Analiza, modeliranje i izvedba sustava za informiranje koji raspoznaje različite uloge korisnika

Kralj, Franjo

Sveučilište u Zagrebu
Fakultet prometnih znanosti

DIPLOMSKI RAD

ANALIZA, MODELIRANJE I IZVEDBA SUSTAVA ZA INFORMIRANJE KOJI RASPOZNAJE RAZLIČITE ULOGE KORISNIKA

ANALSYS, MODELLING AND IMPLEMENTATION OF INFORMATION SYSTEM CAPABLE OF DISTINGUISHING USER ROLES

Mentor: doc. dr. sc. Marko Matulin

Student: Franjo Kralj
JMBAG: 0135234186

Zagreb, rujan 2021.

SVEUČILIŠTE U ZAGREBU
FAKULTET PROMETNIH ZNANOSTI
POVJERENSTVO ZA DIPLOMSKI ISPIT

Zagreb, 13. rujna 2021.

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

DIPLOMSKI ZADATAK br. 6220

Pristupnik: **Franjo Kralj (0135234186)**
Studij: Promet
Smjer: Informacijsko-komunikacijski promet

Zadatak: **Analiza, modeliranje i izvedba sustava za informiranje koji raspoznaje različite uloge korisnika**

Opis zadatka:

Prikazati primjenjivost UML pristupa u modeliranju različitih sustava i informacijsko-komunikacijskih aplikacija. Identificirati i analizirati sudionike sustava za informiranje korisnika i njihove uloge. Na proizvoljnom sustavu za informiranje prikazati međuodnose različitih objekata primjenom UML jezika. Nakon završetka modeliranja proizvoljnog sustava za informiranje, izvesti sustav koristeći web tehnologije za izradu aplikacija i prikazati njegove funkcionalnosti

Mentor:

doc. dr. sc. Marko Matulin

Predsjednik povjerenstva za diplomski ispit:

prof. dr. sc. Štefica Mrvelj

Sažetak

U radu se analiziraju glavni procesi i značajke *web* aplikacije koja raspoznaje dvije vrste korisnika, i ovisno o ulozi korisnika omogućava radnje unutar nje. Sustav je modeliran pomoću UML (*Unified Modeling Language*) modela, a temeljem tih modela izrađena je *web* aplikacija. Pri izradi aplikacije korišten je MVC (*Model-View-Controller*) obrazac razvoja *web* aplikacija, programskim jezicima C# i HTML (*HyperText Markup Language*). Sama aplikacija izrađena je u programskom alatu Microsoft Visual Studio 2019, za izradu baze podataka korišten Microsoft SQL Server Management Studio 2014. UML modeli su izrađeni u programskom alatu Microsoft Visio.

Ključne riječi: UML, MVC, modeliranje, *web* aplikacija

Summary

The paper analyzes the main processes and features of a web application that recognizes two types of users and depending on the role of the user allows actions within the application. The system was modeled using UML (Unified Modeling Language) models, and a web application was created based on them. MVC (Model-View-Controller) form was used for the development of web application. Application was written in C# and HTML (HyperText Markup Language) programming languages. The application itself was created in the Microsoft Visual Studio 2019 software tool, while Microsoft SQL Server Management Studio 2014 was used to create the database. UML models were created in the Microsoft Visio software tool.

Key words: UML, MVC, modeling, web application

Sadržaj:

1. Uvod

Zadatak diplomskog rada je analiza, modeliranje i izvedba sustava za informiranje koji raspoznaje različite uloge korisnika. Sustav je izveden u obliku *web* aplikacije koja raspoznaje dvije vrste korisnika. Ovisno o ulozi koja mu je dodijeljena, korisnik ima pristup određenim funkcionalnostima aplikacije. Svi procesi koji se odvijaju unutar sustava prikazani su pomoću UML (*Unified Modeling Language*) jezika za modeliranje. Za razvoj *web* aplikacije korišten je MVC (*Model-View-Controller*) obrazac razvoja *web* aplikacija s programskim jezicima C# i HTML (*HyperText Markup Language*).

U nastavku rada, sustav je detaljnije opisan analizom i modeliranjem njegovih komponenti, kao i razradom tijeka događaja korištenja sustava.

Diplomski rad se sastoji od osam funkcionalno povezanih cjelina:

1. Uvod
2. Primjena UML jezika u modeliranju sustava
3. Analiza sudionika sustava za informiranje i njihovih uloga
4. Definiranje elemenata (objekata) sustava i njihovih međuodnosa
5. Razrada scenarija korištenja sustava za informiranje korisnika
6. Kratak opis metoda i alata korištenih u razvoju sustava
7. Prikaz funkcionalnosti izvedenog sustava za informiranje korisnika
8. Zaključak.

U drugom poglavlju rada definiran je UML kao jezik modeliranja sustava. Njegova povijest, potreba za njim i primjena prilikom razvoja sustava. U trećem poglavlju je definiran dijagram slučaja uporabe UML-a (use-case). Pomoću use-case dijagrama opisani su sudionici sustava za informiranje koji raspoznaje različite uloge korisnika.

U četvrtom poglavlju obrađena je tematika dijagrama klasa i definirane su klase *web* aplikacije za informiranje korisnika. U petom poglavlju, pomoću dijagrama međudjelovanja, razrađen je scenarij korištenja sustava za informiranje korisnika.

U šestom poglavlju, pomoću dijagrama suradnje, definirana je MVC metoda razvoja *web* aplikacije. U sedmom poglavlju prikazana je funkcionalnosti izvedenog sustava za informiranje korisnika razradom scenarija iz petog poglavlja. U zadnjem poglavlju autor daje svoj konačni komentar na teme prezentirane u radu.

2. Primjena UML jezika u modeliranju sustava

Definiranje pojmova sustava i modela dobar je uvod u shvaćanje UML-a kao koncepta i njegove primjene u modeliranju sustava.

2.1. Sustav

Brojni izvori literature definiraju pojam sustava na svoj način. U [1], Radošević ga definira kao jedino sredstvo u borbi protiv kompliciranosti, metoda pronalaska jednostavnosti u kompleksnosti. Pomalo filozofski, sustav je nešto oprečno kaosu. Pojmu sustava, srodni su i pojmovi reda, poretka ili organizacije u širem smislu. Skoro svaka pojava može predstavljati jedan ili više sustava. Sustav je cjelina koja svojim djelovanjem opravdava svoje postojanje [2].

Više tehnički nastrojenim žargonom, sustav je definiran kao skup elemenata (objekata, dijelova) i veza između njih, koji zajedno čine neki svrsishodan proces (funkciju). Svaki sustav može biti dio, ili u vezi s većim sustavom. Takve sustave se nazivaju podsustavima. Dio cjeline koji nije obuhvaćen sustavom naziva se okolina sustava [3]. Sustav predstavlja skup elemenata povezanih u svrsishodnu cjelinu koji u međusobnoj interakciji, prema određenim zakonima, obavljaju funkciju ostvarenja zajedničkih ciljeva [4]. Svaki sustav karakteriziraju tri stvari:

- *Input* (ulaz) – podrazumijeva skupljene ili dodijeljene veličine koje ulaze u sustav kako bi bile procesirane.
- *Process* (obrada) – predstavlja procese koji pretvaraju *Input* u *Output*.
- *Output* (izlaz) – su transformirane veličine koje su nastale procesom unutar sustava od *Inputa*.

2.2. Model

Model je prikaz stvarnog (ili zamišljenog) sustava u određenom mediju. Model bilježi ključne elemente sustava s određenog gledišta i pojednostavljuje ili izostavlja ostatak suvišnih detalja. Modeliranje se koristi u brojnim disciplinama inženjerstva, arhitekturi i drugim kreativnim područjima. Ovisno i potrebama sustav se promatra s određenog gledišta i modelira u prikladnom mediju za rad [5].

Modeli se izrađuju kako bi se postiglo bolje razumijevanje sustava. Kroz modeliranje se postižu sljedeći ciljevi:

- modeli pomažu u vizualizaciji kompleksnih (stvarnih ili zamišljenih) sustava,
- modeli opisuju strukturu i prikazuju djelovanje sustava,
- modeli predstavljaju predložak sustava koji se izgrađuje,
- modeli dokumentiraju postupke izgradnje sustava.

Svrha modela je u razrada djelovanja sustava i njegovo unaprjeđivanje. Dijeljenje ideja i mogućnost rada na razvoju sustava svih sudionika u razvoju sustava koji model znaju shvatiti. U svrhu rasprostiranja mogućnosti čitanja modela nastaju standardi i jezici modeliranja sustava. Jedan od jezika modeliranja je *Unified Modeling Language*, skraćeno UML [5].

2.3. Unified Modeling Language

UML, na hrvatskom, ujedinjeni jezik za modeliranje je standardizirani jezik opće namjene koji se koristi za modeliranje programskih sustava koji temelje na objektno-orijentiranoj paradigmi. Ujedinjeni, zato što objedinjuje prethodno razvijene tehnike modeliranja programskih sustava koji temelje na objektno-orijentiranoj paradigmi:

- *Object-Oriented Design with Applications* - Grady Booch (1993)
- *Object-Modeling Technique* (OMT) (1990) i
- *Object-Oriented Modeling and Design* - Jim Rumbaugh (1991)
- *Object-Oriented Software Engineering* (OOSE) - Ivar Jacobson (1992).

Booch, Rumbaugh i Jacobson odlučili su objediniti svoje jezike modeliranja stvorivši UML. UML je standardizirala Grupa za upravljanje objektima (eng. *Object Management Group*, kraće: OMG) 1997. godine. Tijekom godina nastajale su brojne inačice UML-a sa zadnjom 2015. godine [6].

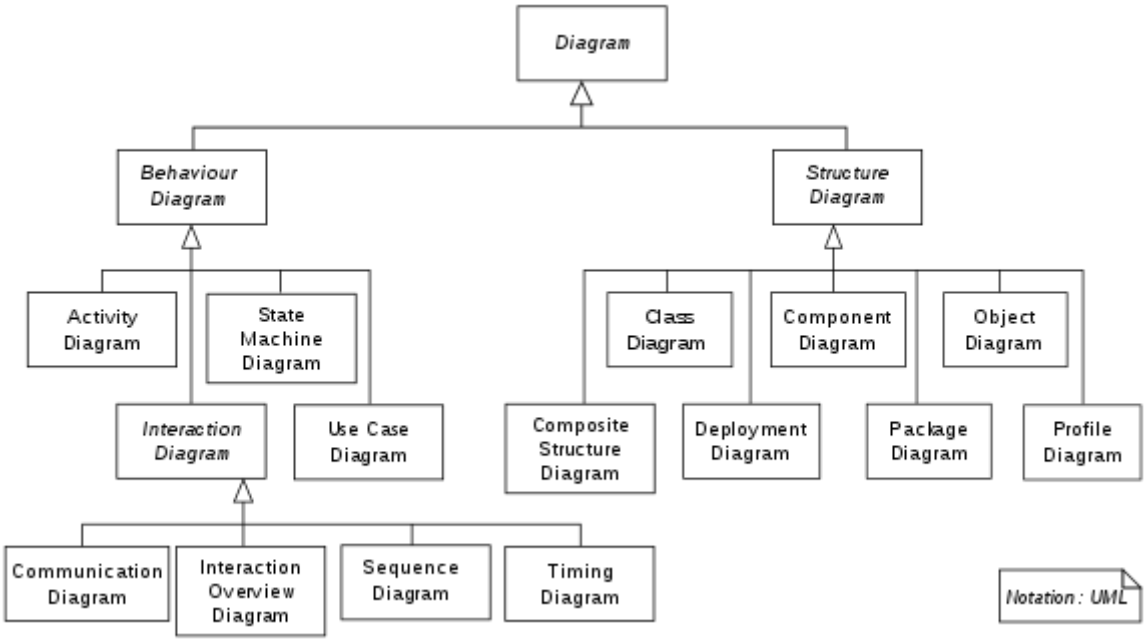
UML uključuje skup metoda kojima se grafički prikazuju objektno-orijentirani računalni sustavi. Računalni sustav se modeliraju raznovrsnim UML dijagramima, od kojih se svaki koristi za prikaz sustava iz nešto drugačije perspektive [6].

