UNIVERZITET U ZENICI
Politehnički fakultet
Softversko inženjerstvo
Razvoj informacijskih sistema

Služba za imovinsko- pravne odnose

Projektni zadatak

Student: **Edina Kaknjo**

Profesor: Doc. dr. sc. Denis Čeke

Zenica, akademska 2023/2024. godina

SADRŽAJ

1. UVOD

U današnjem digitalnom dobu, primjena informacionih sistema postaje ključna u optimizaciji različitih sektora, a posebno u oblasti javnih službi. Kao odgovor na potrebu efikasnijeg upravljanja imovinsko-pravnim odnosima, razvila sam informacioni sistem koji pruža rješenje za ovu kompleksnu oblast. Ovaj sistem ima za cilj poboljšati pristup relevantnim informacijama, pojednostaviti procese, te podržati rad zaposlenih u općinskim uredima i omogućiti transparentniju komunikaciju s građanima.

Kroz razvoj sistema, istražujem ključne zahtjeve imovinsko-pravnog ureda, posebno naglašavajući potrebu za sigurnom autentifikacijom, ali i jednostavnim pristupom za građane koji žele provjeriti informacije o vlasništvu ili parcelama. Analiziram i mehanizme raspoređivanja poslova unutar ureda, pružajući šefu službe mogućnost učinkovitog praćenja rada službenika, te izvršavanja analiza i statistika.

Kroz različite funkcionalnosti, sistem omogućuje preciznu autentifikaciju korisnika, s posebnim pristupom za zaposlene u uredu, građane i administratore. Počevši od autentifikacije na početnoj stranici, prilagođene različitim vrstama korisnika. Službenici ovog sistema, koji su ključni korisnici, imaju svoje specifične ovlasti, omogućujući im da efikasno obavljaju zadatke poput modificiranja vlasništva nad parcelama. Građani imaju ograničen, ali informativan pristup sistemu,bez ikakvog log-ina. Dok administratori održavaju sistem i dodjeljuju uloge korisnicima. U pogledu autorizacije, naglašavam nužnost ograničenja pristupa određenim dijelovima sistema kako bi se očuvala sigurnost i integritet podataka. Tehnički zahtjevi izrade sistema osiguravaju funkcionalnost, ali i prilagodljivost sistema prema specifičnostima svake općine.

Ovaj informacioni sistem ne samo da olakšava rad službenika, već i pruža transparentnost i pristupačnost građanima. Kroz integraciju modernih tehnologija, ovaj sistem pruža efikasno rješenje za sve izazove vezane uz imovinsko-pravne odnose u lokalnim zajednicama.

2. PREGLED PLANA PROJEKTA

	Name	Duration	Start	Finish	Predecessors	Resource Names
1	**Planiranje i istaživanje**	**15 days?**	**3/4/24 8:00 AM**	**3/18/24 5:00 PM**		**Edina Kaknjo**
2	Prikupljanje zahtjeva	10 days?	3/4/24 8:00 AM	3/13/24 5:00 PM		
3	Planiranje rokova	2 days?	3/13/24 8:00 AM	3/14/24 5:00 PM		
4	Proracun budzeta	2 days?	3/17/24 8:00 AM	3/18/24 5:00 PM		
5	**Dokumentacija sistema**	**2 days?**	**3/18/24 8:00 AM**	**3/19/24 5:00 PM**		**Edina Kaknjo**
6	Opis sistema	2 days?	3/18/24 8:00 AM	3/19/24 5:00 PM		
7	**Dizajniranje sistema**	**3 days?**	**3/19/24 8:00 AM**	**3/21/24 5:00 PM**		**Edina Kaknjo**
8	Crtanje dijagrama	2 days?	3/19/24 8:00 AM	3/20/24 5:00 PM		
9	Kreiranje mockups	1 day?	3/21/24 8:00 AM	3/21/24 5:00 PM		
10	**Dizajniranje baze podataka**	**2 days?**	**3/21/24 8:00 AM**	**3/22/24 5:00 PM**		**Edina Kaknjo**
11	Crtanje dijagrama	1 day?	3/21/24 8:00 AM	3/21/24 5:00 PM		
12	Kreiranje relacijskog modela	1 day?	3/22/24 8:00 AM	3/22/24 5:00 PM		
13	**Izrada sistema**	**24 days?**	**3/23/24 8:00 AM**	**4/19/24 5:00 PM**		**Edina Kaknjo**
14	Programiranje(izrada funkcionalnosti)	15 days?	3/23/24 8:00 AM	4/8/24 5:00 PM		
15	Unos podataka u bazu	2 days?	4/4/24 8:00 AM	4/5/24 5:00 PM		
16	Implementacija baze u kodu	5 days?	4/5/24 8:00 AM	4/11/24 5:00 PM		
17	Testiranje sistema	1 day?	4/12/24 8:00 AM	4/12/24 5:00 PM		
18	Rad na frontendu	5 days?	4/13/24 8:00 AM	4/19/24 5:00 PM		
19	Testiranje sistema	24 days?	3/23/24 8:00 AM	4/25/24 5:00 PM		Edina Kaknjo
20	Odrzavanje sistema	364 days?	4/19/24 8:00 AM	9/10/25 5:00 PM		Edina Kaknjo

Slika 1. Prikaz plana projekta

Na slici 1. se vidi plan projekta koji se sastoji od 7 većih faza.

Prva faza je faza ‘Planiranje i istraživanje’ koja se sastoji od prikupljanja zahtjeva od korisnika, planiranja rokova i shodno tome na kraju računanje budžeta za projekat na osnovu prethodno obavljenih aktivnosti.

Sljedeća faza je faza ‘Dokumentacija sistema’ u kojoj se vrši opis sistema. Ova faza je bitna za planiranje dizajna i dublje shvatanje projekta.

Faza ‘Dizajna sistema’ se sastoji iz crtanja use case dijagrama i dijagrama aktivnosti i kreiranje mockups-a sistema.

Faza ‘Dizajniranje baze podataka’ se sastoji od crtanja dijagrama klasa i komponenti i kreiranja relacijskog modela. ER dijagram ćemo generisati iz gotove MySQL baze pri završetku projekta. Gore navedeni dijagrami su dovoljni za kreiranje sistema.

Uspješnim završetkom prethodne faze nastupa faza implementacije ili ‘Izrada sistema’ u kojoj se vrši kodiranje dizajniranog sistema, te usporedno traje faza ‘Testiranje sistema’ kako bi se mogle uočiti greške i izvršiti isparvke na vrijeme.

Faza izrade sistema se sastoji od programiranja, tj kodiranja, unošenja podataka u bazu podataka, implementacije baze podataka u kod, tj povezivanje baze podataka sa funkcionalnostima sistema, jednog velikog testitanja svih funkcionalnosti sistema, te tek onda

nastupa rad na frontendu. Tj tek onda nastupa rad na poboljšanju izgleda informacionog sistema.

Nakon uspješne implementacije i testiranja sistema, po ugovoru, nastupa period faze ‘Održavanje sistema’ i uklanjanja naknadno otkrivenih greški i dokumentovanje izmjena u periodu od godinu dana (365 dana).

3. PRIKUPLJANJE I ODREĐIVANJE KORISNIČKIH ZAHTJEVA

U službi imovinsko-pravnih odnosa imamo šefa službe i obične radnike. Po hijararhiji radnici. tj službenici, su svi na istom nivou, ispod šefa. Još jedna vrsta ljudi kojima je bitan pristup podacima ove službe jesu obični građani, te i njih uzimamo u obzir.

Autentifikacija predstavlja prvi korak, pružajući različite razine pristupa za službenike, građane i administrator (one koji održavaju sistem i šefa službe).

Admin je šef službe. Nakon što se šef loguje sa svojim dodjeljenim user-om i šifrom-njegov glavni posao je -raspoređivanje poslova. Ovo omogućava šefu službe unos predmeta(po id-broju predmeta koji se nalazi na fizičkom dokumentu), te zatim dodjeljivanje predmeta nekom od službenika Službenik je dodan od strane administratora (šefa).

Službenik treba raditi izmjene i ‘rješavati’ predmet. Šef također nakon službenikovog završenog rada na predmetu, treba taj predmet pregledati, pa tek onda ‘ostaviti ’ u arhivu. Šef može raditi izmjene vlasništva na predmetu. Šef također ima pristup arhivi (svim predmetima i izmjenama urađenim nad njima).

