OBLIKOVANJE LOGIKE PROGRAMA I PROGRAMSKO RJEŠENJE ZA SUSTAV EVIDENCIJE ČLANOVA TERETANE

Škoro, Filip

Sveučilište Josipa Jurja Strossmayera u Osijeku
Ekonomski fakultet u Osijeku
Preddiplomski studij *Poslovna informatika*

Filip Škoro

OBLIKOVANJE LOGIKE PROGRAMA I PROGRAMSKO RJEŠENJE ZA SUSTAV EVIDENCIJE ČLANOVA TERETANE

Završni rad

Osijek, 2022.

Sveučilište Josipa Jurja Strossmayera u Osijeku
Ekonomski fakultet u Osijeku
Preddiplomski studij *Poslovna informatika*

Filip Škoro

Oblikovanje logike programa i programsko rješenje za sustav evidencije članova teretane

Završni rad

Kolegij: Oblikovanje i implementacija informacijskog sustava

JMBAG: 0010230431

e-mail: fskoro@efos.hr

Mentor: prof.dr.sc. Josip Mesarić

Komentor: dr.sc. Dario Šebalj

Osijek, 2022.

Josip Juraj Strossmayer University of Osijek

Faculty of Economics in Osijek

Undergraduate Study Business IT

Filip Škoro

DESIGNING THE LOGIC OF THE PROGRAM AND SOFTWARE SOLUTION FOR THE GYM MEMBERS' RECORDS SYSTEM

Final paper

Osijek, 2022.

IZJAVA

O AKADEMSKOJ ČESTITOSTI,

PRAVU PRIJENOSA INTELEKTUALNOG VLASNIŠTVA,

SUGLASNOSTI ZA OBJAVU U INSTITUCIJSKIM REPOZITORIJIMA

I ISTOVJETNOSTI DIGITALNE I TISKANE VERZIJE RADA

1. Kojom izjavljujem i svojim potpisom potvrđujem da je _____završni_____ (navesti vrstu rada: završni / diplomski / specijalistički / doktorski) rad isključivo rezultat osobnoga rada koji se temelji na mojim istraživanjima i oslanja se na objavljenu literaturu. Potvrđujem poštivanje nepovredivosti autorstva te točno citiranje radova drugih autora i referiranje na njih.
2. Kojom izjavljujem da je Ekonomski fakultet u Osijeku, bez naknade u vremenski i teritorijalno neograničenom opsegu, nositelj svih prava intelektualnoga vlasništva u odnosu na navedeni rad pod licencom Creative Commons Imenovanje –Nekomercijalno – Dijeli pod istim uvjetima 3.0 Hrvatska.
3. Kojom izjavljujem da sam suglasan/suglasna da se trajno pohrani i objavi moj rad u institucijskom digitalnom repozitoriju Ekonomskoga fakulteta u Osijeku, repozitoriju Sveučilišta Josipa Jurja Strossmayera u Osijeku te javno dostupnom repozitoriju Nacionalne i sveučilišne knjižnice u Zagrebu (u skladu s odredbama Zakona o znanstvenoj djelatnosti i visokom obrazovanju, NN br. 123/03, 198/03, 105/04, 174/04, 02/07, 46/07, 45/09, 63/11, 94/13, 139/13, 101/14, 60/15).
4. izjavljujem da sam autor/autorica predanog rada i da je sadržaj predane elektroničke datoteke u potpunosti istovjetan sa dovršenom tiskanom verzijom rada predanom u svrhu obrane istog.

Ime i prezime studenta/studentice: Filip Škoro

JMBAG: 0010230431

OIB: 86358238595

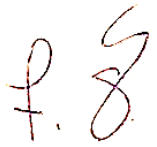
e-mail za kontakt: f.skoro06@gmail.com

Naziv studija: Poslovna informatika

Naslov rada: Oblikovanje logike programa i programsko rješenje za sustav evidencije članova teretane

Mentor/mentorica rada: Josip Mesarić

U Osijeku, _____2022._____ godine



Potpis _____ **_____**

Oblikovanje logike programa i programsko rješenje za sustav evidencije članova teretane

SAŽETAK

U radu je izložen proces izrade aplikacije za evidenciju članova teretane, osoblja, opreme i vanjskih suradnika, poput trenera. Proces izrade se sastoji od 3 faze. U prvoj fazi su izrađeni UML dijagrami pomoću alata Astah koji prikazuju logiku programskog rješenja. U drugoj fazi oblikovano je sučelje aplikacije kojom će upravljati administrator, a skice su oblikovane pomoću alata Balsamiq Wireframes. U trećoj fazi izrađeno je programsko rješenje korištenjem objektno orijentiranog jezika C#. U prvoj fazi logika programa ilustrirana je pomoću 5 vrsta dijagrama: dijagram slučajeva korištenja, dijagram klasa, dijagram objekata, sekvencijski dijagram i dijagram aktivnosti. Razvojno okruženje (IDE) koje je korišteno pri izradi programskog rješenja je Microsoft Visual Studio 2022 te će rješenje biti prikazano kao Windows Forms aplikacija koja se izvodi na desktopu. Svrha rada je unaprijediti poslovanje teretane evidencijom članstva i članarine i angažmana osoblja i vanjskih suradnika.

Ključne riječi: UML **d**ijagrami, Skica, C#, Windows forms

Design of program logic and software solution for the gym members' records system

ABSTRACT

The paper presents the process of creating an application for the records of gym members, staff, equipment and external associates, such as trainers. The development process consists of 3 stages. In the first, UML diagrams were created using the Astah tool that show the logic of the program solution. In the second phase, the interface of the application, which will be managed by the administrator, was designed, and the sketches were designed using the Balsamiq Wireframes tool. In the third phase, a programming solution was created using the object-oriented language C#. In the first phase, the logic of the program is illustrated using 5 types of diagrams: use case diagram, class diagram, object diagram, sequence diagram and activity diagram. The development environment (IDE) used to create the software solution is Microsoft Visual Studio 2022, and the solution is presented as a Windows Forms application running on the desktop. The purpose of the work is to improve the business of the gym by recording membership and membership fees and the engagement of staff and external associates.

Keywords: UML diagrams, Sketch, C #, Windows forms

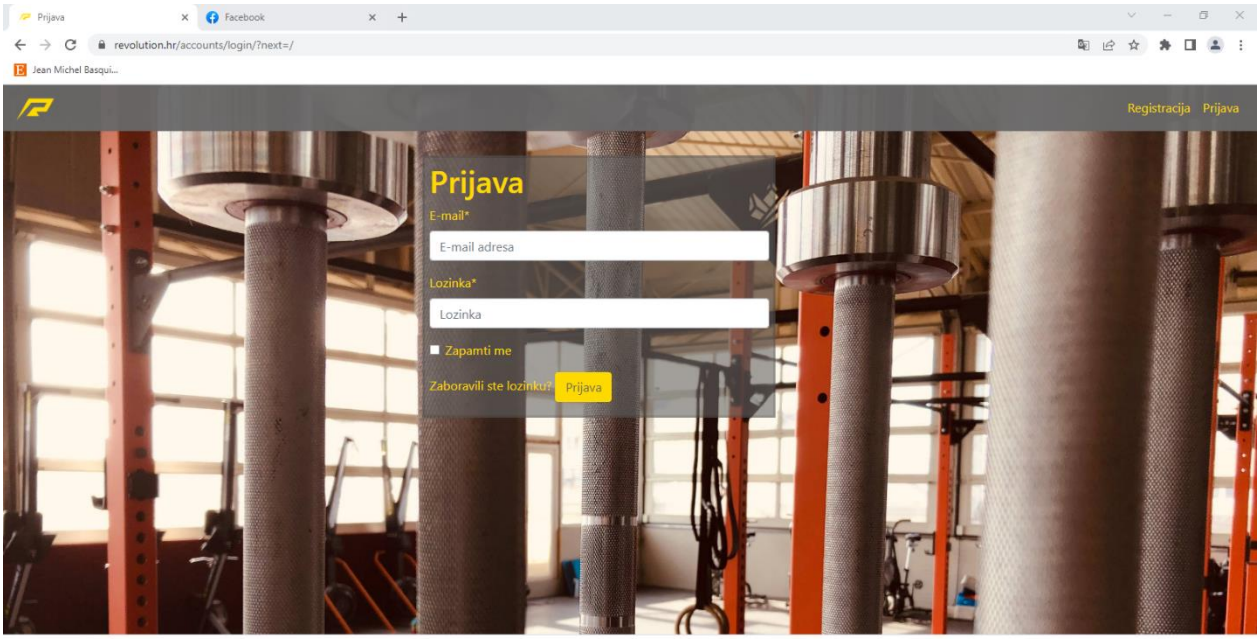
Sadržaj

1. Uvod

Danas je bilo kojem poslovnom subjektu gotovo nemoguće poslovati bez digitalnih alata, poslovnih informacijskih sustava i softvera pa tako i teretanama koje zadnjih godina broje sve više korisnika. Iz tog razloga javlja se potreba za sustavima pomoću kojih će djelatnici teretane lakše voditi evidenciju o članovima, osoblju i opremi. Zahtjevi koji se postavljaju pred aplikaciju po pitanju entiteta i njihovih atributa su stvaranje tablica i primarnih ključeva za svaki entitet te također stvaranje vanjskih ključeva ukoliko će se neki entiteti vezati s drugima. Npr. svaki član bi se mogao vezati uz određenog trenera ukoliko se isti odluči za personalizirane treninge. Svrha rada je olakšati proces evidencije članova njihovih obveza, te evidenciju i korištenje opreme teretane i vođenje evidencije o radu trenera i osoblja s čime se povećava ukupna efikasnost poslovanja. Cilj rada je kreirati grafička rješenja koja opisuju sustav s više različitih aspekata poput unosa, brisanja, pretrage i uređivanja članova, osoblja, opreme i trenera te naposljetku izraditi cjelovitu i funkcionalnu aplikaciju. U radu će detaljno biti opisan proces izrade aplikacije za evidenciju članova teretane koja će administratoru, osim unosa, brisanja i uređivanja članova, također omogućavati iste akcije za osoblje, opremu i vanjske suradnike. Prije izrade programskog rješenja u C# programskom jeziku, sama logika aplikacije će biti prikazana putem UML dijagrama, a skice aplikacije će biti izrađene pomoću alata Balsamiq Wireframes. Aplikacija je namijenjena isključivo za interno korištenje odnosno zaposlenik teretane bi bio administrator te posjedovao svoje korisničko ime i lozinku potrebnih za prijavu u aplikaciju. Svrha rada leži u danas sveprisutnoj digitalizaciji poslovanja i sve većoj popularizaciji teretana s čime se javlja potreba za sustavima koji omogućavaju lako evidentiranje članova i plaćenih članarina.