UML tvore četrnaest dijagrama koje, s obzirom na perspektivu gledanja, prema [6], UML dijagrami su podijeljeni u dvije kategorije:

- Statički dijagrami ne uzimaju u obzir vremensku komponentu sustava, već daju prikaz sustava kakav postoji u nekom trenutku. Statička struktura sustava se opisuje pomoću objekata, atributa, operacija i odnosa. Primjeri statičkih dijagrama su dijagram klase (eng. *class diagram*) i dijagram objekta (eng. *object diagram*).
- Dinamički dijagrami u opisu sustava uključuju i vremensku komponentu međudjelovanja elemenata kako bi se modelirali sljedovi događaja unutar sustava. Primjeri dinamičkih dijagrama su dijagram aktivnosti (eng. *activity diagram*) i dijagram stanja (eng. *state machine diagram*, ili *structure diagram*).

Modernija podjela dijagrama dijeli UML-dijagrame s obzirom na to da li se njima modelira struktura sustava (eng. *structure diagram*) ili ponašanje sustava (eng. *behavior diagram*) [7]. Takva podjela se otprilike podudara s podjelom na statičke (strukturni) i dinamičke (ponašajni) dijagrame, s iznimkom dijagrama slučaja uporabe (eng. *use case diagram*) koji, iako modelira ponašanje, ne modelira vremenske značajke sustava [6].

Neki dijagrami predstavljaju strukturne informacije, a ostali predstavljaju opće tipove ponašanja, uključujući nekoliko koji predstavljaju različite aspekte interakcija unutar sustava. Podjela UML-dijagrama prema normi UML 2.5 prikazana je na slici 1.



Slika 1. Pregled UML-dijagrama, norma UML 2.5, [7]

U nastavku rada, kroz poglavlja i primjene dijagrama detaljnije će biti opisani dijagrami slučaja uporabe, klasa, međudjelovanja i suradnje.

3. Analiza sudionika sustava za informiranje i njihovih uloga

Cilj je razviti jednostavnu *web* aplikaciju za informiranje koja raspoznaje različite uloge korisnika i sukladno dodijeljenoj ulozi, prikazuje korisniku sadržaj. Sadržaj aplikacije je dostupan samo registriranim korisnicima. Registriranom korisniku može biti dodijeljena uloga Korisnika sadržaja ili Administratora sadržaja. Administratoru je omogućen pristup svim funkcionalnostima aplikacije koje ima korisnik, ali mu je omogućen i pristup dodatnim funkcionalnostima preko kojih upravlja sadržajem aplikacije i korisničkim računima. Sve funkcionalnosti aplikacije bit će detaljnije opisane u nastavku rada pomoću UML dijagrama.

Nakon prijave u aplikaciju, korisniku je prikazana naslovna stranica (eng. *Home page*). Na naslovnoj stranici se nalazi oglasna ploča u obliku *carousel slidera* s ikonama objavljenih članaka. Odnosno, pokretna traka fotografija s naslovima, koja ujedno služi kao poveznica za stranicu gdje se nalazi cijeli članak, objava. Korisniku sadržaja dozvoljeno je samo čitanje objava, a administratoru sadržaja omogućeno je uređivanje objava.

Primjenom dijagrama slučaja uporabe napravljena je analiza sudionika sustava za informiranje i njihovih uloga.

3.1. Use Case dijagram

Dijagrami slučaja uporabe prikazuje ponašanje sustava, podsustava ili klase kako s perspektive gledišta korisnika. On dijeli funkcionalnost sustava na transakcije koje su od značaja korisnicima sustava. Dijelovi interaktivne funkcionalnosti nazivaju se slučajevima upotrebe. Slučaj uporabe opisuje interakciju sustava s korisnicima kao slijed poruka između sustava te jednog ili više aktera. Pojam akter uključuje ljude, kao i druge računalne sustave i procese [5].

Osnovna namjena dijagrama slučajeva uporabe je definiranje funkcija sustava. Također, dijagrami slučaja uporabe služe tome da se razvojni tim i korisnici usuglase po pitanju ponašanja korisnika pri komunikaciji sa sustavom [8]. Dijagrami slučaja uporabe su

statički UML-dijagrami, a također pripadaju i skupini dijagrama koji opisuju ponašanje sustava budući da modeliraju moguće ponašanje korisnika sustava [6].

Sudionici ili akteri su idealizacija korisnika sustava, procesa ili stvari u interakciji sa sustavom, podsustavom ili klasom. Sudionik karakterizira interakcije koje korisnici mogu imati sa sustavom. Tijekom izvođenja, jedan korisnik može biti vezan za više sudionika unutar sustava. Različiti korisnici mogu biti vezani za istog aktera i stoga predstavljaju više primjeraka iste definicije aktera. Svaki akter sudjeluje u jednom ili više slučajeva upotrebe. U interakciji je sa slučajem uporabe (i stoga sa sustavom ili klasom koja posjeduje slučaj upotrebe) razmjenom poruka. Implementacija sudionika u dijagramu je dovoljna okarakterizirati nizom atributa koji definiraju stanje sudionika [8].

Slučaj uporabe je koherentna jedinica vanjski vidljive funkcionalnosti sustava koju izvodi jedna ili više komponenta sustava. Svrha slučaja uporabe je definirati dio koherentnog ponašanja bez otkrivanja unutarnje strukture sustava. Definicija slučaja uporabe uključuje sva ponašanja sustava. Ono uključuje nizove glavnih funkcionalnosti sustava, različitih varijacija normalnog ponašanja i svih iznimnih okolnosti koje se mogu pojaviti kao njihov rezultat. Sa stajališta korisnika to su neuobičajene pojave ali s gledišta sustava, to su dodatne varijacije koje se moraju opisati i odrediti način postupanja s njima [5].

U modelu, izvršavanje svakog slučaja uporabe neovisno je o ostalim implementacijama slučajeva upotrebe koji mogu stvoriti implicitne ovisnosti između njih zbog zajedničkih objekata. Svaki slučaj uporabe predstavlja zasebnu funkcionalnost čije izvršavanje može utjecati na izvršavanje drugih slučajeva uporabe [5].

U grafičkom prikazu, na lijevoj strani dijagrama se nalaze sudionici ili akteri koji izravno utječu na izvođenje slučaja uporabe. Slučajevi uporabe se nalaze u sredini prikaza, a na desnoj strani se nalaze ostali sudionici. Linijama se prikazuju veze između sudionika i slučaja uporabe [9]. Slučajevi uporabe su prikazani kao elipse unutar pravokutnika koji predstavlja granice sustav. Opis funkcionalnosti sustava može biti prikazan na više načina. Uglavnom se opisuje neformalnim tekstom, formalnim tekstom s opisom tijeka događanja i pseudo kodom.

Opis unutar slučaja uporabe mora egzaktno definirati odnose između slučaja uporabe i korisnika. U tekstu se ne opisuje na kako se ostvaruju pojedine funkcije sustava. On mora biti napisan na takav način da je lako razumljiv krajnjem korisniku i autoru modela [10].

Tekstom se mogu opisati razni tijekovi rada sustava, a prema [9] definirane su tri vrste:

- Osnovni tijek događaja opisuje stanja kroz koje prolaze sudionik i sustav u normalnim okolnostima. Od početka do kraja se izvodi kako je opisan, predstavlja poželjan slijed događaja za koji se pretpostavlja da korisnici i sustav neće pogriješiti.
- Zamjenski tijek opisuje dodatna ponašanja sustava. Proširenje osnovnog tijeka.
- Iznimni tijek događaja se izvodi prilikom pojave greške ili neispunjavanja zadanih uvjeta. Slučaj uporabe uvijek sadrži barem jedan iznimni tijek događaja. Alternativa osnovnom tijeku događaja.

Veze između elementa modela predstavljaju relacije. Relacija opisuje odnose između elemenata sustava. U dijagramu slučaja uporabe, [5], definira četiri vrste veza:

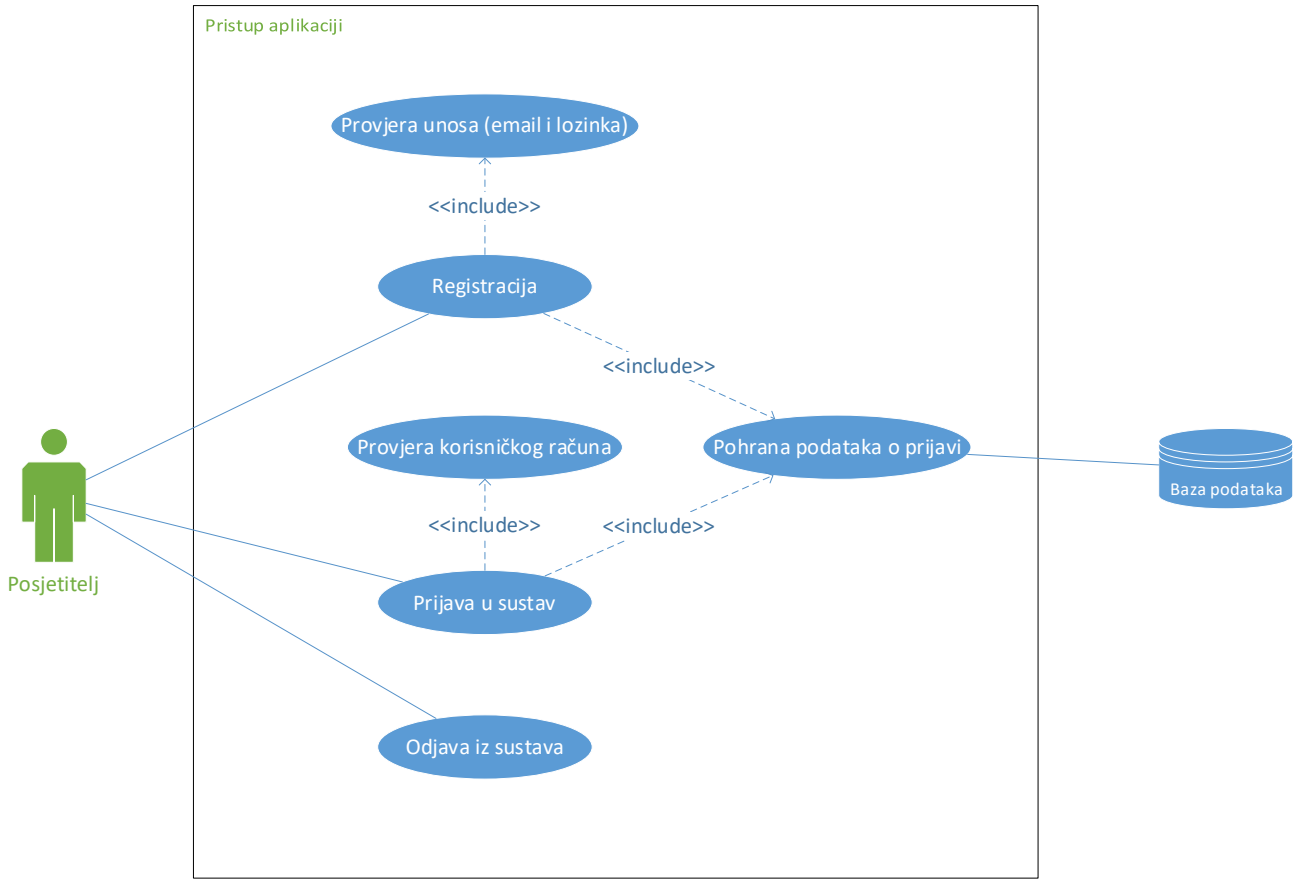
1. Asocijacija (eng. *Association*) - predstavlja komunikacijsku putanju između aktera i slučaja uporabe u kojem sudjeluje. Međusobno povezivanje aktera ili međusobno povezivanje obrazaca uporabe ovom vrstom veze nije dozvoljeno. U slučaju da se želi naglasiti koji akter inicira određeni obrazac uporabe (inicijator) to se može napraviti dodatkom strelice na vezu asocijacije.
2. Proširenje (eng*. Extend*) – relacijom proširenja se označava dodavanje dodatnih funkcionalnosti osnovnom slučaju uporabe. Prošireni slučaj uporabe predstavljaju izvršavanje dodatnih funkcionalnosti osnovnog slučaja uporabe ili funkcionalnosti koje se izvršavaju samo ako su zadovoljeni prethodno definirani uvjeti [11]. Osnovni slučaj uporabe mora moći funkcionirati samostalno bez uporabe proširenog slučaja.
3. Poopćavanje (eng. *Generalization*) – relacija poopćavanja ili relacija nasljeđivanja se koristi kada postoji podslučaj uporabe (dijete) koji koristi svojstava, operacije i odnose od višeg slučaja uporabe (roditelj) [12]. Jedan slučaj uporabe može imati više podslučaja. U podslučaju uporabe samo se definiraju svojstava, operacije i odnosi koji nisu naslijeđeni od višeg slučaja uporabe, [11]. Poopćavanje se može primijeniti i