Službenici imaju privilegije za modificiranje podataka o predmetu, te pristup arhivi. Ukoliko se predmet tek unosi u sistem službenik unosi podatke poput lokacije, dimenzija i vlasnika (ili udjela u vlasništvu od strane više vlasnika). Nakon unesenog predmeta (ili gledajući onaj koji je već u sistemu) službenik treba biti u mogućnosti izmjeniti predmet dok ga ne učini ‘završenim’. .Nakon završetka rada na predmetu, službenik predmet šalje na pregled nazad šefu.

Šef pregleda završene preglede, radi izmjene ukoliko su potrebne, te potom predmet arhivira. Tek nakon što je predmet arhiviran dostupan je u pretraživanju po arhivi.

Obični građani, čiji pristup sistemu odobravamo bez opcije ‘Login’ imaju ograničen pristup, omogućujući im pregled informacija o parcelama. Dakle građani ukucavajući id parcele, tj broj parcele koji imaju na nekom fizičkom dokumentu, mogu doći do informacija o vlasništvu, lokaciji, veličini parcele, itd.

Arhiviranje podataka i analiza omogućuju zaposlenicima pregled vlastitih aktivnosti, analizu podataka, čime se poboljšava efikasnost rada ureda.

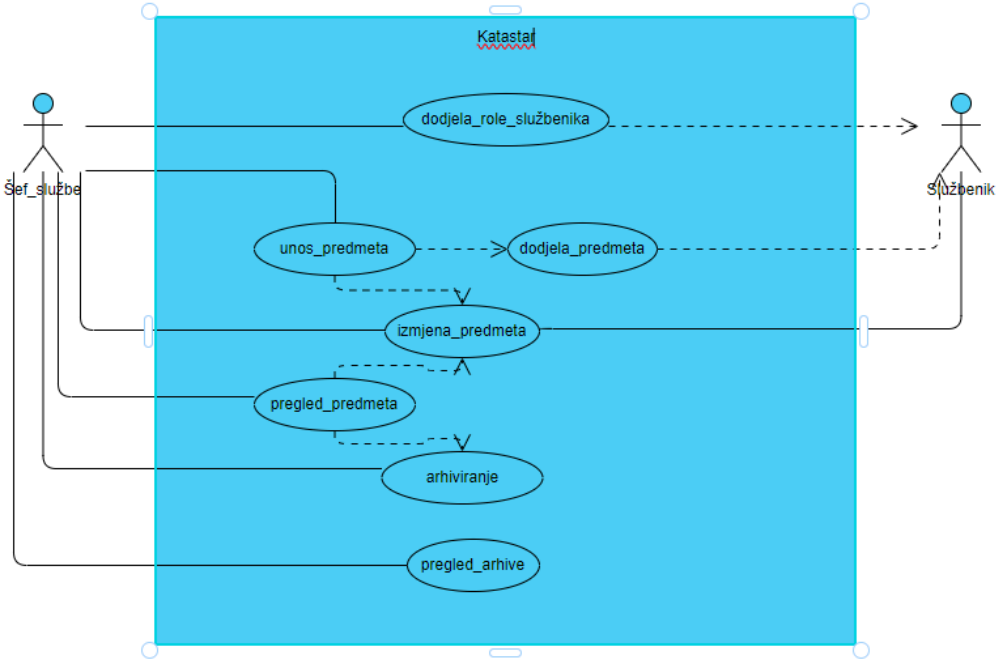
Kroz ove funkcionalnosti, informacioni sistem postaje ključan alat za evidenciju, upravljanje i olakšavanje procesa u općinskim uredima, pridonoseći transparentnosti i bržem rješavanju imovinsko-pravnih pitanja.

4. SISTEMSKA ANALIZA PROBLEMA

U narednoj oblasti projekta će biti prikazani i opisani pojedini dijagrami koji prikazuju dijelove i funkcionalnosti sistema.

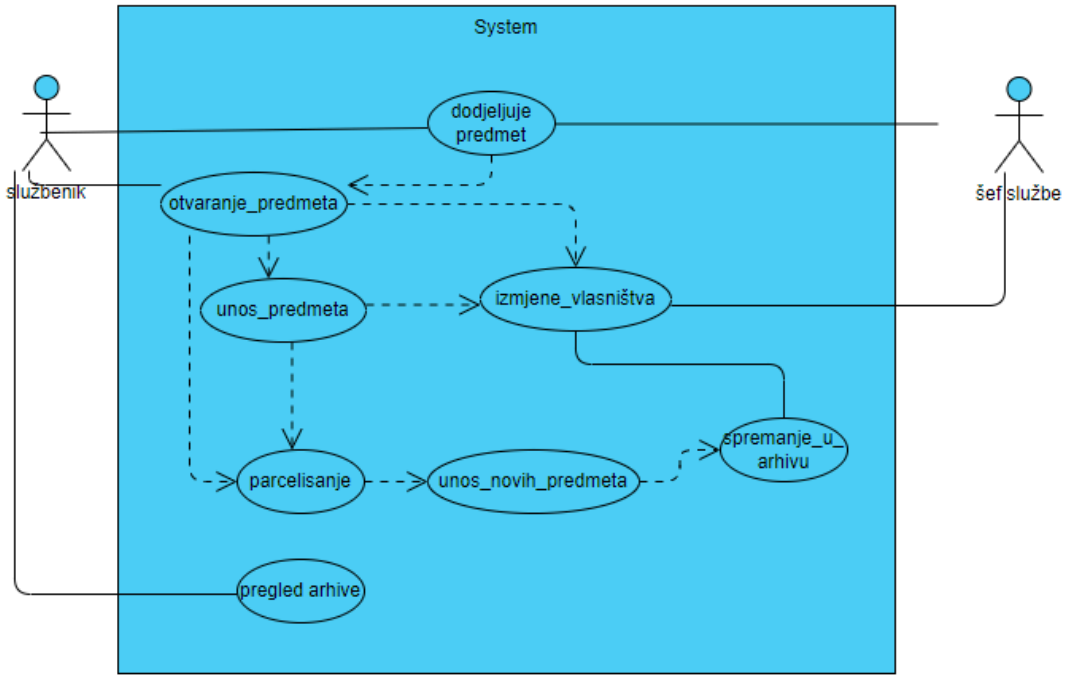
4.1. Use case dijagrami

Use case dijagrami su vrsta dijagrama u analizi softvera koji prikazuju interakcije između korisnika i sistema te opisuju funkcionalnosti sistema iz perspektive korisnika. Koriste se kako bi se razumjelo ponašanje sistema i identificirali glavni slučajevi korištenja. U ovom poglavlju ću prikazati par use case dijagrama koji pokazuju ponašanje korisnika i interakcije između korisnika prilikom rada u sistemu.



Slika 2. Prikaz šef službe-službenik dijagram

Na slici 2. je prikazan odnos šef službe-službenik. Kao što je navedeno u 3. poglavlju šefov glavni posao je -raspoređivanje poslova. Šef službe ima mogućnost unosa predmeta po id-broju predmeta koji se nalazi na fizičkom dokumentu , te zatim dodjeljivanje predmeta nekom od službenika Službenik je dodan od strane nekog od administratora- tvorca sistema ili šefa. Dakle šef službe ima i ovu mogućnost. Nakon službenikovog završenog rada na predmetu šef predmet pregleda i ukoliko ne želi izvršiti nikakve izmjene- sprema ga u arhivu. Šef može raditi izmjene vlasništva na predmetu. On također ima pristup arhivi, tj svim predmetima i izmjenama urađenim nad njima.