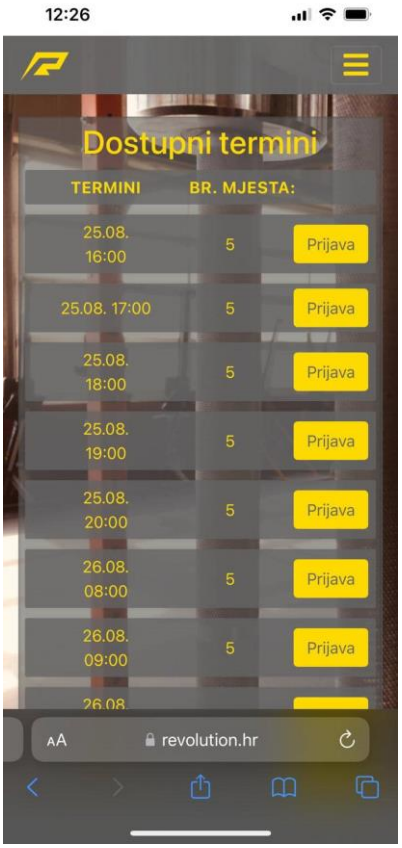
2. Prethodna istraživanja

Zadnjih godina zdrav način života i fizička aktivnost dobivaju na popularnosti jer se ljudi žele suprotstaviti kako užurbanom načinu života, tako i sedentarnom radnom okruženju. Sve to utječe na sve veći broj korisnika teretane te se javlja potreba za sustavima koji će osoblju omogućavati evidentiranje istih. Primjer korištenja sličnih sustava može se pronaći u osječkoj teretani City fitness čiji sustav za evidenciju na dnevnoj bazi kontrolira dolaske članova, u koje vrijeme su došli, a u koje vrijeme su otišli iz fitnessa, provlačenjem kartice kroz čitač. Isto tako sustav evidentira koliko dana je preostalo do isteka članarine, te koju članarinu posjeduju od ponuđenih. Zagrebačka teretana "Revolution" koristi aplikaciju za organizaciju grupnih treninga putem online poveznice revolution.hr. Aplikacija radi na principu da korisnici na spomenutoj web lokaciji unesu email i lozinku te se prijavljuju u željeni termin. Pristupne podatke dobiju na mail adresu. Na slici 1 je prikazano sučelje za prijavu u sustav koje korisniku omogućuje prijavu ili registraciju.



Slika 1. Revolution - sučelje za prijavu, obrada autora

Nakon prijave u sustav, korisnicima su prikazani termini u koje se mogu prijaviti. Sučelje za prijavu u željeni termin je prikazano na slici 2.



Slika 2. Revolution – sučelje za prijavu termina, obrada autora

Baloo Solutions nudi nešto složenije programsko rješenje. Korisnik kroz brojna sučelja može voditi evidenciju o dodanim korisnicima, uređivati njihove podatke te ih raspodijeliti u grupe, pregledati plaćene članarine, pregledati liječničke potvrde, evidentirati prisutnost članova itd. Poduzeće svojim klijentima nudi nekoliko različitih paketa koji se razlikuju po cijeni i dostupnim opcijama. Detaljnije informacije se mogu pronaći na njihovom web sjedištu - https://www.baloo-solutions.com/. U radu će se ponuditi programsko rješenje u obliku Windows Forms aplikacije koja će imati proširene funkcionalnosti u vidu evidencije opreme, osoblja i vanjskih suradnika.

3. Metodologija rada

Aplikacija će biti izrađena u C# objekto orijentiranom programskom jeziku, a korišteno programsko okruženje je Microsoft Visual Studio 2022 koji omogućuje laku izradu Windows Forms aplikacija koje se izvode na desktopu. C# je jezik opće namjene te se koristi u izradi mobilnih i web aplikacija, softverskih rješenja te računalnih igara. Prije izrade programskog rješenja, isto će biti skicirano u aplikaciji Balsamiq Wireframes, a logika detaljno opisana korištenjem UML dijagrama. Odnosi između dijelova sustava će biti prikazani korištenjem pet UML dijagrama: 1. Dijagram slučajeva korištenja, 2. Klasni dijagram, 3. Objektni dijagram, 4. Sekvencijski dijagram i 5. Dijagram aktivnosti. Svaki dijagram će biti posebno prikazan i objašnjen na primjeru aplikacije. Konačan izgled aplikacije tj. sučelja će biti izrađen korištenjem intuitivnog alata Balsamiq Wireframes, koji omogućuje korisniku lako i brzo slaganje idejnog rješenja potrebnog kako bi se isto prezentiralo klijentu i na lak način napravile korekcije.

4. Rezultati istraživanja

Rezultati istraživanja će biti prikazani u tri dijela:

1. Prikaz logike programa pomoću UML dijagrama: kroz 5 dijagrama biti će prikazani odnosi između dijelova aplikacije te njihove interakcije.
2. Finalni izgled biti će skiciran pomoću Balsamiq Wireframe alata.
3. Programsko rješenje će biti izrađeno pomoću C# programskog jezika koristeći Microsoft Visual Studio 2022 kao razvojno okruženje, a aplikacija će biti u obliku Windows Forms aplikacije koja će se izvoditi na desktopu.

4.1. Dijagrami i modeliranje

Izrada nekog poslovnog informacijskog sustava je izuzetno složen i kompliciran proces stoga se prije kodiranja pristupa izradi skica i dijagrama ili modela koji će kako razvojnom timu tako i klijentu pobliže predstaviti proizvod. Modeli prikazuju pojednostavljenu sliku stvarnosti te pružaju jasnu i lako razumljivu sliku proizvoda. Prema Miles i Hamilton "UML tj. Unified Modeling Language (jedinstveni jezik za modeliranje) predstavlja standardni jezik za modeliranje razvoja softvera i sustava te koristi grafički prikaz za izradu apstraktnog modela sustava" (Miles, R., Hamilton, K., 2006). Prednosti UML-a su sažetost, skalabilnost, standardnost i sveobuhvatnost. Jezik za modeliranje je zapravo skup koda, pseudo koda, slika ili dijagrama koji pomažu opisati sustav na više načina i to kao:

- skicu,
- nacrt i
- programski kod.

Kako bi se dovoljno detaljno opisao sustav, koristi se više dijagrama, a u radu će se aplikacija opisati kroz njih pet: 1. Dijagram slučajeva korištenja, 2. Dijagram klasa, 3. Dijagram objekata, 4. Sekvencijski dijagram i 5. Dijagram aktivnosti. Svi će biti pojedinačno detaljno opisani kroz primjer aplikacije. UML dijagrami se mogu podijeliti na dvije temeljne skupine, a to su dijagrami ponašanja i strukture. Od navedenih dijagrama koji će se koristiti u radu, dijagram klasa i dijagram objekata pripadaju dijagramima strukture dok su ostali dijagrami ponašanja.

4.1.1. Dijagram slučajeva korištenja

Dijagram slučajeva korištenja se kreira u početnim fazama, najčešće kao prvi dijagram. On prikazuje koje su funkcionalnosti sustava te odgovara na nekoliko pitanja:

- Što se opisuje?
- Tko su sudionici?
- Što sudionici mogu činiti?

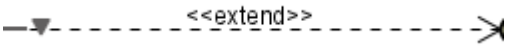
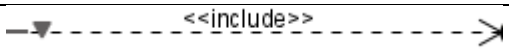
Isti se sastoji od nekoliko elemenata:

- Sudionik
- Slučaj korištenja
- Scenarij

Sudionik predstavlja ulogu koju može igrati korisnik ili drugi sustav. Čini određene akcije ali nije povezan sa sustavom. Kako je već spomenuto, sudionik ne mora biti stvarna osoba, prema Miles i Hamilton "to može biti i sustav treće strane, kao što je u poslovnoj aplikaciji (B2B). Zamislite sudionika kao crnu kutiju: ne možete promijeniti sudionika i ne zanima vas kako funkcionira, ali mora biti u interakciji s vašim sustavom" (Miles, R., Hamilton, K., 2006). Piše ga se kao imenicu u jednini te je poželjno staviti kratak opis. Slučajevi korištenja prikazuju cjelovito funkcioniranje sustava, predstavljaju zadatke koje izvršava sudionik, pišu se kao glagol i definirani su u korisničkim zahtjevima. Slijed logičkih koraka koji opisuju interakciju između sudionika i sustava predstavlja scenarij. Osim navedene tri komponente valja spomenuti i veze između slučajeva korištenja i sudionika. Razlikuju se četiri tipa veze koji su prikazani u tablici 1:

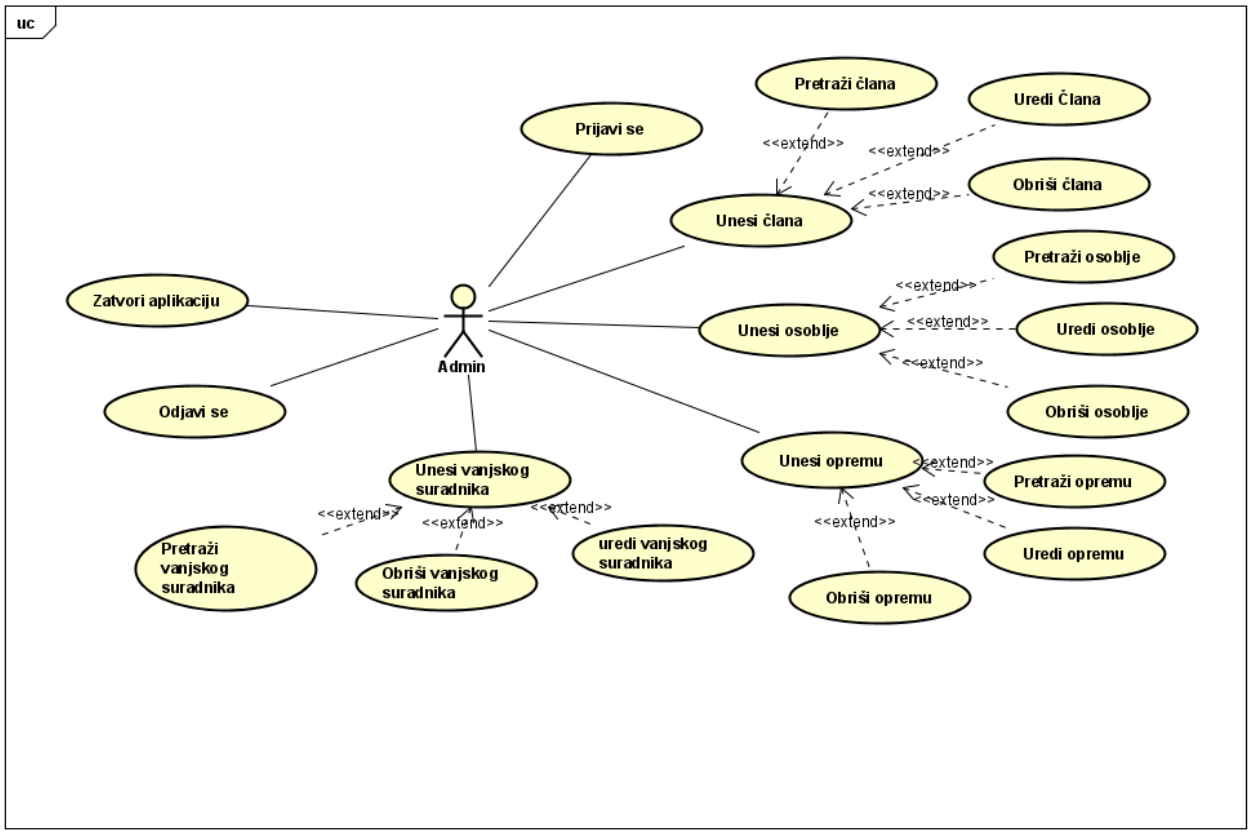
1. Asocijacija
2. Generalizacija
3. Proširenje
4. Uključivanje

Tablica 1. Vrste veza – dijagram slučajeva korištenja, obrada autora

Vrsta veze	Način prikaza
Asocijacija	
Generalizacija	
Proširenje	
Uključivanje	

Asocijacija predstavlja vezu kojom se sudionik i slučaj korištenja povezuju, opisuje razloge za odnos i pravila koja upravljaju odnosom. Veza u kojoj se jedan element modela (dijete) temelji na drugom elementu modela (roditelju) naziva se generalizacija. Koristi se kako bi se naznačilo da podelement prima sve atribute, operacije i odnose koji su definirani u nadelementu. Uključivanje je odnos u kojem jedan slučaj upotrebe uključuje funkcionalnost drugog slučaja upotrebe, a proširenje povezuje dva slučaja korištenja s time da jedan slučaj upotrebe proširuje ponašanje drugog slučaja upotrebe.