između aktera sustava. Nasljeđivanje svojstva podrazumijeva i nasljeđivanje prava pristupa istim slučajevima uporabe kao i akter od kojeg su naslijeđena svojstva.

4. Obuhvaćanje (eng. *Include*) – relacija obuhvaćanja se primjenjuje kada osnovni slučaj uporabe izričito uključuje funkcionalnosti drugog slučaja uporabe. Obuhvaćeni slučaj uporabe ne postoji samostalno, već ovisi o jednom ili više osnovnih slučaja uporabe. Njime se izdvajaju zajedničke funkcionalnosti više slučaja uporabe [13].

3.2. Modeliranje sustava za informiranje Use Case dijagramom

U nastavku, modelirana su dva slučaja uporabe *web* aplikacije sustava za informiranje. Posjetitelji, odnosno neregistrirani korisnici ne mogu pristupiti sadržaju sustava za informiranje. Njima je omogućena registracija u sustav, gdje popunjavaju pristupni obrazac s korisničkim imenom, lozinkom i potvrdom lozinke. Kredencijali novog korisnika pohranjuju se u bazi podataka. Nakon registracije, posjetitelji postaju korisnici sadržaja i omogućen im je pristup naslovnoj stranici. Ako posjetitelj već posjeduje korisnički račun unutar aplikacije za informiranje, on u sučelje aplikacije unosi svoje korisničko ime i lozinku. Nakon provjere kredencijala u bazi podataka, korisnik je preusmjeren na naslovnu stranicu *web* aplikacije. Korisnicima je omogućena i odjava iz aplikacije. Opisani slučaj pristupa *web* aplikaciji, modeliran je *use case* dijagramom prikazanom na slici 2.



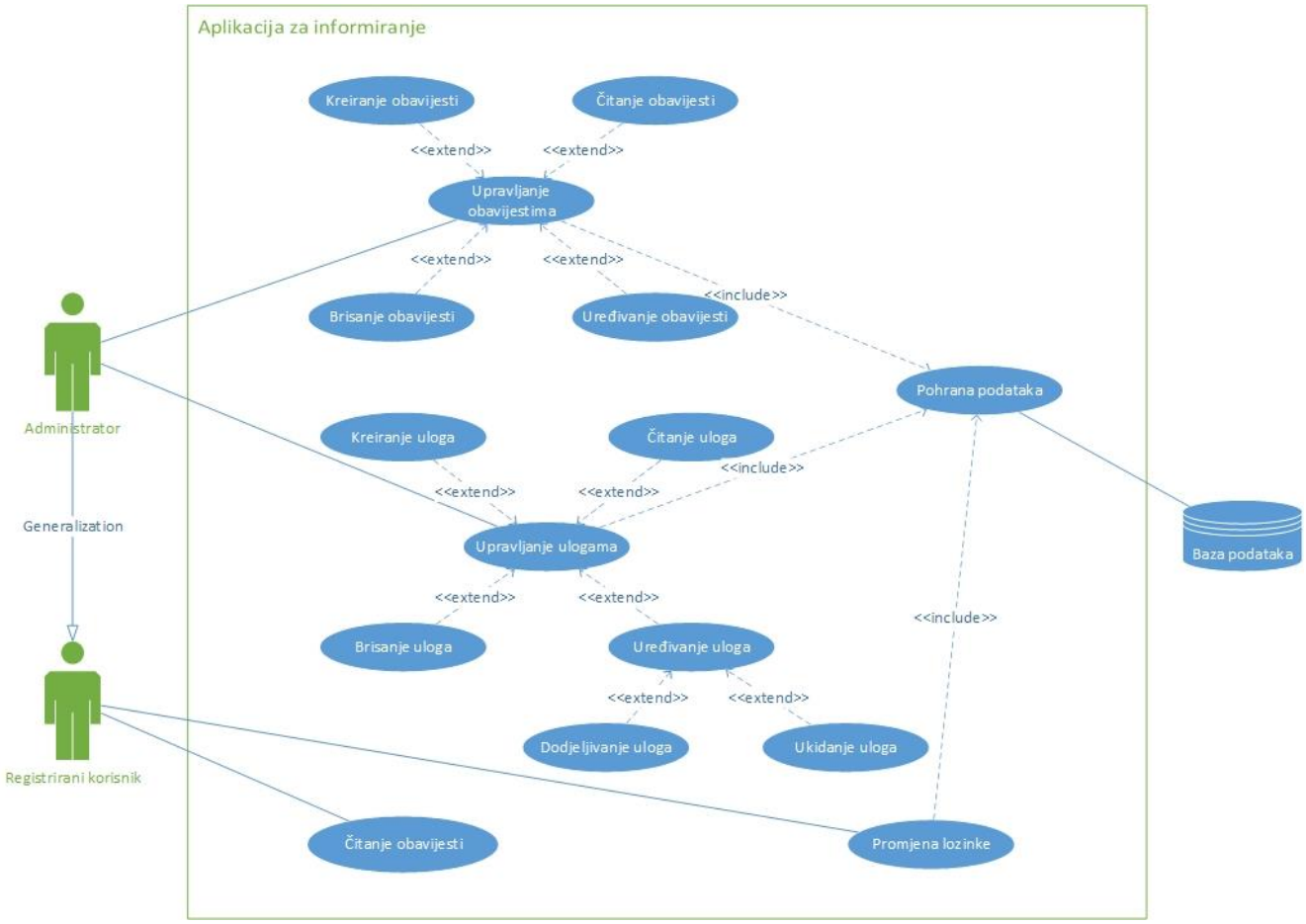
Slika 2. Use case dijagram pristupa aplikaciji

Prilikom prijave u sustav, provjerava se korisnički račun s ciljem provjere uloge dodijeljene registriranom korisniku. Registriranom korisniku može biti dodijeljena uloga Korisnika sadržaja ili Administratora sadržaja.

Kao korisnik sadržaja, korisnik sustava može upravljati vlastitim korisničkim računom, odnosno promijeniti lozinku svojeg korisničkog računa. Također, može pristupiti sadržaju sustava za informiranje koji se sastoji od oglasne ploče gdje su izlistane obavijesti i stranicama samih obavijesti gdje se nalaze dodatni sadržaj obavijesti.

Kao Administrator sadržaja, korisnik sustava može obavljati sve radnje kao i korisnik sadržaja. Zadaća Administratora sadržaja jest pravovremeno ažuriranje podataka koji se nalaze u sustavu za informiranje, stoga on uz čitanje obavijesti u *web* aplikaciji ima i pristup funkcionalnostima aplikacije gdje može upravljati obavijestima. Administrator sadržaja ima CRUD (*Create – Read – Update – Delete*) funkcije nad obavijestima. Odnosno Administrator sadržaja, u aplikaciji može kreirati obavijesti, čitati ih, ažurirati i brisati. Preko sučelja mu je

omogućeno upravljanje ulogama sustava. On ima pristup popisu uloga unutar sustava, mogućnost kreiranja uloga, uređivanja i njihovog brisanja. Administrator sadržaja može dodjeljivati ili ukidati uloge ostalim korisnicima sustava. Opisani slučaj, modeliran je *use case* dijagramom prikazanom na slici 3.



Slika 3. Use case dijagram sustava za informiranje

4. Definiranje elemenata (objekata) sustava i njihovih međuodnosa

U svrhu boljeg razumijevanja sustava i njegove izgradnje, UML dijagramima modelira se i statička struktura sustava. Modeliranje sustava statičkim dijagramima opisuje se sustav ne uzimajući u obzir vremensku komponentu sustava. Statičkim dijagramima opisuju se objekti sustava, što podrazumijeva definiranje stanja sustava, vrste i svojstva. Jedan od statičkih UML dijagrama je i dijagram klasa. Dijagramom klasa opisuje se vrsta, svojstva i međusobni odnosi između objekata sustava.

4.1. Dijagram klasa

Dijagram klasa (eng. *class diagram*) opisuje klase i njihove međusobne veze. Jednako tako, dijagram klasa opisuju vrste objekata unutar nekog sustava i njihove međusobne statične odnose. Dijagram klasa pripada strukturnoj skupini UML-dijagrama (eng. *structure diagram*) [14].

Klasa ili razred (eng. *class*) je osnovni tvorbeni element UML-dijagrama klasa. Zato su drugi nazivi za dijagrame klasa dijagram razreda, ili *class* dijagram. Za razumijevanje definicije klase važno je prvo odrediti značenje objekta. Objekt je entitet iz stvarnog svijeta ili neki koncept, odnosno apstrakcija nečega što ima dobro definirane granice i smisao u sustavu. Stoga je klasa opis grupe objekata sa sličnim svojstvima, a svaki objekt zasebno je obvezno instanca jedne klase [6].

Za svaku klasu definiran je naziv, a moguće je i odrediti popis atributa i operacija. Nije nužno definirati atribute i operacije ali bez njih razred nema implementacijsku svrhu. Za atribute se određuje naziv i tip podataka, a za operacije njihovu definiciju koja uključuje naziv operacije, te sve ulazne i izlazne parametre [6].

U dijagramima klasa, klasa je prikazana pravokutnikom koji se može podijeliti na tri dijela. U prvom odjeljku je upisan naziv klase. Naziv klase mora biti jednosmislen naziv koji se od problemsko područje promatranja. Najčešće se koriste imenice za imenovanje klasa.

