Универзитет у Београду
Електротехнички факултет

Апликација за управљање геодетксим бироом

Дипломски рад

Ментор:

др Милош Цветановић, професор

Кандидат:

Немања Станојковић
2019/0604

Београд, Јун 2024

САДРЖАЈ

1.Увод

Управљање подацима и ефикасно пословање малих фирми представља велики изазов у савременом свету. Без обзира на делатност, свака фирма мора имати систем који ће омогућити праћење свих релевантних информација и података и то све на једном месту. Овај приступ не само да олакшава управљање информацијама већ и обезбеђује да се све промене и ажурирања аутоматски одразе на све повезане податке. У овом контексту, апликација за локалну и интернет употребу представља идеално решење за све мале и велике фирме које се баве геодезијом.

Предмет овог рада је израда апликације која ће омогућити евиденцију и манипулацију кључним подацима Геодетског бироа ЦАГИ. Ова апликација има за циљ да искористи савремене технологије како би креирала интуитиван и лак за коришћење кориснички интерфејс. Апликација ће омогућити централизовано управљање информацијама што ће значајно побољшати ефикасност и продуктивност пословања

Апликација ће бити доступна корисницима преко интернета омогућавајући већем броју људи да приступе и виде све релевантне информације о фирми и њеним услугама. Свака фирма, укључујући и геодетске фирме, мора имати одговарајући веб сајт. Он омогућава потенцијалним клијентима лак приступ информацијама о фирми и њеним услугама, што је кључно за привлачење нових клијената и одржавање добрих односа са постојећим. Поред тога, апликација ће нудити могућност пријаве на посебан део намењен искључиво радницима фирме. Овај део је намењен за локалну употребу у самој фирми, што је веома корисно за чување битних информација и обезбеђивање њихове приватности.

У наредном поглављу ће бити речи о проблемима са којима се Геодетски биро ЦАГИ сусреће у свакодневном пословању. Размотриће се како предложена апликација може да реши те проблеме и унапреди пословне процесе. Треће поглавље ће обухватити детаље о бази података која ће се користити у апликацији. Биће објашњен избор технологије за израду базе података, структура табела и поља, као и начин на који ће подаци бити организовани и приступани. Четврто поглавље ће се фокусирати на технологије које су коришћене за развој апликације. Биће објашњено како се креира *Angular* апликација, како се додају компоненте у *Angular*-у, као и процес креирања *Spring Boot* апликације. Ово поглавље ће такође покрити архитектуру апликације и разлоге за избор конкретних технологија. Пето поглавље ће детаљно описати функционалности система.Такође у овом делу моћи ћемо да видимо нешто више о предметима који представљају кључ свих података. Биће представљени изглед странице, примери програмског кода и *UML* дијаграми који илуструју функционалности апликације. Осим тога, биће описан и мени апликације и начин његовог коришћења.

2.АНАЛИЗА ПРОБЛЕМА

Анализа проблема која је послужила као основа за развој ове апликације за унапређење Геодетског бироа ЦАГИ указује на неколико кључних изазова:

- Недостатак централизованог система за праћење података: Пре имплементације ове апликације, радници су се ослањали на више извора података, што је отежавало ефикасност у раду. Без централизованог система, информације су биле распршене, што је доводило до губитка времена приликом претраге и освежавања података.
- Потешкоће у управљању клијентима и предметима: Без одговарајућег система за праћење, управљање клијентима и њиховим предметима постало је сложено. Радницима је недостајала ефикасна платформа која би им омогућила да лако прате историју комуникације, плаћања и посета терену за сваког клијента и његов предмет.
- Потреба за повратном информацијом од клијената: Без механизма за прикупљање повратних информација од клијената, фирма није имала увид у задовољство клијената и евентуалне проблеме које су могли имати. Недостатак ових информација могао је ограничити способност фирме да побољша своје услуге и задовољи потребе клијената.
- Немогућност ефикасног управљања људским ресурсима: Фирма није имала централизован систем за управљање информацијама о запосленима, као ни ефикасан процес за запошљавање нових радника. Ово је могло отежати процес рекрутације и смањити ефикасност тима.

Ова анализа показује да је апликација развијена како би решила ове проблеме и унапредила ефикасност и производност Геодетског бироа ЦАГИ. Имплементација ове апликације омогућава централизован приступ подацима, олакшава управљање клијентима и предметима, омогућава прикупљање повратних информација од клијената и пружа ефикасан систем управљања људским ресурсима. Ове функционалности доприносе ефикаснијем пословању, бољем задовољству клијената и унапређењу пословних процеса у фирми.

Осим решавања оперативних изазова, ова апликација пружа и значајне предности за маркетирање Геодетског бироа ЦАГИ путем веб сајта:

- Централизована платформа за информације: Интеграција веб сајта са апликацијом омогућава корисницима да добију све потребне информације о фирми на једном месту. То укључује информације о услугама, контакт подацима, као и могућност остављања коментара. Ово олакшава клијентима да се информишу о фирми и њеним услугама, што може побољшати препознатљивост бренда и повећати број потенцијалних клијената.
- Побољшана интеракција са клијентима: Кроз могућност остављања коментара на веб сајту, клијенти имају платформу за пружање повратних информација о својим искуствима са фирмом. Ово не само што омогућава фирми да добије корисне информације о томе како побољшати своје услуге, већ такође ствара позитивну интеракцију са клијентима, што може допринети изградњи лојалности и позитивног имица бренда.
- Повећана видљивост преко интернета: Интеграција апликације са веб сајтом може побољшати *SEO* (оптимизација за претраживаче) перформансе фирме.

Кроз редовно ажурирање веб сајта новим садржајем и активностима, као што су коментари клијената или објаве о новим услугама, фирма може побољшати своју видљивост на интернету и привући нове клијенте.

- Демонстрација технолошке компетенције: Имплементација софистициране апликације за управљање пословним процесима показује да фирма користи најновије технологије како би побољшала своје услуге и пословање. Ово може ојачати перцепцију фирме као технолошки освештене и иновативне, што може привући нове клијенте и инвеститоре.

Кроз ове бенефите за маркетинг путем веб сајта, апликација доприноси ширењу свести о фирми, побољшава интеракцију са клијентима и гради позитиван имиџ бренда на тржишту. Ова синергија између веб сајта и апликације представља снажан инструмент за маркетиншке активности, пружајући комплетан увид у пословање и услуге фирме на ефикасан и привлачан начин.

Кроз унапређење интеракције са клијентима, побољшану видљивост на интернету и демонстрацију технолошке компетенције, Геодетски биро ЦАГИ ствара трајне везе са клијентима и позиционира себе као лидера у индустрији. Овај интегрисани приступ маркетингу и оперативним процесима омогућава фирми да постигне већу конкурентност и стабилност на тржишту.

3.БАЗА ПОДАТАКА

Пројектна одлука је да се за развој апликације користи *MySQL* база података. У питању је релациона база података што значи да се разликује од *NoSQL* база података.

3.1. *MySQL*, Релационе базе података

Релационе базе података користе *SQL* (*Structured Query Language*) за комуникацију и имају табеле са структуром. Овакав тип базе података дизајниран је за интеракцију са апликацијама. *MySQL* је једна од најпопуларнијих релационих база података и пружа подршку за динамичке упите, висок ниво доступности и репликацију, што значи да се подаци могу реплицирати на више сервера. Може вертикално да се скалира (скалирање унутар једног сервера додавањем више ресурса), док хоризонтално скалирање (додавање више сервера) обично није једноставно као код неких *NoSQL* база података, али је могуће уз одговарајућу конфигурацију. Такође, подржава атомичне модификације и сложене трансакције. Погодна је за чување великих количина података и подржава *ACID* (*Atomicity*, *Consistency*, Isolation, *Durability*) својства.