Prvi dijagram slučajeva korištenja prikazuje odnos administratora i sustava (slika 3). Postoji šest slučajeva korištenja te ih sve odrađuje administrator: prijava, unos člana, unos osoblja, unos opreme, odjava i zatvaranje aplikacije. Zbog ponavljajućih aktivnosti neće biti objašnjen svaki slučaj pojedinačno.



Slika 3. Dijagram slučajeva korištenja, obrada autora

Sljedeća tablica prikazuje proces prijave administratora u sustav.

Tablica 2.Slučaj korištenja-prijava u sustav, obrada autora.

Naziv:	Prijava u sustav
Kratki opis:	Administrator se prijavljuje u sustav
Preduvjet:	Administrator posjeduje korisničko ime i lozinku
Uspješno stanje:	Administrator se prijavio u sustav
Neuspješno stanje:	Administrator se nije prijavio u sustav
Sudionici:	Administrator
Okidač:	Administrator se želi prijaviti u sustav

Glavni scenarij:	1. Administrator se želi prijaviti u sustav 2. Unosi korisničko ime i lozinku 3. Klika na gumb „Prijava“
Alternativni scenarij:	1. Administrator nije unio točne korisničke podatke

U prvom slučaju korištenja jedini sudionik je administrator kojemu je cilj prijaviti se u sustav kako bi mogao izvršiti ostale aktivnosti. Preduvjet za prijavu je poznavanje korisničkih podataka tj. korisničkog imena i lozinke. Ukoliko korisnik unese točne podatke biti će uspješno prijavljen u sustav, a alternativni scenarij je neuspješna prijava i upozorenje od strane aplikacije kako su uneseni netočni podaci. Tablica 3 prikazuje proces unosa člana u sustav.

Tablica 3. Slučaj korištenja-unos člana, obrada autora.

Naziv:	Unos člana
Kratki opis:	Administrator ispunjava formu za unos podataka o novome članu
Preduvjet:	Administrator je prijavljen i posjeduje podatke za unos
Uspješno stanje:	Administrator je obavio unos novog člana
Neuspješno stanje:	Administrator pri dodavanju člana nije unio točan format podataka u formu te je upozoren od strane aplikacije
Sudionici:	Administrator
Okidač:	Administrator želi unijeti novog člana
Glavni scenarij:	1. Administrator se prijavljuje u sustav 2. Administrator klika na gumb „Novi član“ 3. Ispunjava formu za unos novog člana 4. Klika na gumb „Spremi“ ili „Poništi“ 5. Aplikacija obavještava administratora jesu li podaci spremljeni ukoliko je kliknuo na gumb „Spremi“
Alternativni scenarij:	1. Administrator nije unio točan format podataka u formu

Drugi slučaj korištenja opisuje postupak u kojem administrator unosi novog člana u aplikaciju. Prvo se prijavljuje u sustav te klika na gumb "Novi član" i ispunjava formu za unos člana. Podaci koje unosi su ime, prezime, spol, dob, adresa, datum rođenja, mobilni telefon, e-mail, datum učlanjenja i period izražen u mjesecima za koji se plaća članarina. Nakon što je administrator ispunio formu, klika na gumb "Spremi" te ga aplikacija obavještava ako su podaci uspješno spremljeni. Administratoru je također omogućena opcija da klikom na gumb "Poništi" obriše sve podatke unesene u formu. Proces pretrage člana opisan je u tablici 4.

Tablica 4. Slučaj korištenja-pretraga člana, obrada autora.

Naziv:	Pretraga člana
Kratki opis:	Administrator u tekstni okvir unosi Id člana kako bi ga pronašao u bazi
Preduvjet:	Administrator je prijavljen u sustav zna korisnikov Id
Uspješno stanje:	Administrator je pronašao člana
Neuspješno stanje:	Administrator je unio krivi ili nepostojeći Id
Sudionici:	Administrator
Okidač:	Administrator želi pretražiti člana
Glavni scenarij:	1. Administrator se prijavljuje u sustav 2. Klika na gumb „Novi član“ 3. Klika na gumb „Pretraga“ 4. U tekstni okvir unosi Id člana 5. Klika na gumb „Pretraži“ 6. Traženi član je prikazan u tablici
Alternativni scenarij:	1. Administrator je unio netočan ili nepostojeći Id te ga aplikacija upozorava na isto

U ovome slučaju korištenja administrator želi pretražiti člana kako bi mogao pregledati informacije o istome. Preduvjet je prijava u sustav i da administrator zna Id člana kojeg želi pretražiti. Prvo klika na gumb "Novi član", a potom na gumb "Pretraga" nakon čega se prikazuje forma koja sadrži tekstni okvir za unos Id-a. Nakon što administrator unese Id, član i podaci o istom će biti prikazani

u tablici uz uvjet da je Id točan. Ukoliko je Id netočan ili nepostojeći, aplikacije će upozoriti korisnika. Tablica 5 Prikazuje proces uređivanja člana.

Tablica 5. Slučaj korištenja-uređivanje člana, obrada autora.

Naziv:	Uređivanje člana
Kratki opis:	Administrator u tekstni okvir unosi Id člana kako bi ga pronašao u bazi te klika na gumb „Uredi“ i ispunjava formu
Preduvjet:	Administrator je prijavljen u sustav zna korisnikov Id
Uspješno stanje:	Administrator je pronašao člana i uredio podatke
Neuspješno stanje:	1. Administrator je unio krivi ili nepostojeći Id 2. Administrator je unio krivi format podataka u formu
Sudionici:	Administrator
Okidač:	Administrator želi urediti člana
Glavni scenarij:	1. Administrator se prijavljuje u sustav 2. Klika na gumb „Novi član“ 3. Klika na gumb „Uredi“ 4. U tekstni okvir unosi Id člana 5. Klika na gumb „Pretraži“ 6. Ispunjava formu 7. Klika na gumb „Spremi promjene“ 8. Aplikacija obavještava administratora da su podaci spremljeni
Alternativni scenarij:	1. Administrator je unio netočan ili nepostojeći Id te ga aplikacija upozorava na isto 2. Administrator unosi krivi format podataka u formu

Za uređivanje člana administrator mora biti prijavljen u sustav. Klika na gumb “Novi član” te nakon pojave forme klika na gumb “Uredi”. Pojavljuje se tekstni okvir za unos Id-a člana kojeg se želi urediti te klikom na gumb “Pretraga” član se prikazuje u tablici. Klikom na gumb “Uredi”

pojavljuje se forma za unos podataka za pretraženog korisnika i nakon što je administrator unio odgovarajuće podatke klika na gumb “Spremi promjene” te aplikacija obavještava korisnika ako su podaci uspješno spremljeni. U tablici 6 opisan je proces brisanja člana iz baze podataka.

Tablica 6. Slučaj korištenja-brisanje člana, obrada autora.

Naziv:	Brisanje člana
Kratki opis:	Administrator u tekstni okvir unosi Id člana kako bi ga pronašao u bazi te klika na gumb „Obriši“
Preduvjet:	Administrator je prijavljen u sustav zna korisnikov Id
Uspješno stanje:	Administrator je pronašao člana i obrisao ga
Neuspješno stanje:	Administrator je unio krivi ili nepostojeći Id
Sudionici:	Administrator
Okidač:	Administrator želi obrisati člana
Glavni scenarij:	1. Administrator se prijavljuje u sustav 2. Klika na gumb „Novi član“ 3. Klika na gumb „Obriši“ 4. U tekstni okvir unosi Id člana 5. Klika na gumb „Obriši“ 6. Aplikacija upozorava korisnika da li je siguran u brisanje člana 7. Administrator klika na gumb „Da“ 8. Aplikacija obavještava administratora da je član obrisan
Alternativni scenarij:	Administrator je unio netočan ili nepostojeći Id te ga aplikacija upozorava na isto

Ukoliko osoba prestane biti članom teretane, posao administratora je ukloniti tu osobu iz baze podataka. Proces uključuje administratorovu prijavu u sustav te klik na gumb “Novi član”, a zatim klik na “Obriši”. Klikom na gumb “Obriši” pojavljuje se tekstni okvir za unos Id-a člana kojeg se želi obrisati. Nakon što korisnik klikne na gumb “Obriši” pojavljuje se dijaloški prozor koji pita korisnika je li siguran u brisanje izabranog člana na što korisnik može odgovoriti potvrdno te

obrisati člana ili poništiti brisanje. Slučajevi korištenja za unos, pretragu, uređivanje i brisanje opreme, osoblja i vanjskih suradnika neće biti pojedinačno opisani jer je postupak isti kao i u slučaju članova teretane. Nakon završetka rada korisniku je omogućeno odjaviti se iz aplikacije te je taj proces opisan u tablici 7.

Tablica 7. Slučaj korištenja-odjava, obrada autora.

Naziv:	Odjava iz aplikacije
Kratki opis:	Administrator se odjavljuje iz aplikacije
Preduvjet:	Administrator je prijavljen u sustav
Uspješno stanje:	Administrator se odjavio
Neuspješno stanje:	Administrator se nije odjavio
Sudionici:	Administrator
Okidač:	Administrator se želi odjaviti
Glavni scenarij:	1. Administrator se prijavljuje u sustav 2. Klika na gumb „Odjava“ 3. Pojavljuje se dijaloški okvir koji pita korisnika želi li se odjaviti 4. Korisnik klika na gumb „Da“ 5. Korisnik je odjavljen te se pojavljuje forma za unos korisničkog imena i lozinke
Alternativni scenarij:	Administrator je kliknuo na gumb „Ne“ u dijaloškom okviru te poništio odjavu

Nakon što je administrator obavio planirane operacije, iz sigurnosnih razloga klika na gumb “Odjava” nakon kojeg je upitan želi li se odjaviti. Nakon što klikne na gumb “Da” pojavljuje se forma za unos korisničkog imena i lozinke. Izlaz iz aplikacije kao posljednja aktivnost koju administrator može obavljati opisana je u potonjoj tablici.

Tablica 8. Slučaj korištenja-izlaz, obrada autora.