U drugi odjeljak se upisuju atributi klase. Atributi klase opisuju svojstva objekta. Kod modeliranja uključuju se samo atributi koji su vezani uz problemsko područje i oni trebaju davati dovoljno informacija za opis i definiranje pojedine instance klase. Svaki atribut ima svoju vrstu i vidljivost. Vrsta ili tip atributa je zapisan nakon dvotočke i predstavljaju vrstu podataka. Npr. *integer*, *Boolean*, *string* i *date*. Vidljivost atributa pokazuje je li atribut vidljiv od strane neke druge klase. Za označavanje vidljivosti objekta koriste se simboli + (javna vidljivost), - (privatna vidljivost) i # (zaštićena vidljivost). Treći odjeljak je namijenjen za operacije klasa. Operacije manipuliraju atributima, odnosno vrijednostima atributa instance klase. Operacije se nazivaju i funkcijama. Funkcija može pripadati samo toj klasi u kojoj je definirana i može se primijeniti samo nad objektima iste klase. Iza dvotočke su upisani tip metode koji se vraća i potrebni parametri za izvršavanje pojedine metode, [15].

U dijagramu klasa, odnosi između klasa prikazani su relacijama (vezama). Definirane klase same po sebi nisu previše uporabljive, relacije između klasa čine osnovu modela, [14]. Vrste relacije u dijagramu klasa su:

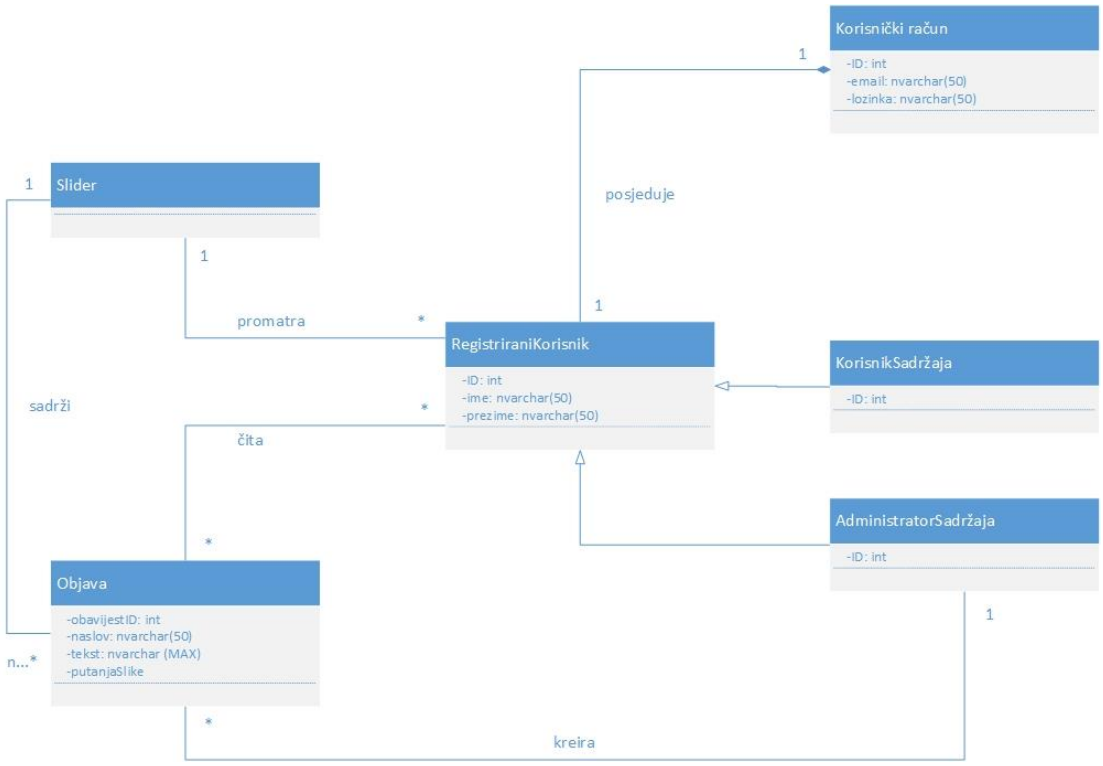
1. Relacija udruživanja (eng. *Association*) - opisuje statične odnose između instanca klasa. Asocijacija je uglavnom dvosmjerna, što znači da ako je jedan objekt na neki način povezan s drugim objektom, oba objekta su svjesni da drugi postoji. Asocijacija omogućava da jedna klasa pristupa atributima druge klase. Veze između klasa osim dvosmjernih, mogu biti jednosmjerne i refleksivne, [15].
2. Relacija sastavljanja (eng. *Aggregation*) – je poseban slučaj relacije udruživanja. Relacija se koristi kada jedna klasa u ovisi o drugoj, tj. predstavlja pod-skup druge klase. Postoje dvije vrste relacije sastavljanja, agregacija i kompozicija. Agregacija je vrsta sastavljanja gdje jedna klasa sadrži drugu. Kompozicija je vrlo slična agregaciji samo što je postojanje pod-klase u zavisnosti o postojanju nad-klase. Odnosno, kada se ukloni nad-klasa uklanja se i pod-klasa, [14].
3. Relacija poopćavanja (eng. *Generalization*) – omogućuje korištenje hijerarhijske klasifikacije. Opisuje veze između klasa ili skupova koji dijele nešto zajedničko. Poopćavanje omogućava nasljeđivanje svojstva od više klase, odnosno od klase roditelja, [15].

Ovisno o smjeru udruživanja, relacije se dijele se na jednosmjerne (unidirekcionalne) i dvosmjerne (bidirekcionalne). Jednosmjernim vezama je definiran smjer samo na jednom vrhu, dok je dvosmjernim relacijama 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].

4.2. Modeliranje sustava za informiranje dijagramom klasa

Klasa Oglasna ploča predstavlja *carousel slidera* s ikonama objavljenih članaka/obavijesti. U aplikaciji postoji jedna oglasna ploča koja sadrži više instanci Objava, odnosno slika i poveznica prema njima. Oglasna ploča u sebi sadrži nekoliko obavijesti. Klasa Objava, definirana je naslovom, tekstom i fotografijom. Odnosno, u kontekstu *web* aplikacije, putanjom koja ukazuje gdje je slika obavijesti pohranjena. Registriranim korisnicima, unutar aplikacije, omogućeno je čitanje obavijesti. Registrirani korisnik, definiran imenom i prezimenom, sastoji se od dva podtipa. To su klasa Korisnik sadržaja i Administrator. Klasa Korisnički račun, definirana korisničkim imenom i lozinkom, osnovni je identifikator korisnika. Svaki registrirani korisnik posjeduje jedan korisnički račun, odnosno kredencijale. Opisani odnosi modelirani su UML dijagramom klasa na slici 4.



Slika 4. Dijagram klasa sustava za informiranje

5. Razrada scenarija korištenja sustava za informiranje korisnika

Do sada se u radu, za opisivanje sustava za informiranje koji raspoznaje različite uloge korisnika, koristili statički UML dijagrami koji prilikom modeliranja ne uzimaju u obzir vremensku komponentu. Odnosno dijagrami prikazuju statičke odnose između vrsta aktera međusobno i između svih klasa međusobno.

U nastavku je opisan rad sustava za informiranje kroz tri uobičajena scenarija. Dijagramima međudjelovanja opisane su funkcije dostupne korisnicima sustava s različitim ulogama. Opisan je rad *web* aplikacije iz perspektive korisnika sadržaja i administratora.

5.1. Dijagram međudjelovanja

Dijagram međudjelovanja opisuje dinamičku strukturu sustava i način na koji objekti međusobno komuniciraju. Njime se prikazuje redoslijed interakcija između objekata, koji u međusobnom djelovanju odrađuju određene funkcije sustava, potaknute pokretanjem jednog slučaja uporabe [16].

Dijagram međudjelovanja je zajednički naziv koji se daje dijagramu vremena (eng. *Timing diagram*), dijagramu pregleda međudjelovanja (eng. *Interaction overview diagram*), komunikacijskom dijagramu (eng. *Communication diagram*) i sekvencijskom dijagramu (eng. *Sequence diagram*) [6]. U ovom radu će se koristiti sekvencijalni dijagram te će samo on biti pobliže definiran.

Sekvencijalni dijagram naglašava vremenski slijed poruka u sustavu. U dijagramu se prikazuje skup objekata i poruke koje ti objekti šalju i primaju [16]. Svaki sekvencijalni dijagram je dijagram međudjelovanja, ali svaki dijagram međudjelovanja ne mora biti sekvencijalni dijagram [16].

Svaki dijagram međudjelovanja koristi sljedeće simbole kojima opisuje sustav:

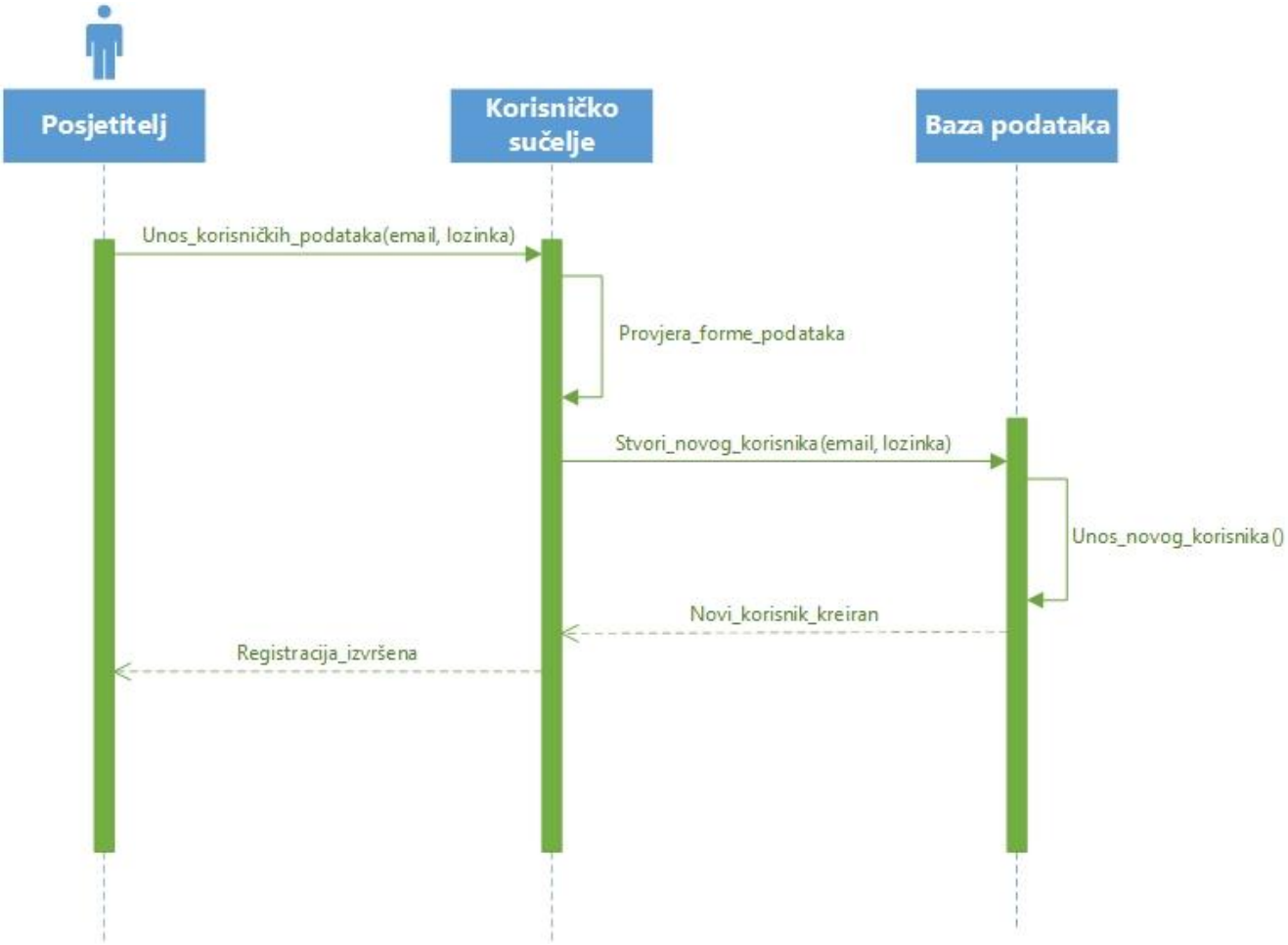
1. Simbol objekta – pravokutnik s nazivom objekta unutar pravokutnika koji se nalazi na vrhu dijagrama.
2. Crte života (eng. *Lifeline*) - vertikalna isprekidana linija koja predstavlja prisutnost objekta u sustavu tijekom određenog vremenskog razdoblja [16].
3. Težište nadzora (eng. *Focus of control*) ili okvir aktivnosti (eng. *Activation bar*) - okvir koji se nalazi na životnoj liniji i označava aktivnost objekta tijekom interakcije s drugim objektima sustava. Duljina okvira aktivnosti označava vrijeme aktivnosti objekta [10].
4. Poruke (eng. *Messages*) - prikazuju radnje/akcije koje se izvršavaju između objekta ili unutar samog objekta.

