječnički pregled. Aplikacija je zamišljena tako da poslužitelj, u ovom slučaju liječnik, otvara slobodne termine za koje se korisnik može predbilježiti putem odgovarajućeg sučelja. Liječnik putem aplikacije ima uvid u sve svoje predbilježene termine te ima mogućnost otvoriti veći broj termina odjednom. Korisnik nakon što stvori svoj račun ima mogućnost pregledavati raspoložive termine i predbilježiti se za njih. Korisnika aplikacija automatski obavještava putem e-pošte, da se predbilježio za određeni termin ili da je termin otkazan.

3.1. Dijagram slučajeva uporabe

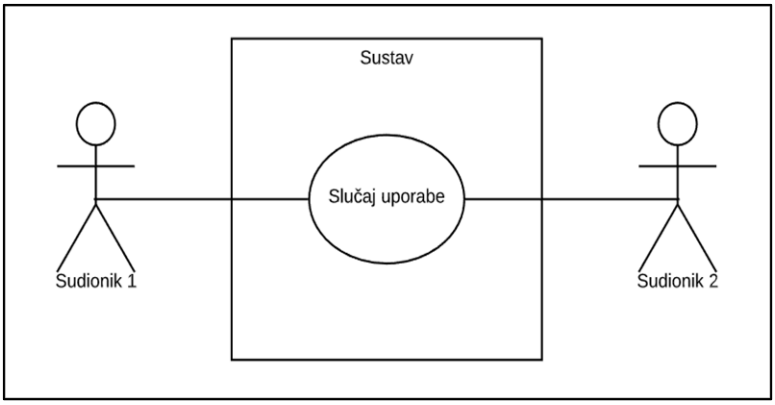
Osnovna namjena dijagrama slučajeva uporabe (eng. *Use case diagrams*) je određivanje funkcija sustava, definiranih od strane internih ili eksternih zahtjeva. Dijagramima slučajeva uporabe se prikazuje što sustav radi i koji su njegovi glavni korisnici. On uvijek mora biti usredotočen na funkcionalnosti koje su od značaja za krajnjeg korisnika. Model slučaja uporabe izrađuje se na temelju komunikacije između modelara i korisnika (krajnji korisnici, naručitelj sustava, itd.) koji se međusobno dogovaraju oko funkcionalnosti sustava. Dijagram slučajeva uporabe stavlja naglasak na brzo prikupljanje korisničkih zahtjeva i lakoj izmjeni početnog modela, [2].

Prikupljanje korisničkih zahtijeva često može biti dugotrajan i opsežan proces, zavisno o veličini i kompleksnosti sustava. Za opis odnosa između korisnika i sustava koristi se UML dijagram slučajeva uporabe koji pomaže u vizualizaciji i opisu zamišljenog ili postojećeg sustav. Dobiveni model slučaja uporabe je od velikog značaja za programsku podršku jer pruža jasan uvid u glave funkcionalnosti koje sustav treba zadovoljavati, što pomaže u planiranju razvoja sustava. On je i od značaja i za marketinški tim jer daje jasan uvid u glavne značajke prema kojima se može napraviti marketinški plan.

Akter predstavlja ulogu korisnika koji komunicira sa sustavom. Korisnik može biti osoba, organizacija, stroj ili drugi vanjski sustav. Jednim akterom se može zastupati više korisnika sustava, a svaki korisnik može imati više uloga. Korisnici pokreću određena ponašanja opisana dijagramom slučajeva uporabe te razmjenjuj podatke sa sustavom. U dijagramima koji opisuju softverske aplikacije, akteri predstavljaju krajnje korisnike, vanjske sustave ili strojeve koji komuniciraju sa sustavom, [11].

Slučaj upotrebe opisuje funkciju koju sustav obavlja kako bi postigao korisnikov cilj. Slučaj upotrebe mora dati vidljiv rezultat koji je koristan korisniku sustava. Slučajevi upotrebe sadrže detaljne informacije o sustavu, korisnicima sustava i odnosima između sustava i korisnik. Slučajevi upotrebe ne opisuju načina na koji je sustav implementiran ili kako se ostvaruje pojedina funkcija sustava. Svaki slučaj upotrebe opisuje određeni cilj za korisnika i način na koji korisnik ostvaruje određeni cilj pomoću interakcije sa sustavom. Slučaj upotrebe opisuje sve načine prema kojim sustav može ili ne može ostvariti zadani cilj od strane korisnika, [11]. Dijagramom slučajeva uporabe prikazuju odnose između sudionika i slučaja uporabe. Sudionici se označavaju pomoću simbola osobe, a slučaj uporabe se označava elipsom i kratim opisom

slučaja uporabe unutar ili pokraj elipse, [12]. Slikom 1 prikazani su simboli slučaja uporabe i sudionika u dijagramu slučaja uporabe.



Slika 1. Simboli slučaja uporabe i sudionika u dijagramu slučaja uporabe

Na lijevu stranu dijagram se stavljaju sudionici ili akteri koji izravno utječu ili izvode slučaj uporabe. Slučaj uporabe se stavlja u središte, a ostali sudionici se stavljaju s desne strane dijagrama. Strelica ili linijama se pokazuje koji sudionici sudjeluju u slučaju uporabe, [12]. Slučajevi uporabe se stavljaju u prazan kvadrant koji predstavlja granice sustav i obuhvaća sve slučaje uporabe koji se nalaze u njemu.

Ponašanje svakog sustava se može opisati na više načina. Najčešće se opisuje jednim od tri načina: neformalnim tekstom, formalnim tekstom s opisom tijeka događanja i pseudo kodom. Opis sustava mora točno specificirati odnose između slučaja uporabe i korisnika. U tekstu se ne opisuje na koji način se ostvaruju pojedine funkcije sustava. On mora biti napisan na takav način da je lako razumljiv krajnjem korisniku i modelaru, [2].

Tekstom se mogu opisati različiti tijekovi rada sustava, a to su: osnovni tijek događaja, zamjenski tijek događaja i iznimni tijek događaja. Osnovni tijek događaja opisuje kroz koje okolnosti prolaze sudionik i sustav u normalnim okolnostima. Od početka do kraj se izvodi kako je opisan, predstavlja poželjan tijek događaja i pretpostavlja da korisnici i sustav ne rade pogreške. Zamjenski tijek opisuje dodatna ponašanja sustava. Iznimni tijek događaja se izvodi u slučaju pojave pogreške ili neispunjavanja zadanih uvjeta. Slučaj uporabe uvijek sadrži barem jedan iznimni tijek događaja, [12].

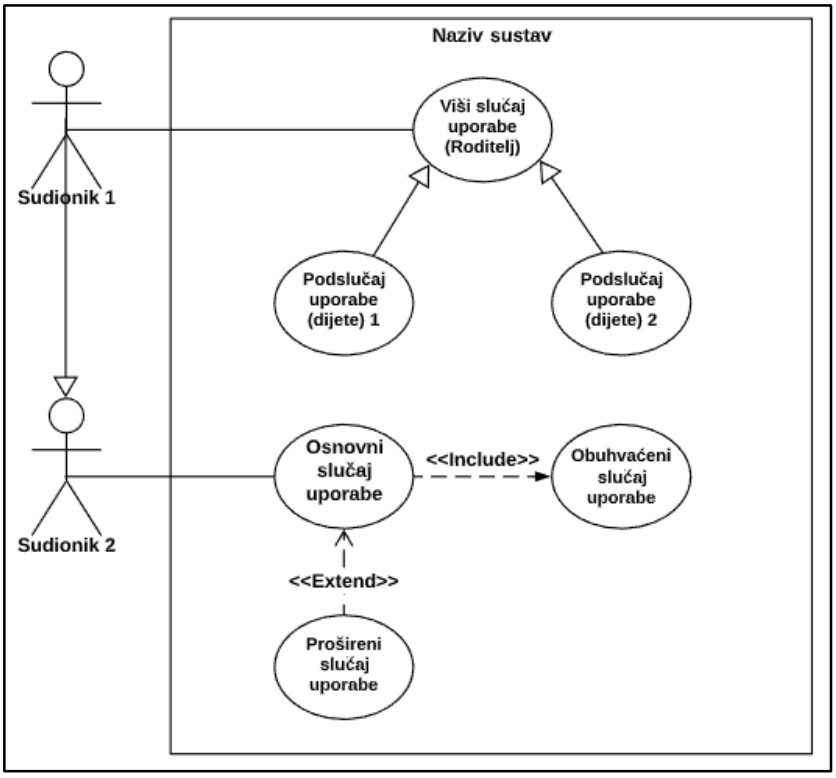
Veze između elementa modela u UML-u predstavljaju relacije. Relacija je vrsta veze koja daje značenje modelu te opisuje značenje i strukturu između elementa/komponenta sustava, [11]. U UML-u postoje tri vrste relacija:

1. Obuhvaćanja (eng. *Include*) – relacija obuhvaćanja se koristi kada osnovni slučaj uporabe izričito obuhvaća funkcionalnosti drugog slučaja uporabe. Obuhvaćeni slučaj uporabe ne postoji samostalno, već ovisi o jednom ili više osnovnih slučaja uporabe. Njime se izdvajaju funkcionalnosti koje su zajedničke za više slučaja uporabe, [12].

2. Proširenja (eng. *Extend*) – relacijom proširenja se dodaju određene funkcionalnosti osnovnom slučaju uporabe. Prošireni slučaj uporabe predstavljaju dodatne funkcionalnosti osnovnog slučaja uporabe ili funkcionalnosti koje se izvršavaju samo kada se zadovolje specifični zahtjevi, [11]. Osnovni slučaj uporabe mora biti sposoban funkcionirati samostalno bez proširenog slučaja uporabe. Prošireni slučaj uporabe se izvodi samo kada se za to pojavi potreba, u protivnom, osnovni slučaj uporabe ga potpuno zanemaruje, ali je svjestan da postoji mogućnost za njegovo korištenje, [2].

3. Poopćavanja (eng. *Generalization*) – relacija poopćavanja ili relacija nasljeđivanja se koristi kada podslučaj uporabe (dijete) koristi određena svojstava, operacije i odnose od višeg slučaja uporabe (roditelj), [1]. Jedan viši slučaj uporabe može imati više podslučaja uporabe. U podslučaju uporabe definiraju se samo svojstava, operacije i odnosi koji nisu naslijeđeni od višeg slučaja uporabe, [11]. Poopćavanje se može primijeniti i između aktera sustava. U tom slučaju akter koji nasljeđuje svojstva drugog aktera ima pravo pristupiti istim slučajevima uporabe kao i akter od kojeg je naslijedio svojstva.

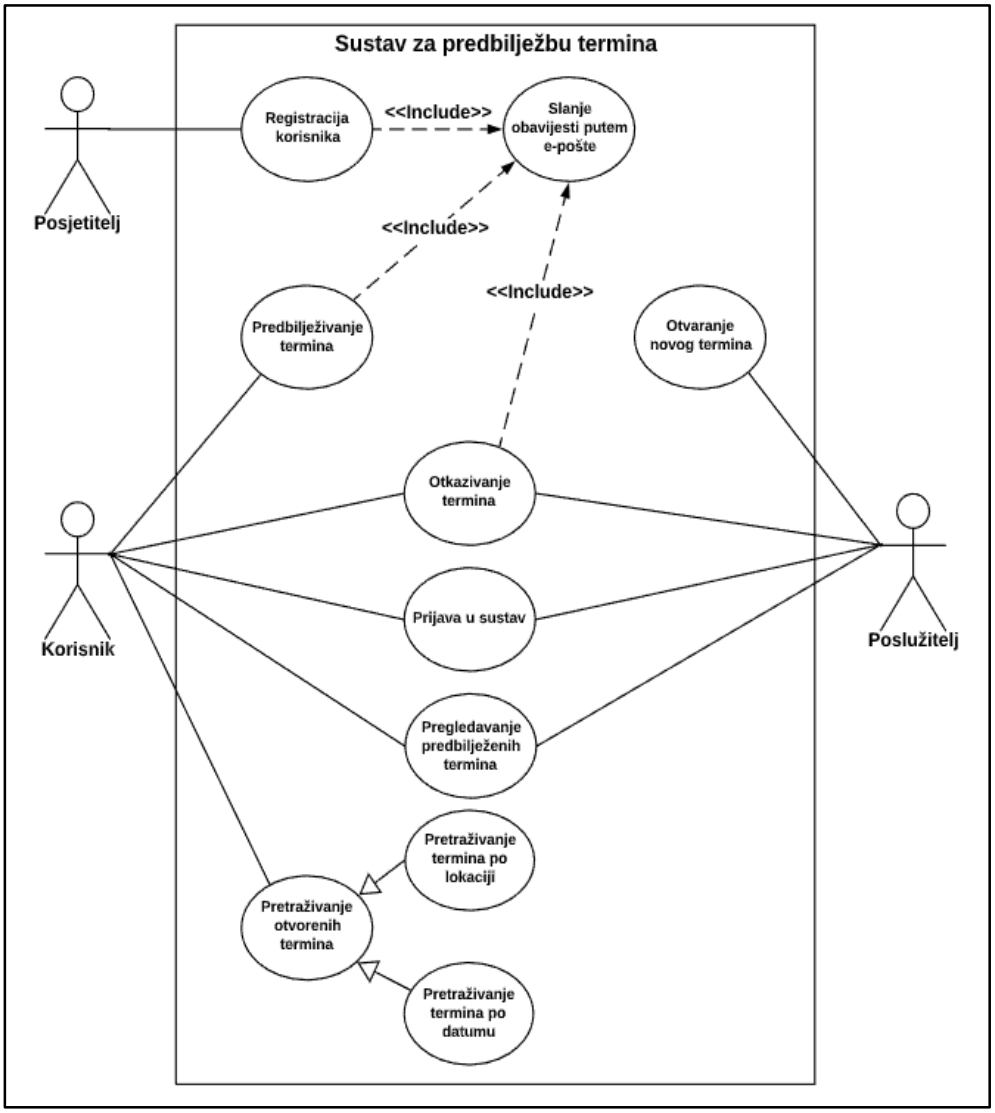
Slikom 2 prikazan je način označavanja relacija u dijagramu slučajeva uporabe. Između osnovnog slučaja uporabe i obuhvaćenog slučaja uporabe upotrijebljena je veza obuhvaćanja. Veza između osnovnog slučaja uporabe i proširenog slučaja uporabe opisana je relacijom proširenja. Relacija poopćavanja upotrijebljena je između višeg slučaja uporabe i podslučaja uporabe 1 i podslučaja uporabe 2. Iz slike se još može vidjeti da se relacija poopćavanja ili nasljeđivanja koristiti između Sudionika 1 i Sudionika 2.