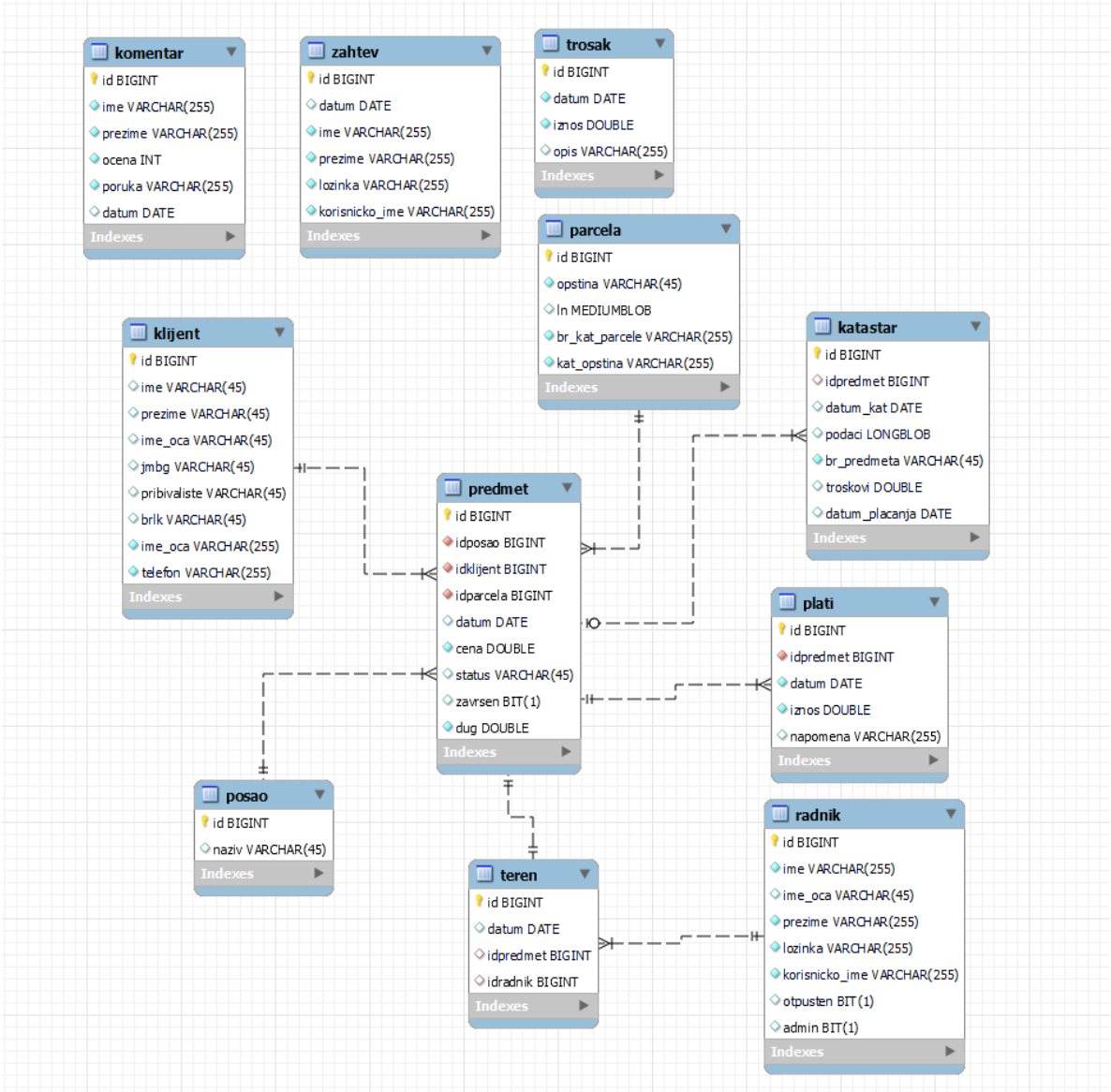
Предности релационих база података укључују јасно дефинисане шеме које омогућавају структурирану и конзистентну организацију података. Ово може бити корисно када се ради на пројектима где је структура података добро дефинисана и константна током времена. Релационе базе података дизајниране су тако да подржавају паралелизацију операција, што омогућава брзу обраду података на више сервера.

MySQL база података чува податке у табелама које се састоје од редова и колона. Свака табела представља један ентитет у бази података, а редови у табели представљају записе. За развој базе података коришћено је окружење *MySQL Workbench*. Ова база се једноставно интегрише у пројекте веб технологија као што је *LAMP* стек (Linux, *Apache*, *MySQL*, *PHP*). Библиотеке као што су *Sequelize* и *TypeORM* могу се користити за објектно-релационо мапирање у апликацијама које користе *MySQL*. У *MERN* стеку (*MongoDB*, *Express*.js, *React*, *Node.js*), *MongoDB* је систем за управљање базама података, тако да *MySQL* није типично коришћен у том контексту.

MySQL нуди низ алата и функционалности које помажу у управљању базама података, као што су *MySQL Workbench*, који је графички алат за дизајн и управљање базом података. *MySQL Workbench* омогућава корисницима да визуализују структуру базе података, креирају и измене табеле, као и да генеришу сложене *SQL* упите и скрипте. Овај алат је веома користан за развој и одржавање базе података, посебно у сложеним пројектима.

3.2. Шема базе података

Табеле које подесује база: *Radnik, Parcela, Posao, Katastar, Teren, Predmet, Klijent, Komentar, Zahtev, Trosak*, и *Plati* .



Слика 3.2. Релациона шема базе података

УМЛ дијаграм је добијен у програму *MySQL Workbench*. Потребно је и рећи да су табеле које видимо на слици креиране на основу *Java* класа ентитета. Користећи анотације за означавање ентитетских класа у *Java* програмском језику, *Hibernate*, који је део *Java Spring Boot* пројекта, може аутоматски превести тај код у одговарајуће ентитете.

```
@Entity
@Data
@NoArgsConstructor
@AllArgsConstructor
@Builder

public class Predmet {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;

    @ManyToOne
    @JoinColumn(
            name = "idparcela",
            referencedColumnName = "id"
    )
    private Parcela parcela;

    @ManyToOne
    @JoinColumn(
            name = "idklijent",
            referencedColumnName = "id"
    )
    private Klijent klijent;

    @ManyToOne
    @JoinColumn(
            name = "idposao",
            referencedColumnName = "id"
    )
    private Posao posao;

    @Column
    private Date datum;

    @Column(nullable = false)
    private double cena;
```

```
    @Column(nullable = false)
    private double dug;

    @Column
    private boolean zavrsen;

}
```

Исечак кода 3.2 (Java) - Класа *Predmet* ентитет

Овај исечак кода је једн пример и представља *Java* класу која је мапирана на табелу у бази података користећи *JPA* анотације. Нећемо приказати остале кодове за ентитете. Класа *Predmet* има везе са другим ентитетима (*Parcela,Klijent,Posao*), што је приказано коришћењем анотација *@ManyToOne* и *@JoinColumn.* Анотација *@Entity* означава да је класа ентитет и да ће бити мапирана на табелу у бази података. Анотације из *@Loombok* библиотеке *(@Data, @NoArgsConstructor, @AllArgsConstructor, @Builder*) помажу у смањењу количине кода тако што аутоматски генеришу уобичајене методе и конструкторе.

Анотација *@Id* означава поље које представља примарни кључ табеле, док *@GeneratedValue(strategy = GenerationType.IDENTITY*) обезбеђује да база података аутоматски генерише вредност примарног кључа. Анотација *@Column* се користи за означавање поља која ће бити мапирана на колоне у табели, при чему се атрибути као што је *nullable = false* користе за додатна подешавања.

Анотација *@ManyToOne* означава везу "много-један" између ентитета, где више инстанци ентитета може бити повезано са једном инстанцом ентитета *Parcela, Klijent* или *Posao*. Анотација *@JoinColumn* дефинише колону која се користи за ову везу, указујући која колона у табели *Predmet* референцира колону у другим табелама.

