

*Univerzitet u Novom Sadu,  
Fakultet tehničkih nauka*

# **SEMINARSKI RAD**

**Služba smeštaja**

**Tehnologije i sistemi eUprave**

Student,  
Mateja Rilak SR 38/2021

Jun, 2024.

## **1. Sažetak**

U ovom radu je opisan sistem za upravljanje domovima, koji omogućava pregled, upravljanje i generisanje entitetima vezanih za domove. Za realizaciju projekta korišćeni su UML dijagrami klasa i slučajeva korišćenja, single sign-on za autentifikaciju i autorizaciju korisnika i mikroservisna arhitektura koja omogućava komunikaciju između servisa. Korisnici sistema su podeljeni u dve grupe, administratori sistema eUprave i studenti. Studenti mogu da pošalju prijavu za dom, pogledaju evidenciju o svojoj sobi, takođe i da pogledaju sve moguće domove i njihove prijave, dok administratori sistema mogu da upravljaju svim entitetima sistema.

## **2. Ključne reči**

- veb aplikacija
- upravljanje sistemom domova
- mikroservisna arhitektura
- autentifikacija i autorizacija korisnika
- QR code očitavanje informacija

### 3. Uvod

Živimo u vremenu kada je digitalizacija postala sastavni deo naših života. Svakodnevno koristimo pametne telefone, tablete i računare kako bismo obavili razne poslove – od plaćanja računa do komunikacije sa prijateljima. Upravo zbog toga, nije iznenađujuće što se i javna uprava polako, ali sigurno, digitalizuje. Ovo donosi mnoge prednosti, a jedna od oblasti gde se eUprava posebno ističe jeste upravljanje službom smeštaja.

Ako ste ikada prošli kroz proces prijavljivanja za smeštaj u studentskom domu, znate koliko to može biti stresno. Puno papirologije, dugotrajni administrativni procesi i beskrajno čekanje na rezultate – sve to može biti veoma naporno. Zato je ideja o eUpravi koja preuzima ovaj zadatak tako privlačna. Elektronski sistem za upravljanje fakultetskim domovima može značajno olakšati život studentima, ali i administrativnim radnicima.

Ovaj seminarski rad bavi se upravo tim – kako eUprava može poboljšati proces upravljanja fakultetskim domovima. Razmotrićemo kako digitalni alati mogu pojednostaviti prijavu, ubrzati obradu zahteva i omogućiti pravičniju raspodelu smeštajnih kapaciteta. Takođe ćemo se osvrnuti na tehničke izazove koje treba prevazići, kao i na to kako sve ove promene utiču na studente i zaposlene na fakultetima.

Na kraju, cilj ovog rada je da pokaže da je moguće stvoriti efikasniji, transparentniji i jednostavniji sistem za upravljanje fakultetskim domovima. Kada se procesi digitalizuju, svi učesnici – od studenata do administrativnog osoblja – mogu imati koristi, a studentski život postaje mnogo jednostavniji i organizovaniji.

## 4. Pregled korišćenih tehnologija

Ovaj odeljak pruža pregled i objašnjenja tehnologija pomoću kojih je predstavljeno rešenje.

### GoLang

GoLang, poznatiji kao Go, je programski jezik koji su razvili inženjeri iz Google-a. Predstavlja kompajlirani jezik, dizajniran da bude jednostavan, efikasan i pouzdan. Go je posebno prilagođen za razvoj skalabilnih i performantnih aplikacija, te je stoga popularan među developerima koji rade na server-side aplikacijama, cloud usluga i distribuiranim sistemima.

Go je statički tipiziran jezik koji koristi garbage collector za upravljanje memorijom, čime se smanjuje složenost koda u odnosu na druge jezike.

Go nudi moćne alate za konkurentno programiranje koristeći goroutine i kanale, što omogućava jednostavno i efikasno upravljanje paralelnim procesima.

Go je open-source jezik, što znači da je njegov razvoj i održavanje u rukama zajednice developera širom sveta. Zahvaljujući svojoj snažnoj zajednici i korporativnoj podršci od strane Google-a, Go se neprestano razvija i unapređuje.

### ReactJS

React je popularna JavaScript biblioteka za izradu korisničkih interfejsa koju je razvila kompanija Facebook. Kao front-end tehnologija, React omogućava developerima da kreiraju dinamične i responzivne web aplikacije na jednostavan i efikasan način. React je poznat po svojoj brzini, fleksibilnosti i skalabilnosti, što ga čini idealnim za razvoj modernih web aplikacija.

Jedna od ključnih karakteristika React-a je upotreba komponenti. Komponente su osnovni gradivni blokovi u React-u i omogućavaju developerima da aplikaciju podele na manje, ponovo upotrebljive delove koda. Svaka komponenta može imati svoj state (stanje) i props (svojstva), što olakšava upravljanje podacima i interakcijama u aplikaciji.

React koristi virtualni DOM (Document Object Model), što značajno poboljšava performanse aplikacija. Umesto da direktno manipuliše stvarnim DOM-om, React kreira virtuelnu kopiju DOM-a i primenjuje minimalne promene kako bi ažurirao korisnički interfejs. Ovaj pristup smanjuje broj skupih operacija manipulacije DOM-om i omogućava brže renderovanje.