Slika 2.Prikaz označavanja relacija u dijagramu slučajeva uporabe

3.2. Dijagram slučaja uporabe web aplikacije podvorbenog sustava s mogućnošću predbilježbe

Prilikom analiziranja i modeliranja nekog stvarnog ili zamišljenog sustava, prvo se razrađuje model slučaja uporabe. Važno je u što ranijoj fazi projekta ili ujedinjenog procesa prepoznati ključne zahtjeve koji se postavljaju na promatrani sustava i definirati problematiku sustava. Dijagrami slučaja uporabe omogućuju programskoj podršci da na lak i razumljiv način prepoznaju glavne funkcionalnosti sustava na koje se trebaju usredotočiti prilikom razvoja sustava. Na slici 3 prikazan je dijagram slučajeva uporabe podvorbenog sustava s mogućnošću predbilježbe, odnosno web aplikacije s mogućnošću predbilježbe korisnika.



Slika 3. Dijagram slučajeva uporabe za sustav s mogućnošću predbilježbe

Iz slike 3 se može vidjeti da sustav ima tri sudionika, to su posjetitelj, korisnik i poslužitelj. Da bi posjetitelj sustava odnosno aplikacije mogao predbilježiti termin najprije mora samostalno otvoriti korisnički račun. Sustav automatski generira e-poštu kojom obavještava posjetitelja da je uspješno otvorio svoj korisnički račun. Poslužitelj se najprije mora prijaviti u sustav kako bi imao mogućnosti otvoriti novi termin i pregledavati svoje predbilježene termine. I poslužitelj i korisnik imaju mogućnost otkazati, odnosno maknuti predbilježbu za određeni termin. Putem e-pošte se obavješćuje korisnika da je maknuta predbilježba za njegov termin. Korisnik nakon uspješne prijave može pretraživati slobodne termine po mjestu i datumu termina te imati uvid u svoje predbilježbe. Nakon što se korisnik predbilježi za određeni slobodni termin, sustav automatski obavještava korisnika da je uspješno predbilježen za željeni termin.

U slučaju da posjetitelj pokuša upisati ne odgovarajuće podatke prilikom stvaranja računa npr. pokuša upisati lozinku dulju od pedeset znakova, aplikacija javlja grešku i traži ponovni upis podataka. Prilikom prijave korisnika u sustav ako dođe do pogrešnog unosa podataka od strane korisnika, aplikacija javlja da je prijava neuspješna i daje korisniku pravo da pokuša ponovno upisati odgovarajuće korisničke podatke. Ako neki od korisnika pokušava pristupiti stranici za koja nema odgovarajuća prava npr. korisnik pokušava pristupiti stranici za unos novih termina, a nema ulogu poslužitelja, onda aplikacija prosljeđuje korisnika na odgovarajuću stranicu za prijavu, u ovom slučaju je to stranica za prijavu poslužitelja.

4. Modeliranje svojstva, ponašanja i stanja objekta

Prije početka izgradnje sustava potrebno je odrediti njegovu statičku strukturu. Statička struktura sustava usredotočena je na definiranje objekta, što uključuje definiranje njegovih stanja, vrsta i svojstva, a vremenska komponenta sustava se zanemaruje. Za vizualni opis statičke strukture sustava koriste se dijagrami klasa i stanja. Dijagram klasa opisuje vrstu, svojstva i međusobne odnose između klasa koje predstavljaju objekte sustava. Dijagram stanja prikazuje ponašanje objekta na određenu aktivnost. Ponašanje objekta ovisi o stanju u kojem se nalazi, a svako stanje traje neki određeni vremenski period, u kojem može zadovoljavati jedan ili više uvjeta.

4.1. Dijagram klasa

Dijagram klasa služi za opisivanje statičke strukture sustava. Opisuje svojstva, vrstu i međusobne odnose između klasa koje se nalaze u istom sustav. Zanemaruju se sva stanja, događaji, aktivnosti i promjena svojstva koja su vezana uz vremensku komponentu sustava. Dijagram klasa opisuje sustava samo u trenucima kada je vrijeme statična komponenta sustava.

Objekti predstavljaju stvari iz stvarnog svijeta koji imaju neki smisao i ulogu u nekom zamišljenom ili stvarnom sustavu. Klase i objekti su vrlo slični pojmovi i usko su vezani jedan uz drugog, klase prikazuju grupe objekta sa sličnim svojstvima. U objektno-orijentiranom programiranju klasa sama po sebi nije objekt, ali opisuju koja svojstva će objekt posjedovati. Glavna razlika između objekta i klase je ta da objekti, osim svojstva, posjeduju određenja stanja i ponašanja, [13]. Svrha dijagrama klasa je modeliranje statičkog prikaza sustava ili aplikacije. Dijagrami klase jedini su dijagrami koji se mogu izravno povezati s objektno-orijentiranim programskim jezicima te se iz tog razloga često koriste prilikom izgradnje aplikacije. Svaka klasa se prikazuje pravokutnikom koji ima tri odjeljka, kao što je prikazano na slici 4.

Slika 4. Prikaz klase u UML-u

U prvi odjeljak se upisuje naziv klase. Naziv klase se treba odnositi na problemsko područje koje se promatra i mora biti ne dvosmislen. U praksi se najčešće koriste imenice za naziv klase. Drugi odjeljak je namijenjen za upis atributa klase. Atributi klase opisuju svojstva objekta. Prilikom modeliranja potrebno je uključiti samo atribute koji su vezani uz problemsko područje i trebaju davati dovoljno informacija za opis i definiranje pojedine instance klase. Svaki atribut ima svoju vrstu i vidljivost. Vrsta ili tip atributa se piše poslije dvotočke i u pravilu predstavljaju tipove podataka kao što su: *integer*, *Boolean*, *string* i *date*. Vidljivost atributa pokazuje može li se neki atribut vidjeti od strane neke druge klase. Za označavanje vidljivosti objekta koriste se simboli + (javna vidljivost), - (privatna vidljivost) i # (zaštićena vidljivost). Zadnji odjeljak je namijenjen za operacije klasa. Operacije manipuliraju atributima ili mogu izvoditi neke druge radnje. Operacije se obično nazivaju funkcijama, ali pripadaju samo toj klasi i mogu se primijeniti samo nad objektima iste klase. Iza dvotočke se upisuje tip metode koji se vraća i potrebni parametri za izvršenje pojedine metode, [2].

Dijagram klasa se sastoji od klasa i relacija (veza) između klasa. Klase same po sebi nisu previše uporabljive, već relacije između klasa čine osnovu modela, [14]. Vrste relacije u dijagramu klasa su sljedeće :

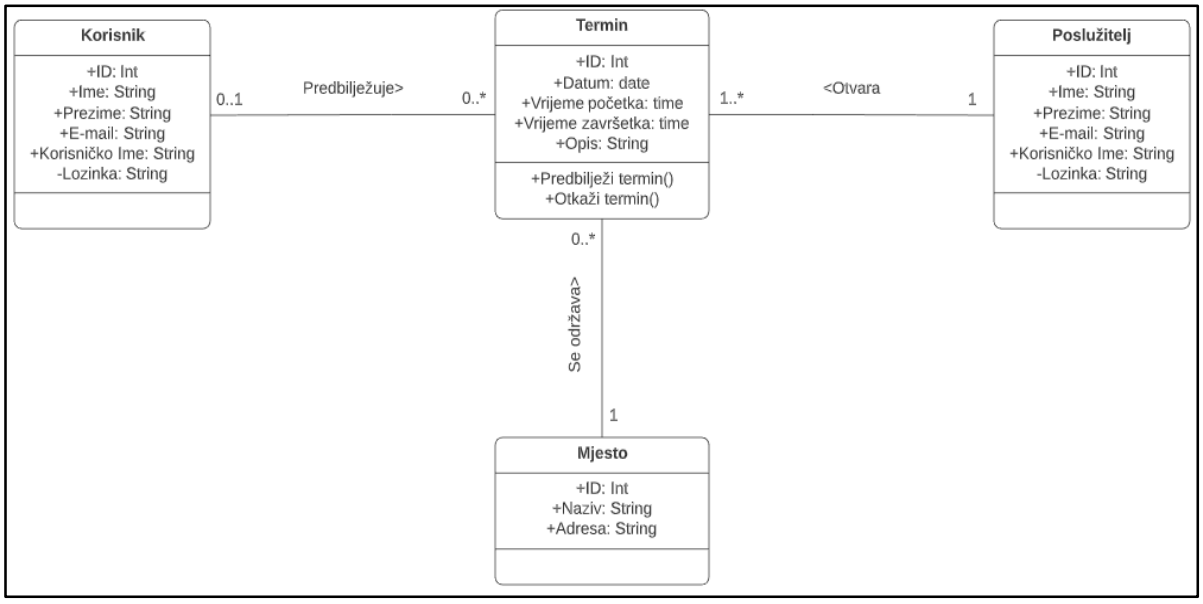
1. Relacija udruživanja (eng. *Association*) - opisuje statične odnose između pojedinaca, odnosno instance klasa. Asocijacija je obično dvosmjerna, što znači da ako je jedan objekt povezan s drugim objektom, oba objekta su svjesni da drugi postoji. Ona omogućava da jedna klasa pristupa atributima druge klase. Veza između klasa osim dvosmjerne može biti još jednosmjerna i refleksivne, [2].

2. Relacija sastavljanja (engl. *Aggregation*) – je poseban slučaj relacije udruživanja. Ona se koristi kada jedna klasa ovisi o drugoj klasi, tj. predstavlja pod-skup druge klase. Razlikuju se dvije vrste relacije sastavljanja, agregacija i kompozicija.

Agregacija je vrsta sastavljanja koja govori da jedna klasa sadrži drugu. Kompozicija je vrlo slična agregaciji samo kada se uništi nad-klasa uništava se i pod-klasa, [14].

3. Relacija poopćavanja (eng. *Generalization*) – omogućuje upotrebu hijerarhijske klasifikacije. Opisuje vezu između klasa ili skupova koji imaju nešto zajedničko. Omogućuje nasljeđivanje svojstva od više klase, odnosno od klase roditelja, [2].

Veze ovisno o njihovom smjeru udruživanja dijele se na jednosmjerne (unidirekcionalne) i dvosmjerne (bidirekcionalne). Jednosmjerne veze imaju smjer definiran samo na jednom vrhu, dok dvosmjerne imaju smjer definiran na oba vrha. Ako smjer nije eksplicitno definiran, smatra se da je veza nepoznata (nedefinirana) ili dvosmjerna. Ako je veza jednosmjerna (unidirekcionalna), onda njezini smjerovi moraju biti međusobno inverzni., [14]. Slikom 5 prikazan je dijagram klasa za aplikaciju sustava s mogućnošću predbilježbe korisnika.



Slika 5. Dijagram klasa sustava s mogućnošću predbilježbe

Slika 5 prikazuje dijagram klasa u kojem je glavni objekt „Termin“ za primjer sustava s mogućnošću predbilježbe. Sporedni objekti su „Korisnik“, „Poslužitelj“ i „Mjesto“. Poslužitelj je sudionik koji u ovom primjeru kreira novi termin. U dijagramu klasa postavljene su oznake za multiplikativnost koje se nalaze pored relacije udruživanja. Iz relacije udruživanja između termina i poslužitelja može se iščitati da se jedan termin otvara od strane jednog poslužitelja, a da jedan poslužitelj može otvoriti jedan ili više termina. Na jednom mjestu se može održavati

niti jedan ili više definiranih termina, dok se jedan specifični termin može održavati na samo jednom mjestu. Kada je termin u stanju slobodan korisnik se može predbilježiti za taj specifični termin. Tada termin mijenja svoje stanje i prelazi u stanje predbilježen. Jedan korisnik se može predbilježiti za niti jedan ili više termina, dok jedan termin može biti predbilježen od niti jednog ili jednog korisnika. U slučaju da korisnik otkaže termin se vraća u stanje slobodan.

4.2. Dijagram stanja

Ponašanje objekta ne ovisi samo o izravnim promjenama njegovi ulaza, odnosno vanjskih aktivnosti, već ovisi i o stanju u kojem se našao prije nego li je došlo do neke vanjske aktivnosti. UML dijagram stanja pokazuje sva stanja u kojima se određeni objekt može naći. Objekt različito reagira na isti događaj, ovisno o stanju u kojem se nalazi. Također mogu pokazati kako objekt reagira na različite događaje mijenjajući se iz jednog u drugo stanje. Dijagram stanja je UML dijagram koji se koristi za modeliranje dinamičke prirode sustava. Pretežno se koristi za prikaz stanja objekta, ali to nije nužno. Može se upotrijebiti na svakom elementu sustava koji ima različita ponašanja kada je u vezi s nekim drugim elementom sustava, [15].