Ова структура омогућава једноставно мапирање и манипулацију подацима у бази, обезбеђујући јасне и ефикасне везе између различитих ентитета у апликацији.

Табела *Predmet* садржи следећа поља:

- *id (long)* - примарни кључ *auto incremental* вредност која се аутоматски повећава;
- *datum (Date)* – представља датум када је предмет заведен. Поље се аутоматски додаје када се дода нови предмет на тренутни датум.
- *cena (double)* – представља цену датог предмета који клијент треба да плати у динарима, поље је обавезно.
- *zavrsen(Boolean)* – показује да ли је предмет завршен или није. Подразумевана вредност је да предмет није завршен.
- *dug (double)* – показује колико је клијент дужан за дати предмет и подразумевана вредност је једнака вредности цене датог предемта, поље је обавезно.
- *status(string)* – представља неки опис датог предмета. Поље није обавезно
- *idklijent (long)* – представља страни кључ из табеле *Klijent*, оно што је битно је да један клијент може да буде укључен у 0, 1 или више предмета.

- *idparcela (long)* – представља страни кључ из табеле *Parcela*, оно што је битно је да једна парцела може да буде укључена у 0, 1 или више предмета.

Табела *Posao* садржи следећа поља:

- *id (long)* - примарни кључ *auto incremental* вредност која се аутоматски повећава.
- *naziv(string)* -представља назив датог посла и он је јединствен и обавезан.

Табела *Radnik* садржи информације о радницима који користе ову апликацију. Она служи за смештање података о налозима датих радника и служи за пријављивање и аутентификацију датих радника. Табела садржи следећа поља:

- *id (long)* - примарни кључ *auto incremental* вредност која се аутоматски повећава.
- *ime (string)* - име радника, обавезно поље.
- *prezime (string)* - презиме радника, обавезно поље.
- *korisnicko_ime (string)* - корисничко име које је јединствено и обавезно.Потребно је приликом логовања датог радника, обавезно поље.
- *lozinka (string)* - лозинка радника, потребно приликом логовања, обавезно поље.
- *otpusten (boolean)* - ово поље нам говори да ли је дати радник отпуштен или не и то директно утиче на то да ли може да се пријави. Подразумевана вредност је *false*.
- *admin (boolean*) - ово поље нам говори да ли је радник админ и ако је админ, има веће привилегије од других радника.

Табела *Parcela* садржи све потребне информације о самој парцели, а то су следећа поља:

- *id (long)* - примарни кључ *auto incremental* вредност која се аутоматски повећава.
- *br_kat_parcele(string*)- може да сарджи само све цифре и „/“ с тим да не може почети цифром „0“,. Ово поље је обавезно.
- *kat_opstina(string)*- представља катастарску општину на којој се налази парцела. Ово поље је обавезно.
- *opstina(string)* - преставља општину на којој се налази парцела. Заједно са *kat_opstina* и *br_kat_parcele* чине јединствену комбинацију. Ово поље је обавезно.
- *ln(blob)* - представља лист непокретности парцеле који је типа *.pdf* . Ово поље није обавезно.

Табела *Klijent* садржи све битне информације о клијентима нашег система који могу бити повезани са одређеним предметом али и не мора. Ова табела садржи следећа поља:

- *id (long)* - примарни кључ *auto incremental* вредност која се аутоматски повећава.
- *ime (string)* - име клијента, обавезно поље.
- *prezime (string)* - презиме клијента, обавезно поље.
- *ime_oca (string)* - име оца клијента, обавезно поље.

- *jmbg(string)* – представља јединствени матични број грађанина који мора да садржи тачно 13 цифре и ово поље је јединствено и обавезно.
- *brlk(string)* – представља број личне карте грађанина који мора да садржи тачно 9 цифре и ово поље је јединствено и обавезно.
- *prebivaliste(string)* – адреса или место на коме одређени клијент живи. Ово поље је обавезно.
- *telefon(string)* – представља контакт одређеног клијента. Ово поље је обавезно.
- *troskovi(double)* – трошкови које дати клијент мора да плати у динарима.
- *datum_placanja(Date)* – датум када је клијент платио део или све своје трошкове.

Табела *Plati* чува информације када је плаћен неки део или цео износ који је био потребан за одређени предмет. Табела садржи следећа поља:

- *id (long)* - примарни кључ *auto incremental* вредност која се аутоматски повећава.
- *iznos (double)* -представља износ у динарима који је плаћен, обавезно поље.
- *datum (Date)* – представља датум када је плаћање извршено. Аутоматски се додаје и има вредност тренутног датума , када додајемо нови датум.
- *idpredemt (long)* – представља страни кључ из табеле *Predmet*. Један предмет може више пута да буде плаћен све док цео његов дуг не буде исплаћен.

Табела *Teren* описују када је одређени радник извршио терен за одређени предмет. Радник може да изврши терен само једном за одређени предмет. Табела садржи следећа поља:

- *id (long)* - примарни кључ *auto incremental* вредност која се аутоматски повећава;
- *datum (Date*) – датум када је радник извршио терен за одређени предмет, аутоматски се додаје на тренутан датум када додајемо нови терен.
- *idpredemt (long)* – страни кључ из табеле *Predmet*. Један предмет може да има само један терен , такође али може и да нема терен.
- *idradnik(long)* - страни кључ из табеле *Radnik*. Један радник може да извршава нула један или више терена.

Табела *Komentar* је она која садржи све коментаре који остављају посетиоци веб сајта и који оцењују услуге фирме. Табела садржи следећа поља:

- *id (long)* - примарни кључ *auto incremental* вредност која се аутоматски повећава;
- *ime (string)* – име особе која оставља коментар, обавезно поље;
- *prezime (string)* – презиме особе која оставља коментар, обавезно поље;
- *poruka (string)* – кратка порука коју оставља особа. Обавезно поље
- *ocena (int*) – број од 1-5 који престављa оцену . Обавезно поље.
- *datum (Date)* – представља датум када је коментар остављен. Аутоматски се додаје и има вредност тренутног датума када додајемо нови датум.

Табела *Zahtev* предтавља захтев за пријем новог корисника:

- *id (long)* - примарни кључ са *auto incremental* вредност која се аутоматски повећава.
- *ime (string)* – име особе која шаље захтев, обавезно поље.
- *prezime (string)* – презиме особе која шаље захтев, обавезно поље.
- *korisnicko_ime (string)* - корисничко име које је јединствено и обавезно.
- *lozinka (string*) - лозинка особе која шаље захтев, потребно приликом логовања, обавезно поље.

Табела *Trosak* садржи једна трошак датог систем који може да унесе само радник који има привилегију , тј.радник који је админ. Табела садржи следећа поља:

- *id (long)* - примарни кључ са *auto incremental* вредност која се аутоматски повећава.
- *datum (Date)* – датум када је трошак додат. Аутоматски се додаје и има вредност тренутног датума , када додајемо нови датум.
- *iznos (double)* – представља износ у динарима која је неопходна за овај трошак, обавезно поље.
- *opis (string)* – представља кратак опис због чега је трошак настао. Ово поље није обавезно.

3.3 Повезивање базе

Повезивање на базу података обавља се кроз конфигурациони фајл *application.properties*, како се може видети у следећем исечку кода:

```
spring.jpa.hibernate.ddl-auto=update
spring.datasource.url=jdbc:mysql://localhost:3306/cagi
spring.datasource.username=root
spring.datasource.password=Nemanja123!!!!
spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver


spring.jpa.show-sql=true
```

Исечак кода 3.3 Повезивање базе

Ево шта представља дати исечак кода:

- *spring.jpa.hibernate.ddl-auto=update*: Аутоматски ажурира шему базе података на основу *JPA* ентитета.
- *spring.datasource.url: URL MySQL* базе података.
- *spring.datasource.username:* Корисничко име за базу података.
- *spring.datasource.password*: Лозинка за базу података.
- *spring.datasource.driver-class-name*: Класа драјвера за *MySQL*.
- *spring.jpa.show-sql=true*: Омогућава приказивање *SQL* упита у конзоли.