React je open-source biblioteka, što znači da je njegov razvoj i održavanje u rukama zajednice developera širom sveta. Zahvaljujući snažnoj podršci zajednice i korporativnoj podršci od strane Facebook-a, React se neprestano razvija i unapređuje. Takođe, React ima bogat ekosistem alata i biblioteka, kao što su React Router za upravljanje rutama i Redux za upravljanje stanjem aplikacije, što dodatno olakšava razvoj složenih aplikacija.

Zahvaljujući svojoj jednostavnosti, moćnim funkcionalnostima i podršci za komponente, React je postao jedan od najpopularnijih alata za razvoj web aplikacija, koriste ga mnoge kompanije širom sveta za kreiranje interaktivnih i visokoperformantnih korisničkih interfejsa.

## **MongoDB**

MongoDB je popularan NoSQL sistem za upravljanje bazama podataka, poznat po svojoj fleksibilnosti i skalabilnosti. Razvijen od strane MongoDB Inc., MongoDB je dizajniran da omogući rad sa velikim količinama podataka i da podrži različite tipove podataka, čime se razlikuje od tradicionalnih relacionih baza podataka.

Jedna od glavnih karakteristika MongoDB-a je njegov dokumentno-orijentisani pristup. Podaci se čuvaju u formi BSON (Binary JSON) dokumenata unutar kolekcija. Ovaj model omogućava razvijanje aplikacija sa kompleksnim podacima bez potrebe za složenim šemama, što daje veću fleksibilnost prilikom dizajna i razvoja baze podataka.

MongoDB nudi visoku skalabilnost i performanse zahvaljujući podršci za horizontalno particionisanje (sharding). Sharding omogućava distribuciju podataka preko više servera, što čini MongoDB pogodnim za rad sa ogromnim količinama podataka i visokim zahtevima za propusnost.

Jedna od ključnih prednosti MongoDB-a je njegov bogat set funkcionalnosti koje podržavaju različite operacije sa podacima. Ove funkcionalnosti uključuju napredne upite, agregacije, indeksiranje i replikaciju, što omogućava efikasno pretraživanje i manipulaciju podacima. Replikacija obezbeđuje visoku dostupnost i pouzdanost podataka, čime se smanjuje rizik od gubitka podataka i omogućava kontinualni rad aplikacija.

MongoDB je open-source baza podataka, što znači da je njen kod dostupan javnosti za korišćenje i modifikaciju. Aktivna zajednica developera i podrška kompanije MongoDB Inc. doprinose stalnom unapređenju i razvoju novih funkcionalnosti. Takođe, MongoDB Atlas, kao upravljani cloud servis, pruža korisnicima mogućnost da jednostavno implementiraju i upravljaju svojim bazama podataka na različitim cloud platformama.

Zahvaljujući svojoj fleksibilnosti, skalabilnosti i bogatom setu funkcionalnosti, MongoDB je postao izbor mnogih kompanija i developera za izgradnju modernih aplikacija koje zahtevaju rad sa velikim količinama podataka i visoku performansu.

## **Docker**

Docker je platforma za kontejnerizaciju koja omogućava developerima da kreiraju, distribuiraju i pokreću aplikacije unutar kontejnera. Razvijen od strane Docker Inc., Docker je revolucionisao način na koji se softver razvija, testira i distribuira, pružajući jednostavno i efikasno rešenje za upravljanje aplikacijama u različitim okruženjima.

## 5. Specifikacija zahteva

U ovom poglavlju možete videti objašnjenje funkcionalnih i nefunkcionalnih zahteva i njihova rešenja.

### 5.1 Slučajevi korišćenja

Tabela 1 prikazuje opis slučaja korišćenja “**Kreiranje doma**”.

Naziv	Kreiranje doma
Učesnici	Administrator
Preduslovi	Administrator je ulogovan kao administrator e-uprave za službu smeštaja
Koraci	Unos bazičnih informacija Unos cena za smeštaj
Rezultat	Kreiran dom
Izuzeci	Problem u kreiranju doma

*Tabela 1 - Opis slučaja korišćenja “Kreiranje doma”*

**Tabela 2** prikazuje opis slučaja korišćenja zahteva za kreiranje sobe.

Naziv	Kreiranje soba
Učesnici	Administrator
Preduslovi	Administrator je ulogovan kao administrator e-uprave za službu smeštaja Dom za koji se kreira soba postoji kao entitet unutar sistema
Koraci	Izaberi dom za koji kreiras sobu Unesi potrebne informacije vezane za sobu
Rezultat	Kreirana soba
Izuzeci	Problem u kreiranju sobe

*Tabela 2 - Opis slučaja korišćenja “Kreiranje soba”*

**Tabela 3** prikazuje opis slučaja korišćenja zahteva za kreiranje konkursa za dom.

Naziv	Kreiranje konkursa za dom
Učesnici	Administrator
Preduslovi	Administrator je ulogovan kao administrator e-uprave za službu smeštaja Dom za koji se kreira soba postoji kao entitet unutar sistema

Koraci	Izabrati dom za koji se kreira konkurs Odabrati datume konkursa
Rezultat	Kreiran konkurs za dom
Izuzeci	Problem u kreiranju konkursa

*Tabela 3 - Opis slučaja korišćenja "Kreiranje konkursa za dom"*

**Tabela 4** prikazuje opis slučaja korišćenja zahteva za "administraciju prijava".

Naziv	Administracija prijava
Učesnici	Administrator
Preduslovi	Administrator je ulogovan kao administrator e-uprave za službu smeštaja Dom kao entitet postoji Konkurs kao entitet postoji Aplikacija ima konkurse
Koraci	Izabrati dom za koji se kreira konkurs Odabrati datume konkursa
Rezultat	Kreiran konkurs za dom
Izuzeci	Problem u kreiranju konkursa

*Tabela 4. - Opis slučaja korišćenja "Administracija prijava"*

U **tabeli 5.** opisana je funkcionalnost prijave za dom.