Slika 3. Prikaz službenik-šef službe dijagrama

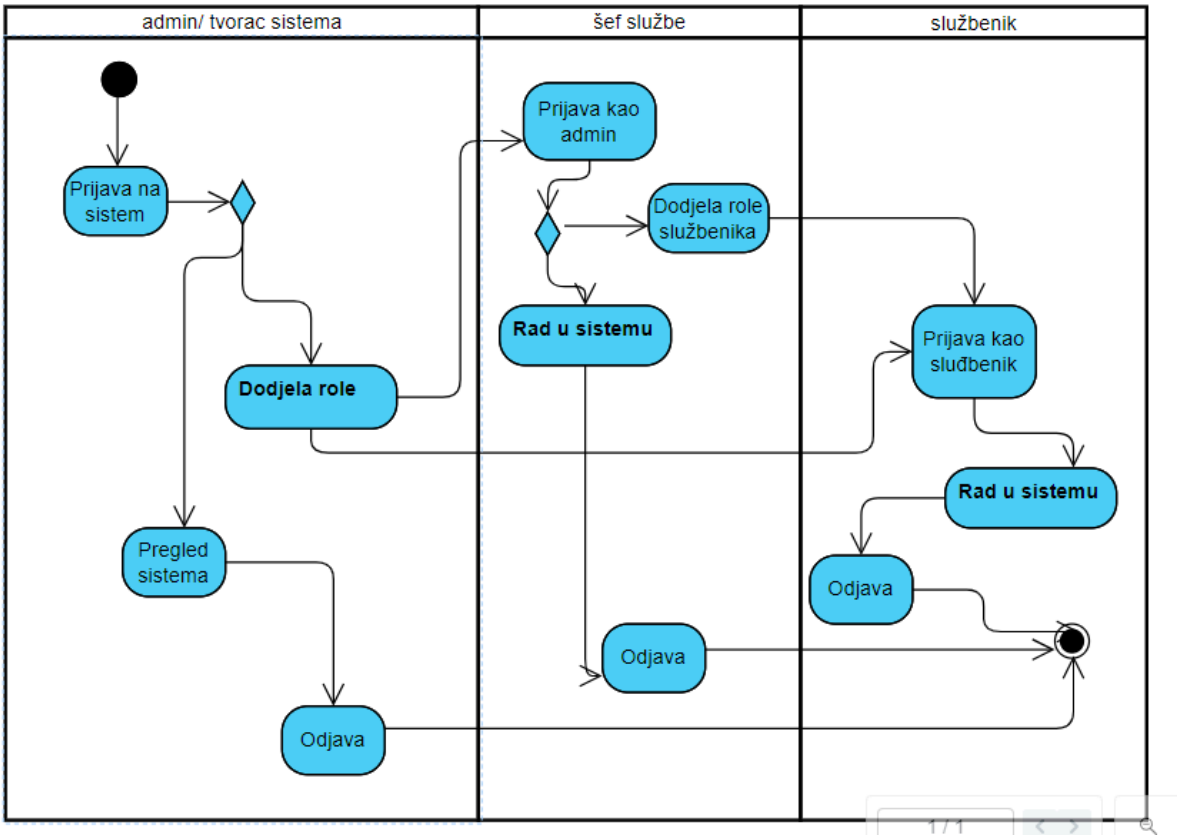
Na slici 3. je prikazana slična stvar kao i na slici 2. s većim fokusom na rad službenika i njegove mogućnosti u ovom sistemu. Razmjena informacija i poslovi koji se dešavaju između službenika i šefa službe su već objašnjeni iznad (ispod slike 2.) Te ću se u ovom dijelu bazirati samo na pojašnjenje uloge službenika.

Službenici imaju privilegije za modificiranje podataka o predmetu, te pristup arhivi. Ukoliko se predmet tek unosi u sistem službenik unosi podatke poput lokacije, dimenzija i vlasnika (ili udjela u vlasništvu od strane više vlasnika). Nakon unesenog predmeta (ili gledajući onaj koji je već u sistemu) službenik treba biti u mogućnosti samo moći izmjeniti vlasnika/vlasnike. Nakon završetka rada na predmetu, službenik predmet šalje na pregled nazad šefu,koji predmet arhivira ukoliko je sve ispravno.

4.2. Dijagrami aktivnosti

Dijagram aktivnosti je vrsta dijagrama u UML-u koji opisuje tok aktivnosti ili procesa u softverskom sistemu, koristeći grafičke elemente poput akcija, odluka i tokova. Koristi se za modeliranje složenih procesa i vizualizaciju slijeda aktivnosti u softverskim aplikacijama.

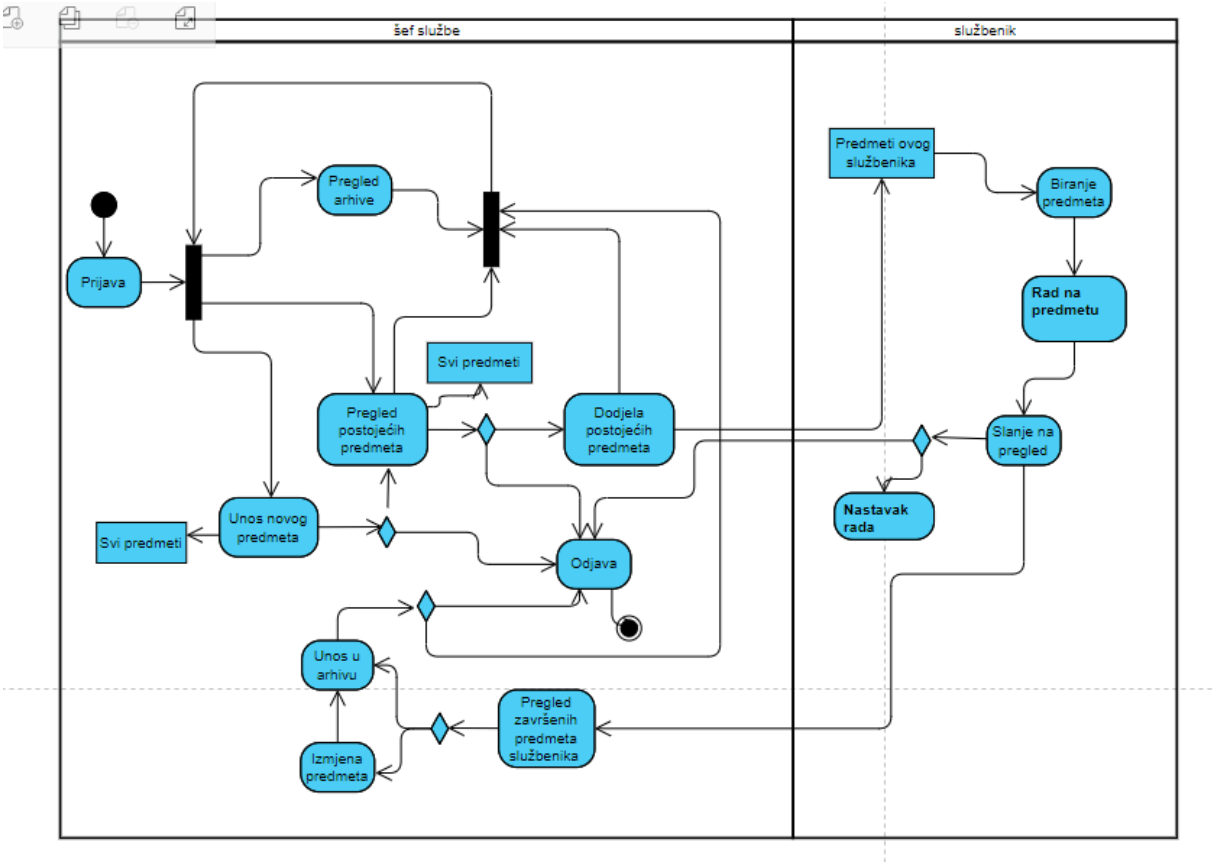
U ovom poglavlju ću prikazati par dijagrama aktivnosti koji se odnose na najsloženije procese unutar ovog sistema.



Slika 4. Prikaz procesa dodjele rola

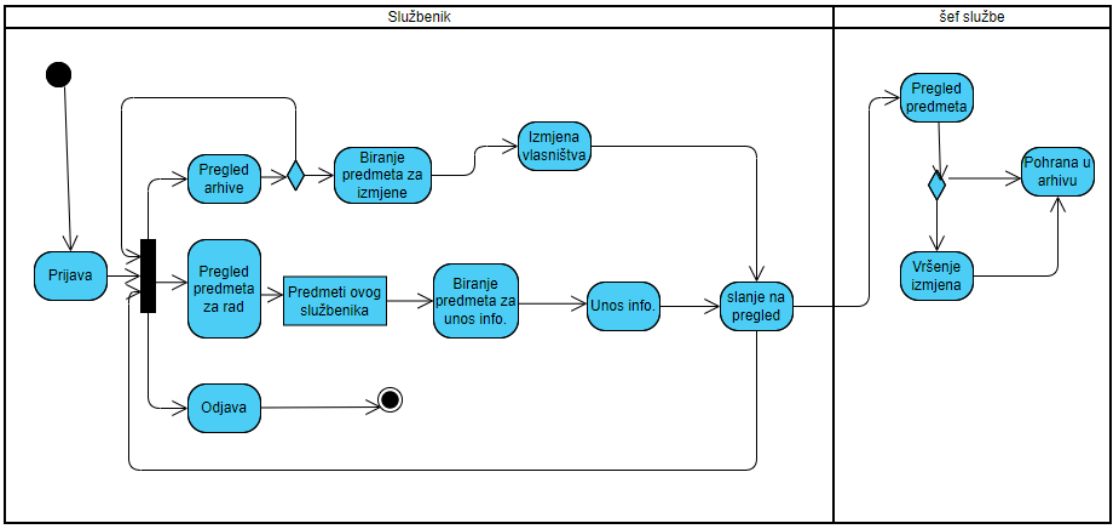
Na slici iznad je prikazan proces dodjeljivanja role . Tvorac sistema dodjeljuje rolu admina šefu službe te se isti može prijaviti na sistem kao administrator. Prijavom kao administrator dobija mogućnost dodjeljivanja iste role nekom drugom, te dodjeljivanja role službenika svojim službenicima..