4.ТЕХНОЛОГИЈЕ И ИМПЛЕМНТАЦИЈА

У овом делу ћемо детаљно описати развојно окружење и алате који су коришћени за имплементацију апликације користећи *Spring Boot* за серверску страну и *Angular* за клијентску страну.

Избор *Spring Boot* за *backend* део апликације образложен је његовом популарношћу и широком подршком у развојној заједници. *Spring Boot* пружа ефикасно окружење за развој, обезбеђујући брзину и сигурност. Интеграција са различитим сервисима и базама података је лака, што олакшава дизајн и склопљење комплексних апликација. Уз то, *Spring Boot* омогућава перформансе и скалабилност које су кључне за модерне веб апликације. Коришћењем *Angular* за *frontend*, омогућујемо развој интерактивних и употребљивих корисничких интерфејса који одговарају стандардима савременог веба.

4.1 Spring Boot

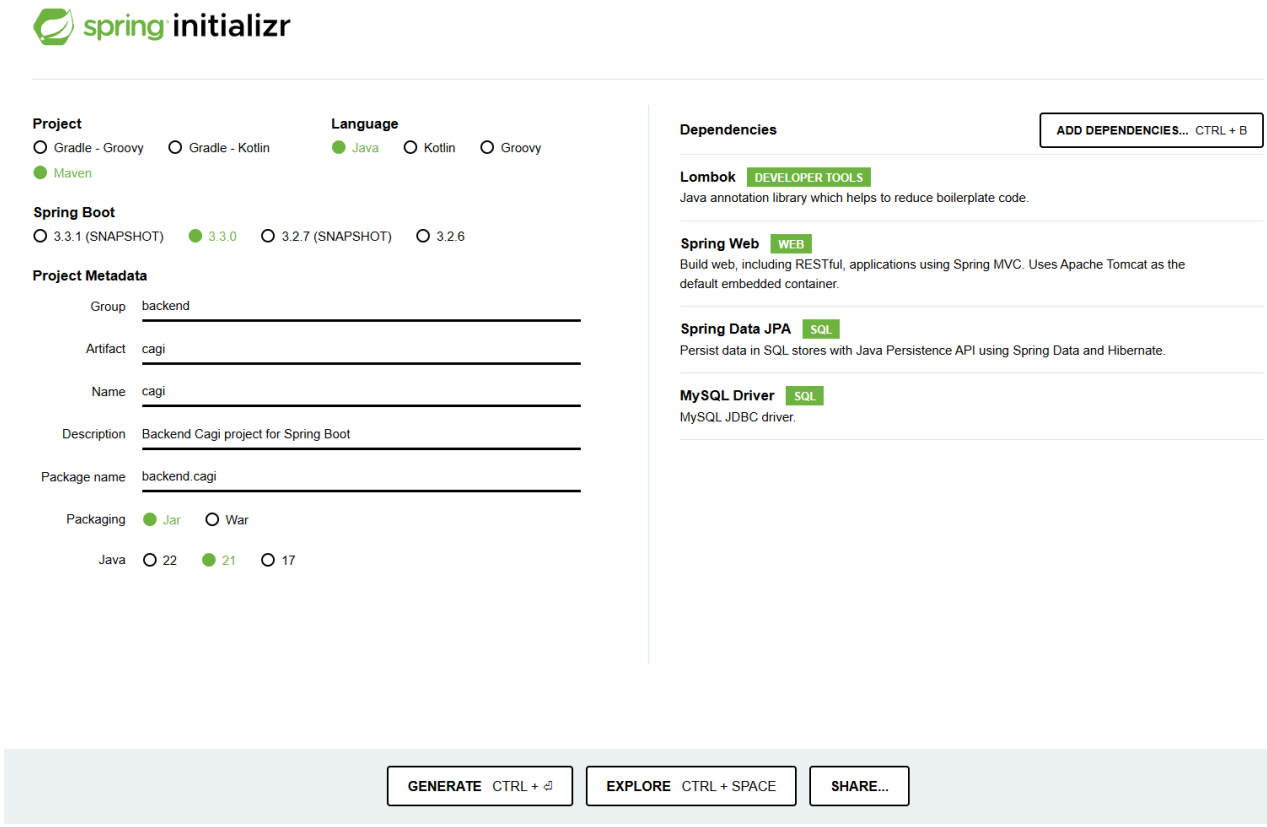
Spring Boot је модеран *Java* оквир који олакшава развој самосталних, производних апликација заснованих на *Spring* екосистему. Дизајниран је да поједностави конфигурацију и постављање апликација, омогућавајући програмерима да се фокусирају на пословну логику. Користећи приступ "конвенције над конфигурацијом", *Spring Boot* аутоматски подешава потребне компоненте, смањујући време потребно за ручну конфигурацију.

Развој *Spring* апликације је извршен у развојном окружењу *IntelliJ IDEA*, који пружа алате за продуктивност и једноставну интеграцију са *Spring Boot* оквиром (*frame*). *IntelliJ IDEA* омогућава лако управљање зависностима, аутоматско комплетирање кода и алате за дебаговање који су били од велике помоћи током развоја апликације.

4.1.1 Почетно стање пројекта

Почетно стање пројекта је генерисано на сајту https://start.spring.io/ . Као што се види на слици испод могуће је бирати:

- Пројекат - *Maven*
- Програмски језик – *Java*
- *Spring boot* верзија– 3.3.0
- *Java JDK* верзија– 3.3.0
- Dependencies:
  - *Loombook* :*Java* библиотека за анотације која помаже у смањењу шаблонског кода.
  - *Spring Web* : Користи се за изградњу веб апликације, укључујући *RESTful*, користећи *Spring MVC*. Као подразумевани уграђени контејнер користи *Apache Tomcat*.
  - *Spring DATA JPA* : Користи се за чување података у *SQL* користећи *JPA* користећи *Spring Data* и *Hibernate*.
  - *MySQL Driver* : *MySql JDBC driver*



Слика 4.1.1 Иницијализација *Spring* пројекта

4.1.2 Архитектура Spring пројекта

Структура пројекта прати архитектуру *model-view-controller* и она се може видети у програмском исечку испод.

```
demo
├── src
│   ├── main
│   │   ├── java
│   │   │   └── backend
│   │   │       └── cagi
│   │   │           └── controller
│   │   │           └── entitity
│   │   │           └── repository
│   │   │           └── service
│   │   │           ├── CagiApplication.java
│   │   │
│   │   │
│   │   └── resources
│   │       ├── application.properties
│   └── test
│       └── java
│           └── backend
│               └── cagi
│                   └── CagiApplicationTests.java
├── mvnw
├── mvnw.cmd
├── pom.xml
└── README.md
```

Исечак кода 4.1.2 Структура *Spring* пројекта

У архитектури *Spring* апликације, постоје четири фолдера, сваки са својим респективним компонентама:

1) *Entity* – овде се налазе класе које представљају табеле из базе података и њихове
2) *Repository* – у овом фолдеру се налазе класе које представљају репозиторијуме и служе за комуникацију са базом података
3) *Service* – овај фолдер садржи класе са функционалностима целе апликације
4) *Controller* – овде се налазе класе које представљају *REST* контролере са *REST* методама

Нећемо приказивати све фајлове из фолдера *Repository* , али даћемо један пример интерфејс *PosaoRepository*:

```
@Repository
public interface PosaoRepository extends JpaRepository<Posao,Long> {


   Posao findPosaoByNaziv(String naziv);


 @Query("SELECT p FROM Predmet p WHERE p.posao.id = :posaoId")
 List<Predmet> findPredmetiByPosaoId(@Param("posaoId") Long posaoId);


}
```

Исечак кода 4.1.2 (Java) - *Repository* интефејс *PosaoRepository*

Исечак кода представља *Java* интерфејс *PosaoRepository* који је анотиран са *@Repository*, чиме се декларише као репозиторијум компоненте у *Spring*-у. Интерфејс проширује *JpaRepository*, омогућавајући *CRUD* операције за ентитет *Posao* са идентификатором типа *Long*. Метод *findPosaoByNaziv* омогућава проналажење ентитета *Posao* по његовом називу. Додатно, метод *findPredmetiByPosaoId* користи прилагођени *JPQL* упит за проналажење свих ентитета *Predmet* који су повезани са задатим *Posao* путем идентификатора, при чему се користе *@Query* и *@Param* анотације за дефинисање и параметризацију упита.