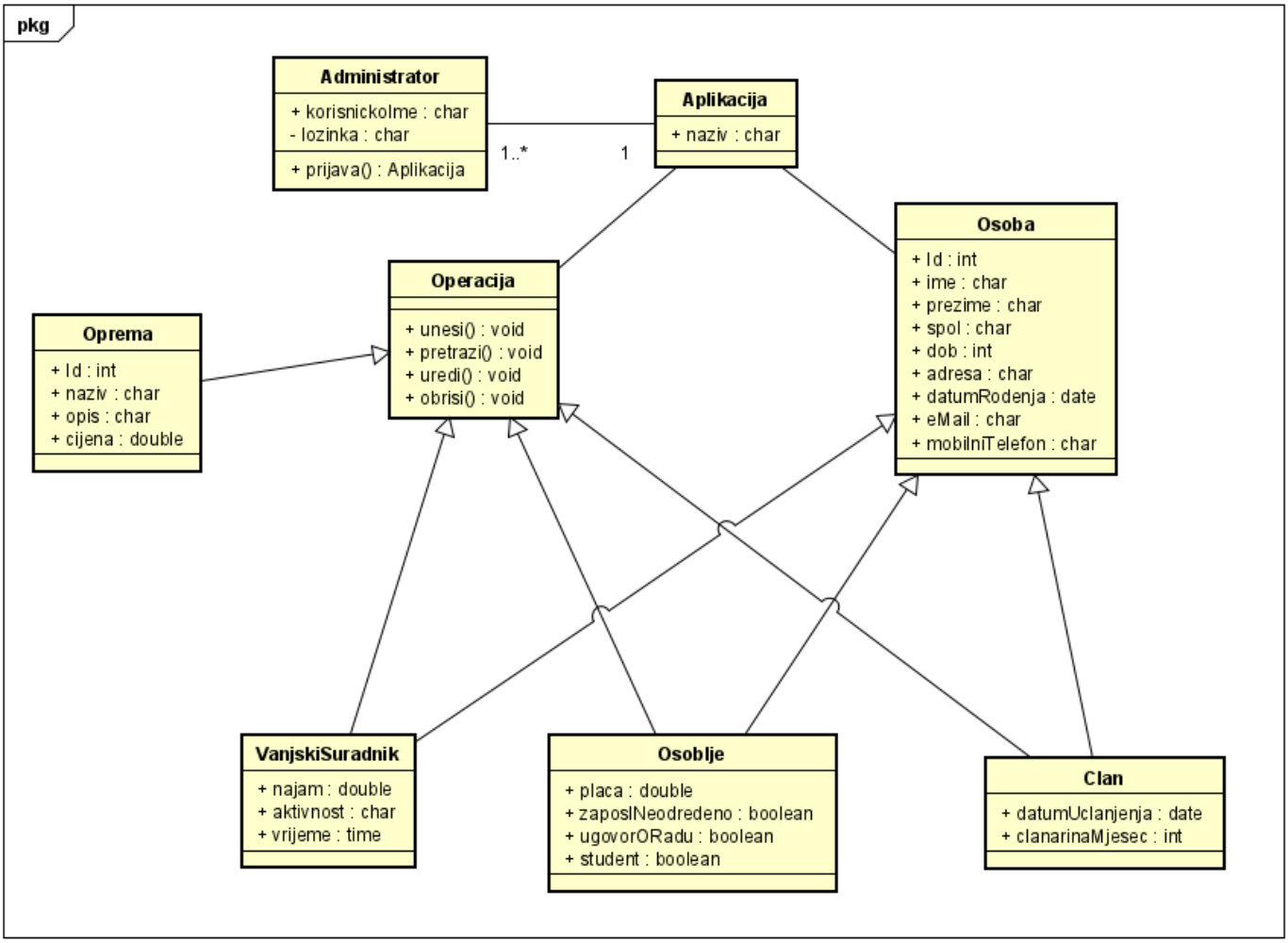
Naziv:	Izlaz iz aplikacije
Kratki opis:	Administrator izlazi iz aplikacije
Preduvjet:	Administrator je prijavljen u sustav
Uspješno stanje:	Administrator je izašao iz aplikacije
Neuspješno stanje:	Administrator nije izašao iz aplikacije
Sudionici:	Administrator
Okidač:	Administrator želi izaći iz aplikacije
Glavni scenarij:	2. Administrator se prijavljuje u sustav 4. Klika na gumb „Izlaz“ 5. Pojavljuje se dijaloški okvir koji pita korisnika želi li izaći iz aplikacije 6. Korisnik klika na gumb „Da“ 7. Korisnik je ugasio aplikaciju
Alternativni scenarij:	Administrator je kliknuo na gumb „Ne“ u dijaloškom okviru te poništio izlaz

Ukoliko administrator više neće koristiti aplikaciju, omogućeno mu je kliknuti na gumb “Izlaz” koji će uz prethodno upozorenje ugasiti aplikaciju. Preduvjet je da je administrator već prijavljen u sustav.

4.1.2. Dijagram klasa

Dijagram klasa je najčešće korišteni dijagram, a razlog tome leži prvenstveno u korištenju principa objektno orijentiranog programiranja u razvoju softvera. Naime, objektno orijentirano programiranje se temelji na klasama tj. objektima i odnosima među njima, a dijagram klasa najbolje opisuje iste. Prvo valja definirati što je klasa. Klase opisuju objekte sustava odnosno jedinke iz realnog svijeta, predstavljaju obrazac ponašanja prema kojem će se voditi svi objekti. (Miles i Hamilton, 2006. Str 89.) navode da klase sadrže dva tipa informacija – u kakvom stanju su njeni objekti te kako će se ponašati. Klasa u UML dijagramu je prikazana kao pravokutnik podijeljen u

tri sekcije. Prva sadrži ime klase, druga njene atribute, a u posljednjem dijelu su operacije koje izvodi. Na sljedećoj slici može se vidjeti dijagram koji prikazuje 8 klasa koje čine aplikaciju.



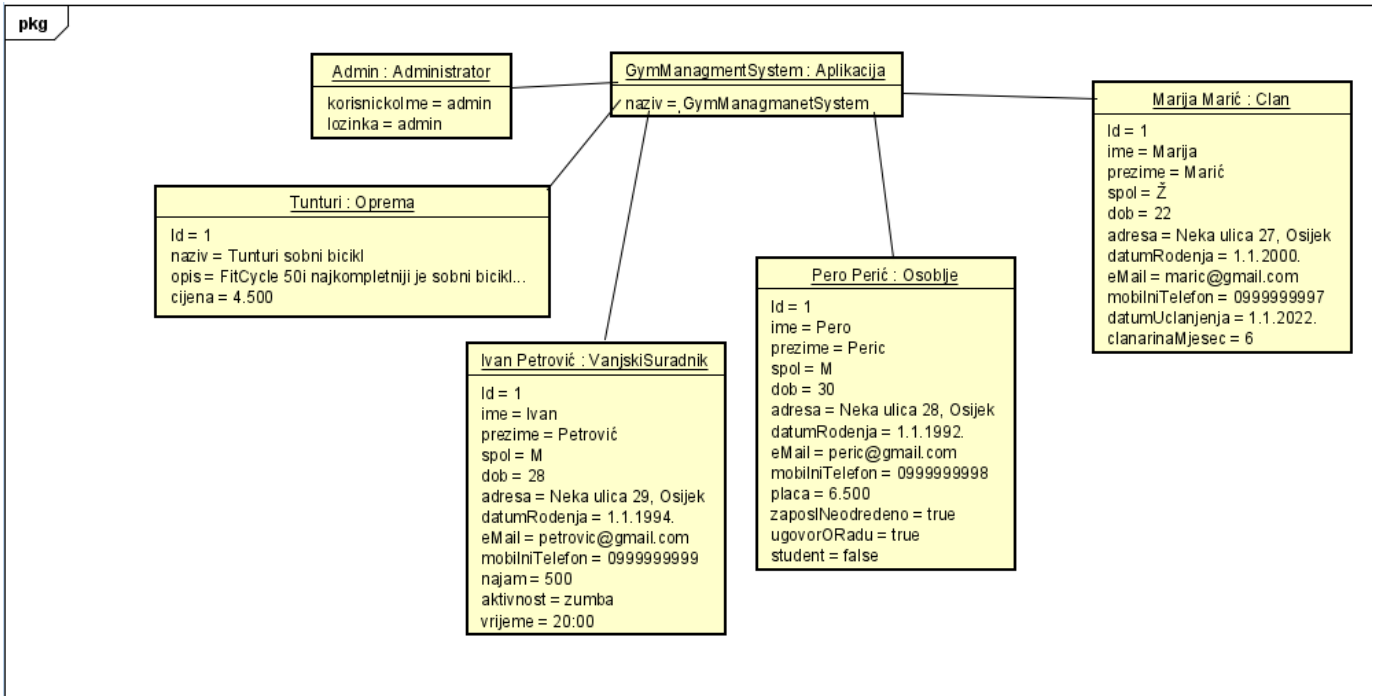
Slika 4. Dijagram klasa, obrada autora

Klasa Aplikacija predstavlja cjelokupan sustav te u sebi sadrži atribut „naziv“ ispred kojeg stoji “+” što naznačava kako je riječ o javno vidljivom atributu. S druge strane atribut lozinka je skriven i samo je korisnik može vidjeti. Administrator klasa je povezana s klasom Aplikacija dvosmjernom asocijacijom jedan ili više naprama jedan što znači da aplikacija može imati više administratora, no korisnik može biti administrator samo jedne aplikacije te sadrži atribute korisnickoIme tipa char

i lozinka, a od operacija prijava. Klase Operacija i Osoba su apstraktne klase, “klasa bez ikakvih instanci. Predstavljaju zajedničke klase za druge izvedene podklase.” (Softwareideas.net, 2022). Dakle u sebi sadrže atribute ili operacije koji su zajednički nekim drugim klasama koje će ih nasljeđivati. U promatranom primjeru podklase VanjskiSuradnik, Osoblje i Clan su povezane vezom agregacije, vezom jačom u odnosu na asocijaciju te naznačuje kako podklasa nasljeđuje atribute nadklase. Nadklasa Osoblje sadrži atribute Id, ime, prezime, spol dob, adresa, datumRodenja, email i mobilni telefon dakle sve one podatke koje će imati svaka osoba povazana sa sustavom neovisno o ulozi koju ima dok se u podklasi mogu naknadno definirati atributi ili operacije koje će imati samo ta klasa. Na primjer, podklasa Osoblje ima atribute placa, ugovorORadu itd. Osim što navedene klase nasljeđuju nadklasu Osoba, također nasljeđuju klasu Operacija koja u sebi ima opisane metode koje će koristiti svaka klasa, a to su operacije unesi, uredi, pretrazi i obrisi. Klasa Oprema nije spojena s nadklasom Osoba jer nema atribute takve prirode, za razliku od ostalih klasa njene instance će biti predmeti tj. oprema poput sprava za vježbanje, a ne žive osobe.

4.1.3. Dijagram objekata

Sljedeći dijagram proizlazi iz dijagrama klasa jer objekti predstavljaju instance klasa odnosno jedinke iz realnog svijeta. Prema Miles i Hamilton (2006. Str., 95.) objekti su srž svakog objektno orijentiranog sustava jer objekti čine njegove dijelove i oživljavaju klase. Na slici 5 je vidljivo kako su objekti naslijedili atribute koji su definirani u klasama.



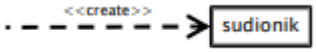
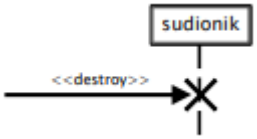
*Slika 5. D*ijagram objekata, obrada autora

Prikaz objekta se sastoji od naziva instance klase npr. Pero Perić pored čijeg naziva stoji ime klase koju taj objekt nasljeđuje. Zatim se u tijelu objekta nalaze definirani atributi s jedinstvenim vrijednostima bez definiranog tipa podatka. Ono što valja istaknuti jest to da nema instanci apstraktnih klasa već su njihovi atributi instancirani u podklasama.