Dakle, nakon prijave i dodjela rola, administrator-šef može nastaviti raditi u sistemu ili dodjeliti rolu službeniku ili odjaviti se, a službenik može da se odjavi ili radi u sistemu. Pod “radom u sistemu” se podrazumjeva niz funkcionalnosti koje će biti detaljnije prikazane u nekim drugim dijagramima.



Slika 5. Prikaz rada šefa službe u sistemu

Funkcionalnosti ovog aktera su već pojašnjene u prethodnim poglavljima te mislim da nema potrebe u detalje prelaziti. Bitno je napomenuti da su u dijagramu aktivnosti iznad- object nodes korišteni kao prikazi dijelova baze podataka kojima ovi akteri imaju pristup u vremenu izvršavanja određene funkcionalnosti. To jest- za funkcionalnost iznad- kojim dijelovima baze podataka akteri pristupaju.



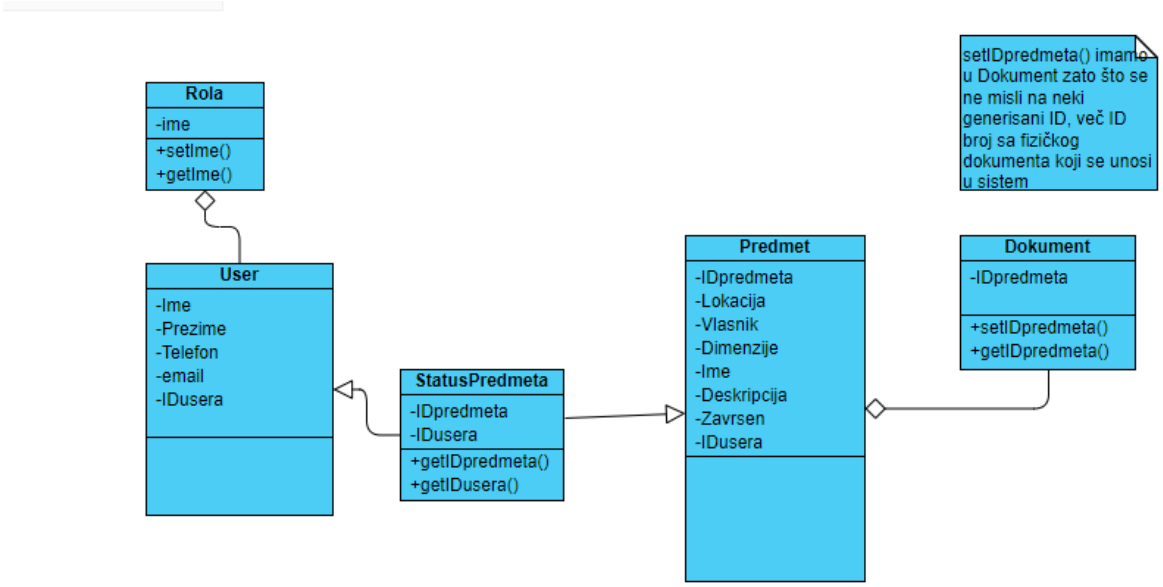
Slika 6. Prikaz rada službenika u sistemu

Na slici iznad je prikazan rad službenika u sistemu koji je detaljno opisan u prethodnim poglavljima i detaljno prikazan na slici iznad.

4.3. Dijagram klasa

Dijagram klasa je vrsta dijagrama u UML-u koji prikazuje strukturu softverskog sistema, identificirajući klase, njihove atribute i odnose među njima, čime omogućuje vizualizaciju organizacije podataka i logičkih veza unutar aplikacije. Koristi se za modeliranje objektno orijentiranih programa radi boljeg razumijevanja njihove arhitekture i dizajna

Na slici ispod je prikazan dijagram klasa ovog sistema.



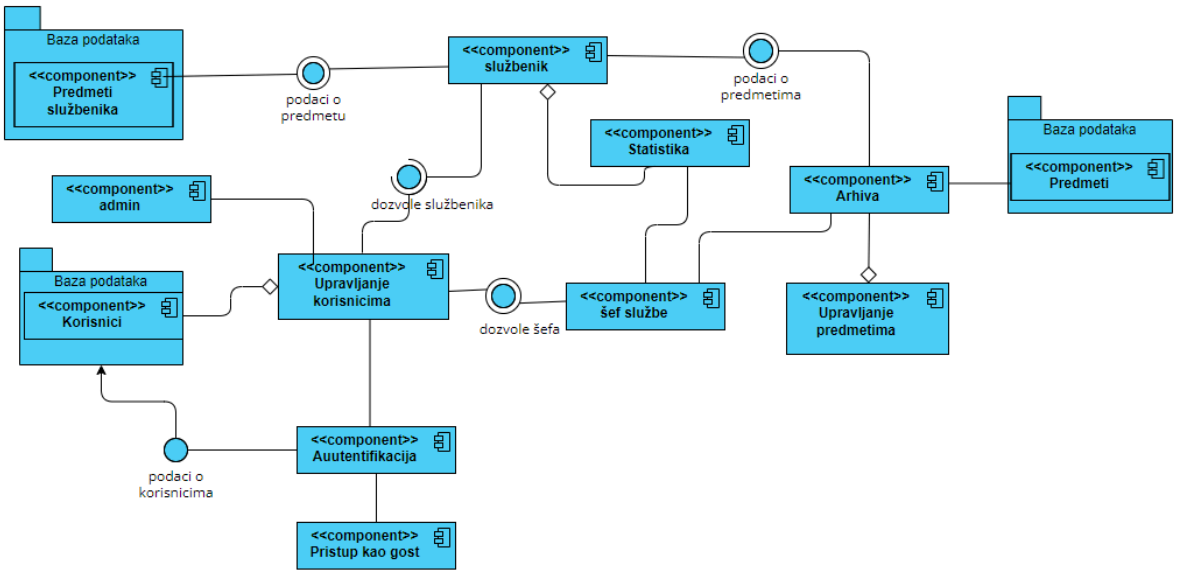
Slika 7. Dijagram klasa sistema

Na slici iznad vidimo da sistem ima 6 klasa, a to su: User (korisnik), Predmet, Rola, Dokument, StatusPredmeta i StatistikaUsera. StatusPredmeta i StatistikaUsera ne postoje bez klasa User i Predmet, te s obzirom na to su na iste klase povezane vezom agregacije. Veza agregacije označava odnos gdje jedna klasa predstavlja cjelinu, dok druge klase predstavljaju dijelove te cjeline. Klase dijelova mogu postojati neovisno o cjelini, ali cjelina ne može postojati bez barem jednog od svojih dijelova. U ovom slučaju- npr. Predmet je cjelina, a njen dio (u sistemu, ne u fizičkom svijetu) je klasa Dokument i ne može postojati bez klase Predmet.Klasa User i StatistikaUsera su povezane vezom kompozicije. Ova veza je slična agregaciji ali ne ista. Kompozicija je tip veze između klasa u dijagramu klasa koji označava "čvršću" vezu gdje jedna klasa (cjelina) posjeduje i kontrolira druge klase (dijelove). Dijelovi ne mogu postojati neovisno o cjelini te se obično koristi za modeliranje odnosa gdje je cjelina odgovorna za stvaranje, upravljanje i brisanje dijelova. Veza se označava punim rombom koji pokazuje prema cjelini. Kompozicija implicira da dijelovi ne mogu postojati neovisno o cjelini, dok agregacija dopušta da dijelovi postoje i bez cjeline.