Svi objekti imaju stanje koje je rezultat prethodnih aktivnosti koje obavlja objekt i obično se određuje vrijednostima njegovih atributa i vezama s drugim objektima. Klasa može imati određeni atribut koji određuje stanje ili se stanje može odrediti prema vrijednostima atributa klase. Objekt prelazi (mijenja se) iz jednog stanja u drugo stanje kada se nešto dogodi, što se naziva događajem, [2]. Prema [16] događaji stanja se dijele u sljedeće četiri skupine:

1. Signal - asinkroni način komunikacije između objekata.

2. Događaj poziva (eng. *event call*) – sinkroni način komunikacije između objekata, gdje objekt može pozvati svoju metodu, pozvati metodu nekog drugog objekta i ima jednaki učinak kao poziv akcije.

3. Vremenski događaj (eng. *time event*) – objekt mijenja svoje stanje nakon proteka određenog vremenskog perioda. Najčešće se izražava pomoću riječi „poslije“ iza koje slijedi oznaka vremena.

4. Događaj promjene (eng. *change event*) – objekt mijenja svoje stanje nakon što se zadovolje određeni uvjeti. Najčešće se izražava pomoću riječi „kada“ iza koje slijedi Bool-ov izraz.

Objekt tijekom svojeg životnog vijeka može biti u više stanja u kojem može obavljati razne aktivnost, čekati na ispunjavanje uvjeta ili čekati na pojavu novog događaja. Kada se ispune određeni uvjeti objekt mijenja svoje stanje što se naziva prijelazom iz jednog stanja u drugo. Prijelaz je odnos između dva stanja koji ukazuje da će objekt u prvom stanju izvesti određene radnje i ući u drugo stanje kada se dogodi određeni događaj i ispune određeni uvjeti. Za pravilan opis i određivanje prijelaza stanja objekta iz jednog stanja u drugo, koristi se slijedećih pet stanja, događaja i uvjeta:

1. Početno (izvorišno) stanje - stanje na koju utječe tranzicija, ako je objekt u početnom stanju, prijelaz iz stanja se može aktivirati kada objekt primi događaj prijelaza i ako je ispunjen čuvani uvjet, koji stanje može i ne mora imati.

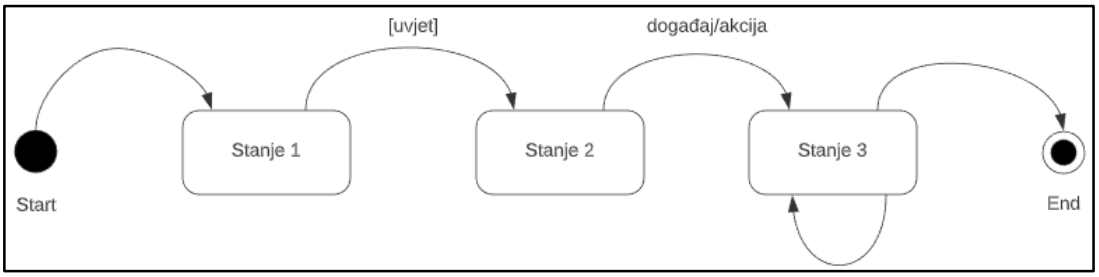
2. Pokretač događaja (eng. *event trigger*) - je pojava podražaja koji može pokrenuti prijelaz stanja. Pokretači događaji mogu biti: događaji poziva, vremenski događaji, promjena u trenutnom stanja i signali poziva. Signal ili poziv mogu imati parametre čije su vrijednosti dostupne prijelazu, uključujući izraze za čuvane uvjete. Također je moguće izvršiti prijelaz bez pokretača događaja. Ti prijelazi, koji se također nazivaju prijelazi dovršetka, pokreću se implicitno kada je izvorno stanje dovršilo svoju aktivnost.

3. Čuvani uvjeti (eng. *guard conditions*) - Boolov izraz koji se procjenjuje kada se prijelaz aktivira prijemom okidača događaja. U slučaju da je Boolov izraz istinit (eng. *true*), uvjet prijelaza je ispunjen i objekt mijenja svoje stanje. Ako je Boolov izraz neistina (eng. *false*), prijelaz se ne aktivira. Ako nema drugog prijelaza koji bi mogao biti pokrenut istim događajem, objekt ostaje u trenutnom stanju.

4. Akcije – je neka radnja ili događaj koji direktno djeluju na objekt, što direktno uzrokuje promjenu njegovog stanja, u slučaja kada ne postoje čuvani uvjeti. Akcije mogu indirektno djelovati i na objekte koji su vidljivi objektu na kojeg se specifična akcija odnosi.

5. Ciljano stanje (eng. *target state*) – stanje u kojem se objekt nalazi nakon uspješnog prijelaza, [16].

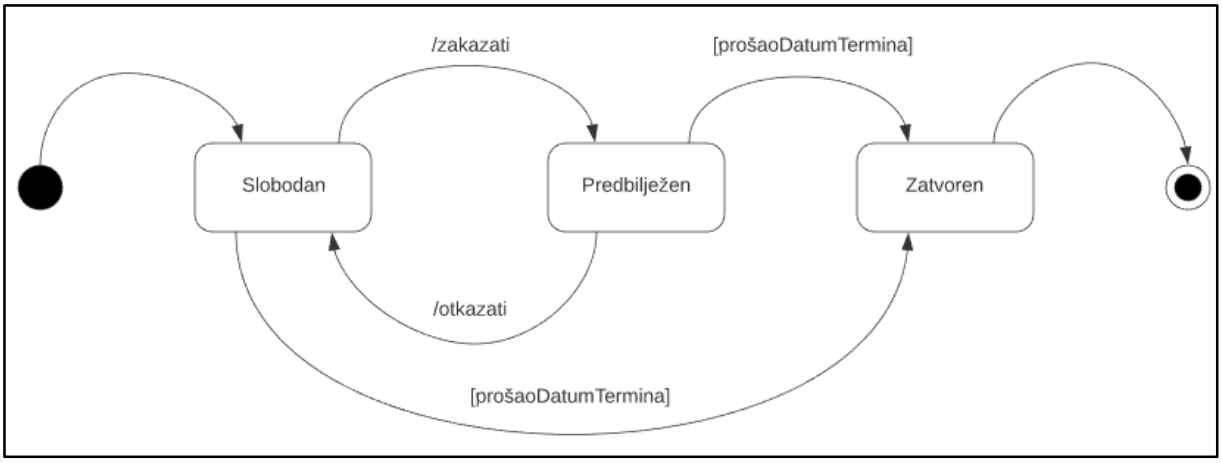
Objekt prelazi iz jednog stanja u drugo kada se zadovolje svi uvjeti ili kada se pojavi događaj ili akcija koji uzrokuju promjenu stanja objekta. Objekt može i bezuvjetno prijeći u drugo stanje, bez pojave događaja ili akcije, ako je objekt izvršio sve radnje i aktivnosti koje je trebao izvršiti u početnom stanju. Na slici 6 prikazani su simboli početka i završetka, stanja i uvjeta prijelaza u dijagramu stanja.



Slika 6. Simboli, stanja i uvjeti dijagrama stanja

Kada objekt prelazi iz jednog stanja u drugo, a pritom postoji uvjet koji se piše u uglatim zagradama, njemu se može pridružiti neka akcija ili događaj. U slučaju da ne postoji čuvani uvjet ili akcija, objekt prijelazi iz jednog stanja u drugo bezuvjetno. U vlastitom prijelazu početno i odredišno stanje je isto.

Svaki dijagram stanja se odnosi na jedan specifični objekt i prikazuje njegovo ponašanje na razne događaje i aktivnosti unutar sustava. Slikom 7 prikazan je dijagram stanja objekta termin, za primjer aplikacije podvorbenog sustava s mogućnošću predbilježbe.



Slika 7. Dijagram stanja objekta termin

Dijagram stanja započinje u simbolu početnog stanja. Objekt bezuvjetno prelazi iz početnog stanja u stanje „Slobodan“. Kada se objekt nalazi u stanju slobodan on može prijeći u stanje predbilježen ako se pojavi akcija „zakazati“, koja predstavlja zakazivanje termina od strane korisnika. U slučaju da je zadovoljen uvjet „prošaoDatumTermina“, objekt prelazi iz stanja „Slobodan“ u stanje „Zatvoren“. Kada se objekt nalazi u stanju „Predbilježen“ on se može vratiti u stanje „Slobodan“ ako se pojavi akcija „otkazati“, koja predstavlja otkazivanje predbilježbe od strane korisnika ili poslužitelja. Objekt prelazi iz stanja „Predbilježen“ u stanje zatvoren ako je ispunjen uvjet „prošaoDatumTermina“. Iz stanja „Zatvoren“ objekt bezuvjetno prelazi u konačno stanje.

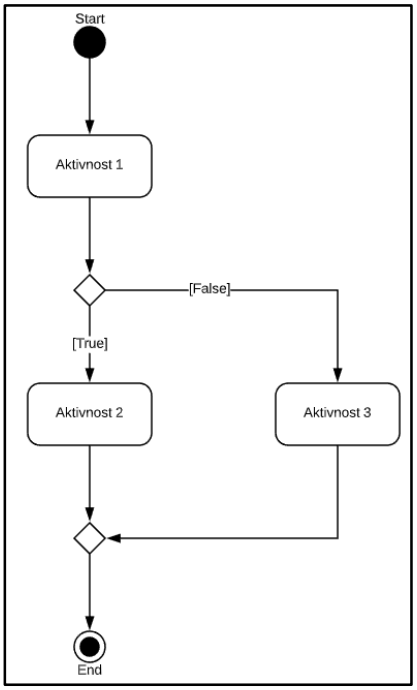
5. Modeliranje toka podataka i toka upravljanja primjenom dijagrama aktivnosti

Dijagram aktivnosti opisuje dinamički aspekt sustava i predstavlja tijek ili redoslijed odvijanja aktivnosti unutar sustava. Aktivnost se može definirati kao operacija ili radnja koju sustav neprekidno obavlja i traje neko određeno vrijeme. Svaka aktivnost ima mogućnost prekida izvršavanja i može se rastaviti u više aktivnosti.

5.1. Dijagram aktivnosti

Dijagram aktivnosti prikazuje protok upravljanja (eng. *flow control*) ili protoka objekta (eng. *object flow*) s naglaskom na redoslijed i uvjete toka. Radnje koordinirane modelima aktivnosti mogu se pokrenuti u slučajevima kada ostale radnje završe svoje izvršavanje, kada se oslobode potrebni podaci i objekti ili kada dođe do pojave nekog događaja koji se nalazi izvan toka, [16]. Dijagrami aktivnosti se mogu koristiti za prikazivanje poslovnog modela ili za modeliranje logike jednog slučaja uporabe, ali se najčešće koriste za prikazivanje opisa rada algoritma, detalja računanja ili u objektno orijentiranom programiranju za prikazivanje toka objekta.

Dijagram aktivnosti obuhvaća račvanje i grananje, prijelaze, te grananje i stapanje aktivnosti koje su pridružene određenom objektu u sustavu koje se promatra. Početno stanje i završno stanje prikazuje se simbolom popunjene kružnice kako je prikazano slikom 8, osim toga početna i završna stanja se dodatno označuju natpisima „Start“ i „End“.



Slika 8. Simboli dijagrama aktivnosti

U objektno orijentiranom modelu aktivnosti se obično neizravno pozivaju kao metode vezane za operacije koje se izravno pozivaju, [16]. Aktivnosti se označavaju simbolom pravokutnika zaobljenih vrhova u kojem se nalazi naziv pojedine aktivnosti, kao što je prikazano na slici 8. Kontrola toka u dijagramima aktivnosti se označava pomoću strelica koje dodiruju rubove simbola aktivnosti. U slučaju da vrh strelice dira simbol aktivnosti tada to predstavlja ulaz u protoku objekta, a u slučaju da strelica kreće od simbola aktivnosti onda to predstavlja izlazni tok.

U većini slučajeva aktivnosti se izvode slijedno, ali ponekad dolazi do potrebe za razdvajanjem toka aktivnosti. Mjesto na kojem dolazi do razdvajanja slijeda aktivnosti u dva ili više pravca (grana) naziva se mjesto grananja (eng. *Branch*), a mjesto gdje se sastavljaju dva ili više pravca (grana) naziva se mjesto stapanja (eng. *Merge*), [17]. Mjesto grananja i stapanja označuje se istim simbolom romba, kako je prikazano na slici 8.