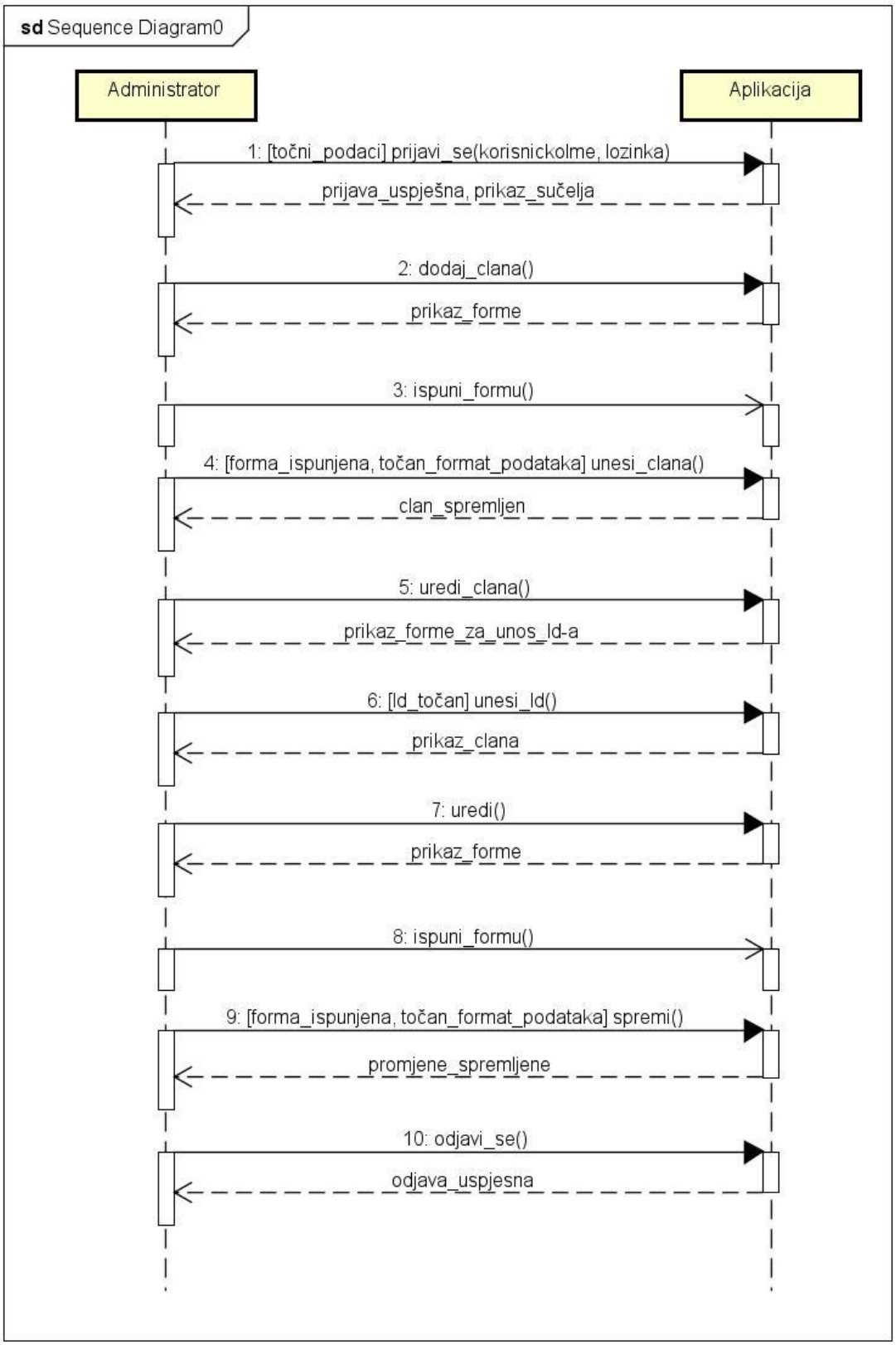
4.1.4. Sekvencijski dijagram

Sekvencijski dijagram spada u kategoriju dijagrama ponašanja te on prikazuje kako sustav obavlja specifične zadatke te definira vremenski redoslijed. Prema Miles i Hamilton (2006., Str. 214.) sekvencijski dijagrami oslikavaju redoslijed interakcija između dijelova sustava. Koristeći sekvencijski dijagram, mogu se opisati koje će se interakcije pokrenuti kada se izvrši određeni slučaj korištenja i kojim će se redoslijedom te interakcije dogoditi. Dijagram se sastoji od sudionika i poruka koje razmjenjuju. Sudionik je definiran nazivom i životnom linijom, a poruke sadrže parametar te mogu biti sinkrone, asinkrone, povratne, poruke kreiranja i brisanja sudionika što je vidljivo u Tablici 9.

Tablica 9. Vrste poruka – Sekvencijski dijagram, obrada autora

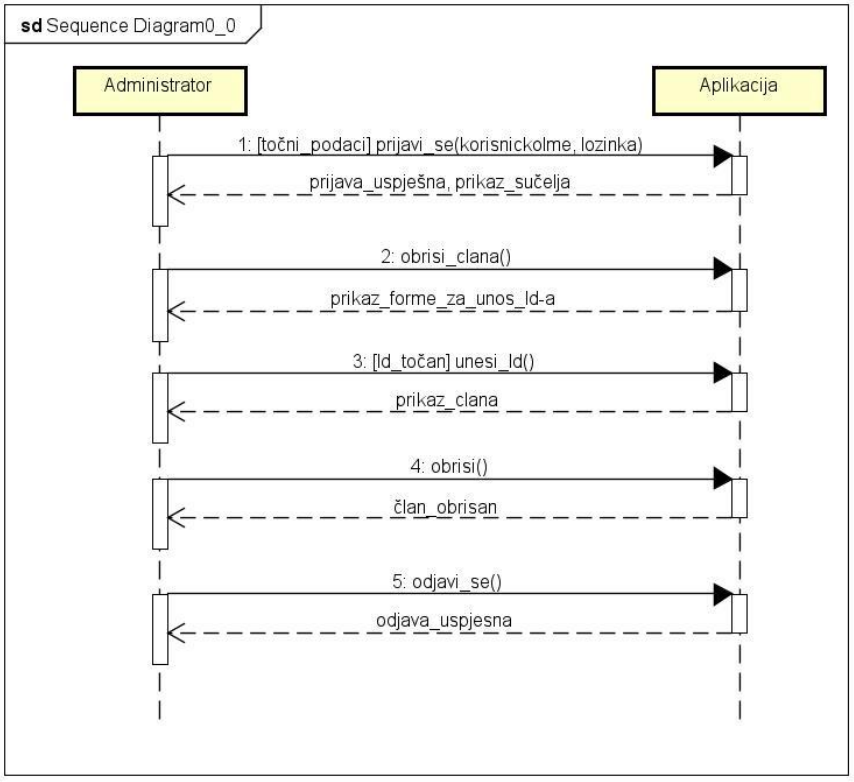
Vrsta poruka	Način prikaza
Sinkrona	
Asinkrona	
Povratna poruka	
Kreiranje sudionika	
Brisanje sudionika	

Sinkrona poruku sudionik šalje onda kada očekuje odgovor koji dolazi u obliku povratne poruke te je okidač za daljnji događaj dok se kod asinkrone poruke ne očekuje nikakav odgovor. Neće svi sudionici biti u procesu do kraja, to ovisi o samim procesima stoga će se neki od njih ukloniti ili kreirati, a to omogućuju poruke kreiranja i brisanja. Prvi dijagram prikazuje kronološki slijed operacija koje izvodi administrator pri prijavi, unosu, uređivanju i odjavi iz aplikacije (slika 6).



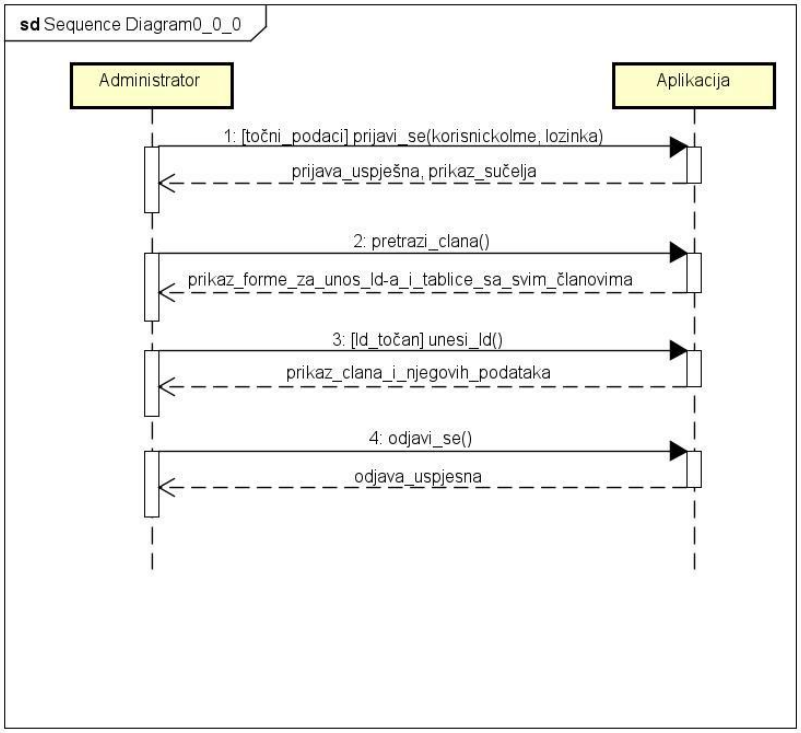
Slika 6. Sekvencijski dijagram - unos i uređivanja člana, obrada autora

Za prijavu su mu potrebni korisničko ime i lozinka te nakon unosa istih pod uvjetom da su točni, u aplikaciji se prikazuje sučelje s mogućim opcijama što predstavlja povratnu poruku na sinkronu vezu. Korisnik klikom na gumb "Dodaj člana" dobiva odgovor od aplikacije u vidu forme za unos podataka. Nakon što je korisnik ispunio formu i kliknuo na gumb "Unesi člana", aplikacije šalje poruku kako je član spremljen u bazu pod uvjetom da je forma ispunjena te da su podaci u točnom formatu. Sljedeći proces je vrlo sličan prvome, kod uređivanja je potrebno u tekstni okvir upisati Id člana kojeg se želi uredit te ukoliko je Id točan, aplikacija odgovara prikazom istog. Nakon klika na "Uredi" proces ispune forme je identičan kao i kod unosa te na kraju administrator sprema promjene u bazu. Zadnja operacija je odjava iz sustava koja je predstavljena sinkronom vezom u kojoj aplikacija nakon odjave prikazuje početno sučelje pri pokretanju aplikacije. Sljedeći dijagram prikazuje proces brisanja člana iz baze podataka (slika 7).



Slika 7. Sekvencijski dijagram - brisanje člana, obrada autora

Nakon postupka prijave koji je objašnjen u prethodnom primjeru, slijedi brisanje člana. Korisnik klika na gumb “Obriši člana” te mu aplikacija odgovara prikazom tekstnog okvira za unos Id-a člana kojeg se želi obrisati. Ukoliko je Id točan, član će biti obrisan iz baze klikom na gumb “Obriši”, a aplikacija će o tome porukom obavijestiti korisnika. Posljednji dijagram prikazuje pretragu člana u bazi (slika 8).



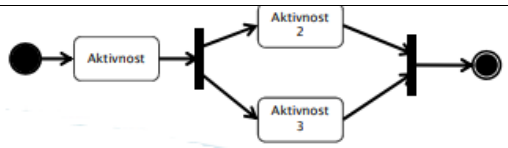
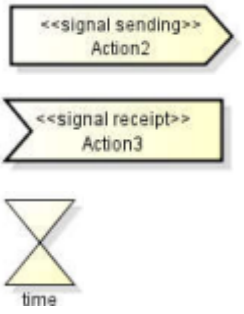
Slika 8. Sekvencijski dijagram - pretraga člana, obrada autora

Korisnik unosom Id-a šalje sinkronu poruku na koju aplikacija, ukoliko je Id točan i postojeći, odgovara prikazom traženog člana u tablici. Proces unosa, pretrage, uređivanja i brisanja je isti za osoblje, opremu i vanjske suradnike te stoga oni neće biti pojedinačno objašnjavani.