Naziv	Prijava za dom
Učesnici	Student
Preduslovi	Student je ulogovan kao student Student je aktivan u novoj školskoj godini Dom postoji kao entitet Konkurs postoji kao aktivan entitet.
Koraci	Izabrati dom za koji se konkurise Odabrati način prijave za aplikaciju
Rezultat	Kreirana prijava
Izuzeci	Problem u kreiranju prijave

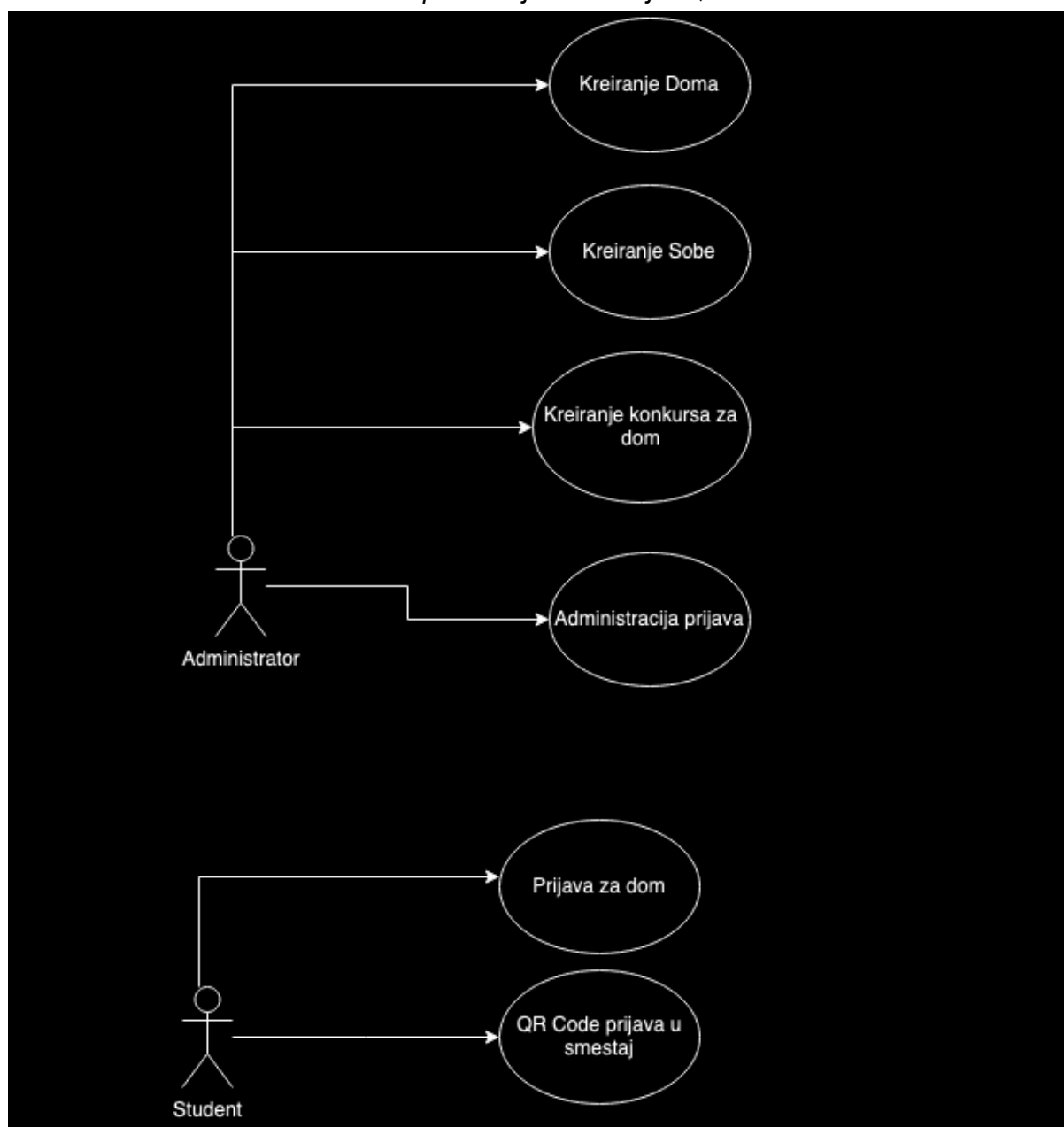
*Tabela 5. - Opis slučaja korišćenja "Prijava za dom"*

**Tabela 6.** opisuje funkcionalnost korišćenja QR Code za ulazak u dom i ostale funkcionalnosti unutar sistema.

Naziv	QR Code
Učesnici	Student

Preduslovi	Student je ulogovan kao student Student se nalazi u domu Student se nalazi u određenoj sobi u domu
Koraci	Student kada se uloguje na stranici doma vidi QR code
Rezultat	Student kada se uloguje na stranici doma vidi QR code
Izuzeci	Problem u kreiranju QR koda.