Da bi se grananje i stapanje moglo primijeniti potrebno je zadovoljiti čuvane uvijete. Rezultat čuvanih uvjeta je uvijek Bool-ov izraz koji može poprimiti vrijednost istina (eng. *True*) ili laž (eng. *False*). Nastavljanje praćenja grane zavisi o vrijednosti Bool-ovog izraza. Skup svih grana iz mjesta grananja mora obuhvaćati sve mogućnosti slijeda, a da pritom ne dolazi do

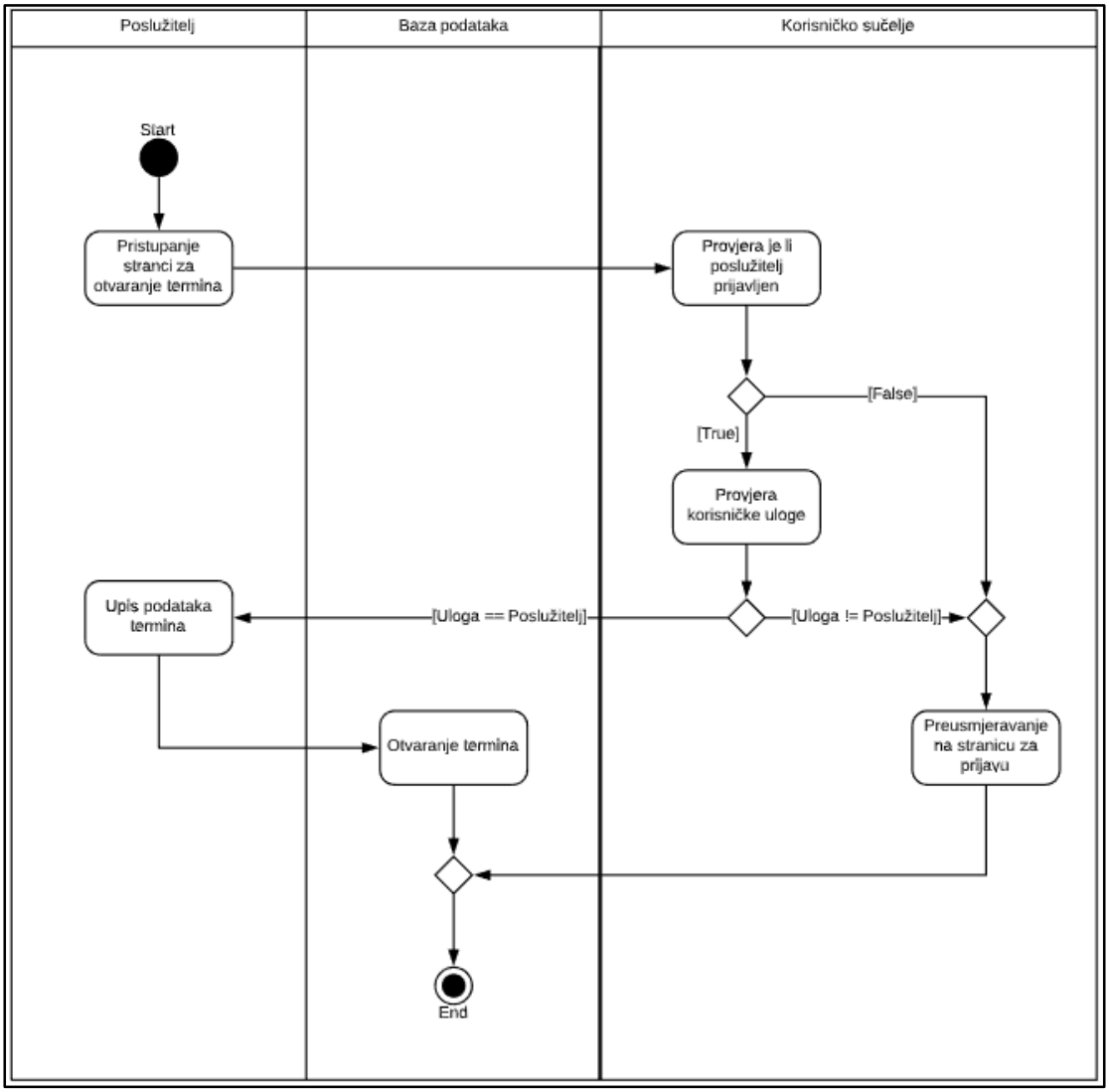
preklapanja čuvanih uvjeta. Uvjet se uvijek upisuje u uglate zagrade uz granu na koju se odnose, [17].

Često se aktivnosti u dijagramu aktivnosti grupiraju kako bi olakšali njihovo praćenje. Najčešće se aktivnosti grupiraju prema dijelovima organizacije ili prema objektima u sustavu. Grupiranje se vizualno prikazuje pomoću plivaćih staza (eng. *Swimlane*). Osim grupiranja u dijagramima aktivnosti moguće je aktivnosti rastaviti u dva ili više nezavisna i sinkrona (istodobna) tijeka, što se naziva račvanjem (eng. *Fork*). Mjesto spajanja (eng. *Join*) je mjesto na kojem se usklađuju dva ili više tijekova izvođenja u jedan. Najčešće se koriste kod modeliranja programske niti ili procesa, [17]. Mjesto račvanja i spajanja označuje se uskim dugim crnim pravokutnikom, koji se još naziva i sinkronizacijska crta (eng. *Synchronization bar*).

Dijagramima aktivnosti se ne prikazuju svi detalji odvijanja aktivnosti unutar sustava. Oni prikazuju tijek aktivnosti koje se odvijaju unutar sustava, ali zanemaruju objekte koji izvršavaju aktivnosti. Često je dijagram aktivnosti među prvim modelima koji se modelira u novom sustavu. Svaka aktivnost opisana dijagramom aktivnosti se implementiraju kao jedna ili više operacija, za čiju implementaciju je zadužena specifična klasa objekta, [1].

5.2. Dijagrami aktivnosti aplikacije za podvorbeni sustav s mogućnošću predbilježbe

Slikom 9 prikazan je dijagram aktivnosti stvaranja novog termina od strane poslužitelja kako bi se korisnik mogao predbilježiti za željenu uslugu u nekom od definiranih termina od strane poslužitelja.



Slika 9. Dijagram aktivnosti stvaranja novog termina

Dijagramom na slici 9 prikazan je tijek aktivnosti stvaranja novog termina od strane poslužitelja. Dijagram započinje početnim stanjem. Zatim poslužitelj pokušava pristupiti stranici za stvaranje novog termina. Korisničko sučelje poslužitelja, nakon što zaprimi zahtjev za pristupom stranici za stvaranje novog termina, obavlja provjeru je li korisnik prijavljen u sustav kako bi se spriječio pristup nelegitimnom korisniku. Nakon uspješne provjere prijave poslužitelja u sustav, korisničko sučelje poslužitelja obavlja dodatnu provjeru uloge korisnika. U slučaju da uloga nema vrijednost „Poslužitelj“ ili korisnik nije prijavljen u sustav, korisnika ili poslužitelja se preusmjeruje na stranicu za prijavu poslužitelja u sustav. Za vrijeme trajanja provjere pristupa poslužitelj cijelo vrijeme čeka na odgovor. Nakon što je poslužitelju odobren pristup stranici za stvaranje novog termina, on upisuje podatke o novom terminu, koji se potom stvara (otvara) u bazi podataka i vidljiv je korisnicima sustava te ga mogu predbilježiti.

6. Modeliranje interakcije između sudionika

6.1. Dijagram međudjelovanja

Dijagrami međudjelovanja opisuju dinamičku strukturu sustava i način na koji objekti komuniciraju među sobom u jednom slučaju uporabe. Njima se vizualno predstavlja redoslijed interakcije između objekata, koji u međusobnom djelovanju odrađuju određene funkcije sustava, potaknute pokretanjem jednog slučaja uporabe.

Dijagram međudjelovanja se može prikazati u dva oblika, jedan je u generičkom obliku, a drugi je oblik instance. U generičkom obliku se uključuju sve moguće alternative u scenariju, što može uključivati petlje, grananje i posebne uvjete. Oblik instance se odnosi na samo jedan specifični slučaj. U njemu se ne prikazuju petlje, grananja i posebni uvjeti, već se prikazuje samo interakcija objekata za odabrani slučaj, [2].

Svaki dijagram međudjelovanja koristi sljedeće oznake elementa za opis scenarija:

1. Oznaka objekta - stavljaju se na vrh dijagrama i obično se prikazuju kao pravokutnik sa zaobljenim rubovima i nazivom objekta unutar pravokutnika.

2. Crte života (eng. *Lifeline*) - je vertikalna isprekidana linija koja predstavlja postojanje objekta tijekom određenog vremenskog razdoblja. Većina objekata koji se pojavljuju u dijagramu interakcije postojat će tijekom trajanja interakcije, tako da su svi ovi objekti usklađeni na vrhu dijagrama, s crtama pravca od vrha dijagrama do dna, [18].

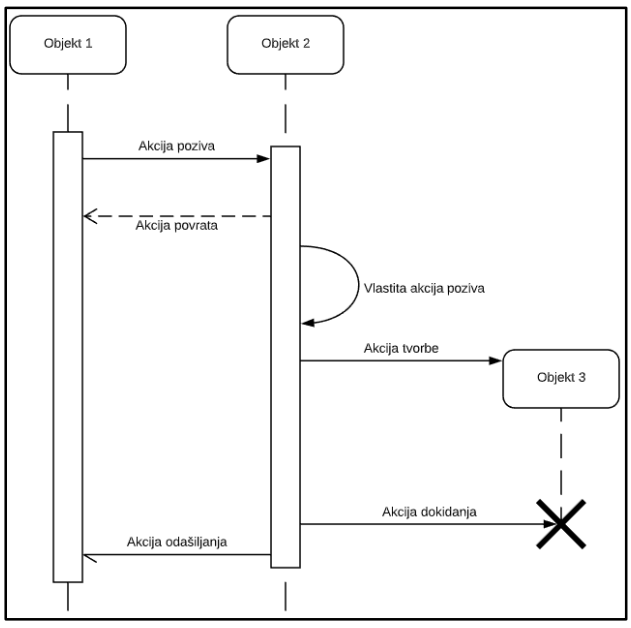
3. Težište nadzora (eng. *Focus of control*) ili okvir aktivnosti (eng. *Activation bar*) - je okvir koji se nalazi na životnoj liniji. Koristi se za označavanje aktivnosti objekta tijekom interakcije dva objekta. Duljina pravokutnika označava trajanje objekata koji ostaju aktivni, [19].

4. Poruke (eng. *Messages*) - prikazuju koje akcije se izvršavaju između objekta ili unutar samog objekta.

Poruke ili akcije u dijagramima međudjelovanja predstavljaju izvršavanje naredbi koje mogu rezultirati promjenom jednog ili više atributa objekta, povratnom vrijednosti objektu koji

je uputio poruku i promjenom i povratom vrijednosti. Poruke se mogu slati u bilo kojem smjeru odnosno prema bilo kojem objektu, uključujući i samog sebe. U UML dijagramima se razlike pet vrsta akcija odnosno poruka.

Akcija poziva (eng. *Call action*) je sinkrona poruka, koja priziva metode odredišnog objekta. Objekt pošiljatelj pretpostavlja da je objekt primatelj spreman primiti poruku, a pošiljatelj prije nego krene s pokretanjem druge metode čeka odgovor odredišnog objekta. Akcija povrata (eng. *Return action*) predstavlja odgovor pošiljatelju na njegovu akciju poziva. Svaka akcija poziva mora imati odgovarajuću akciju povrata. Akcija tvorbe (eng. *Create action*) se koristi kada se želi naglasiti da se određeni objekt, odnosno instanca klase stvara tek kada se izvrši akcija tvorbe. Akcija dokidanja (eng. *Delete action*) se može opisati kao suprotna akcija od akcije tvorbe. Njome se uništava (briše) određeni objekt kada više nema potrebe za tim objektom. Akcija dokidanja može pozvati samu sebe. Posljednja akcija je akcija odašiljanja (eng. *Send action*). Ona odašilje signal poruku, za koju pošiljatelj ne očekuje odgovor.

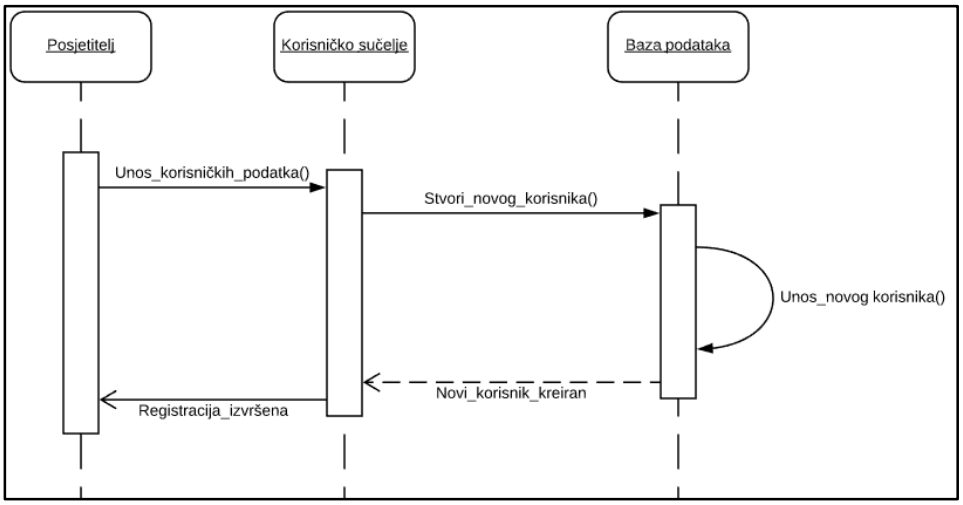


Slika 10. Simboli, oznake i akcije dijagrama međudjelovanja

Na slici 10 prikazani su svi navedeni i objašnjeni elementi i akcije dijagrama međudjelovanja. Slikom su odgovarajućim simbolima prikazani objekti, crte života objekta, okviri aktivnosti i svih pet ranije definiranih vrsta akcija u dijagramima međudjelovanja.