Poruke ili akcije u dijagramima međudjelovanja označavaju provođenje naredbi koje rezultiraju promjenom jednog ili više atributa objekta, povratnom vrijednosti objektu koji je uputio poruku i promjenom i povratom vrijednosti. Poruke se mogu slati od jednog objekta prema drugome ili poruka može imati isti izvor i odredište. Poruka objekta poslana je samom sebi. U UML dijagramima međudjelovanja postoji pet vrsta akcija odnosno poruka [16]:

1. Akcija poziva (eng. *Call action*) je poruka koja priziva metode odredišnog objekta. Objekt pošiljatelj šalje poruku pod pretpostavkom da je objekt primatelj spreman primiti poruku. Pošiljatelj prije pokretanja sljedeće metode čeka odgovor odredišnog objekta.
2. Akcija povrata (eng. *Return action*) predstavlja odgovor primatelja pošiljatelju na akciju poziva. Svaka akcija poziva ima odgovarajuću akciju povrata.
3. Akcija tvorbe (eng. *Create action*) se koristi kada se želi istaknuti da se određeni objekt, odnosno instanca klase kreira tek kada se izvrši akcija tvorbe.
4. Akcija dokidanja (eng. *Delete action*) je suprotna akcija od akcije tvorbe. Njome se uklanja određeni objekt kada više nije potreban. Akcija dokidanja može pozvati samu sebe.
5. Akcija odašiljanja (eng. *Send action*) je signalna poruka na koju pošiljatelj ne očekuje odgovor.

5.2. Razrada scenarija korištenja *web* aplikacije dijagramom međudjelovanja

Prvi promatrani scenarij je registracija posjetitelja *web* aplikacije sustava za informiranje. Slikom 5, prikazan je dijagram međudjelovanja za vremenski tok razmjena poruka između objekata tijekom registracije novog korisnika na aplikaciju sustava.

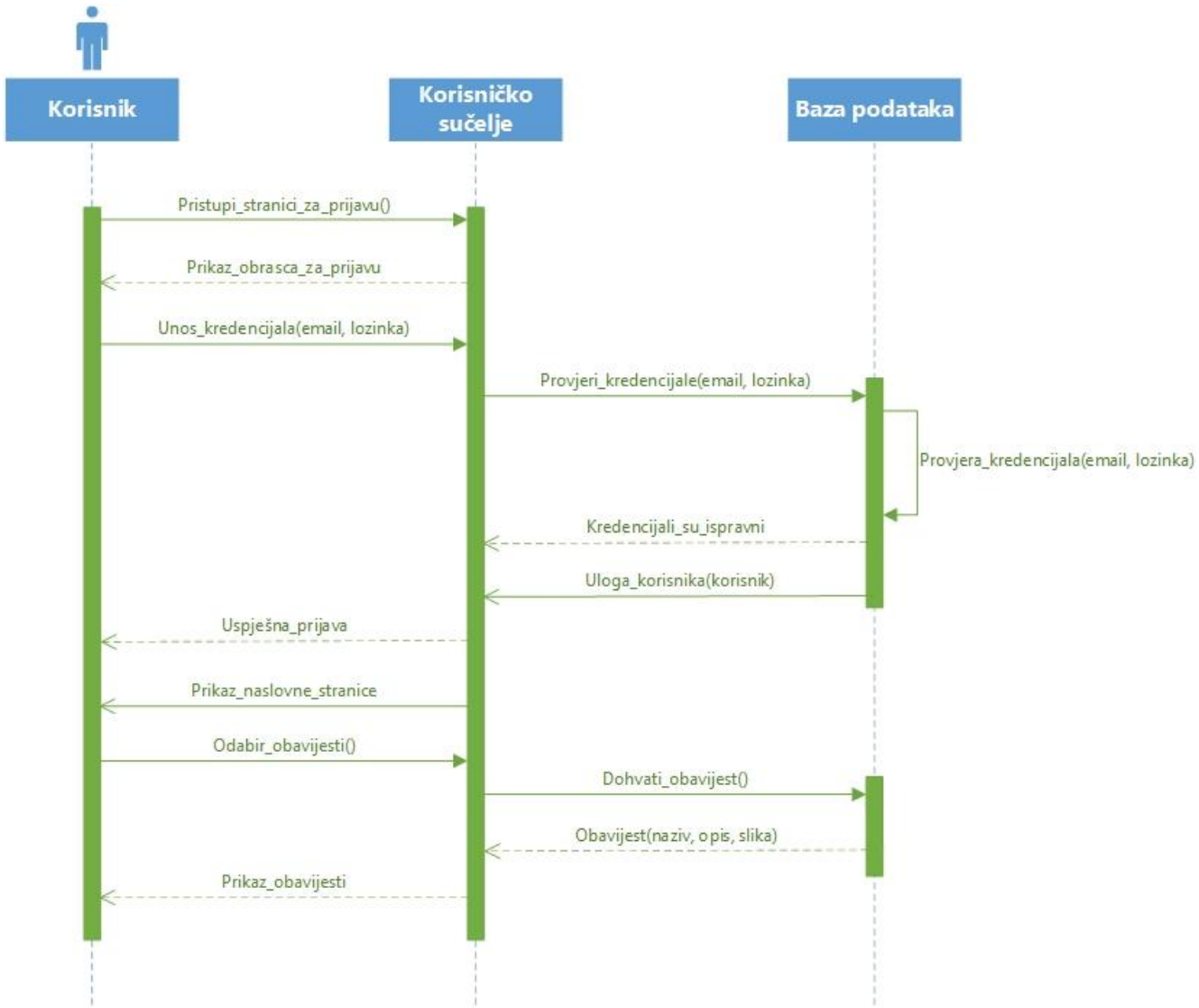


Slika 5. Dijagram međudjelovanja registracije korisnika u aplikaciju sustava za informiranje

Sadržaj sustava za informiranje dostupan je samo registriranim korisnicima. Registraciju posjetitelj aplikacije može napraviti unosom svoje email adrese i proizvoljne lozinke u obrazac za registraciju na korisničkom sučelju *web* aplikacije sustava za informiranje. U korisničkom sučelju, aplikacija provjerava da li uneseni podaci zadovoljavaju formu traženih veličina. Provjerava se da li unesena email adresa zadovoljava formu stvarne

adrese i da li lozinka zadovoljava zahtjeve kompleksnosti. Kada se ustvrdi da je forma unesenih podataka zadovoljena, uneseni podaci se prosljeđuju bazi podataka. U bazi podataka se kreira novi korisnički račun s proslijeđenim atributima. Nakon uspješnog kreiranja korisničkog računa, baza podataka šalje obavijest korisničkom sučelju da je kreiran novi korisnik. Korisničko sučelje obavještava posjetitelja da je njegov zahtjev za registracijom uspješno proveden.

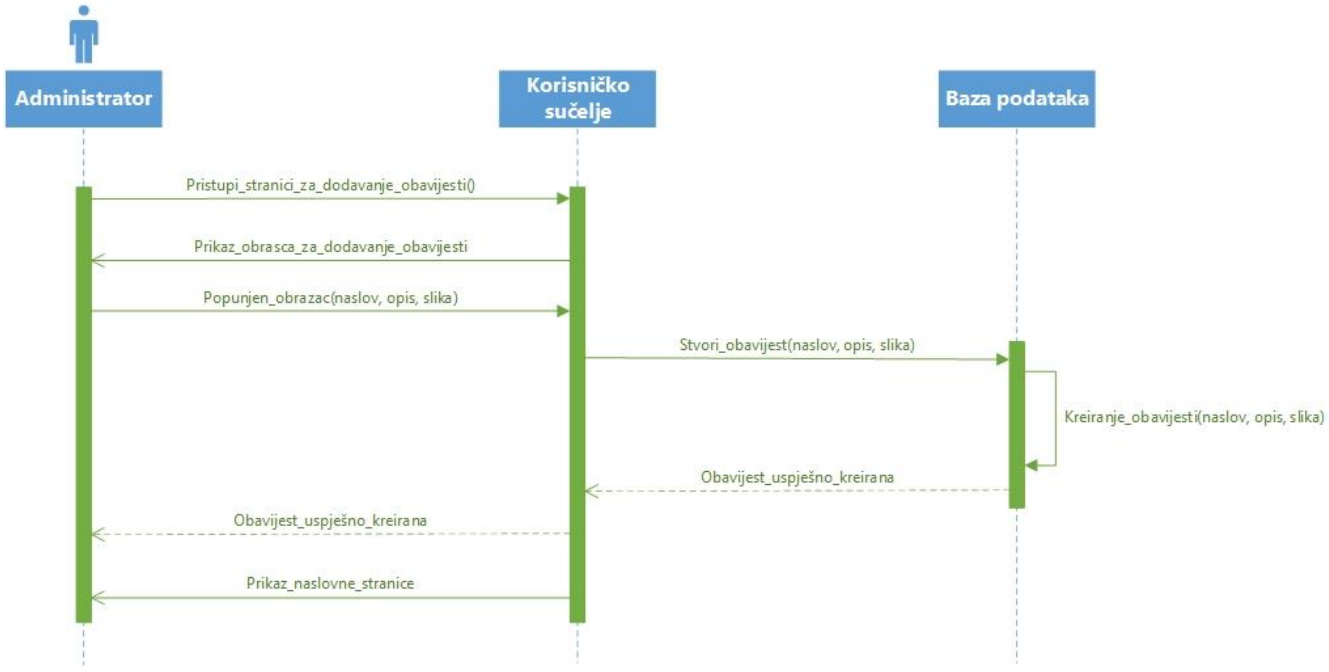
Registrirani korisnik za pristup sadržaju aplikacije ne treba svaki puta prolaziti proces registracije. Slikom 6, je prikazan dijagram međudjelovanja prijave registriranog korisnika na aplikaciju i čitanje objave unutar sustava.