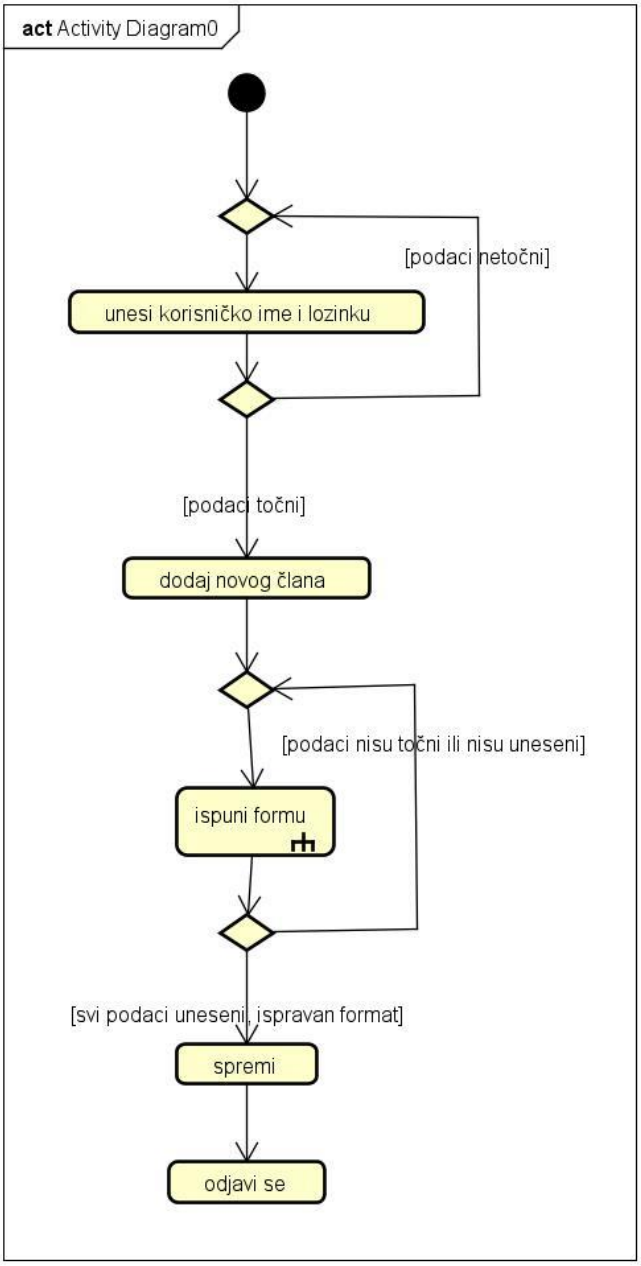
4.1.5. Dijagram aktivnosti

Dijagram aktivnosti vizualno predstavlja niz radnji i izgledom podsjeća na algoritam te se iz tog razloga često koristi u modeliranju poslovnih procesa. Temeljni elementi dijagrama aktivnosti su sljedeći – početno i konačno stanje, akcije, prijelaz između aktivnosti, odluka i spajanje, račvanje i skupljanje te signal. Potonja tablica prikazuje kako su isti kreirani u UML-u.

Tablica 10. Temeljni dijelovi – Dijagram aktivnosti, obrada autora

Naziv	Način prikaza
Početno i konačno stanje	
Akcije	
Prijelaz između aktivnosti	
Odluka i spajanje	
Račvanje i skupljanje	
Signal 1. Šaljući 2. Primajući 3. Vremenski	

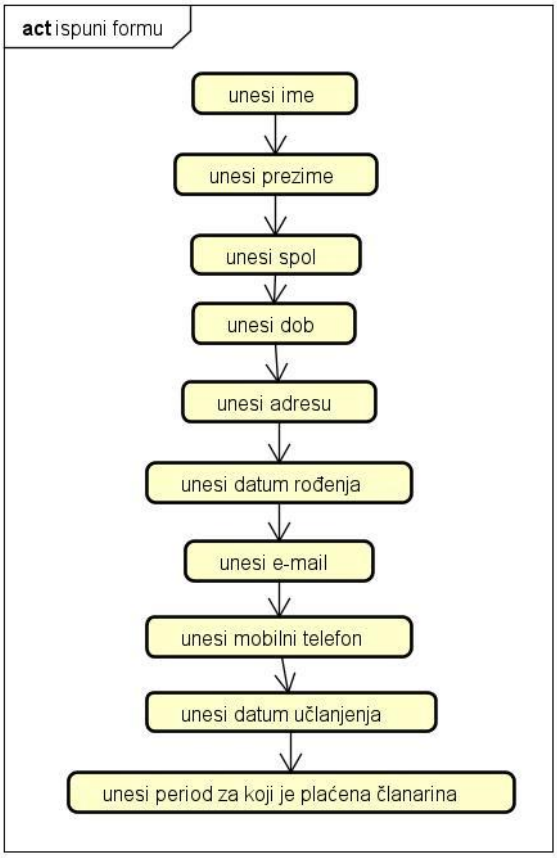
U radu će biti prikazana dva dijagrama aktivnosti, opisani procesi će biti prijava, unos, pretraga i brisanje člana te odjava. Slika 9 prikazuje prvi dijagram aktivnosti koji opisuje proces prijave i unosa novog člana u sustav.



Slika 9. Dijagram aktivnosti - dodavanje novog člana, obrada autora

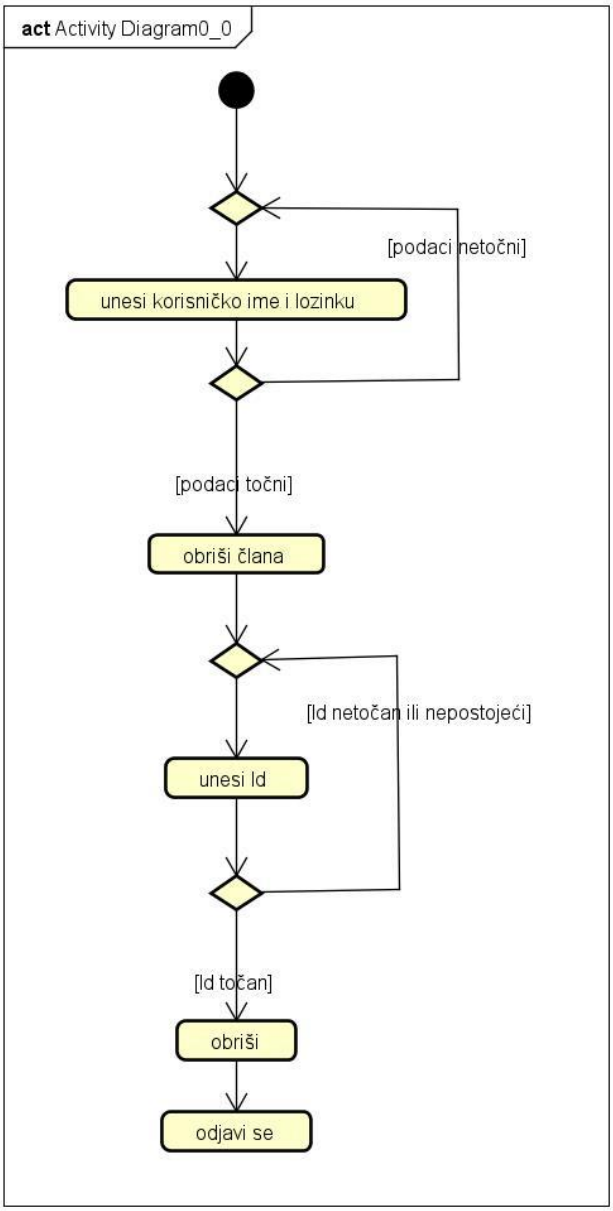
Prvi korak je prijava koja od korisnika zahtjeva unos točnog korisničkog imena i lozinke. Ukoliko potonji nisu točni, aplikacija upozorava korisnika te je prisiljen upisati pristupne podatke ponovo. Sljedeći korak je klik na gumb “Dodaj novog člana” nakon čega se pojavljuje forma za unos.

Aktivnost ispuni formu poziva druge aktivnosti, a to su unos imena, prezimena, spola i ostalih podataka o korisniku. Podaci koje administrator treba upisati vidljivi su na slici 10.



Slika 10. Dijagram aktivnosti - ispuni formu, obrada autora

Nakon što je administrator upisao sve podatke u formu, može nastaviti dalje na spremanje korisnika u bazu te se na kraju odjavljuje. Sljedeća aktivnost prikazana je na slici 11 te predstavlja brisanje člana iz baze podataka.

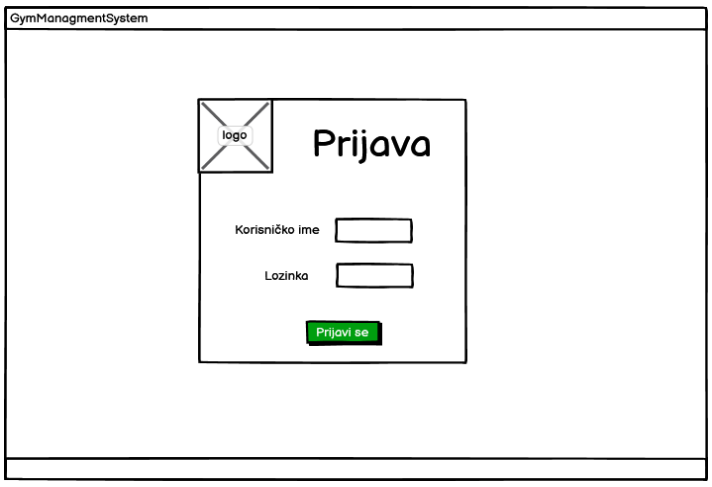


Slika 11. Dijagram aktivnosti - brisanje člana, obrada autora

Nakon već opisanog procesa prijave, korisnik klika na gumb ”Obriši člana” te se pojavljuje tekstni okvir za unos Id-a korisnika kojeg se želi obrisati. Ukoliko Id nije točan korisnik ne može prijeći na sljedeću fazu, a to je potvrda brisanja i odjava iz aplikacije nakon čega se vraća na početno sučelje.