6.2. Primjeri dijagrama međudjelovanja za aplikaciju podvorbenog sustava sa mogućnošću predbilježbe

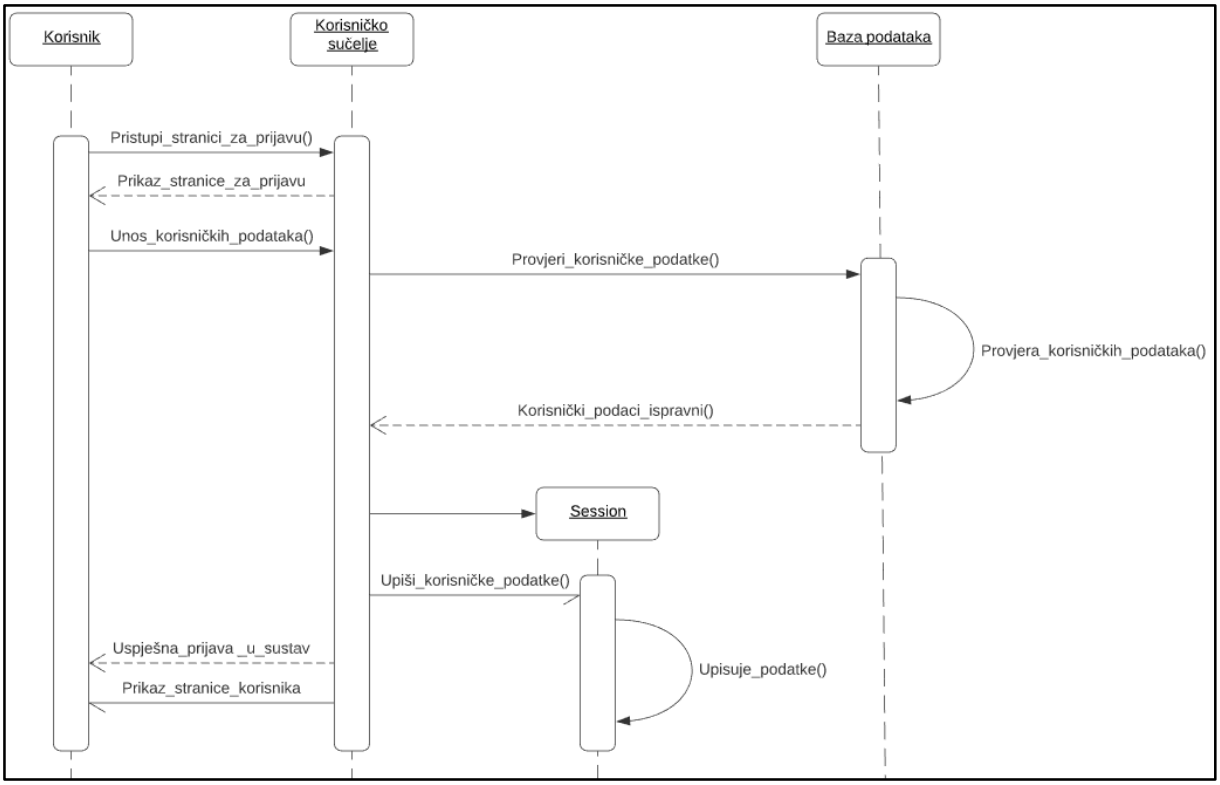
Slikom 11 prikazan je dijagram međudjelovanja za vremenski slijed razmjena poruka između objekata za slučaj uporabe registracije korisnika.



Slika 11. Dijagram međudjelovanja za slučaj uporabe registracije korisnika

Kako bi se korisnik mogao uspješno prijaviti u sustav, najprije je potrebno izvršiti registraciju korisnika odnosno otvaranje korisničkog računa. Posjetitelj putem korisničkog sučelja upisuje podatke o svojem korisničkom računu što uključuje njegovo ime, prezime, e-mail, korisničko ime i lozinku. Korisničko sučelje zatim prosljeđuje podatke korisnika bazi podataka, koja potom upisuje korisnikove podatke u bazu podataka. Baza podataka po uspješnom završetku unosa korisnika, šalje poruku korisničkom sučelju da je unos korisnika uspješan. Korisnik kroz korisničko sučelje vidi da je registracija uspješno izvršena. Korisnik se tada može uspješno prijaviti u sustav s podacima koje je predao sustavu.

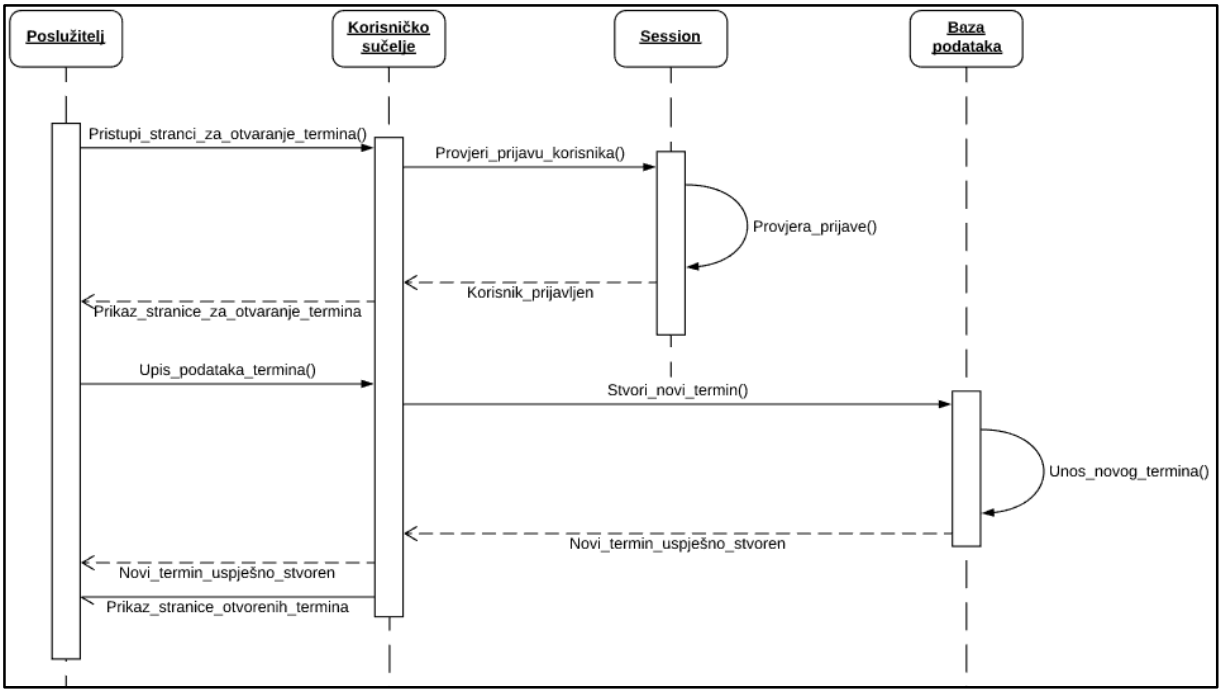
Slikom 12 prikazan je dijagram međudjelovanja za vremenski slijed razmjene poruka između objekta za slučaj uporabe prijave korisnika u sustav.



Slika 12. Dijagram međudjelovanja za slučaj uporabe prijave korisnika u sustav

Korisnik putem korisničkog sučelja pokušava pristupiti stranici za prijavu korisnika. Korisničko sučelje na korisnikov zahtjev prikazuje stranicu za prijavu, na kojoj korisnik potom unosi svoje korisničke podatke (korisničko ime i lozinku). Zatim korisničko sučelje šalje poruku bazi podataka da provjeri postoji li korisnik s unesenim korisničkim podacima. Baza podataka poslije uspješne provjere korisničkih podataka šalje poruku korisničkom sučelju da postoji korisnik s predanim podacima. Nakon zaprimanja poruke korisničko sučelje stvara objekt „Session“ te mu predaje upisane korisničke podatke. Podaci koji se unose u objekt „Session“ su korisnikov ID, korisničko ime, korisnička uloga i adresa e-pošte korisnika. Korisnik kroz korisničko sučelje vidi da je uspješno prijavljen u sustava te ga je korisničko sučelje automatski preusmjerilo na stranicu za korisnike.

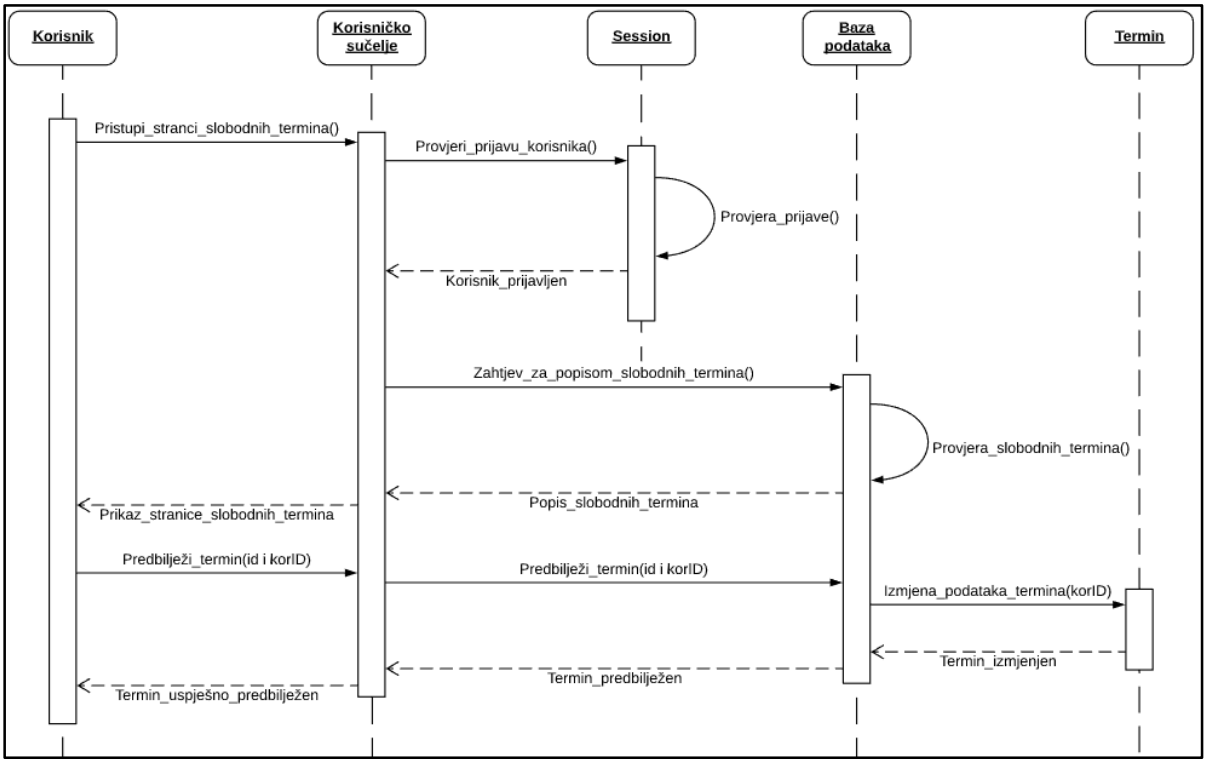
Slikom 13 prikazan je dijagram međudjelovanja za vremenski slijed razmjene poruka između objekta za slučaj uporabe stvaranja novog termina.



Slika 13. Dijagram međudjelovanja za slučaj uporabe otvaranja novog termina

Poslužitelj pokušava putem korisničkog sučelja pristupiti stranici za otvaranje novog termina. Korisničko sučelje zaprima zahtjev poslužitelja i šalje poruku objektu „Session“ da provjeri prijavu korisnika, kako bi se spriječili nelegitimni korisnici. Nakon što objekt „Session“ uspješno provjeri da postoje podaci o prijavi i da korisnik (poslužitelj) ima odgovarajuću ulogu, šalje poruku korisničkom sučelju da je korisnik prijavljen. Potom korisničko sučelje preusmjerava korisnika na stranicu za otvaranje termina gdje korisnik unosi podatke o terminu što uključuje datum, vrijeme početka, vrijeme završetka, opis i mjesto termina. Korisničko sučelje zatim prosljeđuje podatke bazi podataka, koja potom unosi novi termin u bazu podataka. Nakon uspješnog unosa termina baza podatka šalje poruku korisničkom sučelju da je termin uspješno stvoren, što poslužitelj i vidi putem korisničkog sučelja. Korisničko sučelje na kraju preusmjerava poslužitelja na stranicu otvorenih termina.

Slikom 14 prikazan je dijagram međudjelovanja za vremenski slijed razmjene poruka između objekta za slučaj uporabe predbilježivanja termina od strane korisnika.



Slika 14. Dijagram međudjelovanja za slučaj uporabe predbilježbe termina

Nakon što je poslužitelj uspješno otvorio novi termin, korisnik se može predbilježiti za njega. Korisnik najprije želi pristupiti stranici otvorenih termina putem korisničkog sučelja. Korisničko sučelje provjerava prijavu korisnika na isti način kao i u slučaju uporabe stvaranja novog termina. Korisničko sučelje nakon uspješne provjere prijave šalje zahtjev bazi podataka da joj preda popis slobodnih termina. Baza podataka predaje popis slobodnih termina nakon što ih sve pronađe u bazi podataka, koje potom korisničko sučelje prikazuje korisniku. Korisnik odabire željeni termin pritiskom na gumb, čime predaje id termina korisničkom sučelju. Korisničko sučelje šalje poruku bazi podataka da predbilježi korisnika za termin te mu predaje id termina i korisnički ID koji je dohvaćen iz objekta „Session“. Baza podataka pronalazi termin pod predanim id-em i upisuje korisnički ID u podatke termina, čime se označuje da je korisnik s tim korisničkim id-em predbilježen za taj termina. Baza podataka potom javlja korisničkom sučelju da je termin uspješno predbilježen, što korisnik i vidi putem korisničkog sučelja.