Slika 6. Dijagram međudjelovanja prijave korisnika u aplikaciju sustava za informiranje

Za čitanje sadržaja, registrirani korisnik mora se prijaviti u sustav. S početne stranice korisničkog sučelja, korisnik odabire poveznicu prema formi za prijavu u sustav. Korisničko sučelje aplikacije prezentira korisniku formu prijave. Korisnik obrazac popunjava sa svojim kredencijalima, odnosno email adresom i lozinkom. Korisničko sučelje dobivene podatke prosljeđuje bazi podataka na provjeru. Nakon verifikacije dobivenih kredencijala, baza podataka šalje obavijest korisničkom sučelju da su kredencijali ispravni. Baza podataka, također, šalje korisničkom sučelju dodijeljenu ulogu korisnika koji je obavio prijavu. Korisničko sučelje obavještava korisnika o uspješnosti prijave u sustav i prikazuje mu naslovnu stranicu aplikacije. Na naslovnoj stranici, s oglasne ploče, korisnik odabire sliku objave o kojoj se želi dodatno informirati. Korisničko sučelje zaprima zahtjev korisnika i ovisno o poveznici koju je korisnik odabrao prosljeđuje zahtjev za podacima objave bazi podataka. Prema dobivenom zahtjevu, baza podataka šalje korisničkom sučelju naziv, opis i sliku tražene objave. Korisničko sučelje dobivene podatke uređuje i prikazuje ih korisniku.

Treći scenarij opisuje proces kreiranja objave u aplikaciji od strane administratora. Administratoru, koji je prijavljen u sustav, na naslovnoj stranici uz oglasnu ploča je ponuđen pristup funkcijama sustava koji nisu dostupni običnim korisnicima. Administrator ima mogućnost dodavanja objave i upravljanja korisničkim računima i ulogama. U ovoj situaciji, administrator se odlučuje kreirati novu objavu. Slikom 7, je prikazan dijagram međudjelovanja kreiranja objave na *web* aplikaciji.



Slika 7. Dijagram međudjelovanja kreiranja objave u aplikaciju sustava za informiranje

Na naslovnoj stranici, administrator odabire Dodaj objavu. Korisničko sučelje administratoru prezentira obrazac za kreiranje objave. Obrazac se sastoji od polja za unos Naziva, Opisa i forme preko koje odabire sliku koju želi *uploadati*. Dobivene vrijednosti, korisničko sučelje prosljeđuje bazi podataka. Baza podataka s dobivenim veličinama kreira novu instancu klase Objava. Baza podataka šalje poruku o uspješnom kreiranju objave korisničkom sučelju. Korisničko sučelje obavještava administratora o uspješnom kreiranju objave i vraća ga na naslovnu stranicu. Na naslovnoj stranici, oglasnoj ploči, dodana je novokreirana objava.

6. Kratak opis metoda i alata korištenih u razvoju sustava

Web aplikacija za informiranje je izrađena po MVC (eng. *Model-View-Controller*) arhitekturi. Koncept MVC-a jasniji je ako ga se modelira dijagramom suradnje.

6.1. Dijagram suradnje

Dijagram suradnje, poznat i kao komunikacijski dijagram, prikaz je odnosa i interakcija između objekata sustava. Dijagram se mogu koristi za prikaz dinamičkog ponašanja određenog slučaja uporabe i definiranje uloge svakog objekta [17].

Dijagrami suradnje nastaju tako što se prvo identificiraju strukturni elementi potrebni za obavljanje funkcionalnosti interakcije. Model se zatim gradi pomoću odnosa između tih elemenata.

Dijagram suradnje nalikuje dijagramu toka koji prikazuje uloge, funkcionalnost i ponašanje pojedinih objekata, ali i cjelokupni rad sustava u stvarnom vremenu. Postoje četiri glavne komponente dijagrama suradnje [17]:

1. Objekti - U dijagramu, objekti su prikazani kao pravokutnici s upisanim nazivom. Ako objekt ima svojstvo ili stanje koje posebno utječe na međudjelovanje s ostalim objektima, ispod naziva se unosi to svojstvo.
2. Korisnici ili akteri (eng. *Actors*) - Korisnici su pokretači svih interakcija u dijagramu.
3. Asocijacije ili linkovi (eng. *Links*) - Linkovi povezuju objekte s akterima i prikazani su punom linijom između dva elementa.
4. Poruke - U dijagramu suradnje, poruke između objekata prikazane su kao zapisi na linkovima sa strelicom koje označavaju njihov smjer. Ove su poruke komunikacija između objekata koje prenose informacije o aktivnosti i mogu uključivati redni broj.

Najvažniji objekti smješteni su u središte dijagrama, a svi ostali sudjelujući objekti se granaju. Nakon što se svi objekti postave, između njih treba dodati veze i poruke.

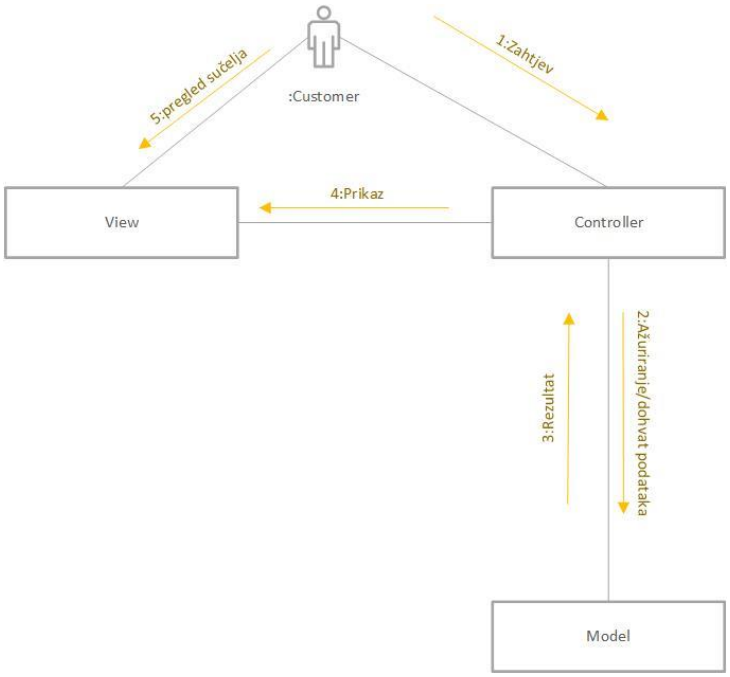
6.2. MVC

MVC je kratica za *Model-View-Controller*, te predstavlja softversku arhitekturu za razvoj *web* aplikacija. Arhitektura je prvi puta korištena 1970-ih, s ciljem pojednostavljenja ranih aplikacija s grafičkim korisničkim sučeljem (*Graphical User Interface* – GUI). Razvojem Interneta i pojavom *web* aplikacija, MVC postaje ponovno aktualna zbog svoje usklađenosti s osnovnim principima *web*a aplikacija. MVC razdvaja podatke, programsku logiku i korisničko sučelje, te tako olakšava testiranje i održavanje aplikacija [9].

MVC se sastoji od tri osnovna dijela:

- *Model* - Sadrži podatke koje program koristi i kojima korisnik upravlja. Središnja komponenta uzorka. To je dinamička struktura podataka aplikacije, neovisna o korisničkom sučelju, [18]. Izravno upravlja podacima, logikom i pravilima aplikacije. Dobiva input od *Controllera*.
- *View* - Generirani rezultat programa, odnosno *web* stranica koja sadrži podatke modela. Definira kako će grafičko sučelje aplikacije izgledati.
- *Controller* - Zaprima korisničke zahtjeve (eng. *requests*), izvršava operacije na modelu, odabire i šalje rezultat, odnosno *View* koji se generira prema korisniku [14].

Uz podjelu na ove komponente, dizajn MVC-a definira i interakcije između tih komponenti [16]. U razvoju sustava za informiranje korisnika, MVC arhitektura je primijenjena u ASP.Net frameworku. ASP.NET *framework* otvorenog koda (eng. *open source*) koji se koristi za izradu *web* aplikacija. Razvijen je od strane Microsofta kako za izradu dinamičkih *web* stranica, aplikacija i usluga. U dijagramu suradnje na slici 8, prikazana je implementacija MVC arhitekture u ASP.net-u.



Slika 8. Dijagram suradnje komponenti MVC-a

Zahtjevi korisnika dolaze do *Controllera* koji može upravljati podacima modela. Promjene stanja modela trajno se zapisuju u relacijskoj bazi podataka. *Controller* s podacima modela poziva odgovarajući *View*, koji se generira od strane takozvanog *view enginea* i šalje kao rezultat korisniku, odnosno korisnik vidi rezultat u *Viewu*.

6.3. Korištene tehnologije u izradi aplikacije

Za kreiranje pogleda *web* aplikacije za informiranje korištena je sintaksa Razor. Sama datoteka *viewa* ima ekstenziju .cshtml jer se koristi HTML-a (eng. *HyperText Markup Language*) u kombinaciji s C#-om. Programski jezik C# je objektno orijentirani jezik razvijen od strane Microsofta, namijenjen za izradu aplikacija. Za pohranu i dostupnost podataka *web* aplikacije, koristi se SQL (eng. *Structured Query Language*) relacijska baza podataka. SQL je programski jezik za izradu, traženje, ažuriranje i brisanje podataka iz relacijskih baza podataka. C# i SQL su razvijeni od strane Microsofta.

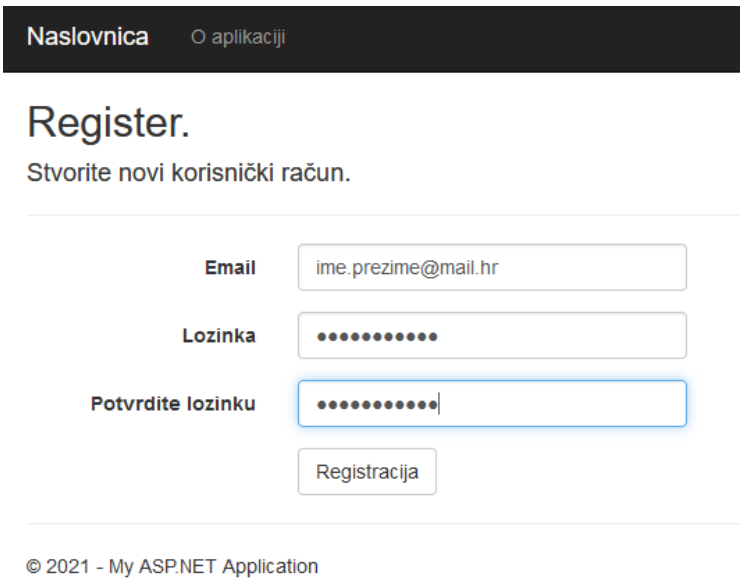
Web aplikacija je izrađena pomoću programskog alata Microsoft Visual Studio 2019, a za izradu baza podataka korišten je Microsoft SQL Server Management Studio 2014. Za izradu modela sustava za informiranje korišten je program Microsoft Visio.

7. Prikaz funkcionalnosti izvedenog sustava za informiranje korisnika

Implementacija sustava za informiranje koji raspoznaje različite uloge korisnika koji je opisivan u ovom radu implementiran je u obliku turističke *web* aplikacije. Sadržaj u aplikaciji dostupan je samo registriranim korisnicima koji posjeduju korisničke račune. Aplikacija raspoznaje dvije uloge koje mogu biti dodijeljene korisničkim računima, uloga korisnika sadržaja i uloga administratora. Ovisno o ulozi koja mu je dodijeljena, korisnik ima pristup određenim funkcionalnostima aplikacije.