*Tabela 6. - Opis slučaja korišćenja "QR Code"*



*Slika 1 - UML dijagram slučaja korišćenja*



## 5.2 Specifikacija nefunkcionalnih zahteva sistema

Nefunkcionalni zahtevi:

- **Kontejnerizacija:** Sve činioce jednog sistema potrebno je pokrenuti unutar alata za kontejnerizaciju i virtuelizaciju poput Docker-a.
- **Parcijalni otkazi:** Aplikacija treba da ostane živa čak i ukoliko se dese parcijalni otkazi.
- **UI/UX:** Sama aplikacija treba da bude laka i prilagodljiva za korišćenje svim starosnim dobima.

## 6. Specifikacija dizajna

Ovo poglavlje objašnjava dizajn softverskog rešenja za sistem za upravljanjem studentskim domovima.

### 6.1 Arhitektura sistema

Ceo sistem eUprave, dela za univerzitete, je osmišljen da funkcioniše kao jedna mikroservisna aplikacija. Sistem se sastoji od više komponenti koje zajedno komuniciraju preko API interfejsa.

Dijagram klasa, predstavljen Slikom 2, prikazuje samu strukturu sistema za upravljanje studentskim domovima.

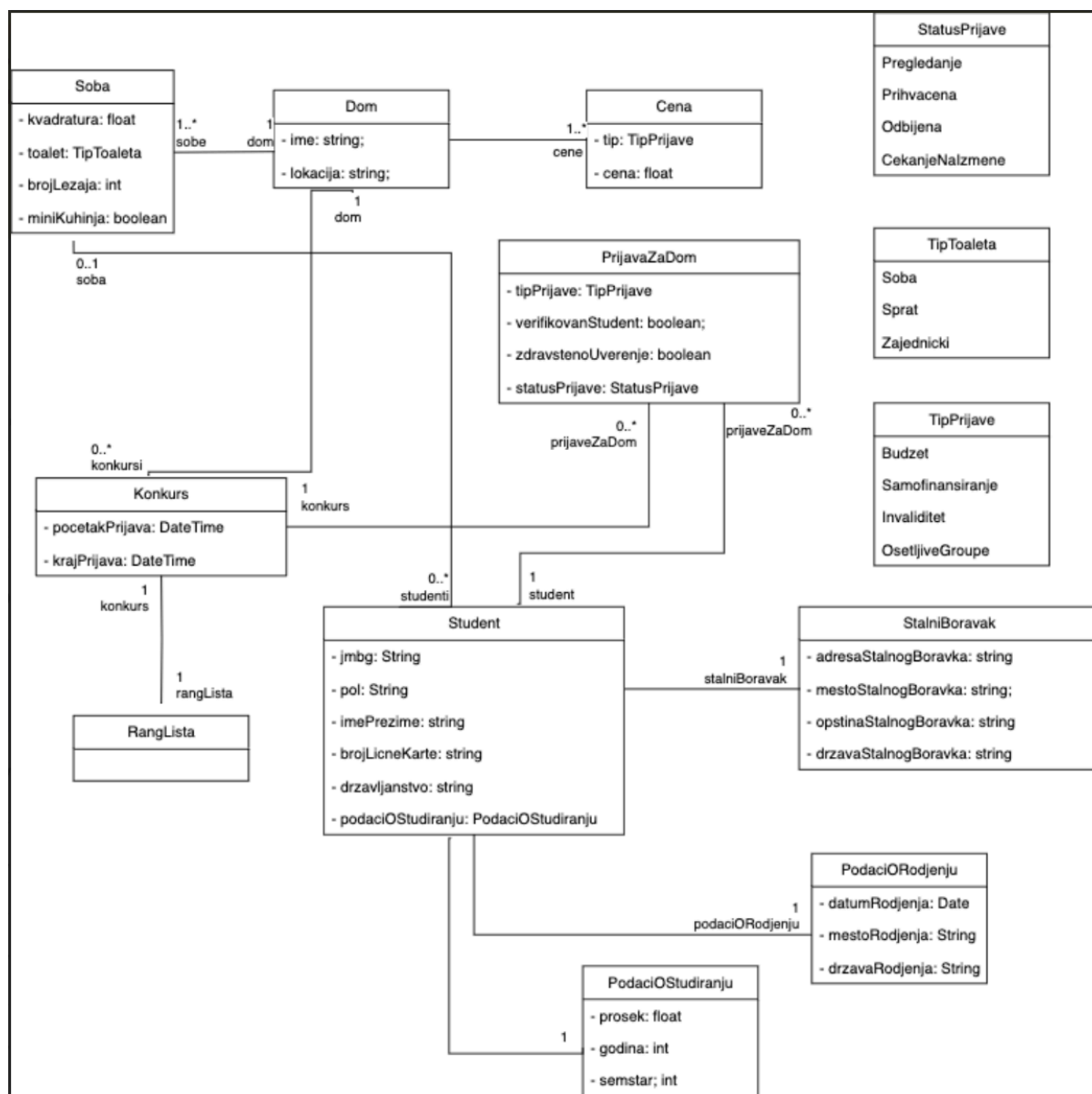
Centralni deo sistema oko koga se gradi ceo sistem čini klasa Dom, koja predstavlja domove za smeštaj studenata i sadrži njihove ključne podatke kao što su ime i lokacija samog doma. Klasa dom je usko povezana klasom Cena, koja varira od različitosti prijave.

Svaka prijava može da bude jedan od sledećih tipova: Budzet, Samofinansiranje, Invaliditet, OsetljiveGrupe, to jest, TipPrijave predstavlja enumeraciju.