6.3. Dijagram suradnje

Dijagram suradnje prikazuje strukturalnu povezanost objekta. On može prikazati dinamičko ponašanje objekta za jedan slučaj uporabe. Iako je vrlo sličan dijagramu međudjelovanja glavna razlika je u tome što se njime ne prikazuje crta života i okvir aktivnosti.

Dijagram suradnje sliči dijagramu toka koji prikazuje uloge, funkcionalnost i ponašanje pojedinih objekata, kao i cjelokupni rad sustava u stvarnom vremenu. Četiri glavne komponente dijagrama suradnje su:

1. Objekti – prikazuju se pravokutnikom koji ima oznaku naziva unutar pravokutnika. Naziv objekta je najčešće jednak nazivu klase koju predstavlja, ali u nekim slučajevima može nositi i naziv stanja.

2. Korisnici ili akteri (eng. *Actors)* – oni pokreću cijeli slučaj uporabe. Svaki akter ima ime i ulogu te je on pokretač svih interakcija u dijagramu.

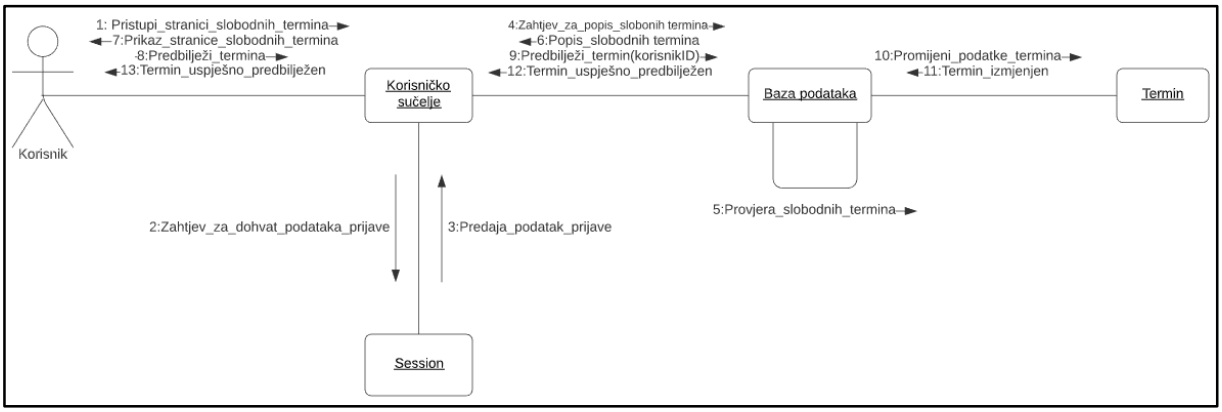
3. Asocijacije ili linkovi (eng. *Links*) – povezuju objekte s glumcima i prikazuju se pomoću pune crte između dva elementa dijagram. Svaka poruka se može poslati, ako postoji asocijacija odnosno veza između dva elementa.

4. Poruke – označuju se pomoću popunjene strelice koja se obično nalazi iznad ili pored linka. Naziv akcije, odnosno poruke, se obično piše iznad ili pored strelice te sadržava sekvencijski broj poruke, [22].

Budući da dijagrami suradnje i dijagrami međudjelovanja koriste iste podatke i predstavljaju iste informacije, može se reći da su semantički jednaki. Iz tog razloga moguće je zamijeniti dijagrame bez gubitka informacija, ali to ne znači da dijagrami vizualno prezentiraju informaciju na identičan način, [18].

6.4. Primjer dijagrama suradnje za aplikaciju podvorbenog sustava s mogućnošću predbilježbe

Slikom 15 prikazan je dijagram suradnje za slučaj uporabe predbilježbe termina od strane korisnika nakon što se prijavio u sustav.



Slika 15. Dijagram suradnje za slučaj uporabe predbilježbe termina

Korisnik putem korisničkog sučelja pokušava pristupiti stranici slobodnih termina. Korisničko sučelje zaprima podatke prijave korisnika koje je tražilo od objekta „Session“ te potom šalje zahtjev za popisom slobodnih termina bazi podataka. Baza podataka nakon što pronađe slobodne termine šalje popis istih korisničkom sučelju, koje potom korisniku prikazuje stranicu slobodnih termina. Korisnik putem korisničkog sučelja bira jedan od slobodnih termina i pokušava ga predbilježiti. Korisničko sučelje šalje poruku bazi podataka da predbilježi korisnika za željeni termin te mu predaje korisnički ID koji je ranije dohvaćen iz objekta „Session“. Baza podataka mijenja podatke termina i javlja korisničkom sučelju da je željeni termin uspješno predbilježen, što korisnik može i vidjeti putem korisničkog sučelja.

7. Modeliranje implementacije sustava

Prilikom projektiranja kompleksnih sustava potrebna dokumentacija može biti vrlo opširna, iz tog razloga se koriste UML dijagrami komponenata i rasporeda. Oni prikazuju fizičke komponente sustava koje su implementirane u fizičke jedinice sustava, u realnom vremenu. Dijagramom rasporeda prikazuje se konačna topologija sustava, dok dijagram komponenti prikazuje logičku strukturu i ovisnost fizičkih komponenta sustava.

7.1. Dijagram komponenti i rasporeda

Dijagram komponenti predstavlja logičku ovisnost između fizičkih komponenata i artefakta sustava. Pod fizičke komponente sustava se ubrajaju svi izvršni programi, datoteke, dokumenti, biblioteke, itd. Koje se nalaze implementirane u čvoru sklopa. Artefakti sustava mogu biti izvorni (eng. *Source*) i izvršni (eng. *Executable*). Izvorni artefakti sustava su od velikog značaja za implementaciju sustava. To su obično datoteke, odnosno izvorni kodovi, koji su zaduženi za stvaranje jedne ili više klasa. Izvršni artefakti su izvršne programske datoteke koje povezuju sve binarne komponente, [2].

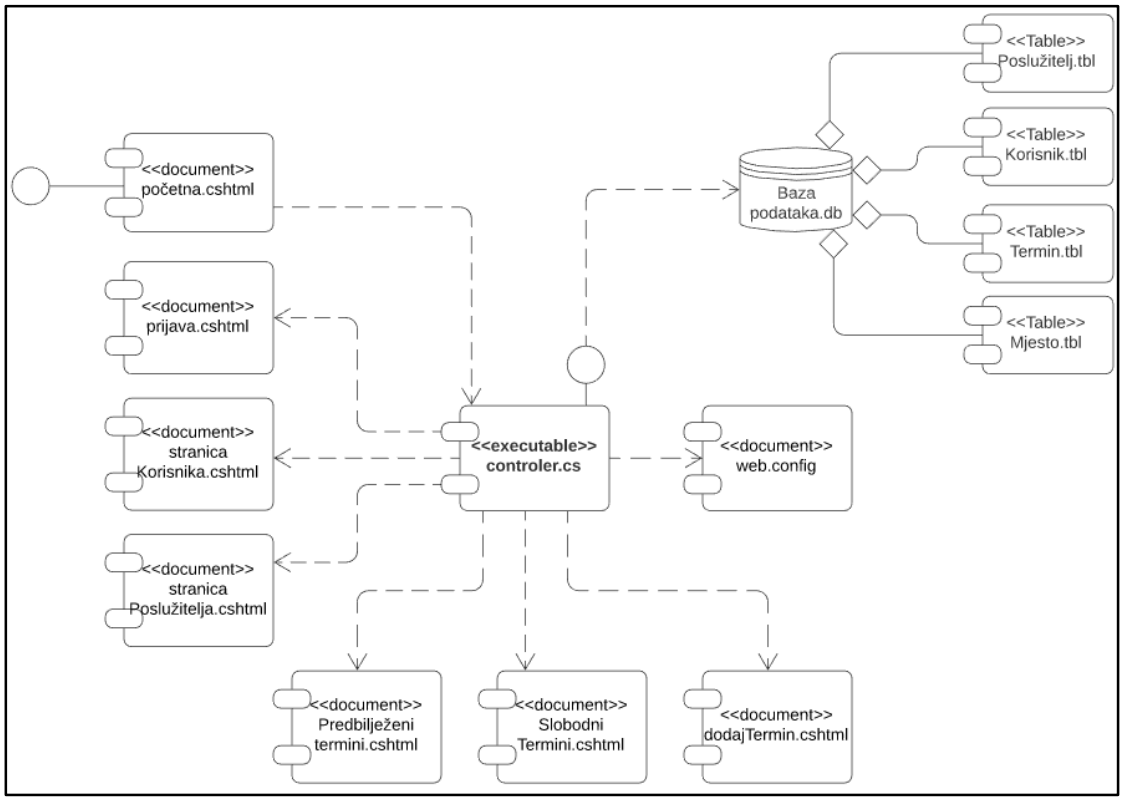
Dijagram komponenti se koristi za prikaz implementacije statičke strukture sustava. Prilikom modeliranja implementiranja statičke strukture sustava, dijagrami komponenti se najčešće koriste za jedan od četiri načina modeliranja, a to su: modeliranje izvornog koda, modeliranje završne verzije sustava, modeliranje fizičkih baza podataka i modeliranje prilagodljivih sustava, [18].

Dijagram rasporeda prikazuje arhitekturu izvršnih uređaja, što uključuje i njihovu okolinu i artefakte. Prikazuje konačni fizički opis topologije sustava, strukturu hardvera i softvera koji se nalazi na svakoj fizičkoj jedinici unutar sustava. Kroz tako opisanu strukturu sustava lako se može primijetiti koje fizičke i logičke komponente su implementirane u pojedinom čvoru sustava. Čvorovi predstavljaju fizičke jedinice na kojoj mogu biti implementirani artefakti sustava te imaju sposobnost obrade podataka. Najčešće su to osobna računala, čitači kartica, mobilni terminalni uređaji, itd. Čvorovi se u dijagramu označuju simbolom kocke koja na prednjoj stranici ima upisan naziv čvora, [2].

Dijagrami rasporeda se obično primjenjuju u većim i kompleksnim sustava gdje postoji veliki broj čvorova. Primjeri takvih sustava su ugrađeni sustavi (eng. *Embedded*), klijent/poslužitelj i distribuirani sustavi. Za manje sustava na kojima se baza podataka, aplikacija i korisničko sučelje nalazi na jednom ili malom broju računala se obično izrađuje samo dijagram komponenata.

7.2. Primjer dijagrama komponenta web aplikacije sustava s mogućnošću predbilježbe

Slikom 16 prikazan je dijagram komponenta za aplikaciju podvorbenog sustava s mogućnošću predbilježbe. Njime su prikazane sve programske komponente koje su potrebne za pravilan rad sustava.



Slika 16. Dijagram komponenata za aplikaciju sustava s mogućnošću predbilježbe

Korisnik pomoću odgovarajućeg sučelja pristupa početnoj stranici. Zavisno o korisnikovim zahtjevima i predanim parametrima izvršna datoteka „controler.cs“ prosljeđuje korisnika na jednu od sljedećih stranica prijava, korisnika, poslužitelja, predbilježi termin, slobodni termini ili dodaj termin. U dokumentu „web.config“ se pohranjuju podaci o uspostavljenoj vezi s bazom podataka. Izvršna datoteka pristupa bazi podataka preko odgovarajućeg sučelja. U bazi podataka se nalaze tablice poslužitelj, korisnik, termin i mjesto. Veza između baze podatak i tablica je takva da kada se uništava baza podataka zajedno s njom se uništavaju i sve tablice koje se nalaze u njoj.

8. Web aplikacija podvorbenog sustava s mogućnošću predbilježbe

Na temelju model procesa, objašnjenih pomoću UML-a, napravljena je aplikacija za specifični slučaj predbilježbe termina za liječnički pregled, koja prati opisan proces. U izradi aplikacije korišten je programski jezik C# i Microsoft SQL (eng. *Structured Query Language*) relacijska baza podataka koja primarno koristi jezik za upite Transact SQL (T-SQL).

8.1. Programski jezici, baza podatka i jezik za označavanje aplikacije

Aplikacija je razvijena pomoću obrasca razvoja MVC (eng. *Model-View-Controller*). Modeli upravljaju poslovnim pravilima, podacima i logikom koja se koristi u aplikaciji. Upravljač (eng. *Controller*) prihvaća unose korisnika i pretvara ih u naredbe koje dalje prosljeđuje modelu i pogledu. Pogled (eng. *View*) predstavlja informacije korisniku pomoću korisničkog sučelja. Za izradu pogleda koristio se Razor pogled. On pruža sintaksu označavanja za ugrađivanje poslužiteljskog koda u web stranice. Osnovni jezik koji koristi je HTML-a (eng. *HyperText Markup Language*) u kombinaciji sa serverskim kodom i C#-om. MVC radi na takav način da prvo korisnik inicira upravljač koji manipulira podacima modela. Na temelju izmijenjenih podataka model ažurira podatke u pogledu, koji se potom prikazuje korisniku putem pogleda. Sama aplikacija je izrađena pomoću programskog alata Microsoft Visual Studio 2019, dok je za izradu baza podataka korišten Microsoft SQL Server Management Studio 2014.

8.1.1. Programski jezik C#

Programski jezik C# je viši programski jezik razvijen od strane Microsofta, namijenjen za izradu aplikacija. On je objektno orijentirani jezik što omogućuje upotrebu koncepata enkapsulacije, nasljeđivanja i polimorfizma. Sve aplikacije koje su razvijene pomoću C#-a koriste .NET razvojno okruženje. Ono pruža veliki broj biblioteka klasa, koje programerima omogućuje upotrebu pouzdanog kod za sva glavna područja razvoja aplikacija, [23].