Нећемо приказивати све фајлове из фолдера *Service*, али даћемо један пример класу *PosaoService*:

```
@Transactional
@Service
public class PosaoService {


    @Autowired
    private PosaoRepository posaoRepository;


    public List<Posao> getAllPoslovi(){
        return posaoRepository.findAll();
    }
    public Posao savePosao(Posao p){
        Posao pp=posaoRepository.findPosaoByNaziv(p.getNaziv());
        if(pp!=null){
            return pp;
        }
        return  posaoRepository.save(p);
```

```
    }
    public String removePosao(Long id){
        posaoRepository.deleteById(id);
        return "Uspesno obrisan posao";
    }
    public Posao updatePosao(Posao posao) {
        Posao posaoB=posaoRepository.findById(posao.getId()).get();
        if(!posao.getNaziv().equals("")){
            posaoB.setNaziv(posao.getNaziv());
        }
        return posaoRepository.save(posaoB);
    }
    public List<Predmet> getPredmetiByPosaoId(Long id) {
        return posaoRepository.findPredmetiByPosaoId(id);
    }
}
```

Исечак кода 4.1.2 (Java)- *Repository* класа *PosaoRepository*

@Transactional - Означава да ће методе у овом сервисном слоју бити управљане трансакцијама, омогућавајући аутоматско управљање трансакцијама од стране *Spring Framework-a*.

@Service - Анотација која означава да је класа *Spring bean* који треба да буде управљан од стране *Spring* контењера, обично се користи за означавање сервисних компоненти.

Дађемо један пример из фолдера *Controller* , класу *PosaoController*:

```
@RestController
@CrossOrigin(origins="http://localhost:4200/")
public class PosaoController {
    @Autowired
    private PosaoService posaoService;

    @GetMapping(path = "/poslovi") public List<Posao>getAllPoslovi(){
     return posaoService.getAllPoslovi();
 }
 @PostMapping(path = "/poslovi/dodaj")
 public Posao savePosao(@Valid @RequestBody Posao posao){
     return posaoService.savePosao(posao);
 }
 @PostMapping(path = "/poslovi/promeni")
 public Posao updatePosao(@Valid @RequestBody Posao posao){
```

```
            return posaoService.updatePosao(posao);
        }
        @DeleteMapping (path = "/poslovi/izbrisi/{id}")
        public String removePosao(@Valid @PathVariable("id") Long id){
            return posaoService.removePosao(id);
        }
        @GetMapping("/{id}/posao/predmeti")
        public List<Predmet> getPredmetiByKlijentId(@PathVariable Long id) {
            List<Predmet> predmeti = posaoService.getPredmetiByPosaoId(id);
            return predmeti;
        }
    }
```

Исечак кода 4.1.2 (Java) - *Controller* класа *PosaoController*

@RestController - Означава да је *PosaoController Spring* компонента која обрађује *HTTP* захтеве и генерише одговоре. *@CrossOrigin(origins="http://localhost:4200/")* омогућава приступ контролеру са локације *http://localhost:4200/* путем *CORS-a.* Методе у овом контролеру омогућавају добијање свих послова, додавање новог, ажурирање и уклањање постојећег посла, као и добијање предмета везаних за одређени послов. Методе се мапирају на одговарајуће *HTTP* методе путем анотација *@GetMapping*, *@PostMapping*, *@DeleteMapping*, док се *@Valid* користи за валидацију параметара метода.

4.2 Angular

Angular је један од најпопуларнијих *JavaScript framework*-а који је развјен од стране *Google*-а за израду динамичких веб апликација. Омогућава развој *single-page* апликација (*SPA*) користећи архитектуру *Model-View-Controller* (*MVC*). *Angular* користи *TypeScript*, што доноси статичку типизацију и напредне развојне могућности. Интегрише се са разним алатима и библиотекама, олакшавајући тестирање и одржавање кода. Пружа снажан систем директива и компоненти за модуларно и поново употребљиво програмирање. Због своје робусности и флексибилности. *Angular* је широко прихваћен међу програмерима за велике и сложене пројекте.

4.2.1 Инсталација и пакети

Командни интерфејс за *Angular* се инсталира помоћу команде: *npm install -g @angular/cli*. Ако желимо да покренемо апликацију на клијентској страни потребно је откуцати команду *ng serve* и онда ће наша апликација бити доступна на адреси http://localhost:4200/ или на другој адреси ако је ова адреса заузета.

Пакети који се користе у овој апликацији:

1) *Angular Material* – библиотека која обезбеђује готове компоненте и стилове за *Angular*, као што су дугмад, картице, дијалози, табеле и многе друге компоненте. Инсталира се командом: *ng add @angular/material.*
2) *Bootstrap – CSS* и *JavaScript* оквир за креирање респонзивних и атрактивних веб страница. Инсталира се командом: *npm install bootstrap*.
3) *Animate.css* – библиотека за анимације која се користи за лако додавање анимација у веб пројекте. Инсталира се командом: *npm install animate.css*.
4) *Easing* – библиотека за креирање анимација са различитим функцијама еасинга. Инсталира се командом: *npm install easing*.
5) *Owl Carousel – jQuery carousel plugin* који је интегрисан са *Angular*-ом. Инсталира се командом: *npm install ngx-owl-carousel-o*.
6) *Waypoints* – библиотека која омогућава покретање функција када скроловање дође до одређених тачака. Инсталира се командом: *npm install waypoints*.
7) *WOW.js* – библиотека која омогућава приказивање анимација при скроловању. Инсталира се командом: *npm install wowjs*.

Ови пакети су кључни за креирање модерног, респонзивног и интерактивног корисничког интерфејса у *Angular* апликацији. Омогућавају лако додавање стилова, анимација и функционалности које побољшавају корисничко искуство.

4.2.2 Компоненте (Components)

Компонента је основна градивна јединица у *Angular* апликацијама, која обухвата *HTML template* за структуру странице, *TypeScript* класу која дефинише логику и понашање компоненте, као и *CSS* стилове који примењују дизајн и изглед странице.

Компонента се у *Angularu* обележава са декоратором *@Component()* у којем се дефинише јединствени селектор који идентификује компоненту у *HTML* документу. Овај селектор омогућава уграђивање компоненте у друге делове апликације и приказивање њеног садржаја.

За генерисање компоненте у *Angular*-у користите *Angular CLI*. Потребно је отворити терминал и употребите команду *ng generate component naziv-komponente* , замењујући *naziv-komponente* жељеним именом нове компоненте.