Ostale klase su povezane asocijacijom, dakle mogu postojati i funkcionisati jedna bez drudge u ovom sistemu. Skupa sa prethodnim dijagramima, smatram da su veze između klasa detaljno pojašnjene.

4.4. Dijagram komponenti

Dijagram komponenti je vrsta dijagrama u UML-u koji prikazuje strukturu i međusobne odnose između fizičkih komponenti softverskog sistema, poput biblioteka, modula ili izvršnih datoteka, čime omogućuje vizualizaciju arhitekture sistema na razini implementacije. Koristi se za modeliranje složenih softverskih sistema i identifikaciju komponenti te njihovih međusobnih veza. Na slici ispod je prikazan dijagram komponenti ovog sistema.



Slika 8. Dijagram komponenti sistema

Dijagram uključuje različite komponente kao što su "Baza podataka", "Predmeti", "podaci o službenicima", "podaci o predmetima" i "Statistika"

Dijagram dodatno objašnjava odnose između različitih komponenti, poput "Upravljanja korisnicima", "Arhive" i "Autentifikacije", što ukazuje na kompleksan sistem koji se bavi procesima autentifikacije korisnika, arhiviranja podataka i upravljanja korisnicima. Elementi poput "dozvola službenika", "dozvola šefa" i "Pristup kao gost" impliciraju hijerarhijski mehanizam kontrole pristupa unutar sistema, osiguravajući sigurnost podataka i odgovarajuće nivoe autorizacije za različite korisnike.

Posebno, dijagram koristi ponavljajuće simbole označene sa "<< komponenta >>" kako bi označio modularne dijelove sistema, sugerirajući dizajn koji naglašava ponovnu upotrebljivost i fleksibilnost. Uključivanje uloga "šef službe" i "admin" ističe administrativnu strukturu sistema, koja zadovoljava različite operativne potrebe unutar organizacije. Ovi odnosi jasno definiraju kako različite komponente sistema međusobno interagiraju surađuju u ostvarivanju funkcionalnosti sistema.

5. SISTEMSKI DIZAJN

5.1. ER dijagram

ER dijagram sistema Katastar predstavlja strukturu podataka i veza među entitetima u bazi podataka, analogno preslikavanju klasa u entitete. Svaki entitet ima svoje specifične atribute i veze sa drugim entitetima kako bi se omogućila funkcionalnost sistema.

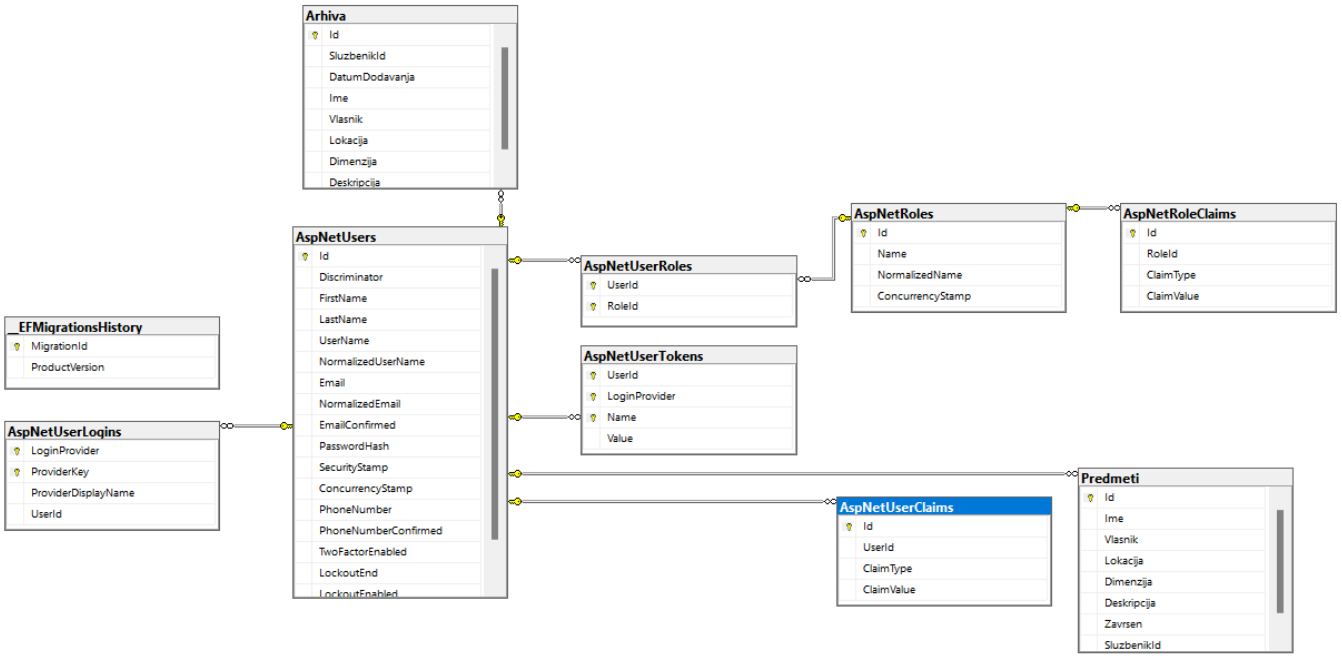
ER dijagram je generisan iz SQL koda. Ovaj SQL kod definiše migraciju za bazu podataka koja se koristi u sistemu Katastar. Evo opisa svake tabele i veza među njima:

- AspNetRoles: Ova tabela čuva informacije o ulogama korisnika. Svaka uloga ima jedinstveni identifikator (Id), ime (Name), normalizovano ime (NormalizedName) i oznaku konkurentnosti (ConcurrencyStamp).
- AspNetUsers: Tabela koja sadrži informacije o korisnicima sistema. Osim osnovnih informacija o korisniku (Ime, Prezime, Korisničko Ime, Email itd.), ova tabela takođe ima polje Discriminator koje se koristi za razlikovanje različitih tipova korisnika.
- AspNetRoleClaims: Ova tabela čuva tvrdnje koje su povezane sa određenom ulogom. Tvrdnje su često korišćene za prava pristupa ili dozvole.
- Arhiva: Tabela koja sadrži podatke o arhiviranim predmetima. Svaki predmet ima jedinstveni identifikator (Id), informacije o službeniku koji je arhivirao predmet (SluzbenikId), datum dodavanja (DatumDodavanja), ime predmeta (Ime), vlasnika predmeta (Vlasnik), lokaciju (Lokacija), dimenziju (Dimenzija) i opis (Deskripcija).
- AspNetUserClaims: Tabela koja čuva tvrdnje koje su povezane sa određenim korisnikom.
- AspNetUserLogins: Ova tabela čuva informacije o prijavljivanju korisnika pomoću različitih provajdera za prijavu, kao što su Facebook ili Google.
- AspNetUserRoles: Tabela koja uspostavlja vezu između korisnika i njihovih uloga. Svaki red u ovoj tabeli predstavlja jednu vezu između korisnika (UserId) i uloge (RoleId).
- AspNetUserTokens: Ova tabela čuva informacije o tokenima koje su generisali provajderi prijave kako bi omogućili autentifikaciju korisnika.
- Predmeti: Tabela koja čuva informacije o predmetima u sistemu. Svaki predmet ima jedinstveni identifikator (Id), ime (Ime), vlasnika (Vlasnik), lokaciju (Lokacija), dimenziju (Dimenzija), opis (Deskripcija) i informaciju o tome da li je završen (Zavrsen). Takođe, veza je uspostavljena sa korisnikom (SluzbenikId) koji je zadužen za taj predmet.

Ovaj SQL kod opisuje strukturu baze podataka za sistem Katastar i omogućava definisanje entiteta i veza među njima.

Kroz ovu strukturu podataka, sistem Katastar omogućava efikasno upravljanje informacijama o parcelama i upravljanje osobljem. Ovakva organizacija podataka omogućava transparentno vođenje evidencije i olakšava procese u okviru službe imovinsko-pravnih odnosa.