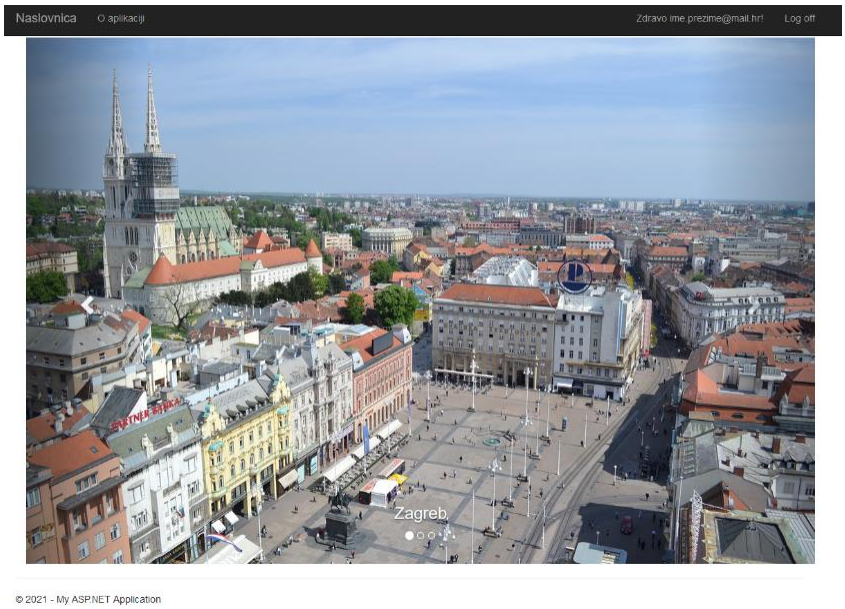
7.1. Registracija i pregled funkcionalnosti korisnika sadržaja

Svaki posjetitelj *web* aplikacije može obaviti registraciju preko sučelja *web* aplikacije. Novom korisniku bit će dodijeljena uloga korisnika sadržaja. Na slici 9, je prikazan obrazac aplikacije za registraciju novog korisnika. Posjetitelj u definirana polja unosi svoju email adresu i lozinku koja zadovoljava kriterije kompleksnosti aplikacije.



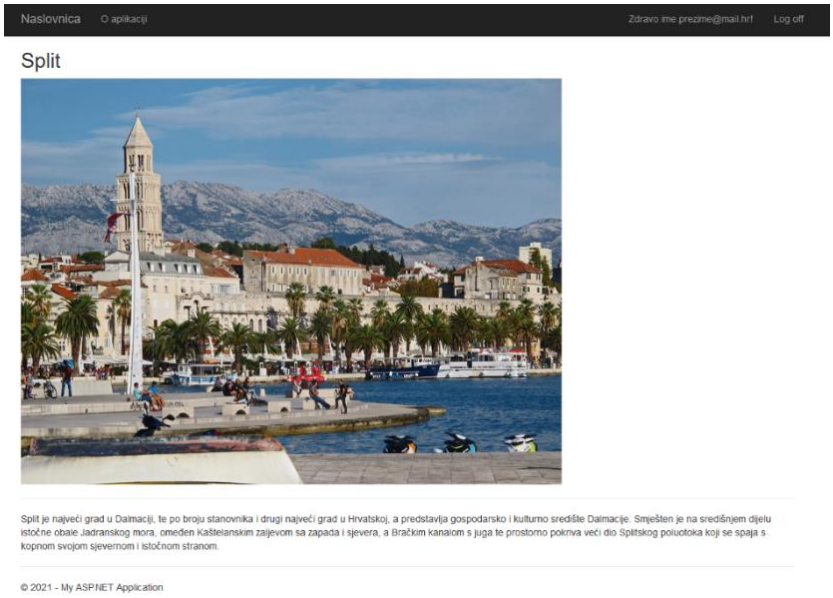
Slika 9. Registracijski obrazac aplikacije

Korisniku sadržaja su dostupne funkcionalnosti prijave i odjave iz aplikacije ali i promjene vlastite lozinke. Nakon uspješne registracije, korisnik se automatski prijavljuje u aplikaciju i biva proslijeđen na naslovnu stranicu. Slikom 10, je prikazana naslovna stranica aplikacije za račune s ulogom korisnik sadržaja.



Slika 10. Naslovna stranica za korisnike bez administratorskih ovlasti

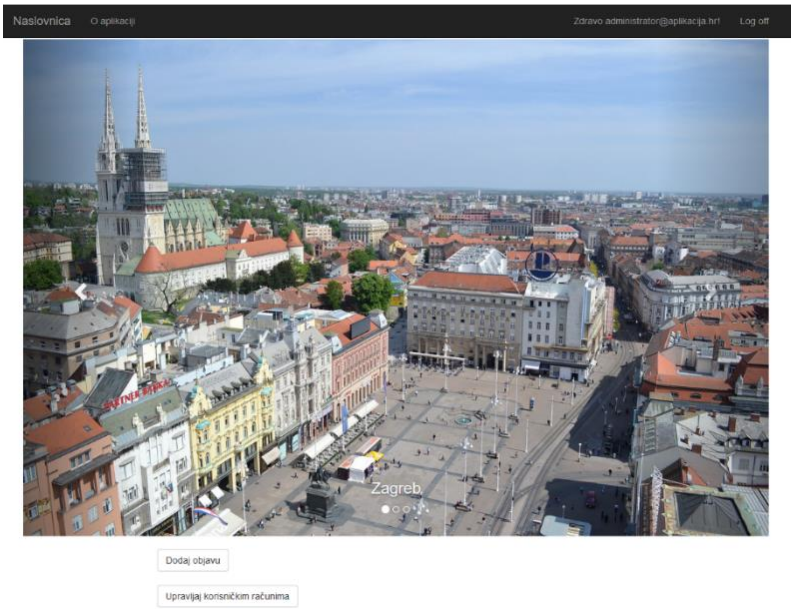
Kao glavna funkcija sustava za informiranje, korisniku je dostupna oglasna ploča sa svim objavama unutar aplikacije. Oglasna ploča je izvedena u obliku *carousel slidera* s ikonama objavljenih članaka. Odnosno, pokretna traka s fotografija gradova, koja ujedno služi kao poveznica za stranicu gdje se nalazi cijeli članak. Odabirom grada na korisničkom sučelju aplikacije, otvara se stranica sa sadržajem objave. Primjer objave koja sadrži naslov, fotografiju i kratak tekst je prikazana slikom 11.



Slika 11. Stranica obavijesti za korisnike bez administratorskih ovlasti

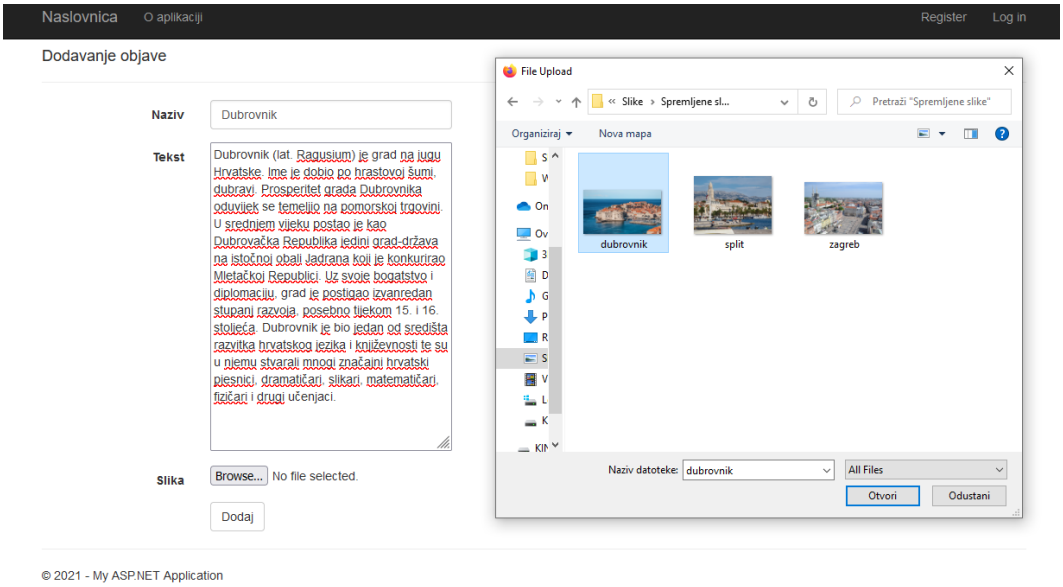
7.2. Kreiranje objave i upravljanje ulogama korisnika

Sljedeći scenariji su opisani sa stajališta administratora. U primjeru, administrator mora kreirati objavu i dodijeliti administratorska prava korisničkom računu kreiranom u prethodnom primjeru. Na slici 12, je prikazana naslovna stranica korisničkog sučelja *web* aplikacije sustava za informiranje kojoj pristupaju administratori nakon prijave u aplikaciju.



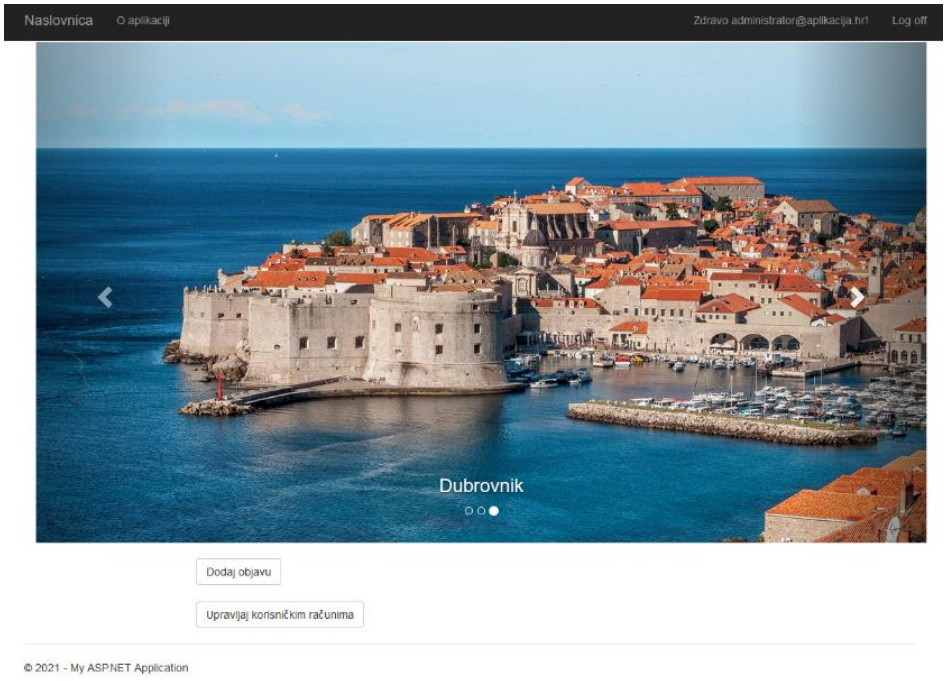
Slika 12. Naslovna stranica za korisnike s administratorskim ovlastima

Za razliku od naslovne stranice običnih korisnika, u ovom sučelju, administratoru je dostupan pristup poveznicama za funkcije dodavanja objave i upravljanja korisničkim računima. Za prvi cilj, administrator ima dodavanje nove stavke na oglasnu ploču i kreiranja nove objave (eng. *post*). Odabire Dodaj objavu i biva preusmjeren na stranicu koja sadrži obrazac za kreiranje objave prikazan na slici 13.