4.2.3 Рутери (Routing)

У *Angular* апликацијама, рутер омогућава навигацију између различитих делова апликације користећи *URL* адресе. Рутер управља која ће се компонента приказивати када корисник посети одређени пут у апликацији. Дефинисање рута у *Angular*-у се ради креирањем објекта са путањама и одговарајућим компонентама, а затим конфигурисањем ових путања у модулу апликације. Пример конфигурације рута можемо видети на следећем исечку кода:

```
const routes: Routes = [
  { path: "", component: MainComponent },
  { path: "registerUser", component: RegisterUserComponent },
  { path: "login", component: LoginComponent },
  { path: "meni", component: MeniComponent },
  { path: "header", component: HeaderComponent },
  { path: "poslovi", component: PosloviComponent },
  { path: "radnici", component: RadnikComponent },
  { path: "klijenti", component: KlijentiComponent },
  { path: "parcele", component: ParceleComponent },
  { path: "katastar", component: KatastarComponent },
  { path: "predmeti", component: PredmetiComponent },
  { path: "blagajna", component: BlagajnaComponent }
];

@NgModule({
  imports: [RouterModule.forRoot(routes)],
  exports: [RouterModule]
})
export class AppRoutingModule { }
```

Исечак кода 4.2 3 (TypeScript) – рутирање *app-routig.module.ts*

Ова конфигурација се налази у *app-routing.module.ts* фајлу. Поред тога, да би апликација функционисала потребно је да *y index.html* фајлу буде следећи таг:

```
<app-root></app-root>
```

Исечак кода 4.2.3 (HTML) - рутирање

Ово је коренски елемент који *Angular* користи за учитавање и приказивање апликације. Укључивање свих ових делова осигурава да је апликација правилно конфигурисана и спремна за навигацију и приказивање садржаја.

4.2.4 Модели (Modules)

У *Angular* апликацијама, модели представљају структуре података и служе за дефинисање и управљање подацима које апликација користи. Модели су обично дефинисани као *TypeScript* класе и садрже својства и методе које описују податке и понашање тих података у апликацији. У овој апликацији, сви модели су смештени у фолдеру *models*.

Следећи исечак кода представља пример модела који се налази у фајлу *Klijent.ts*.

```
export class Klijent {
    id: number;
    ime: string;
    prezime: string;
    ime_oca: string;
    jmbg: string;
    telefon: string;
    pribivaliste: string;
    brlk: string;
}
```

Исечак кода 4.2.4 (TypeScript) – Модел класа klijent.ts

4.2.5 Сервиси (Services)

У *Angular* апликацијама, сервиси се користе за обраду логике која није директно повезана са приказом (*UI*), као што су приступ подацима, комуникација са сервером и руковање пословном логиком.

Сервис се може креирати помоћу *Angular CLI* алата следећом командом: *ng generate service naziv-servisa* .

Следећи исечак кода представља пример како се може креирати сервис у *Angular*-у и користећи *HttpClient* за приступ *API*-ју:

```
@Injectable({
  providedIn: 'root'
})
export class PosaoService {

  private readonly uri = 'http://localhost:8080';

  constructor(private http: HttpClient) { }

  getAllPoslovi(): Observable<Posao[]> {
    return this.http.get<Posao[]>(`${this.uri}/poslovi`);
  }
  dodajPosao(naziv: string): Observable<Posao> {
    const data = { naziv };
    return this.http.post<Posao>(`${this.uri}/poslovi/dodaj`, data);
  }
```

Исечак кода 4.2.5 (Typescript) – Сервис класа PosaoService.service.ts

Из датог исечка кода видимо да променљива *uri* дефинише базни *URL* за *API*. У овом случају то је http://localhost:8080 .

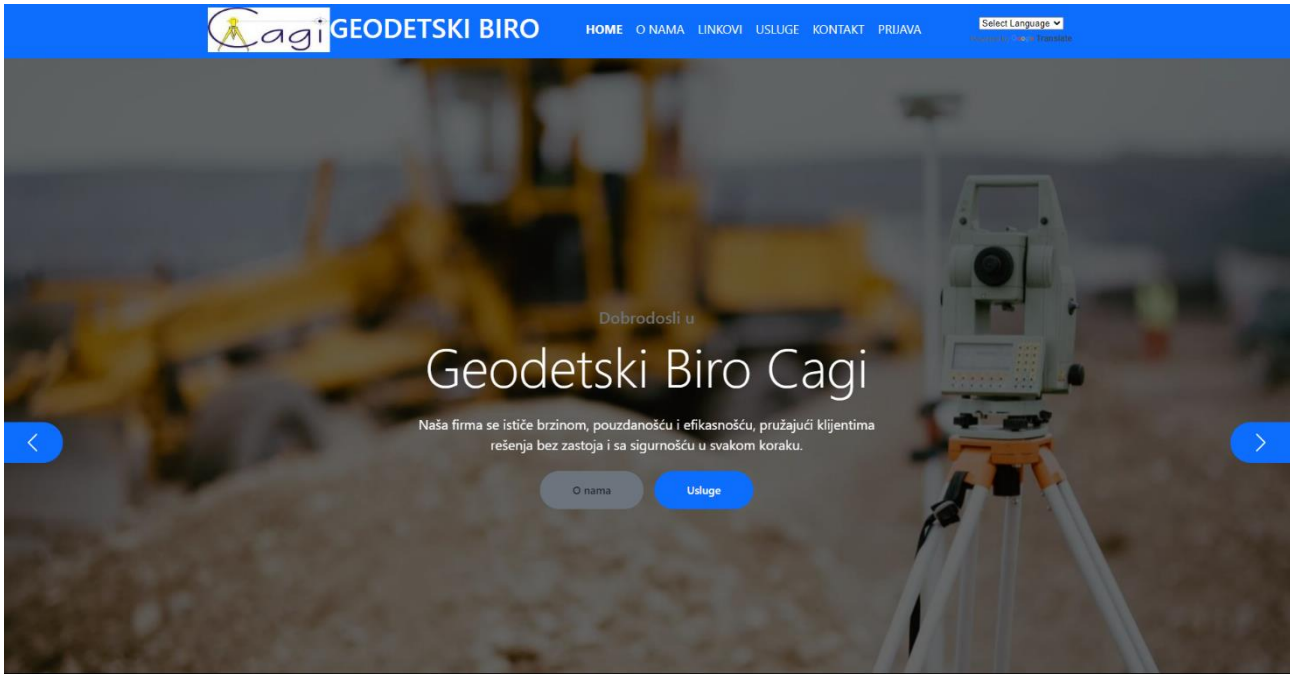
5.ФУНКЦИОНАЛНОСТИ АПЛИКАЦИЈЕ

У следећем поглављу биће приказане главне функционалности датог пројекта. Због комплексности самог пројекта немогуће је приказати и детаљно описати све функционалсти апликације. Такође биће приказани и УМЛ дијаграми, кодови и слике како изгледа сама апликација.

5.1 Презентовање фирме

5.1.1. Почетна страница, мени

На следећој слици се види почетна страница заједно са навигацијом . Навигациони мени омогућава корисницима лако кретање кроз различите секције сајта, као што су „О нама", „Услуге" и „Контакт". Такође види се и могућност бирања језика што је за ову фирму веома битно. *Carousel* са сликама пружа визуелно привлачан начин за представљање главних услуга и вредности фирме, укључујући брзу и поуздану услугу.



Слика 5.1.1 Почетна страница, мени

```
      <div class="container-fluid bg-primary">
          <div class="container">
              <nav class="navbar navbar-dark navbar-expand-lg py-0">
<div class="container-fluid bg-primary">
    <div class="container">
```

```
        <nav class="navbar navbar-dark navbar-expand-lg py-0">
       <img src="assets/cagi.jpg" alt="Logo" class="logo">
            <a href="index.html" class="navbar-brand">
             <h1 class="text-white d-block">GEODETSKI BIRO <span class="text-
secondary"></span></h1>
            </a>
<button type="button" class="navbar-toggler me-0" data-bs-toggle="collapse"
data-bs-target="#navbarCollapse">
             <span class="navbar-toggler-icon"></span>
            </button>
       <div class="collapse navbar-collapse bg-transparent"
id="navbarCollapse">
               <div class="navbar-nav ms-auto mx-xl-auto p-0">
             <a href="index.html" class="nav-item nav-link active">HOME</a>
       <a href="#o-nama-section" class="nav-item nav-link">O NAMA</a>
       <a href="#linkovi-section" class="nav-item nav-link">LINKOVI</a>
             <a href="#usluga-section" class="nav-item nav-link">USLUGE</a>
       <a href="#kontakt-section" class="nav-item nav-link">KONTAKT</a>
             <a href="login.html" class="nav-item nav-link">PRIJAVA</a>
               </div>
            </div>
            <div id="google_translate_element"></div>
        </nav>
    </div>
</div>
```

Исечак кода 5.1.1 (HTML) - Мени

5.1.2 О нама, линкови, услуге и контакт

- О нама - На следећој слици може се видети изглед дела странице који даје више информација о самој фирми. До овог дела може се доћи кликом на дугме „*О нама*“ који се налази на навигационом менију.