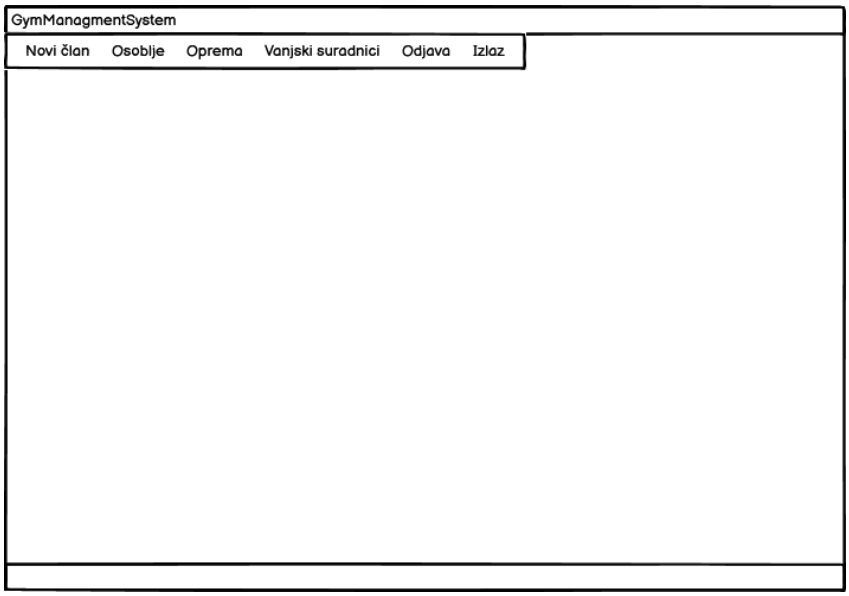
4.2. Izrada skice aplikacije upotrebom Balsamiq Wireframes alata

Prilikom izrade aplikacije, istu je potrebno prvo skicirati upotrebom nekog alata. Skiciranje uvelike pomaže developerima na način da im se jasno vizualizira kako finalni proizvod treba izgledati, a klijentima se može prezentirati zamišljeno rješenje te prepraviti određene stavke na lak način ukoliko to klijent zahtjeva. Proces izrade skica naziva se wireframing te je vrlo važan korak u izradi aplikacije. Prema Tutsplus.com (2020) wireframing omogućuje definiranje hijerarhije informacija unutar dizajna te olakšava planiranje izgleda obrade informacija ovisno o klijentovim željama. Alat za dizajn korisničkog sučelja koji će se koristiti u radu je Balsamiq Wireframe, prema mnogima najbrži i najbolji low fidelity alat u industriji. Balsamiq se koristi za testiranje i prezentaciju rada u početnim fazama, prikaz koncepta rješenja u ranoj fazi, dok se kasnije mnogi UX dizajneri prebacuju na neki kompleksniji alat. Ovisno o razini detalja, razlikuju se wireframe, mock up i prototype. Wireframe je najjednostavniji prikaz te se najčešće zadržava na korištenju blok dijagrama i grey box elemenata. Mockup predstavlja srednju razinu detalja te služi kao dopuna temeljima koji su postavljeni korištenjem wireframe-a. Posljednji pojam je prototype koji omogućuje interakciju korisnika sa skicom putem gumba. U radu će se izraditi mockup-ovi koji predstavljaju izgled stranice pri prijavi u sustav, dodavanju, pretrazi i uređivanju člana teretane. Prva stranica koja se pojavljuje u aplikaciji je forma za upis korisničkog imena i lozinke, a popunjava ju administrator (slika 12).



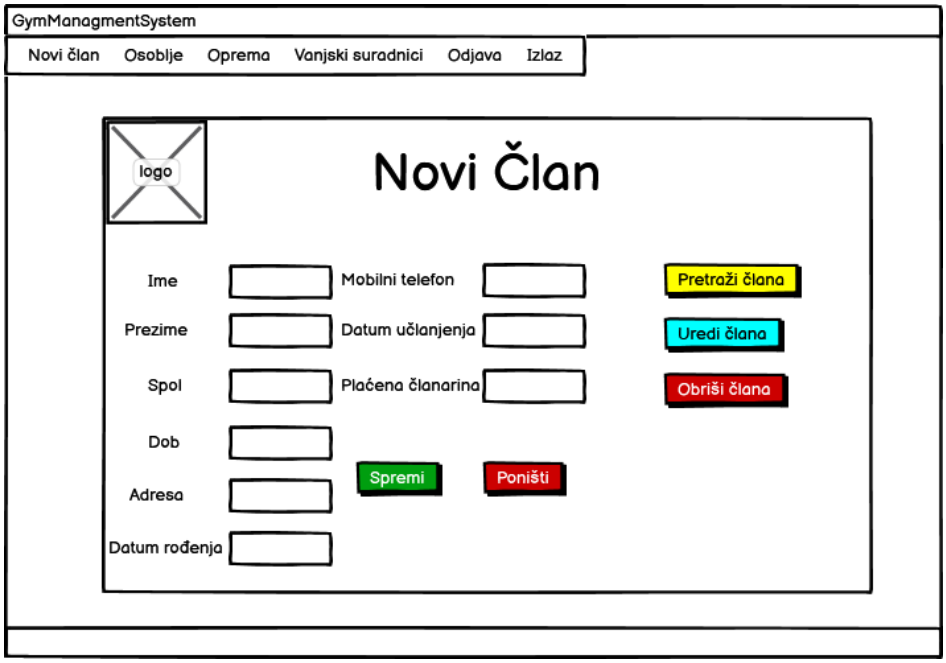
Slika 12. Mockup - prijava, obrada autora

Podaci za upis su hardkodirani u rješenju te su adminu dani na korištenje od strane developera. Sljedeća slika prikazuje izgled početne stranice aplikacije.



Slika 13. Mockup - Početna, obrada autora

Ista administratoru omogućuje klik na druge opcije poput unosa novog člana, osoblja, opreme i vanjskih suradnika te odjavu i kraj rada. Klikom na gumb “Novi član” koji se nalazi u menu bar-u pojavljuje se forma za unos novog člana teretane (slika 14).

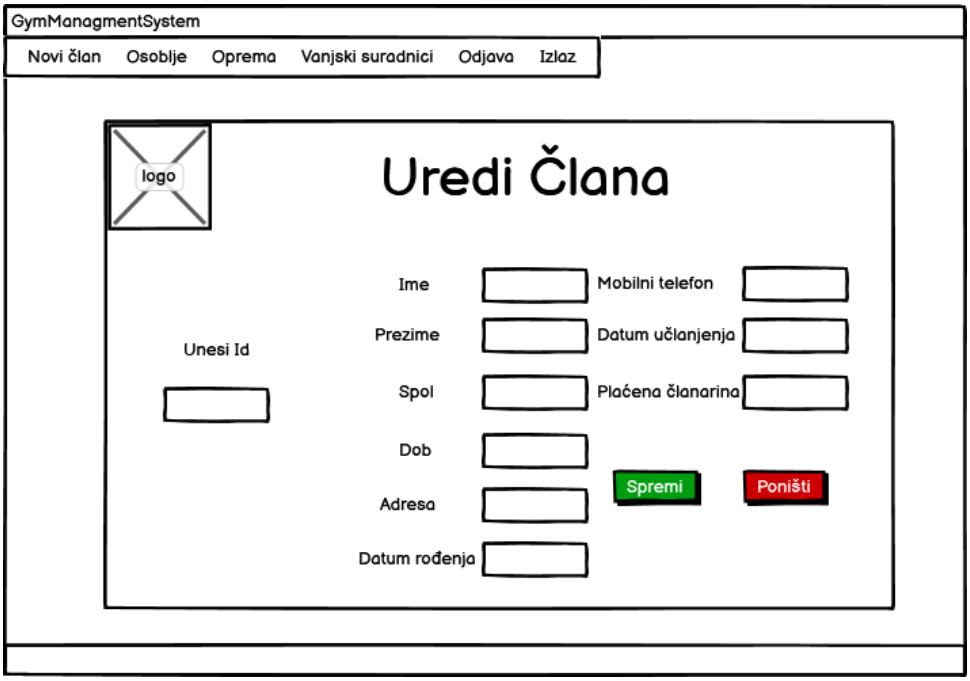


Slika 14. Mockup - novi član, obrada autora

Nakon što je svaki tekstni okvir ispunjen odgovarajućim formatom podataka, administrator klika na “Spremi” te se član sprema u bazu ili na “Poništi” što izbriše tekst iz forme. Također su omogućene opcije pretrage, uređivanja i brisanja člana. Klikom na gumb “Pretraži člana” pojavljuje se sljedeća forma (slika 15):

Slika 15. Mockup - pretraži člana, obrada autora

Vidljiv je tablični prikaz sa svim evidentiranim članovima te se od korisnika traži da unese Id korisnika čije podatke želi pregledati. Nakon unosa točnog Id-a, u tablici se prikazuje samo jedan član. Proces biranja člana je gotovo identičan te se neće pojedinačno objašnjavati. Slika 16 prikazuje izgled sučelja stranice za uređivanje člana teretane.



Slika 16. Mockup - uredi člana, obrada autora

Prilikom uređivanja postojećeg člana, potrebno je upisati odgovarajući Id te upisati podatke u formu. Kao i kod unosa novog člana, korisnik ima opciju spremiti promjene ili poništiti proces. Proces unosa, brisanja, uređivanja i pretrage je isti za osoblje, opremu i vanjske suradnike stoga se neće zasebno pojašnjavati.

4.3. Izrada windows forms aplikacije u C# programskom jeziku

Nakon prikaza logike programa pomoću UML dijagrama i skiciranja istog koristeći Balsamiq Wireframes, prelazi se na izradu aplikacije u C# programskom jeziku. Razvojno okruženje koje će se koristiti je MS Visual Studio 2022, a aplikacija će biti tipa Windows Forms.

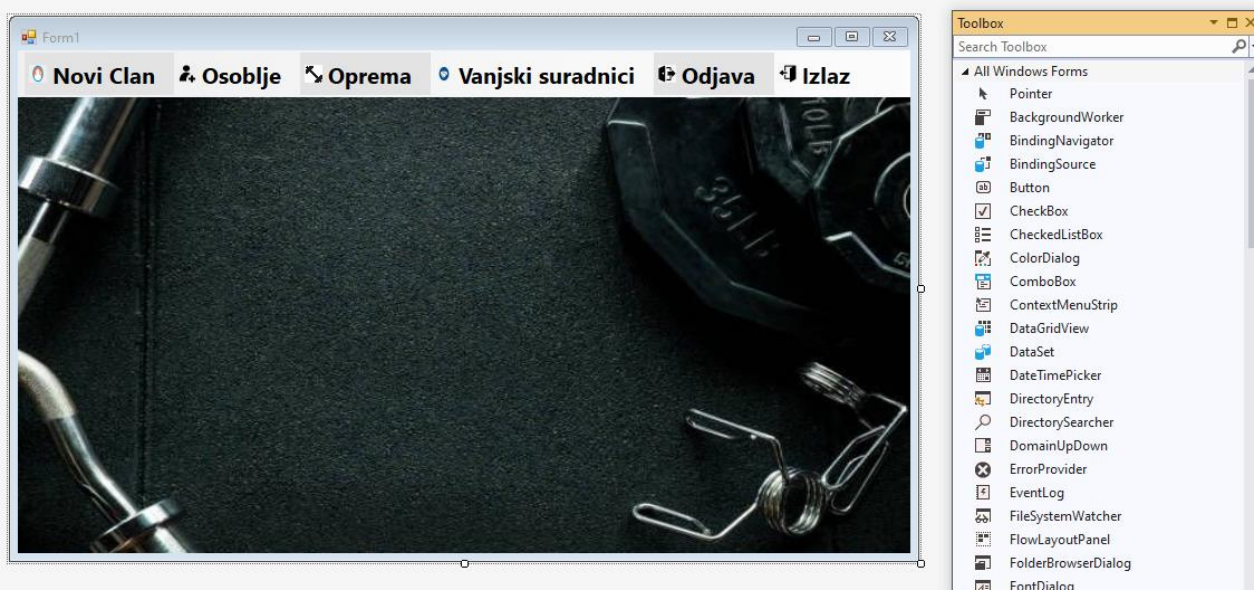
4.3.1. OOP, C# i Windows Forms

OOP ili objektno orijentirano programiranje je jedan od pristupa računalnom programiranju, a težište u projektiranju aplikacije leži na objektima i porukama koje razmjenjuju među sobom. "Objektno orijentirano programiranje je programska paradigma koja se oslanja na koncept klasa i objekata. Koristi se za strukturiranje softverskog programa u jednostavne dijelove koda za višekratnu upotrebu (koji se obično nazivaju klasama), koji se koriste za stvaranje pojedinačnih instanci objekata" (Eduacative.io 2020). Neizostavno je spomenuti i principe OOP-a, a to su apstrakcija, učahurivanje, nasljeđivanje i polimorfizam. C# je moderan objektno orijentirani programski jezik koji omogućuje izradu web aplikacija, softverskih rješenja te je prisutan i u industriji igrica, a "izvodi se na .NET-u, virtualnom sustavu izvršavanja koji se naziva Common Language Runtime (CLR)"(docs.Microsoft.com 2022). Nastao je 2000. Godine i razvila ga je kompanija Microsoft, a najzaslužnijim se smatraju Anders Hejlsberg(dizajner) i Mads Torgersen(developer). Windows Forms je biblioteka klasa grafičkog korisničkog sučelja (GUI) koja se nalazi u paketu .Net Framework. Glavna svrha je korisniku omogućiti brzu i laku izradu aplikacija za stolna računala kroz intuitivno grafičko sučelje. Smatra se dobrim prvim korakom za početnike jer se izrada sučelja aplikacije radi putem "drag and drop" te se klikom na određenu komponentu izrađuje klasa u kojoj se definira što će ista izvršavati.