Svaki dom ima jedan ili više smeštajnih kapaciteta, što je prikazano vezom sa klasom "Soba". Klasa "Soba" beleži informacije o pojedinačnim sobama, uključujući kvadraturu, tip toaleta, broj ležaja i studente koji se nalaze u sobi.

Klasa Konkurs u sebi sadrži dva datuma, početak i kraj prijave. Takođe, u direktnoj vezi je sa klasom Dom, to jest, jedan dom može imati više konkursa, dok jedan konkurs je vezan za jedan dom. Konkurs u sebi sadrži direktnu vezu sa RangListom. Konkurs može da sadrži više prijave za dom.

Ovaj dijagram klasa pruža jasan pregled strukture sistema za upravljanje studentskim domovima, uključujući ključne entitete, njihove attribute i međusobne odnose. Na ovaj način omogućava se efikasno upravljanje podacima o domovima, sobama i prijavama studenata, što doprinosi organizaciji i upravljanju smeštajnim kapacitetima za studente.



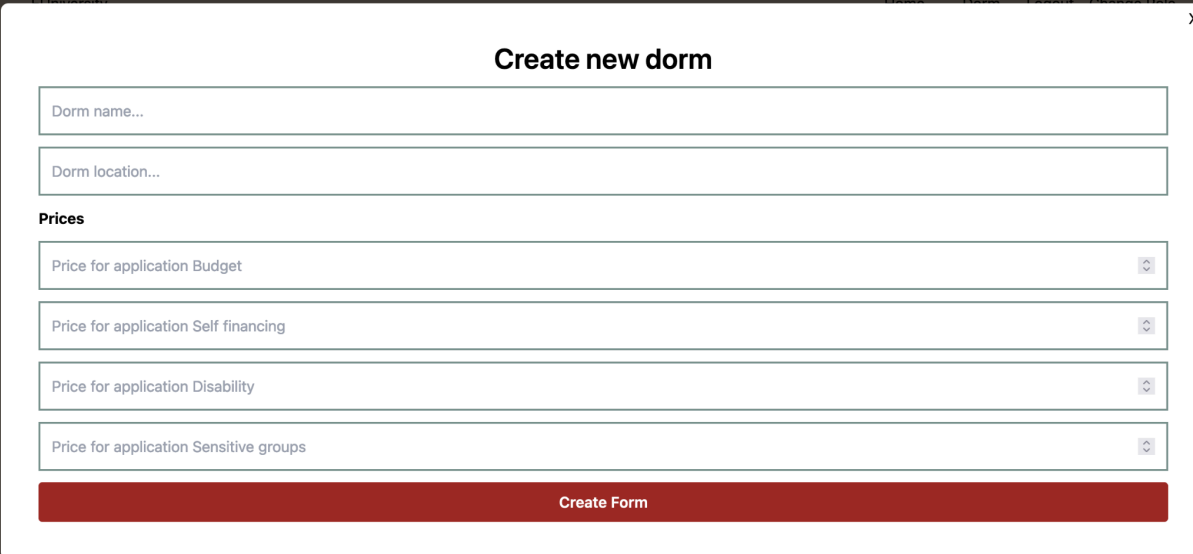
Slika 2 - UML Class Diagram

## 7. Implementacija i demonstracija

Ovo poglavlje prikazuje način na koji su implementirane najbitnije funkcionalnosti sistema za službu smeštaja. Objasnjena je implementacija ključnih komponenti i funkcionalnosti sistema.

### Implementacija kreiranja doma

U ovom odeljku je predstavljena implementacija funkcionalnosti za kreiranje doma. U ovom korisničkom interfejsu, korisnik popunjava formu za kreiranje doma.



Slika 3 - Forma za kreiranje doma

Nakon što zahtev uspešno prođe, podaci se prosleđuju ka serveru, koji te podatke obrađuje tako što kreira novi entitet sa podacima koji su prosleđeni sa klijentove strane.

```
func (ah DormHandler) CreateNewDorm(rw http.ResponseWriter, h *http.Request) {
    var dorm models.Dorm
    if !utils.DecodeJSONFromRequest(h, rw, &dorm) {
        utils.WriteErrorResp("Error while casting", 500, "/dorm", rw)
        return
    }
    createdDorm, err := ah.dormService.CreateNewDorm(dorm)
    if err != nil {
        utils.WriteErrorResp(err.GetErrorMessage(), err.GetErrorStatus(), "path", rw)
        return
    }
    utils.WriteResp(createdDorm, 201, rw)
}
```

Listing 1 - Controller metoda za kreiranje doma

U listingu 2 se nalazi primer upisa podataka u bazu "dorm" i kolekcije "dorm". Baza kreira automatski jedinstvene identifikatore koji se postavljaju za dati entitet.

```
func (dr DormRepository) SaveNewDorm(dorm models.Dorm) (*models.Dorm, *errors.ErrorStruct) {
    dormCollection := dr.cli.Database("dorm").Collection("dorm")
    insertedDorm, err := dormCollection.InsertOne(context.TODO(), dorm)
    if err != nil {
        return nil, errors.NewError(err.Error(), 500)
    }
    log.Println(insertedDorm)
    dorm.ID = insertedDorm.InsertedID.(primitive.ObjectID)
    return &dorm, nil
}
```

Listing 2 - Metoda za upis u bazu unutar Repository sloja.