8.1.2. SQL server baza podataka

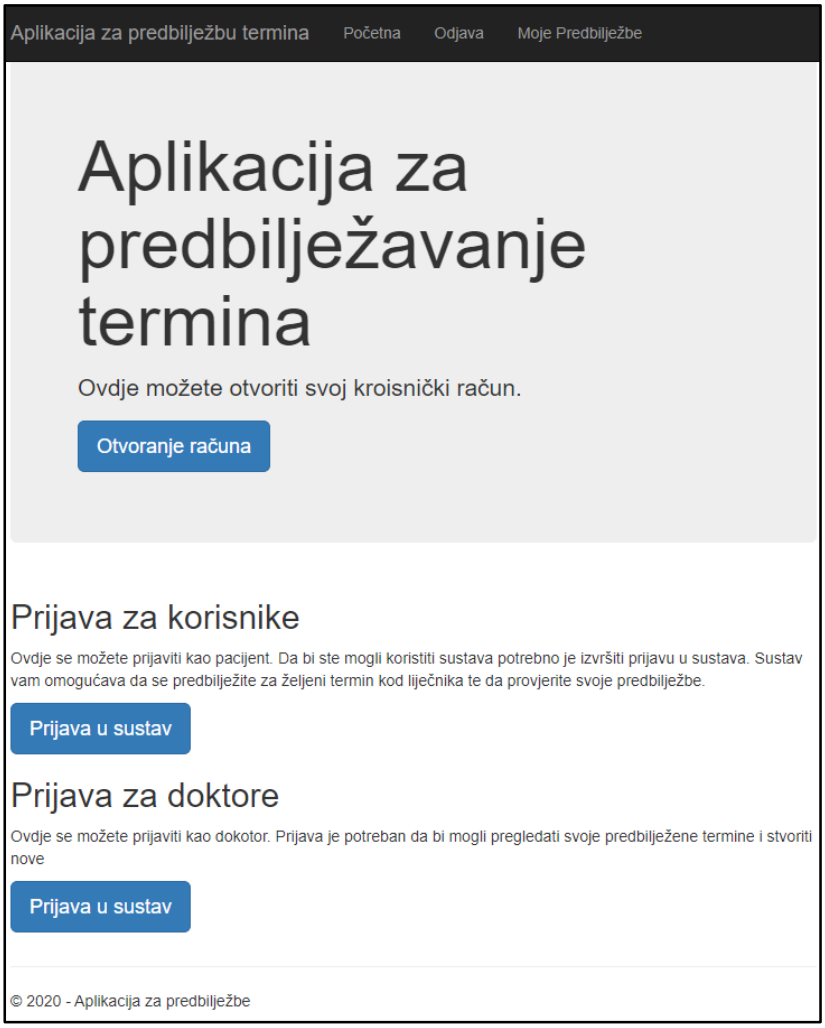
SQL server je relacijska baza podataka razvijena od strane Microsofta, koja se temelji na jeziku za upite T-SQL. Ono što čini SQL server drugačijim od T-SQL je to da ima mogućnost promjene programskog toka, dok T-SQL ima mogućnost samo dohvata podataka iz baze podataka. Također SQL server omogućuje povezivanje baze podataka s aplikacijom, iz tog razloga je vrlo često korišten za izradu baze podataka web aplikacija. Sve aplikacije koje su pisane u .NET okruženju u programskom kodu mogu pozivati funkcije i procedure koje su pohranjene u bazi podataka, [24].

8.1.3. Razor pogled

Razor pogled pruža sintaksu označavanja za ugrađivanje poslužiteljskog koda u web stranice. On koristi sintaksu označavanja u kombinaciji s C# i HTML-om, i prepoznatljiv je po ekstenziji datoteke *.cshtml*, [25]. HTML je kod koji se koristi za strukturu web stranica i njenog sadržaja. Struktura web stranice se postiže korištenjem različitih elementa, odjeljka, tablica, paragrafa i još mnogih drugih elementa, kako bi se neki sadržaj prikazao na određeni način, [26].

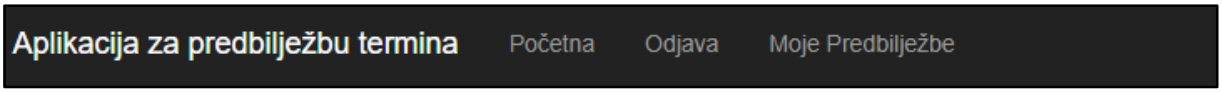
8.2. Izvedba web aplikacije

Glavni korisnici ove aplikacije su pacijenti i liječnici, kojima je dodijeljena odgovarajuća uloga unutar sustava, odnosno aplikacije. Prilikom pristupa web aplikaciji korisnik odnosno posjetitelj sustava je preusmjeren na početnu stranicu. Početna stranica je prikazana slikom 17.



Slika 17. Početna stranica web aplikacije

Iz slike 17 se može vidjeti da je početna stranica podijeljena u tri odjeljka, te se na vrhu stranice nalazi zaglavlje koje je prikazano slikom 18. Prvi dio početne stranice se odnosi na posjetitelje sustava, i u njemu se nalazi poveznica za otvaranje novog korisničkog računa koji je potreban za pristup ostalim funkcionalnostima sustava odnosno aplikacije. Drugi i treći odjeljak se odnose na prijavu korisnika u sustav, gdje se nalaze poveznice koji prosljeđuju korisnike na odgovarajuću stranicu za prijavu.

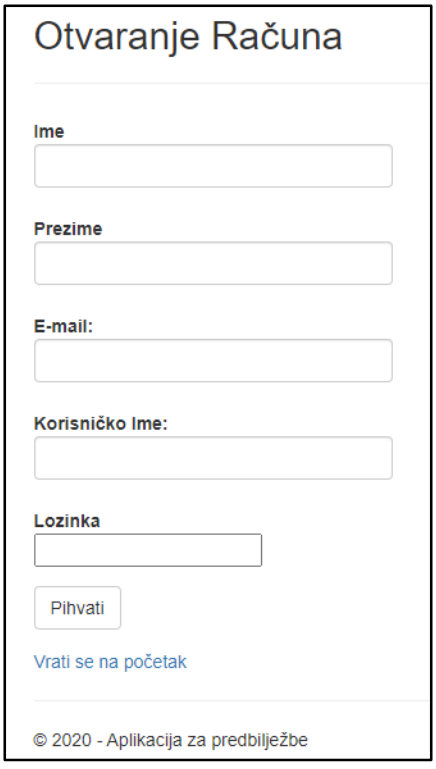


Slika 18. Zaglavlje početne stranice

Slikom 18 prikazano je zaglavlje stranice, koje se nalazi na vrhu svake stranice aplikacije. Zaglavlje je podijeljeno u četiri glavna dijela. Prvi dio zaglavlja nosi naziv same aplikacije, te ono prosljeđuje korisnika na početnu stranicu aplikacije, u slučaju da korisnik pritisne na njega. Drugi dio zaglavlja se naziva „Početna“ te zavisno o korisničkoj ulozi preusmjerava korisnika na početnu stranicu za korisnike ili liječnike. Treći dio zaglavlja je zadužen za odjavu trenutno prijavljenog korisnika. Zadnji dio zaglavlja nosi naziv „Moje Predbilježbe“ te zavisno o korisničkoj ulozi korisnika se preusmjerava na stranicu predbilježenih termina.

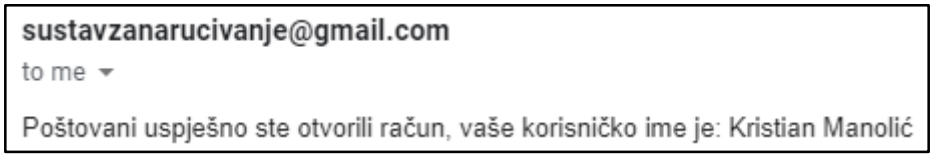
8.2.1. Otvaranje računa i prijava u sustav

Kako bi korisnik mogao pregledavati i predbilježiti termine nužno je da ima otvoreni korisnički račun. Pritiskom poveznice u prvom odjeljku početne stranice preusmjerava se korisnika na stranicu za otvaranje računa, koja je prikazana slikom 19.



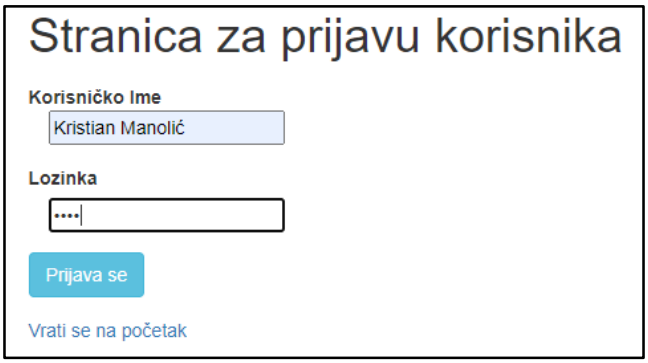
Slika 19. Stranica otvaranja računa

Korisnik popunjaju tražene podatke i predaje ih aplikaciji pritiskom poveznice „Prihvati“. Aplikacija prvo šalje naredbu bazi podataka da upiše korisnika, a potom mu automatski šalje e-poštu koja potvrđuje da je korisnik otvorio svoj račun. Slikom 20 prikazana je e-pošta koju korisnik zaprima nakon uspješnog otvaranja računa.



Slika 20. Prikaz e-pošte uspješnog otvaranja korisničkog računa

Nakon što je korisnik zaprimio e-poštu aplikacija ga preusmjerava na stranicu za prijavu, koja je prikazana slikom 21. Korisnik upisuje svoje korisničke podatke u odgovarajuća polja i pritiskom poveznice „Prijavi se“ predaje podatke aplikaciji. Aplikacija pronalazi korisnika u bazi podataka i posprema podatke o prijavljenom korisniku, koji su potrebni da aplikacija funkcionira ispravno. Nakon uspješne prijave u sustav korisnik može koristi sve funkcionalisti aplikacije, koje zavise o njegovoj ulozi.



Slika 21. Prikaz stranice za prijavu

Postupak prijave liječnika u sustav je isti samo koristi stranicu prijave za liječnike. Glavna razlika je u tome što liječnik ne otvara sam račun, već ga stvara osoba koja je zadužena za upravljanje bazom podataka putem Microsoft SQL Server Management Studija.

8.2.2. Postupak otvaranja i predbilježbe termina

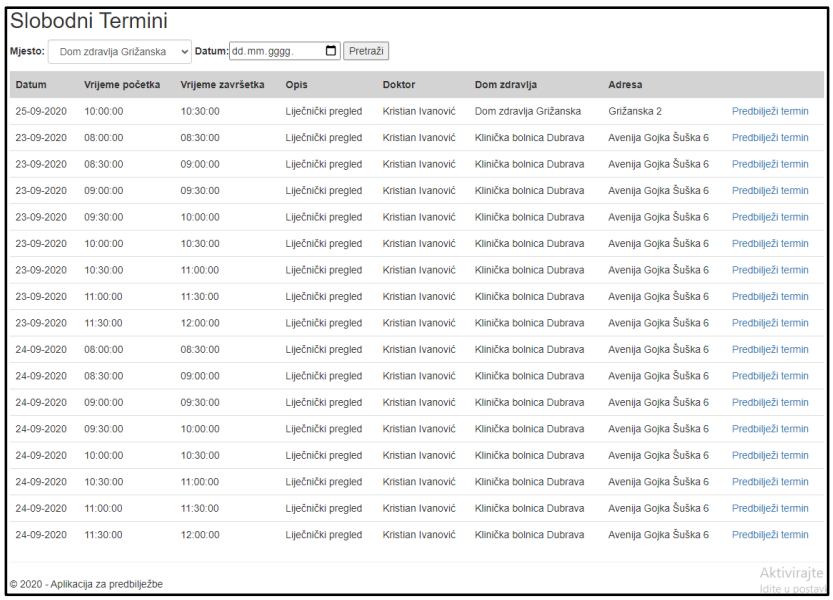
Za otvaranje novih termina je zadužena osoba koja ima ulogu liječnika u sustavu. Liječnik nakon uspješne prijave u sustav je preusmjeren na stranicu za liječnike gdje ima opciju stvaranja termina. Stranica za otvaranje prikazana je slikom 22.

Slika 22. Prikaz stranice za otvaranje termina 1

Liječnik upisuje tražene podatke u stranicu i pritiskom na poveznicu „Otvori termin“ ih predaje aplikaciji. Potom se isti podaci šalju bazi podataka koja potom stvara novi termin, koji je vidljiv korisnicima putem stranice za pretraživanje slobodnih termina. S obzirom na to da postoji potencijalno veliki broj termina koji se trebaju otvoriti, liječniku je omogućeno i otvaranje većeg broja termina putem stranice za unos većeg broja termina, koja je prikazana slikom 24.

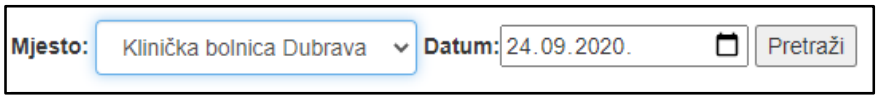
Slika 23. Prikaz stranice za otvaranje termina 2

Postupak otvaranja većeg broja termina je isti kao onaj za otvaranje pojedinačnog termina. Od liječnika se traži da upiše početni datum i završni datum za koji želi stvoriti termine. Početak radnog vremena je vrijeme prvog termina za taj datum, a završetak radnog vremena predstavlja zadnje moguće vrijeme početka termina. Razmak između početka termina zapravo predstavlja trajanje termina. Nakon što liječnik stvori željene termine, korisnici se mogu predbilježiti za njih putem stranice slobodnih termina, koja je prikazana slikom 24.