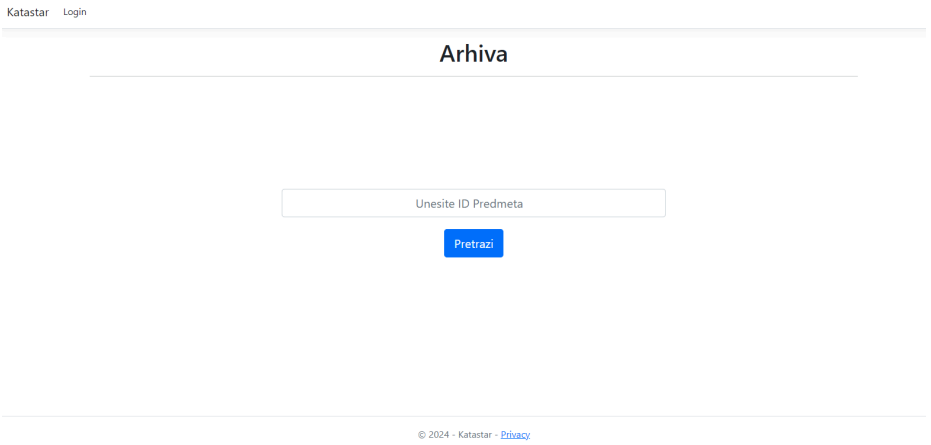
Na slici ispod je prikazan ER dijagram.



Slika 9. ER Dijagram sistema

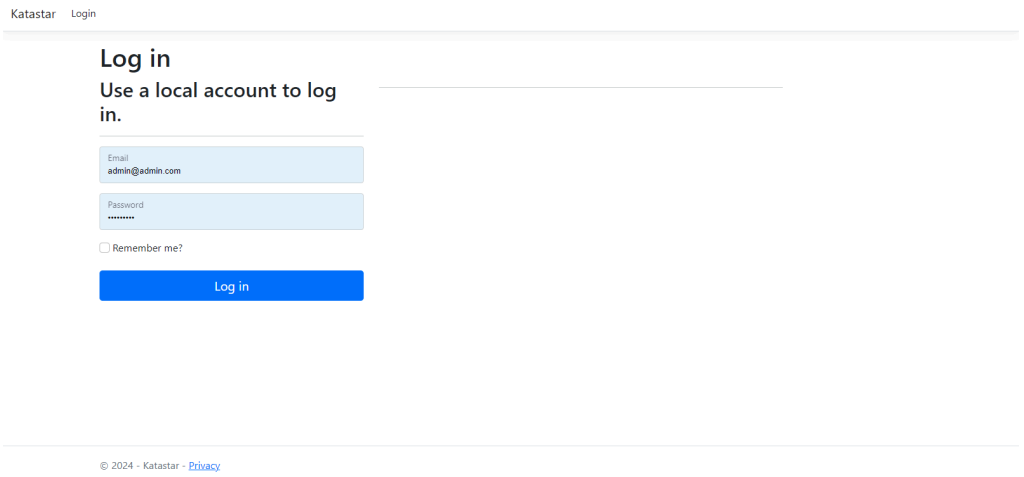
5.2. Dizajn interfejsa, formi, izvještaja i dijaloga

U ovom poglavlju ću prikazati kako izgleda kretanje usera kroz aplikaciju.



Slika 10. Početna stranica

Nakon klika na Login opciju otvara se naredna stranica:



Slika 11. Login stranica

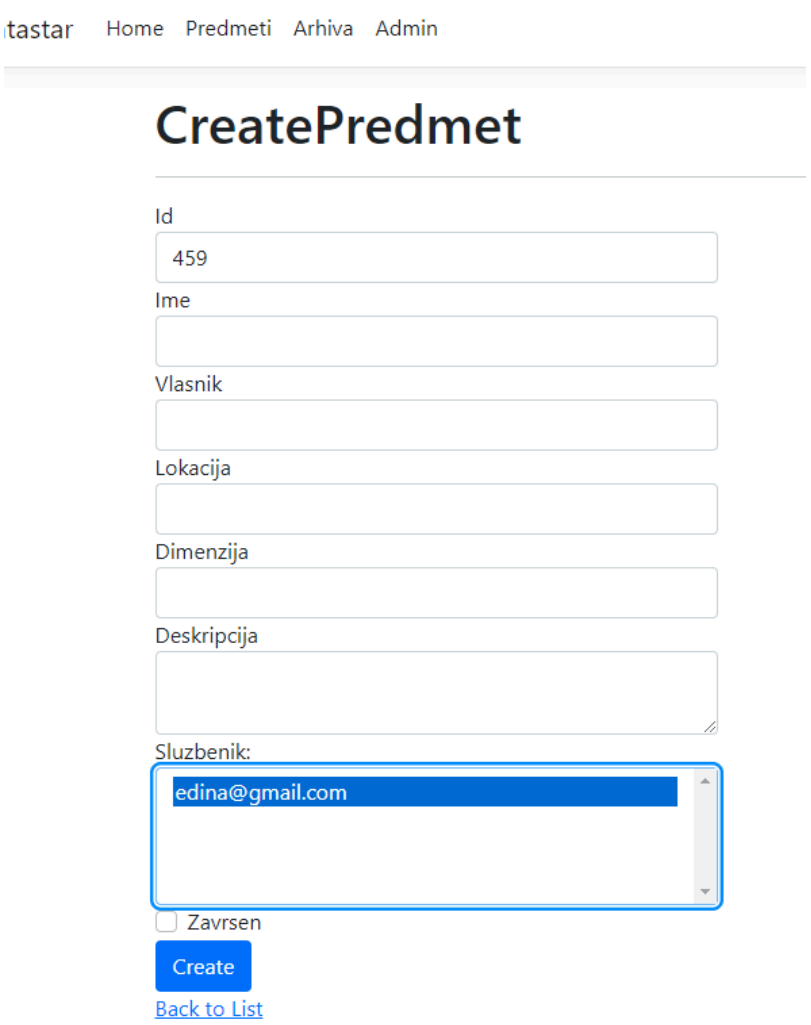
Ispunjeni su admin credentials te se logujemo. Nakon logina nav bar izgleda kao na slici 12.

Slika 12. Nav bar admin korisnika

Klikom na Predmeti u navigaciskoj traci otvara se stranica ispod:

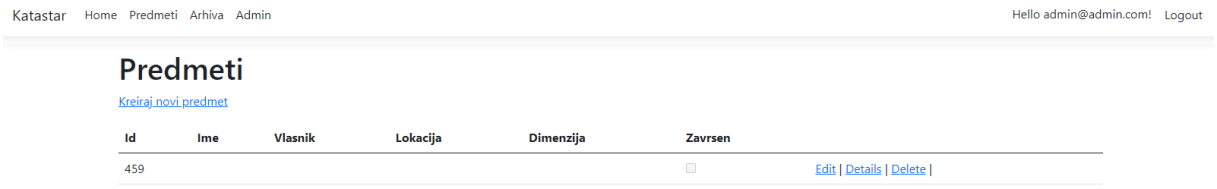
Slika 13. Dostupni predmeti

Na ovoj stranici, admin, tj šef službe- može dodati, tj kreirati novi predmet. Klikom na opciju “Kreiraj novi predmet” otvara se stranica prikazana na slici 14.



Slika 14. Kreiraj novi predmet

Nakon kreiranja predmeta, predmet će se pokazati kao na slici 15.

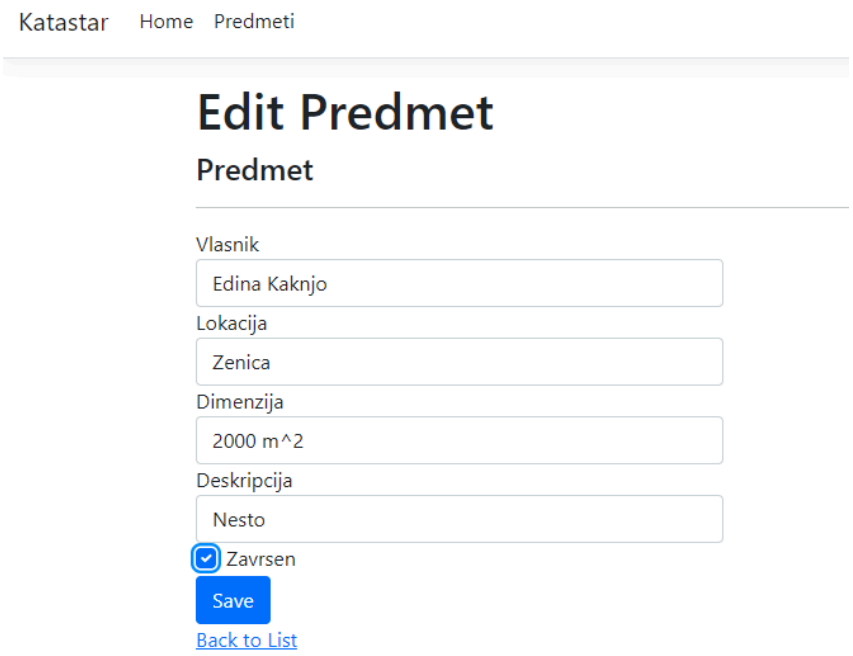


Slika 15. Dostupni predmeti

Na slici 16. vidimo da predmet “459” još nije dostupan u arhivi.