## Implementacija kreiranja doma

Na slici 4. je prikazan grafički interfejs za kreiranje doma, gde korisnik bira dom za koji će se kreirati nova soba, a takođe unosi informacije vezane za veličinu sobe, broj kreveta i tip toaleta.

Slika 4 - Grafički interfejs za kreiranje sobe unutar doma

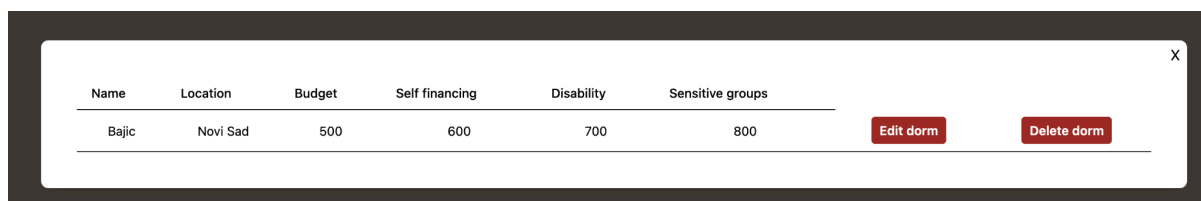
Na slici Listing 3 pokazuje način implementacije čuvanja sobe unutar baze “dorm” i kolekcije “rooms”. Soba se kreira inicijalno kao prazna, to jest, bez studenata.

```
func (rr RoomRepository) SaveNewRoom(room models.Room) (*models.Room, *errors.ErrorStruct) {
    roomsCollection := rr.cli.Database("dorm").Collection("rooms")
    room.Students = make([]models.Student, 0)
    insertedRoom, err := roomsCollection.InsertOne(context.TODO(), room)
    if err != nil {
        return nil, errors.NewError(err.Error(), 500)
    }
    room.ID = insertedRoom.InsertedID.(primitive.ObjectID)
    return &room, nil
}
```

Listing 3 - Implementacija upisa sobe u bazu podataka

### Primer prikaza kreiranih domova

Primer prikaza kreiranih domova se može videti na slici 5. Ovo je primer prikaza iz administracionog panela, gde administrator ima mogućnost da izmeni ili obriše dom.

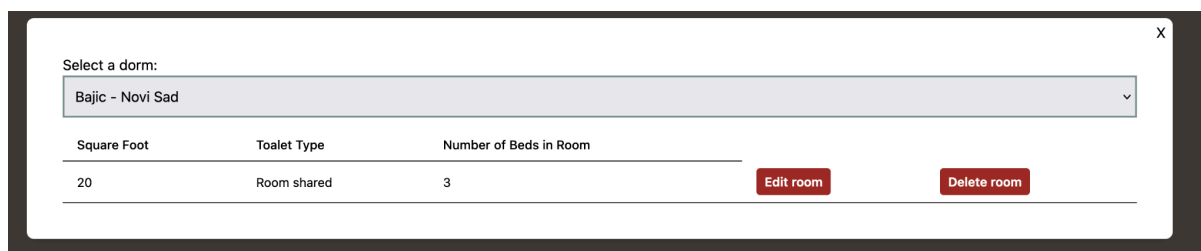


Name	Location	Budget	Self financing	Disability	Sensitive groups		
Bajic	Novi Sad	500	600	700	800	Edit dorm	Delete dorm

Slika 5 - Primer prikaza kreiranih domova iz administracionog pane

### Primer prikaza soba unutar određenog doma:

Primer prikaza soba unutar određenog doma se može videti na slici 6. Ovo je primer prikaza sobe, nakon što je korisnik odabrao ponuđeni dom.

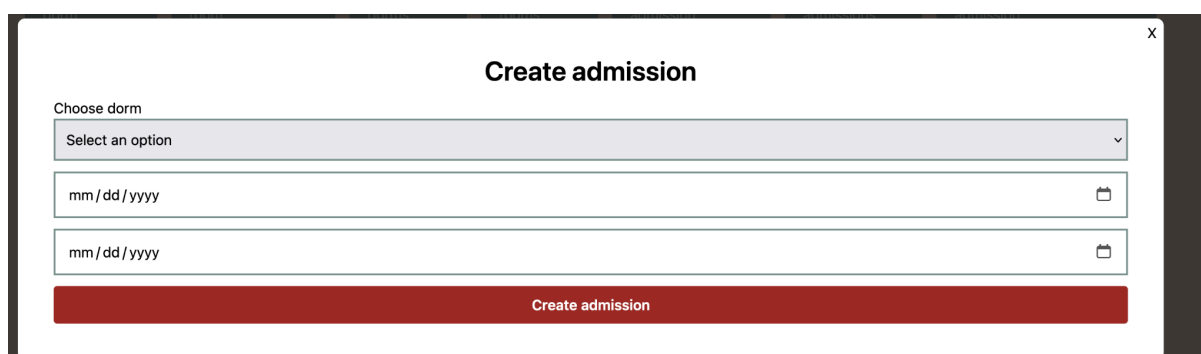


Square Foot	Toalet Type	Number of Beds in Room		
20	Room shared	3	Edit room	Delete room

Slika 6 - Primer prikaza sobe unutar određenog doma

### Primer prikaza forme za kreiranje konkursa:

Na slici 7. se može videti primer prikaza forme za kreiranje konkursa, gde korisnik može izabrati dom za koji specifično želi da otvori konkurs.