O NAMA

O NAMA

Geodetski biro Cagi je licencirani geodetski biro koji se sastoji od tima stručnjaka sa bogatim iskustvom u radu sa najsavremenijim geodetskim softverima i instrumentima. Naša primarna delatnost je precizno snimanje, obrada podataka i kvalitetna izrada geodetskih podloga za potrebe projektovanja. Svaka usluga koju pružamo se razlikuje po obimu posla, a većina usluga zahteva i završavanje administrativnih poslova u službi za katastar nepokretnosti.

U našim prostorijama pružamo besplatne informacije o Vašim nepokretnostima, kao i profesionalnu tehničku podršku i savete u oblasti geodetskih usluga. Spremni smo da svim strankama odgovorimo brzo i kvalitetno, što nas čini jedinstvenim na ovom tržištu.

Слика 5.1.2 О нама

- Линкови – На следећој слици можемо да видимо најбитније линкове који су везани за геодезију и који ће корисницима бити потребни. До овог дела може се доћи кликом на дугме „*Linkovi*“ који се налази на навигационом менију. Приликом клика на дати линк идемо на страницу која је дата у називу линка.

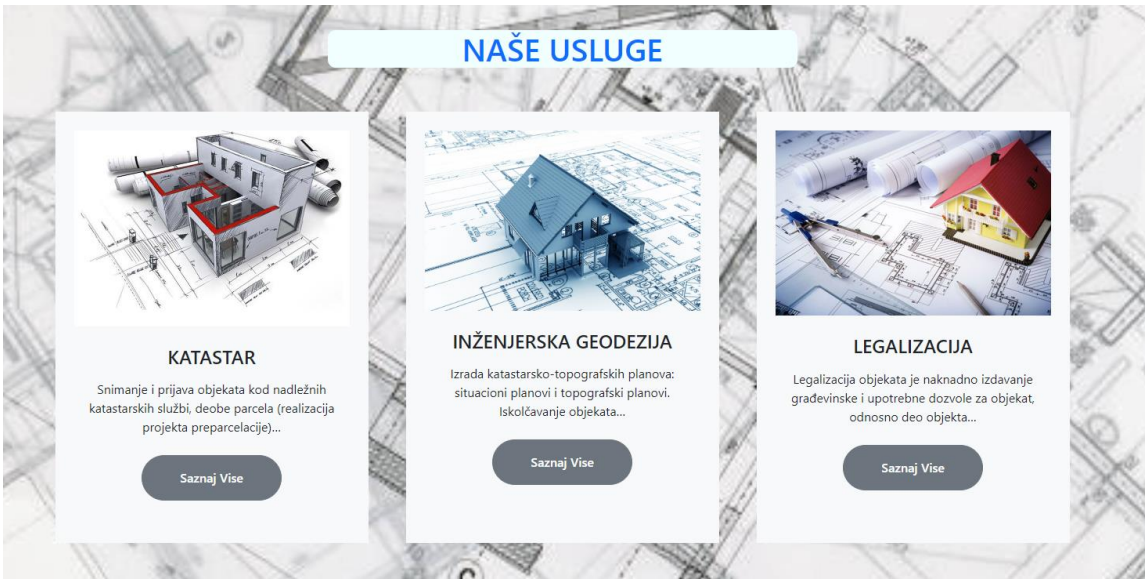
Слика 5.1.2 Линкови

```
      <div class="elementor-element elementor-element-d2601df elementor-
position-left elementor-vertical-align-top elementor-widget elementor-widget-
image-box" data-id="d2601df" data-element_type="widget" data-
widget_type="image-box.default">
          <div class="elementor-widget-container">
              <div class="elementor-image-box-wrapper">
              <figure class="elementor-image-box-img">
                  <a href="https://geosrbija.rs/">
        <img width="165" height="147" src="geosrbija.png" alt="GEO Srbija">
```

```
                </a>
            </figure>
            <div class="elementor-image-box-content">
                <h3 class="elementor-image-box-title">
                    <a href="https://geosrbija.rs/">GEO Srbija</a>
                </h3>
                <p class="elementor-image-box-description">
                    Nacionalna veb GIS aplikacija omogućava prikaz,
pretraživanje, analizu, transformaciju, kreiranje, deljenje i održavanje
geoprostornih podataka Srbije.
                </p>
            </div>
        </div>
    </div>
 </div>
```

Исечак кода 5.1.2 (HTML) – Линк Геосрбија

- Услуге - На следећој слици можемо да видимо услуге које фирма нуди. До овог дела може се доћи кликом на дугме „*Usluge*“ који се налази на навигационом менију. Приликом клика на дугме „*Saznaj vise“* могуће је видети више информација о тој услузи.



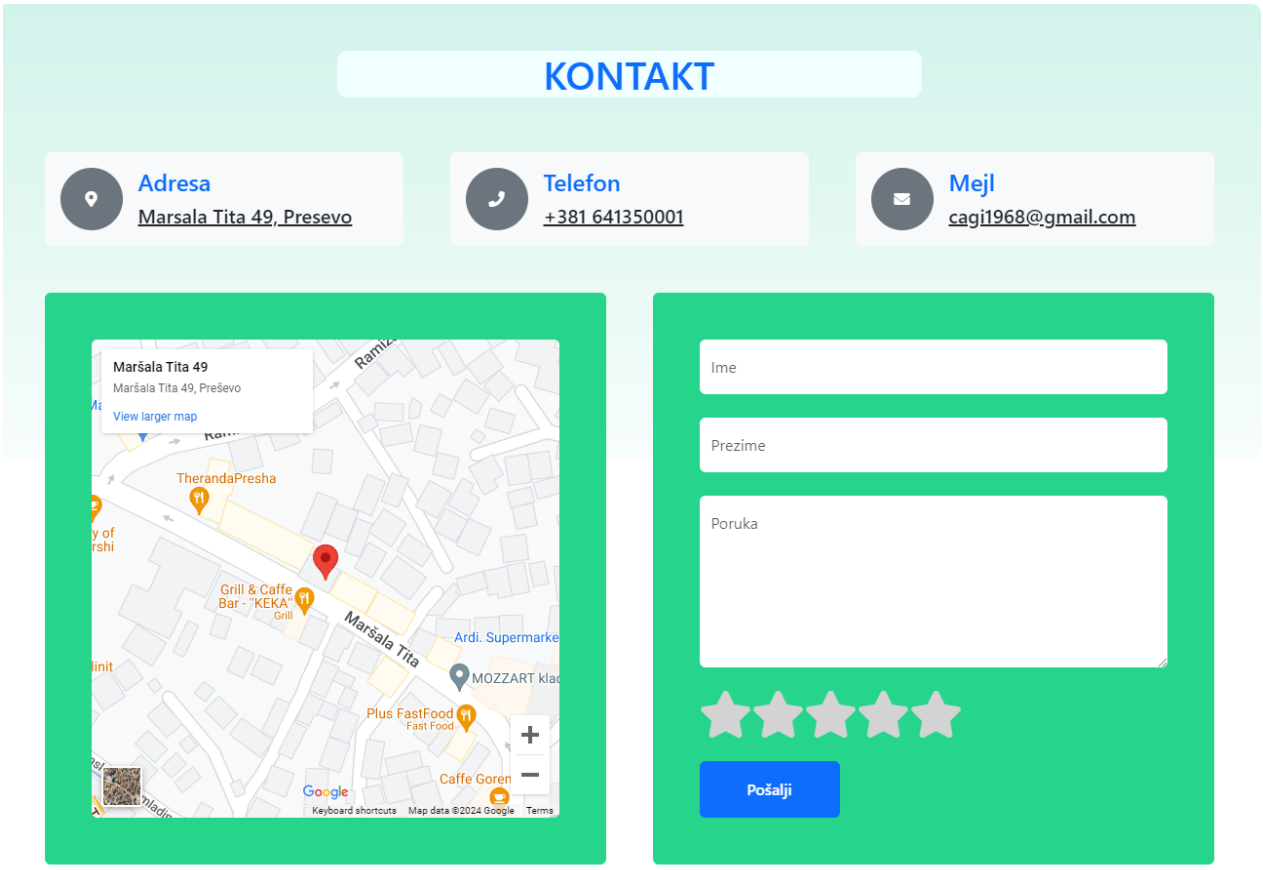
Слика 5.1.2 Услуге

```
  <div class="col-md-6 col-lg-4 wow fadeIn" data-wow-delay=".5s">
<div class=" bg-light">
 <div class="p-4 text-center services-content"  style="min-height: 550px;">
```

```
        <div class="services-content-icon">
  <img src="assets/usluge1.jpg" alt="Opis slike" class="img-fluid mb-4">
              <h4 class="mb-3">INŽENJERSKA GEODEZIJA</h4>
<p class="mb-4">Izrada katastarsko-topografskih planova: situacioni planovi i topografski planovi. Iskolčavanje objekata…</p>
              <a href="#modalUsluge2" class="btn btn-secondary text-white px-5 py-3 rounded-pill"  data-toggle="modal">Saznaj Vise</a>
        </div>
  </div>
  </div>
</div>
```