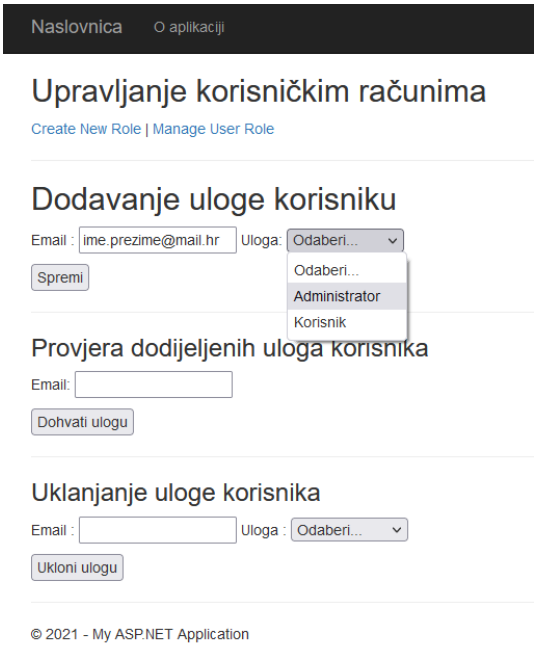
Slika 13. Kreiranje nove objave

Administrator popunjava polja nazivom i kratkim tekstom te *uploada* sliku koja je trenutno pohranjena lokalno na računalu. Administrator završava akciju, dobiva kratku obavijest o uspješnom dodavanju *posta* i biva preusmjeren natrag na naslovnu stranicu aplikacije. Na slici 14, prikazana je novokreirana objava na oglasnoj ploči.



Slika 14. Oglasna ploča s novokreiranom objavom

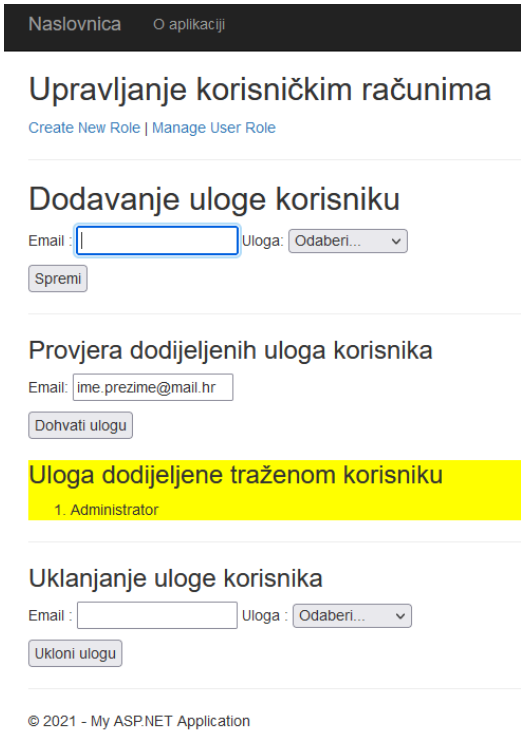
Za pregled cijele objave, administrator treba kliknuti na sliku u oglasnoj ploči. Sljedeći cilj administratora jest podizanje prava korisniku koji je kreiran u prethodnom primjeru. Administrator odabire Upravljaj korisničkim računima i biva preusmjeren na sučelje prikazano slikom 15.



Slika 15. Administratorsko sučelje za upravljanje ulogama i računima

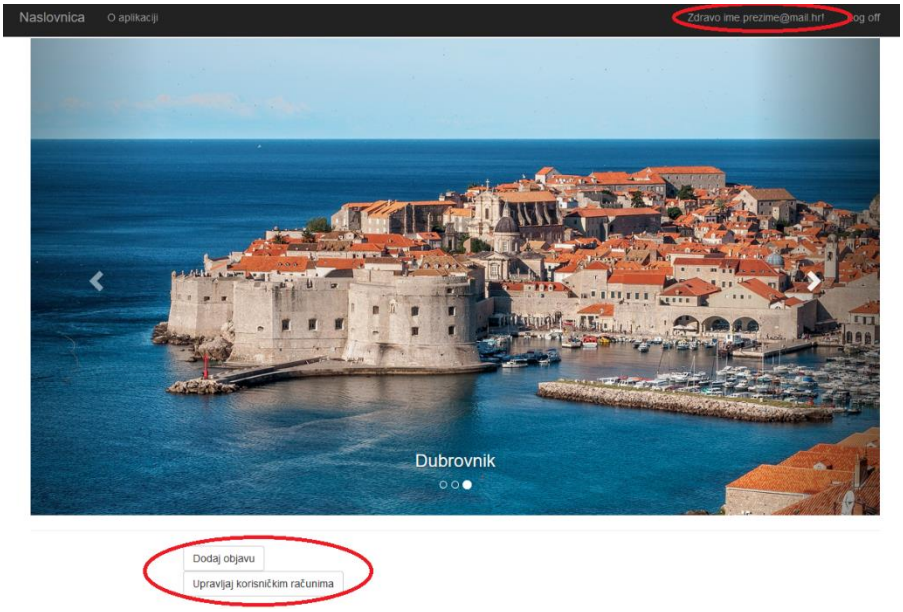
Na prethodno prikazanom sučelju, administrator u Email polje obrasca unosi email adresu „ime.prezime.mail.hr“ i iz padajućeg izbornika odabire koju ulogu želi dodijeliti tom korisničkom računu. Odabire mu dodijeliti administratorske ovlasti nad aplikacijom.

U ovom sučelju, administrator provjerava koje su uloge dodijeljene pojedinim korisničkim računima. U Email polje obrasca Provjera dodijeljenih uloga korisnika administrator upisuje adresu „ime.prezime.mail.hr“ i odabire Dohvati ulogu. Kao rezultat dobiva obavijest prikazanu slikom 16.



Slika 16. Provjera dodijeljenih uloga korisniku

Sa slike 16 je vidljivo da je korisničkom računu „ime.prezime.mail.hr“ dodijeljena uloga administratora. Konačna provjera uspješnosti promjene uloge korisničkog računa je *login* u aplikaciju s kredencijalima korisničkog računa „ime.prezime.mail.hr“.



Slika 17. Naslovna stranica novopromoviranog korisnika

Slika 17, prikazuje naslovnu stranicu nakon prijave u sustav novopromoviranog korisnika „ime.prezime.mail.hr“. Korisnik sada ima mogućnosti pristupa svim administratorskim funkcijama aplikacije. Za usporedbu, na slici 10 je prikazana naslovna stranica korisnika bez administratorskih ovlasti.

8. Zaključak

Razvoj suvremenih informacijskih sustava može biti vrlo kompliciran i zahtjevan proces koji uključuje veliki broj radnih skupina. Zbog lakšeg razumijevanja, sustav se secira na manje komponente koji se prikazuju pomoću modela.

Prilikom razvoja sustava za informiranje koji raspoznaje različite uloge korisnika korišteni su odgovarajući UML dijagrami za opis sustava iz različitih stajališta pogleda kako bi se dobio cjelokupni prikaz sustava, njegovih komponenti i procesa. Dobiveni modeli sustava uvelike su pomogli kod njegovog razvoja i implementacije.

Kao implementacija sustava za informiranje koji raspoznaje različite uloge korisnika, razvijena je *web* aplikacija. *Web* aplikacija je razvijena pomoću MVC obrasca razvoja, te su korišteni programski jezik C#, HTML, relacijska baza podataka SQL Server. Pomoću UML dijagrama prikazan je proces i ideja aplikacije za informiranje koji raspoznaje različite uloge korisnika.

Literatura

[1] Radošević, D.: Osnovne teorije sustava, Nakladni zavod Matice Hrvatske, Zagreb, 2001.

[2] Županović, I.: Tehnologija cestovnog prometa, Fakultet prometnih znanosti, Zagreb, 2002.

[3] Pavlić, M.: Razvoj informacijskih sustava, Znak, Zagreb, 1996.

[4] Stair, R.; Reynolds, G.: Fundamentals of Information Systems, Sixth Edit, Course Technology, Boston, 2012.

[5] Rumbaugh, J.; Jacobson, I.; Booch, G.: The Unified Modeling Language Reference Manual Second Edition, Addison Wesley Longman, Inc., Reading, Massachusetts, 1999.

[6] Jović, A.; Horvat, M.; Grudenić, I.: UML-dijagrami: Zbirka primjera i riješenih zadataka, Fakultet elektrotehnike i računarstva, Zagreb, 2013.

[7] "OMG Unified Modeling Language (OMG UML), Superstructure. Version 2.4.1". Object Management Group. Dostupno na: https://www.omg.org/spec/UML/2.4.1/Superstructure/PDF, [pristupljeno 30. 8. 2021.]

[8] [Rational 2001] Rational Software Corporation, Artifact: Use-Case Model. Dostupno na: http://www.ts.mah.se/RUP/RationalUnifiedProcess/process/artifact/ar_ucmod.html, [pristupljeno 30. 8. 2021.]

[9] Visual Paradigm :https://www.visual-paradigm.com/guide/uml-unified-modeling-language/how-to-model-mvc-with-uml-sequence-diagram/ , [pristupljeno 30. 8. 2021.]

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

[11] IBM. Use-case diagrams. Dostupno na: https://www.ibm.com/support/knowledgecente, [pristupljeno 30. 8. 2021.]

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

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

[14] Code Project:MVC (Model–View–Controller) Architectural Pattern. Dostupno na: https://www.codeproject.com/Articles/25057/Simple-Example-of-MVC-Model-View-Controller-Design [pristupljeno 30. 8. 2021.]

[15] Creately. Sequence Diagram Tutorial: Complete Guide with Examples. Dostupno na: https://creately.com/blog/diagrams/sequence-diagram-tutorial/ [pristupljeno 30. 8. 2021.]

[16] Meunier, R.; Rohnert, H.;Sommerlad, R.; Stal, M.: Pattern-Oriented Software Architecture, Willey, New Jersey, 1996.

[17] Techtarget. Collaboration diagram. Dostupno na: https://searchsoftwarequality.techtarget.com/definition/collaboration-diagram [pristupljeno 30. 8. 2021.]

[18] Burbeck, S.: Applications Programming in Smalltalk-80:How to use Model–View–Controller (MVC), Boston,1992.

[19] OMG About the Unified Modeling Language Specification Version 2.5 (omg.org). Dostupno na: https://www.omg.org/spec/UML/About-UML/ [pristupljeno 30. 8. 2021.]

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

Popis kratica

UML - Unified Modeling Language

MVC - Model–view–controller

SQL - Structured Query Language

HTML - HyperText Markup Language

OMT - Object-Modeling Technique

OOSE - Object-Oriented Software Engineering

OMG - Object Management Group

GUI - Graphical User Interface

CRUD - Create, Read, Update and Delete

Popis slika

Sveučilište u Zagrebu
Fakultet prometnih znanosti
10000 Zagreb
Vukelićeva 4

IZJAVA O AKADEMSKOJ ČESTITOSTI I SUGLASNOST

Izjavljujem i svojim potpisom potvrđujem kako je ovaj diplomski rad isključivo rezultat mog vlastitog rada koji se temelji na mojim istraživanjima i oslanja se na objavljenu literaturu što pokazuju korištene bilješke i bibliografija.

Izjavljujem kako nijedan dio rada nije napisan na nedozvoljen način, niti je prepisan iz necitiranog rada, te nijedan dio rada ne krši bilo čija autorska prava.

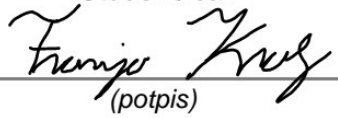
Izjavljujem također, kako nijedan dio rada nije iskorišten za bilo koji drugi rad u bilo kojoj drugoj visokoškolskoj, znanstvenoj ili obrazovnoj ustanovi.

Svojim potpisom potvrđujem i dajem suglasnost za javnu objavu diplomskog rada pod naslovom **ANALIZA, MODELIRANJE I IZVEDBA SUSTAVA ZA INFORMIRANJE KOJI RASPOZNAJE RAZLIČITE ULOGE KORISNIKA**

na internetskim stranicama i repozitoriju Fakulteta prometnih znanosti, Digitalnom akademskom repozitoriju (DAR) pri Nacionalnoj i sveučilišnoj knjižnici u Zagrebu.

U Zagrebu, 13.9.2021

Student/ica:



(potpis)