**Create admission**

Choose dorm

Select an option

mm / dd / yyyy

mm / dd / yyyy

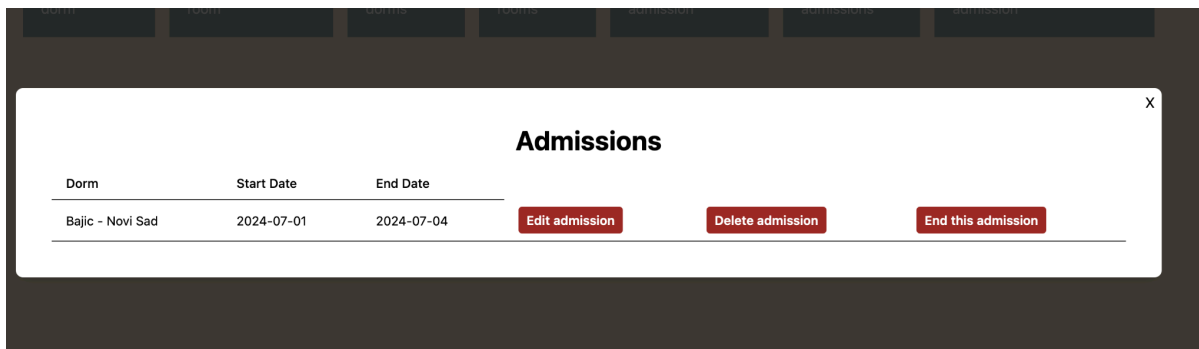
Create admission

Slika 7 - Primer prikaza forme za kreiranje konkursa

### Primer prikaza aktivnih konkursa:

Na slici 8. se može videti primer prikaza trenutno aktivnih konkursa, sa mogućnostima za izmenu, brisanje i završavanje konkursa.

Kada korisnik klikne na završavanje konkursa, aplikacija automatski sortira sve prijave u rang listu i onda krene da upisuje studente unutar soba.



Slika 8 - Primer prikaza svih aktivnih konkursa

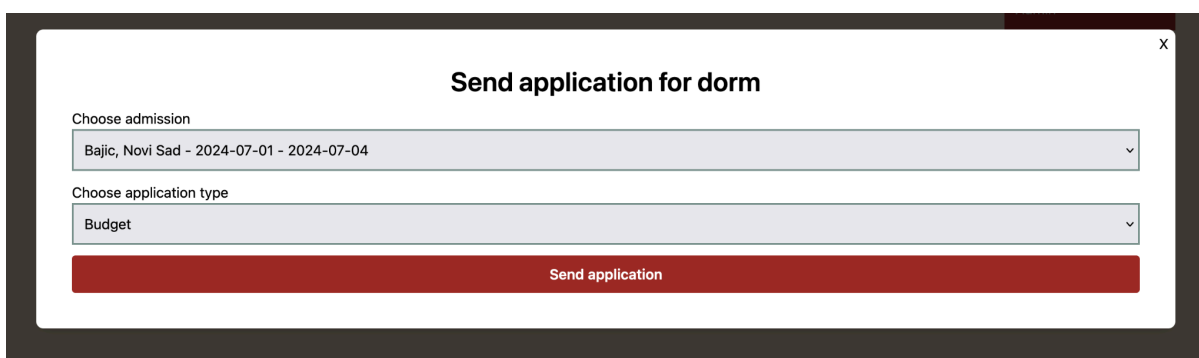
### Implementacija slanja aplikacije za dom:

Implementacija slanja aplikacije za dom prikuplja informacije o trenutno ulogovanom korisniku u sistemu, koji kada pošalje aplikaciju, automatski napuni aplikaciju svojim podacima. Zatim, Dorm service komunicira sa Health Care servisom da provere da li student uradio lekarski pregled. Ukoliko je student uspešno uradio lekarski pregled, njegova aplikacija se stavlja na proces "PRIHVAĆENA", što ne znači nužno da je on upao u dom, već da je njegova aplikacija samo prošla kroz faze validacije i provere.

```
func (as ApplicationsService) CreateNewApplication(application models.ApplicationForDorm) (*models.ApplicationForDorm, *errors.ErrorStruct) {
    valueFromHealthCare, err := as.healthCareClient.GetUserHealthStatusConfirmation(application.Student.PersonalIdentificationNumber)
    if err != nil {
        log.Println(err.GetErrorMessage())
    }
    application.HealthInsurance = valueFromHealthCare
    if !application.HealthInsurance || !application.VerifiedStudent {
        application.ApplicationStatus = models.Pending
    } else {
        application.ApplicationStatus = models.Accepted
    }

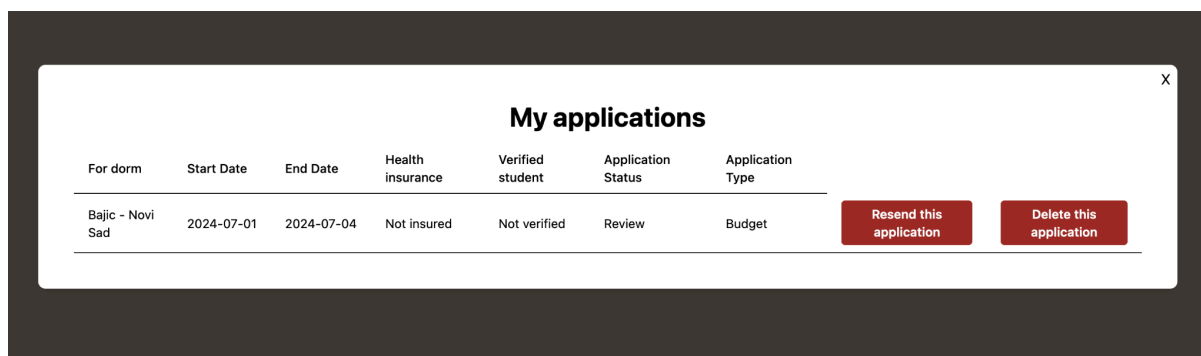
    createdApplication, err := as.applicationsRepository.SaveNewDorm(application)
    if err != nil {
        return nil, err
    }
    return createdApplication, nil
}
```