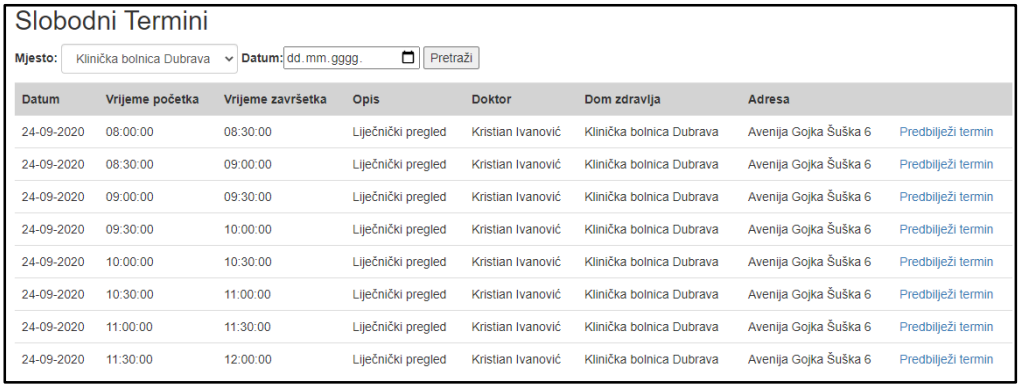
Slika 24. Stranica slobodnih termina

S obzirom na to da stranica slobodnih termina prikazuje sve slobodne termine koji su raspoloživi poslije trenutnog datuma, korisnicima je omogućena pretraga po mjestu i datumu. Tražilica se nalazi pri vrhu stranice i prikazana je slikom 25.



Slika 25. Prikaz tražilice slobodnih termina

Nakon što korisnik unese parametre pretraživanja i stisne poveznicu „Pretraži“ aplikacija vraća popis svih raspoloživih termina za definirane parametre, što je prikazano slikom 26.



Datum	Vrijeme početka	Vrijeme završetka	Opis	Doktor	Dom zdravlja	Adresa	
24-09-2020	08:00:00	08:30:00	Liječnički pregled	Kristian Ivanović	Klinička bolnica Dubrava	Avenija Gojka Šuška 6	Predbilježi termin
24-09-2020	08:30:00	09:00:00	Liječnički pregled	Kristian Ivanović	Klinička bolnica Dubrava	Avenija Gojka Šuška 6	Predbilježi termin
24-09-2020	09:00:00	09:30:00	Liječnički pregled	Kristian Ivanović	Klinička bolnica Dubrava	Avenija Gojka Šuška 6	Predbilježi termin
24-09-2020	09:30:00	10:00:00	Liječnički pregled	Kristian Ivanović	Klinička bolnica Dubrava	Avenija Gojka Šuška 6	Predbilježi termin
24-09-2020	10:00:00	10:30:00	Liječnički pregled	Kristian Ivanović	Klinička bolnica Dubrava	Avenija Gojka Šuška 6	Predbilježi termin
24-09-2020	10:30:00	11:00:00	Liječnički pregled	Kristian Ivanović	Klinička bolnica Dubrava	Avenija Gojka Šuška 6	Predbilježi termin
24-09-2020	11:00:00	11:30:00	Liječnički pregled	Kristian Ivanović	Klinička bolnica Dubrava	Avenija Gojka Šuška 6	Predbilježi termin
24-09-2020	11:30:00	12:00:00	Liječnički pregled	Kristian Ivanović	Klinička bolnica Dubrava	Avenija Gojka Šuška 6	Predbilježi termin

Slika 26. Prikaza rezultata pretraživanja slobodnih termina

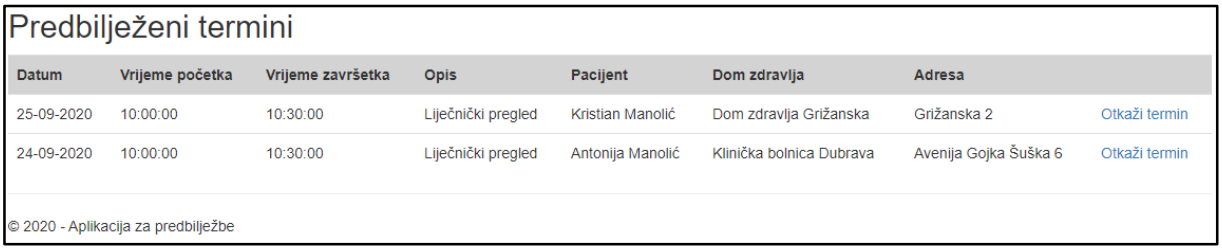
Kada korisnik pronađe slobodni termin koji mu odgovara on se može predbilježiti pomoću pritiskom na poveznicu „Predbilježi termin“ koja se nalazi u krajnjem desnom rubu stranice. Nakon što korisnik pritisne poveznicu aplikacija šalje poruku bazi podataka da predbilježi korisnika za željeni termin. Aplikacija poslije uspješne predbilježbe termina šalje e-poštu korisniku koja je prikazana slikom 27.

Slika 27. Prikaz e-pošte potvrde predbilježbe termina

8.2.3. Analiza podataka dobivenih aplikacijom

Cijela aplikacija je osmišljena kako bi se olakšao proces predbilježbe termina od strane korisnika da on bespotrebno ne bi trebao čekati na uslugu. Aplikacija ima univerzalnost da se može primijeniti u bilo kojem slučaju gdje se korisnici predbilježuju za pružanje neke usluge.

Proces predbilježbe termina se odvija tako da prvo liječnik otvara termin za pružanje usluge na određenom mjestom u određeno vrijeme. Kada se otvori novi termin on je u stanju slobodan odnosno niti jedan korisnik nije predbilježen za njega. Korisnik pretražuje slobodne termine te predbilježuje željeni termin pomoću stranice slobodnih termina. Time termin prelazi u stanje predbilježen. Kada je termin predbilježen, liječnik na stranici predbilježenih termina vidi sve svoje predbilježene termine. Liječnik putem stranice ima uvid na kojem mjestu se pruža usluga, koji pacijent (korisnik) je predbilježio termin, datum, vrijeme početka termina i ima mogućnost otkazati termin. Stranica predbilježenih termina liječnika prikazana je slikom 28.



Slika 28. Prikaz stranice predbilježenih termina liječnika

U slučaju da liječnik ili pacijent ne može stići na određeni termin on ima mogućnost otkazati termin, pritiskom na poveznicu „Otkaži termin“. Time termin prelazi u stanje slobodan. Aplikacija automatski obavještava korisnika o otkazivanju termina putem e-pošte. Primjer e-pošte koju korisnik zaprima prikazana je slikom 29.

Slika 29. Prikaz e-pošte o otkazivanju termina

Svaki otvoreni ili predbilježeni termin kojem je prošao datum održavanja termina se ne prikazuje korisniku, te time prelazi u stanje zatvoren.

9. Zaključak

U današnje vrijeme razvoj sustava može biti vrlo kompliciran i zahtjevan proces te može uključivati veliki broj radnih skupina. Zbog toga se veći sustavi raščlanjuju na manje segmente, koji se prikazuju pomoću modela. Iz tog razloga, u ovom radu su upotrjebljeni odgovarajući UML dijagrami za prikaz modela sustava. Time je omogućen prikaz kompleksnih sustava u više manjih lako razumljivih modela, što omogućuje jednostavniju komunikaciju rješenja, i u konačnici razvoj samog sustava.

Cilj ovog rada bio je pomoću UML dijagrama prikazati zahtjeve i procese koji su postavljeni na podvorbeni sustav s mogućnošću predbilježbe te realizirati web aplikaciju na temelju dobivenih modela. Aplikacijom je omogućeno korisnicima da prijave željeni termin i obavještavanje korisnicima o promjeni stanja predbilježenog termina.

Web aplikacija je razvijena uz pomoć MVC obrasca razvoja, i u izradi iste su korišteni programski jezik C#, relacijska baza podataka SQL Server i Razor pogled. Pomoću UML dijagrama prikazan je proces i zamisao same aplikacije za predbilježbu korisnika, kojoj je glavna zadaća povećanje efikasnost iskorištenja kapaciteta repa i smanjenje vremena čekanja korisnika u repu.

Razvijena web aplikacija ukazuje na to da se pomoću dobivenih UML dijagrama mogu prikazati glavni zahtjevi i procesi unutar podvorbenog sustava s mogućnošću predbilježbe i mogu se iskoristiti u razvoju rješenja takvih sustava. U konačnici aplikacijom se smanjuje vrijeme koje korisnik provede u repu sustava, što može povećati korisničko zadovoljstvo uslugom.

Literatura

[1] Rumbaugh J, Jacobson I, Booch G. The Unified Modeling Language Reference Manual Second Edition, Pearson Education Inc., Boston; 2005.

[2] Eriksson H, Penker M, Lyons B, Fado D. UML 2 Toolkit, Wiley Publishing, Indianapolis; 2004.

[3] Tutorialspoint. UML overview. Dostupno na: https://www.tutorialspoint.com/uml/uml_overview.htm (Pristupljeno: srpanj.2020.)

[4] Progres OpenEdge. Encapsulation. Dostupn na: https://documentation.progress.com/output/ua/OpenEdge_latest/index.html#page/gsdev%2Fencapsulation.html%23 (Pristupljeno: srpanj 2020.)

[5] Progres OpenEdge. Inheritance. Dostupno na: https://documentation.progress.com/output/ua/OpenEdge_latest/index.html#page/gsdev%2Finheritance.html%23 (Pristupljeno: srpanj 2020.)

[6] Microsoft. Polymorphism. Dostupno na: https://docs.microsoft.com/en-us/dotnet/csharp/programming-guide/classes-and-structs/polymorphism (Pristupljeno: srpanj 2020.)

[7] Mrvelj Š. Uvod. Autorizirana predavanja. Fakultet prometnih znanosti Sveučilišta u Zagrebu; 2018./2019.

[8] Sarkara D, Bhalla M. Enhancing Unified Process Workflows using UML, U: 7th International Conference on Cloud Computing, Data Science & Engineering; 2017.

[9] Rational. Rational Unified Process Best Practices for Software Development Teams; 1998.

[10] Anwar A. A Review of RUP (Rational Unified Process), International Journal of Software Engineering (IJSE), 2014;5(2): 8-24.

[11] IBM. Use-case diagrams. Dostupno na: https://www.ibm.com/support/knowledgecenter/SS8PJ7_9.7.0/com.ibm.xtools.modeler.doc/topics/cucd.html ([Pristupljeno: srpanj 2020.)

[12] Mrvelj Š. Prikupljanje zahtijeva na sustav. Autorizirana predavanja. Fakultet prometnih znanosti; 2014/2015.

[13] Visual paradigm. UML Class diagram tutorial. Dostupno na: https://www.visual-paradigm.com/guide/uml-unified-modeling-language/uml-class-diagram-tutorial/ (Pristupljeno: srpanj 2020.)

[14] Mrvelj Š. Dijagram klasa. Autorizirana predavanja. Fakultet prometnih znanosti Sveučilišta u Zagrebu; 2016./2017.

[15]Visual paradigm. What is state machine diagram. Dostupno na: https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-state-machine-diagram/ (Pristupljeno: srpanj 2020.)

[16] UML diagrams. State-machine diagrams. Dostupno na: https://www.uml-diagrams.org/state-machine-diagrams.html (Pristupljeno: kolovoz 2020.)

[17] Mrvelj Š. Opis tijekova podataka. Autorizirana predavanja. Fakultet prometnih znanosti Sveučilišta u Zagrebu; 2019.

[18] Booch G, Rumbaugh J, Jacobson I. The Unified Modeling Language User Guide. Addison Wesley, 1998.

[19] Creately. Sequence Diagram Tutorial: Complete Guide with Examples. Dostupno na: https://creately.com/blog/diagrams/sequence-diagram-tutorial/ (Pristupljeno: kolovoz 2020.)

[20] Mrvelj Š. Opis zajedničkog rada elemenata. Autorizirana predavanja. Fakultet prometnih znanosti Sveučilišta u Zagrebu; 2018.

[21] Mrvelj Š. Dijagrami interakcije (2. dio) dijagram suradnje. Autorizirana predavanja. Fakultet prometnih znanosti Sveučilišta u Zagrebu; 2017/2018.

[22] Techtarget. Collaboration diagram. Dostupno na: https://searchsoftwarequality.techtarget.com/definition/collaboration-diagram (Pristupljeno: kolovoz 2020.)

[23] Microsoft. Introduction to the C# language and the .NET Framework. Dostupno na: https://docs.microsoft.com/en-us/dotnet/csharp/getting-started/introduction-to-the-csharp-language-and-the-net-framework (Pristupljeno: kolovoz 2020.)

[24] SQLserver.tutorial. What is SQL Server. Dostupn na: https://www.sqlservertutorial.net/getting-started/what-is-sql-server/ (Pristupljeno: kolovoz 2020.)

[25] Microsoft. Razor syntax reference for ASP.NET Core. Dostupno na: https://docs.microsoft.com/en-us/aspnet/core/mvc/views/razor?view=aspnetcore-3.1 (Pristupljeno: kolovoz 2020.)

[26] W3schools. What is HTML. Dostupno na: https://www.w3schools.com/whatis/whatis_html.asp (Pristupljeno: kolovoz 2020.)

Popis kratica

UML	Unified Modeling Language
UP	Unified Process
CRM	Customer relationship management
SQL	Structured Query Language
T-SQL	Transact SQL
MVC	Model-View-Controller
HTML	HyperText Markup Language

Popis slika