4.3.2. Programsko rješenje

U ovome poglavlju aplikacija će biti prezentirana kroz nekoliko fotografija programskog koda uz opis što isti omogućuje. Procesi koji će se opisati su unos, pretraga, uređivanje i brisanje novog člana te prijava u sustav. Aplikacija je stvorena uz pomoć drop & drag sustava, nakon stvaranje forme istu je moguće urediti toolbox-om. Toolbox u sebi sadrži sve elemente potrebne za izradu

sučelja aplikacije npr. Gridovi, tekstni okviri, fotografije, menu itd. Slika 17. prikazuje dizajn početnog sučelja koje vodi na sve ostale opcije.



Slika 17. Programsko rješenje - dizajn sučelja

Dvostrukim klikom na jednu od stavki u izborniku npr. “Novi član”, stvara se metoda clickEvent u kojoj se definira da nakon klika na tu stavku korisnik bude odveden na zasebnu formu gdje se upisuje novog člana u evidenciju. Na slici 16 prikazan je programski kod za dodavanje nove klase odnosno forme za unos članova.

```
1 reference
private void dodajToolStripMenuItem_Click(object sender, EventArgs e)
{
    NewMember nm = new NewMember();
    nm.Show();
}
```

Slika 18. Programsko rješenje - prikaz forme

Forma za dodavanje novog člana u bazu će također biti napravljena koristeći toolbox. Nakon ispune svih tekstnih okvira, korisnik klika na gumb "Spremi" što pokreće metodu koja podatke iz forme sprema u zasebne varijable te se ti podaci nakon spajanja s bazom podataka spremaju u istu, a kod koji to omogućuje prikazan je na slici 19.

```
1 rererence
private void btnSave_Click(object sender, EventArgs e)
{
    String fname = txtFirstName.Text;
    String lname = txtLastName.Text;

    String gender = "";

    bool isChacked = radioButton3.Checked;

    if (isChacked)
    {
        gender = radioButton3.Text;
    }
    else
    {
        gender = radioButton1.Text;
    }

    String dob = dateTimePickerDOB.Text;
    Int64 mobile = Int64.Parse(txtMobile.Text);
    String email = txtEmail.Text;
    String joindate = dateTimePickerJoinDate.Text;
    String address = txtAddress.Text;
    String membership = comboBoxMembership.Text;


    //spajanje na bazu podataka u mssms-u
    SqlConnection con = new SqlConnection();
    //con.ConnectionString = "data source = DESKTOP-EQ0AOTO; database = gym; integrated security = True";
    con.ConnectionString = "data source = DESKTOP-EQ0AOTO; database = gymZavrsni; integrated security = True";
    SqlCommand cmd = new SqlCommand();
    cmd.Connection = con;

    cmd.CommandText = "insert into NewMember (Fname, Lname, Gender, Dob, Mobile, Email, JoinDate, Maddress, MembershipTime) values " +
        "('" + fname + "', '" + lname + "', '" + gender + "', '" + dob + "', " + mobile + ", '" + email + "', '" + joindate + "', '" + address + "', '" + membership + "')";

    SqlDataAdapter DA = new SqlDataAdapter(cmd);
    DataSet DS = new DataSet();
    DA.Fill(DS);
    MessageBox.Show("Data saved");
```

Slika 19. Programsko rješenje - spremanje podataka u bazu

Kako bi se obavile bile kakve akcije nad podacima, u kod je potrebno uključiti SQL naredbe. SQL naredba koja se koristila za unos podataka je " insert into 'ImeTablice' " nakon čega je potrebno redom navesti svaki stupac iz baze kako bi se podaci spremili. Naredba "select * from 'ImeTablice' where MID = " + txtSearch.Text + "" (slika 18) omogućuje ispis svih podataka u data grid o jednome članu ovisno o njegovome Id-u koji unosi korisnik.

```
    //cmd.CommandText = "select * from NewMember where MID = " + txtSearch.Text + " OR Lname = " + txtSearchLastName.Text + "";
    cmd.CommandText = "select * from NewMember where MID = " + txtSearch.Text + "";
    SqlDataAdapter DA = new SqlDataAdapter(cmd);

    DataSet DS = new DataSet();
    DA.Fill(DS);

    dataGridView1.DataSource = DS.Tables[0];
}
```

Slika 20. Programsko rješenje - prikaz podataka iz baze

Kod za brisanje članova iz baze je vrlo sličan prethodno opisanom. Potrebna SQL instrukcija glasi "delete from NewMember where MID = " + txtDelete.Text + ""; dakle briše onog korisnika čiji je Id administrator upisao u tekstni okvir s imenom txtDelete.

Proces uređivanja člana i spremanja promjena započinje upisivanjem Id-a člana kojeg se želi urediti a zatim se podaci unose u formu. Potrebna SQL instrukcija je “update” uz dodatak uvijeta “where MID = '" + id + "'” (slika 21). Nakon unosa svih podataka korisnik sprema podatke te ga o uspješnoj pohrani obavještava aplikacija.

```
cmd.CommandText = "update NewMember set " +
    "Fname = '" + fname + "', Lname = '" + lname + "', Gender = '" + gender + "', Dob = '" + dob + "', Mobile = " + mobile + ", " +
    "Email = '" + email + "', JoinDate = '" + joindate + "', Maddress = '" + address + "', MembershipTime = '" + membership + "' where MID = '" + id + "'";

SqlDataAdapter DA = new SqlDataAdapter(cmd);
DataSet DS = new DataSet();
DA.Fill(DS);
MessageBox.Show("Data saved");
```

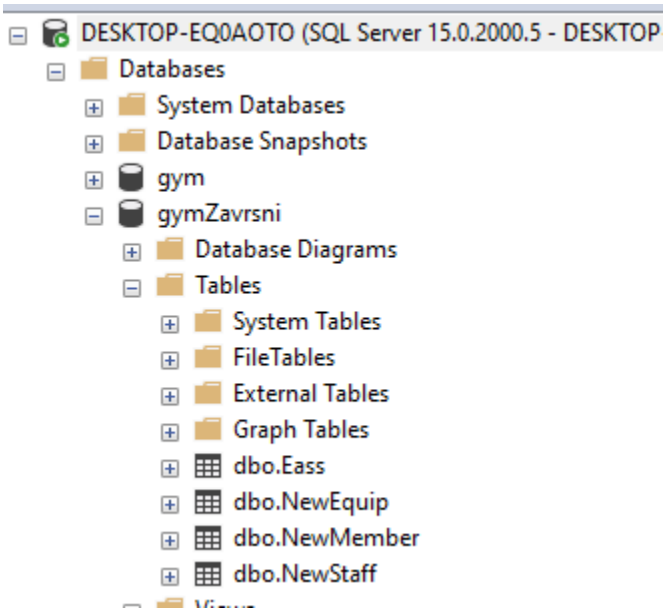
Slika 21. Programsko rješenje - uređivanje člana

Posljednji proces koji će biti opisan je prijava u aplikaciju. Nakon kreiranja forme za prijavu, u datoteku program.cs u Main metodu je potrebno upisati sljedeći kod “Application.Run(new Login());”. Taj kod omogućava da se pri pokretanju aplikacije prvo prikaže forma te nakon upisa točnih podataka klikom na “Prijava” korisnika odvede na glavni izbornik. Programski kod koji stvara clickEvent i vrši provjeru pristupnih podataka prikazan je na slici 22.

```
private void btnLogin_Click(object sender, EventArgs e)
{
    if (txtBoxUserName.Text == "admin" && txtBoxPassword.Text == "admin")
    {
        Form1 fm = new Form1();
        fm.Show();
        this.Hide();
    }
    else
    {
        MessageBox.Show("Incorret user name or Password", "Error", MessageBoxButtons.OK, MessageBoxIcon.Warning);
    }
}
```

Slika 22. Programsko rješenje – prijava

Procesi dodavanja, uređivanja, pretrage i brisanja osoblja, opreme i vanjskih suradnika su identični uz naravno drugačija imena tablica i atributa. Baza podataka "gymZavrsni" je izrađena u alatu Microsoft SQL Server Management Studio (slika 23).



Slika 23. Programsko rješenje - baza podataka

5. Zaključak

U radu je izložen sustav za evidenciju članova teretane kao temeljne svrhe, no dodane su mogućnosti evidentiranja osoblja, opreme i vanjskih suradnika. Logika aplikacije je prezentirana kroz 5 UML dijagrama: dijagram slučajeva korištenja, dijagram klasa i objekata, sekvencijski dijagram te dijagram aktivnosti. Nacrt kako bi sučelje aplikacije trebalo izgledati te ostale skice su izrađeni korištenjem alata Balsamiq Wireframes, po mnogima najbržim i najboljim low-fidelity alatom za izradu wireframe-ova. Slijedeći upute koje su dali dijagrami i skice, napravljena je funkcionalna windows forms desktop aplikacija uz sitne preinake. Koristio se C# objektno orijentirani programski jezik, a razvojno okruženje Microsoft Visual Studio. Uzevši u obzir da su se pri izradi aplikacije koristili C# i Visual Studio odlučeno je da se se za izradu baze podataka također koristiti Microsoftov alat, Microsoft SQL Server Management Studio. Iako je programsko rješenje kompletno, postoji prostora za poboljšanja poput uvođenja opcije promjene pristupnih podataka, metoda koje će obavijestiti administratora ukoliko je nekome istekla članarina ili mogućnost upisa radnih sati pa da aplikacija na kraju mjeseca izračuna plaću ili rad studenta veže uz studentski ugovor.

Literatura

1. All You Need to Know About UML Diagrams
   https://tallyfy.com/uml-diagram/
   [Pristupljeno 10.5.2022.]
2. Balsamiq Wireframes
   https://balsamiq.com/wireframes/
   [Pristupljeno 20.5.2022.]
3. C# programming guide
   https://docs.microsoft.com/en-us/dotnet/csharp/programming-guide/
   [Pristupljeno 20.5.2022.]
4. C# Programming Tutorials: Beginners 05 Windows Forms and Event Handlers
   https://www.youtube.com/watch?v=W6vJ_c9Mt6A
   [Pristupljeno 20.5.2022.]
5. Dukić, B.: Baze podataka i poslovni procesi, Praktikum, Ekonomski fakultet u Osijeku, Osijek 2011.
   [Pristupljeno 10.6.2022.]
6. Kovačević, Ž., Slamić, M., Stojanović, A. (2022). Objektno orijentirano programiranje. Zagreb
   https://www.bib.irb.hr/979189/download/979189.Objektno_orijentirano_programiranje.pdf
   [Pristupljeno 1.6.2022.]
7. Miles, R., Hamilton, K. (2006). Learning UML 2.0. Sebastopol: O'Reilly Media
   https://www.oreilly.com/library/view/learning-uml-20/0596009828/
   [Pristupljeno 10.5.2022.]
8. Mesarić, J., Šebalj, D.: Nastavni materijali za predmet Oblikovanje i implementacija informacijskih sustava, EFOS, ak. god. 2018/19.
   [Pristupljeno 10.5.2022.]

Popis Slika

Popis tablica