Listing 4 - Primer implementacije servisnog sloja za kreiranje aplikacije



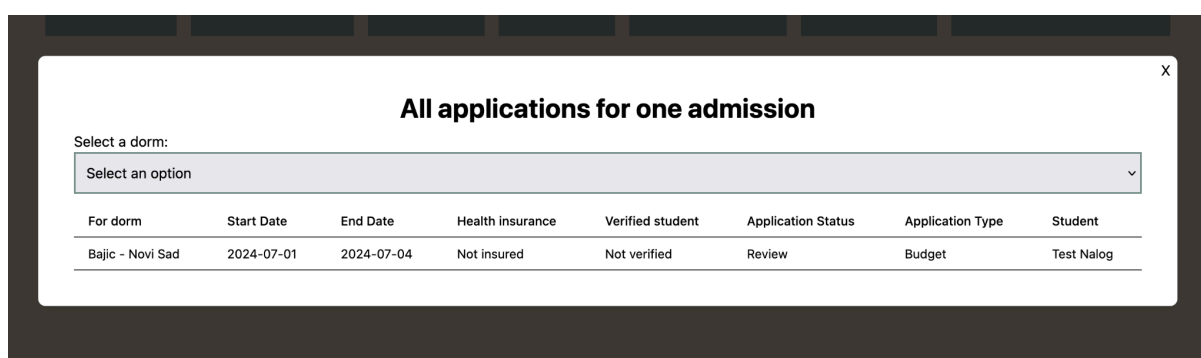
Slika 9 - Primer slanja aplikacije za dom

Korisnik kada pošalje aplikaciju, ukoliko se neki od uslova ne ispunjava, njegova aplikacija je na Pending. Ukoliko je on uspešno ispunio sve uslove nakon roka, može da pošalje ponovo aplikaciju klikom na dugme "Resend this application", gde će aplikacija ponovo automatskim putem proveriti da li su svi uslovi ispunjeni.



Slika 10 - Primer prikaza aplikacija kod studenta

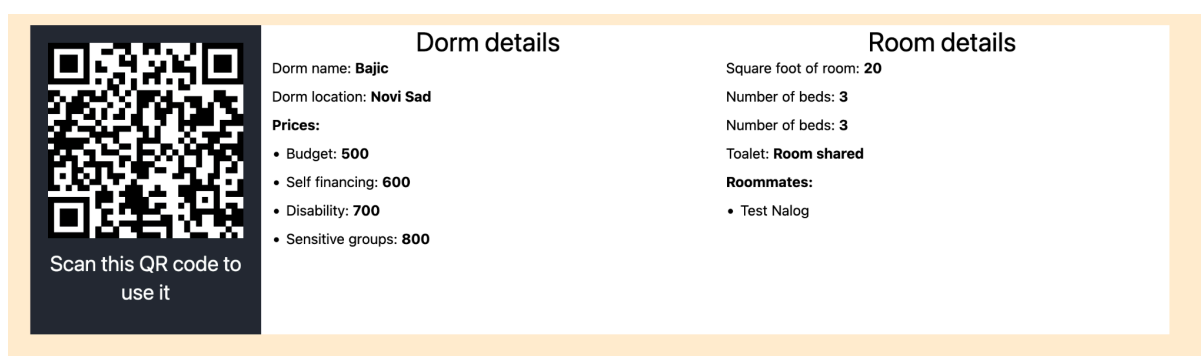
Na slici 11. se može videti primer prikaza svih aplikacija za jedan konkurs sa administratorske strane.



Slika 11 - Primer prikaza svih aplikacija za jedan dom.

### Primer prikaza studentske stranice kada je student u domu:

Na stranici za dom, kada se student nalazi u domu, on na njoj vidi QR kod funkcionalnost i sve informacije vezane za dom i sobu, zajedno sa tim ko je sve u sobi sa njim.



Slika 12 - Primer prikaza studentske stranice kada je student u domu



## 8. Zaključak

U ovom radu je predstavljen sistem za eUpravu, Služba smještaja, razvijen korišćenjem tehnologija GoLang, React, MongoDB i Docker. Sistem omogućava kreiranje domova i omogućavanje elektronskog slanja zahteva za upis i raspoređivanje unutar domova.

Kroz implementaciju različitih funkcionalnosti, poput komunikacija između sistema fakulteta i zdravstva, možemo značajno da ubrzamo sam proces upisa i raspoređivanje u dom, i rasteretimo nadležne institucije.

Ovaj projekat pokazuje kako tehnologija može biti pozitivan član današnjeg društva za unapređivanje procesa.

## 9. Literatura

- React: <https://react.dev/>
- GoLang: <https://go.dev/>.
- Docker: <https://www.docker.com/>.
- MongoDB: <https://www.mongodb.com/>