Slika 16. Predmeti dostupni u arhivi

Nakon logouta, login-at ćemo se kao službenik edina, ulaskom na Predmete vidimo svoj zadani predmet 459, te ga uređujemo kao na slici 17.

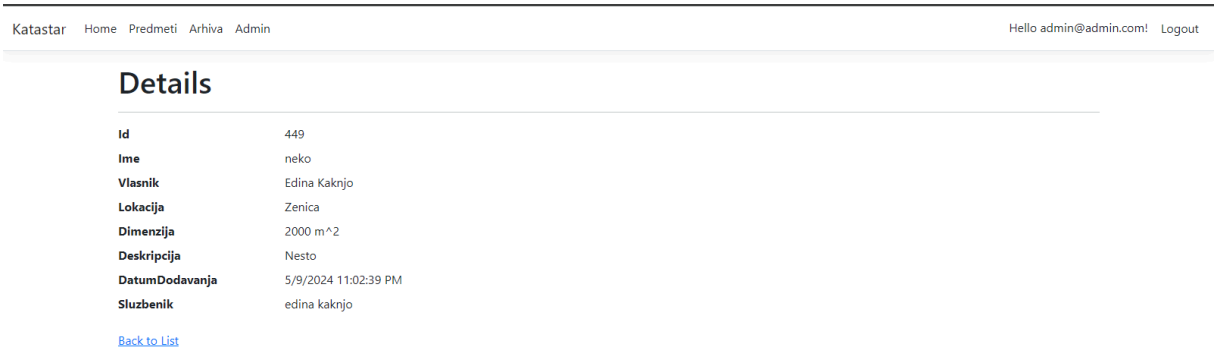


Slika 17. Uređivanje predmeta

Klikom na ‘završeno’ i save, predmet se šalje nazad šefu na pregledavanje i arhiviranje. Nakon toga, izlistavanje predmeta kod šefa izgledea kao na slici ispod:

Slika 18.Predmeti za arhiviranje

Pretraživanjem ID broja i klikanjem na ‘details’ ili ulaskom u arhivu i klikanjem na ‘details’, će ovaj predmet biti dostupan na pregledavanje. Ovo je prikazano na slici 19.



Slika 19. Prikaz detalja unesenog predmeta

6. ARHITEKTURA SISTEMA

U arhitekturi ovog sistema koristi se troslojna struktura radi jednostavnosti i efikasnosti. Ovi slojevi su:

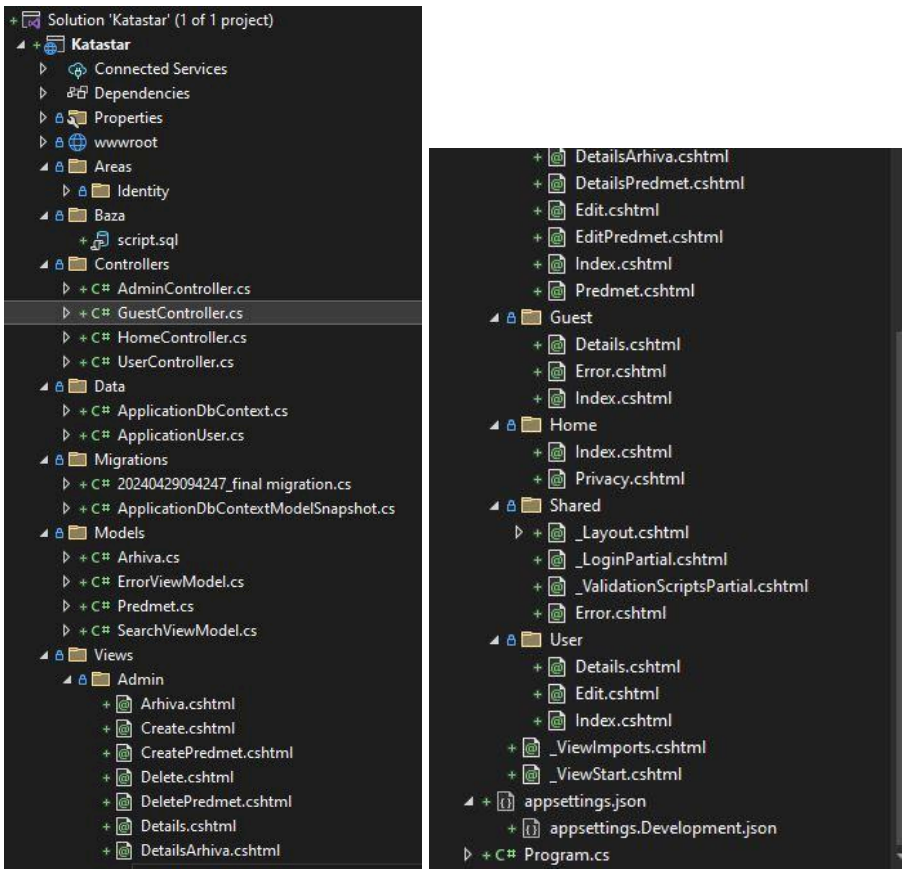
1. **Prezentacijski sloj**: Ovdje se nalazi kod koji formira korisnički interfejs ili frontend aplikacije. Ovaj dio komunicira s poslovnim slojem radi dobijanja podataka za prikaz, ali nije direktno povezan s njim.

2. **Poslovni sloj**: Poslovni sloj se bavi dohvatanjem podataka iz baze i njihovom obradom za prikaz u prezentacijskom sloju. Ovaj sloj komunicira s bazom podataka i korisničkim interfejsom, ali nije zvisan od njih.

3. **Sloj baze podataka**: Ovdje se nalazi sama baza podataka, prikazana kroz ER dijagram. Svi bitni podaci za rad katastra su pohranjeni u ovoj bazi, a poslovni sloj koristi ove podatke za obradu i prikaz u prezentacijskom sloju.

Ovaj odabir arhitekture omogućava odvajanje poslovne logike od grafičkog prikaza interfejsa, što olakšava održavanje, pronalaženje grešaka i izmjene u kodu ili interfejsu bez utjecaja na druge dijelove sistema.

Na slikama ispod je prikazana struktura koda za ovaj informacioni sistem.



Slike 10 i 11. Struktura koda po folderima

Na temelju strukture projekta "Katastar", možemo izvući neke zaključke o arhitekturi sistema:

Korišten je MVC (Model-View-Controller) način kodiranja:

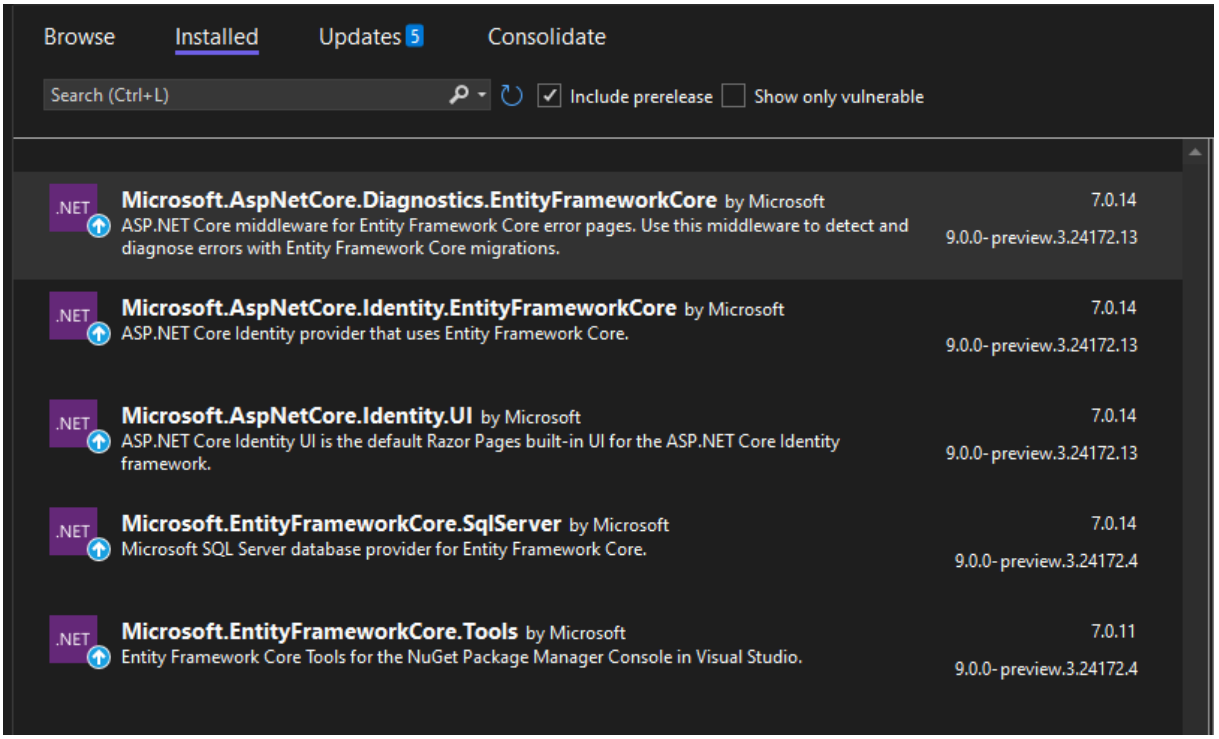
- Struktura foldera "Controllers" sadrži kontrolere koji su odgovorni za obradu zahtjeva i koordinaciju logike aplikacije. Bitno je pomenuti Gost i Korisnik kontrolere: Postoje različiti kontroleri za različite korisničke uloge, poput "GuestController" i "UserController", što sugerira pristup različitim resursima ili funkcionalnostima ovisno o ulozi korisnika.
- Folder "Views" sadrži poglede (HTML datoteke s Razor sintaksom) koji su odgovorni za prikazivanje korisničkog sučelja.Pogledi su organizirani u odgovarajuće foldere prema funkcionalnosti ili korisničkim ulogama. Na primjer, postoje pogledi za "Admin", "Guest", "Home" i "User".
- Folder "Models" sadrži modele koji predstavljaju strukturu podataka aplikacije i poslovnu logiku.

Folder "Areas/Identity" sugerira da aplikacija koristi ASP.NET Core Identity za upravljanje autentikacijom i autorizacijom korisnika.

Što se tiče baze podataka- folder "Data" sadrži DbContext klasu ("ApplicationDbContext.cs") koja predstavlja kontekst baze podataka i upravlja komunikacijom s bazom podataka. Dok, folder "Migrations" sadrži migracijske skripte koje definiraju promjene strukture baze podataka tijekom vremena..

Ovi podaci sugeriraju da je arhitektura sistema bazirana na MVC arhitekturi s primjenom ASP.NET Core Identity za upravljanje autentikacijom i autorizacijom korisnika. Također, struktura projekta ukazuje na pristup različitim resursima ovisno o korisničkim ulogama te na odvojenu obradu baze podataka i poslovne logike aplikacije.

7. TEHNOLOGIJE ZA BACKEND



Slika 12. Instalirani NuGet Packages

Kod je pisan u c# jeziku, a sistem je rađen s ASP.NET Core platformom za razvoj web aplikacija, a za komunikaciju s bazom podataka krišten je Microsoft SQL Server, što je potvrđeno korištenjem Entity Framework Core paketa za pristup i upravljanje podacima u bazi putem .NET okvira.Na slici iznad su prikazani isntalirani paketi. Na osnovu instaliranih NuGet paketa, možemo zaključiti da su korištene sljedeće tehnologije za backend:

- ASP.NET Core Identity: Ovaj paket pruža podršku za upravljanje korisnicima, ulogama i autentikacijom u ASP.NET Core aplikacijama. Koristi se za implementaciju sigurnosnih funkcionalnosti poput registracije korisnika, prijave i upravljanja pravima pristupa.
- Entity Framework Core (EF Core): Ovaj paket pruža ORM (Object-Relational Mapping) alat koji omogućava mapiranje objekata u bazi podataka. Koristi se za interakciju s bazom podataka, uključujući čitanje, pisanje, ažuriranje i brisanje podataka.
- Microsoft.EntityFrameworkCore.SqlServer: Ovaj paket omogućava Entity Framework Core-u da komunicira s Microsoft SQL Server bazom podataka. To je SQL Server specifični pružatelj podataka koji omogućava EF Core-u da izvršava SQL upite na SQL Serveru.
- Microsoft.EntityFrameworkCore.Tools: Ovaj paket pruža alate za Entity Framework Core koji olakšavaju razvoj baze podataka. To uključuje alate za migraciju baze

podataka, generiranje koda iz baze podataka i interakciju s bazom podataka putem konzole za upravljanje paketima u Visual Studiu.

8.TEHNOLOGIJE ZA FRONTEND

Korištena je Razor sintaksa za generisanje HTML-a u ASP.NET Core aplikacijama. Razor stranice su dinamične i omogućavaju ugradnju C# koda direktno unutar HTML-a. Na slici ispod je prikazana stranica Details.cshtml.

```
@model Katastar.Data.ApplicationUser

@{
    ViewData["Title"] = "Details";
}

<h1>Details</h1>

<div>
    <h4>scaffold</h4>
    <hr />
    <dl class="row">
        <dt class = "col-sm-2">
            @Html.DisplayNameFor(model => model.FirstName)
        </dt>
        <dd class = "col-sm-10">
            @Html.DisplayFor(model => model.FirstName)
        </dd>
        <dt class = "col-sm-2">
            @Html.DisplayNameFor(model => model.LastName)
        </dt>
        <dd class = "col-sm-10">
            @Html.DisplayFor(model => model.LastName)
        </dd>
        <dt class = "col-sm-2">
            @Html.DisplayNameFor(model => model.Email)
        </dt>
        <dd class = "col-sm-10">
            @Html.DisplayFor(model => model.Email)
        </dd>
        <dt class = "col-sm-2">
            @Html.DisplayNameFor(model => model.PhoneNumber)
        </dt>
        <dd class = "col-sm-10">
            @Html.DisplayFor(model => model.PhoneNumber)
        </dd>
        <dt class = "col-sm-2">
            @Html.DisplayNameFor(model => model.Id)
        </dt>
        <dd class = "col-sm-10">
            @Html.DisplayFor(model => model.Id)
        </dd>

    </dl>
</div>
<div>
    <a asp-action="Edit" asp-route-id="@Model?.Id">Edit</a> |
    <a asp-action="Index">Back to List</a>
</div>

```

Slika 11. Stranica Details

Ovaj kod će generirati HTML stranicu koja prikazuje detalje korisnika aplikacije.

Zatim slijedi dio gdje se prikazuju detalji korisnika, poput imena (FirstName), prezimena (LastName), e-pošte (Email), broja telefona (PhoneNumber) i identifikatora (Id). Nakon toga, korisniku će biti ponuđene opcije za uređivanje ili povratak na listu korisnika. Opcija "Edit" vodi korisnika na stranicu za uređivanje detalja trenutnog korisnika, dok opcija "Back to List" vraća korisnika na listu svih korisnika.

9.PRAKTIČNA IMPLEMENTACIJA SISTEMA

Dijelovi praktične implementacije sistema su obrađeni u poglavljima 5.2. , 6., 7. i 8

Sistem je dostupan za pregled na linku: https://github.com/edinakaknjo/Katastar

10.ZAKLJUČAK

Sistem službe za imovinsko-pravne odnose ili kako sam ga pominjala u dokumentaciji- 'Katastar' implementiran je koristeći ASP.NET Core tehnologiju, pri čemu se oslanja na MVC arhitekturu kako bi osigurao jasnu organizaciju i odvajanje logike aplikacije.

U projektnom zadatku detaljno su prikazani različiti aspekti informacionog sistema, koristeći tabele, dijagrame i druge alate. Dokumentacija obuhvata organizaciju vremena za dizajn i razvoj, slučajeve korištenja, komponente sistema i strukturu baze podataka. Ovaj sistem igra ključnu ulogu u efikasnom vođenju katastarskih evidencija i procesima upravljanja zemljištem. Iako projektni zadatak predstavlja samo dio kompleksnosti ovog sistema, jasno je da bi njegov detaljniji razvoj zahtijevao dugotrajniji proces analize, dizajna i testiranja.