Исечак кода 5.1.2 (HTML) – Услуга инжењерска геодезија

- Контакт – На следећој слици можемо да видимо контакт фирме на веома леп визуалан начин. Ово је битно како би корисници имали контакт и информације где се налази сама фирма. Такође на овај део странице можемо да дођемо кликом на дугме „*Kontakt*“ који се налази на навигационом менију.



Слика 5.1.2 Контакт

```
        <div id="kontakt-section" class="container-fluid py-5 mb-5">
          <div class="container">
            <div class="contact-detail position-relative p-5">
              <div class="text-center mx-auto pb-5 wow fadeIn" data-wow-
delay=".3s" style="max-width: 600px;">
                <h1 class="text-primary" style="border-radius: 10px;
background-color: azure;">KONTAKT</h1>
              </div>
              <div class="row g-5 mb-5 justify-content-center">
                <div class="col-xl-4 col-lg-6 wow fadeIn" data-wow-delay=".3s">
                  <div class="d-flex bg-light p-3 rounded">
                    <div class="flex-shrink-0 btn-square bg-secondary rounded-
circle" style="width: 64px; height: 64px;">
                      <i class="fa fa-map-marker-alt text-white"></i>
                    </div>
                    <div class="ms-3">
                      <h4 class="text-primary">Adresa</h4>
                      <a
href="https://www.google.com/maps/place/Mar%C5%A1ala+Tita+49,+Pre%C5%A1evo/@42.
3097865,21.6481377,19z/data=!4m6!3m5!1s0x1354f7e22c1066c5:0x26f38c144fe0d814!8m
2!3d42.3097961!4d21.6486153!16s%2Fg%2F11y236f762?authuser=1&entry=ttu"
target="_blank" class="h5">
                        Marsala Tita 49, Presevo
                      </a>
                    </div>
                  </div>
                </div>
                <div class="col-xl-4 col-lg-6 wow fadeIn" data-wow-delay=".5s">
                  <div class="d-flex bg-light p-3 rounded">
                    <div class="flex-shrink-0 btn-square bg-secondary rounded-
circle" style="width: 64px; height: 64px;">
                      <i class="fa fa-phone text-white"></i>
                    </div>
                    <div class="ms-3">
                      <h4 class="text-primary">Telefon</h4>
                      <a class="h5" href="tel:+381 641350001"
target="_blank">+381 641350001</a>
                    </div>
                  </div>
                </div>
                <div class="col-xl-4 col-lg-6 wow fadeIn" data-wow-delay=".7s">
                  <div class="d-flex bg-light p-3 rounded">
                    <div class="flex-shrink-0 btn-square bg-secondary rounded-
circle" style="width: 64px; height: 64px;">
```

```
                    <i class="fa fa-envelope text-white"></i>
                  </div>
                  <div class="ms-3">
                    <h4 class="text-primary">Mejl</h4>
                    <a class="h5" href="mailto:cagi1968@gmail.com"
target="_blank">cagi1968@gmail.com</a>
                  </div>
                </div>
              </div>
            </div>
```

Исечак кода 5.1.2 (HTML) – Контакт

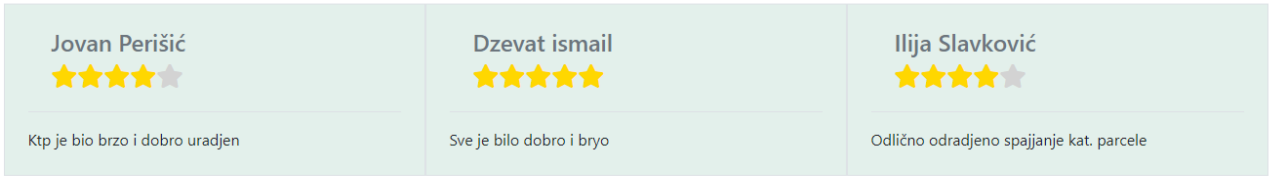
5.1.3 Коментари

Оно што је такође веома битно за фирму је могућност остављања коментара корисника и читање других коментара и оцена. Ова функционалност је значајна јер помаже у стварању заједнице и повећава ангажовање корисника.

Приказаћемо следеће функционалности које су везане за коментаре:

- Приказивање коментара
- Додавање коментара

На слици (Слика 5.1.3) се може видити део где корисник може видети коментаре који су оставили други корисници као и оцене које су корисници оставили



Слика 5.1.3 Коментари

На слици (Слика 5.1.2 Контакт) се може видети део како изгледа форма коју корисник треба да попуни како би оставио коментар. Неопходно је да остави своје име, презиме, поруку као и оцену.

Приликом клика на дугме *Posalji* извршиће се функција *dodajKomentar* (Исечак кода 5.1.3) из фајла *main.ts* , који служи за додавање коментара.

```
        dodajKomentar() {
         this.proveraa = true;
         this.komentarService.dodajKomentar(
           this.komentar.ime,
           this.komentar.prezime,
```

```
        this.komentar.poruka,
        this.komentar.ocena
      ).subscribe(response => {
        console.log('Komentar uspešno dodat:', response);
        alert("Uspešno ostavljen komentar");
        this.komentar = new Komentar();
        this.refreshComments();
      }, error => {
        console.error('Greška prilikom dodavanja komentara:', error);
        alert("Greška: ne može da se doda komentar");
      });
    }
```

Исечак кода 5.1.3 (TypeScript) – Функција *dodajKomentar() main.ts*

Следећи исечак кода представља функцију *dodajKomentar* из сервиса *komentarService.* Сервис *komentarService* шаље *HTTP POST* захтев серверу са подацима о коментару.

```
  dodajKomentar( firstnameForm, lastnameForm,messageForm,gradeForm,){
      const data = {
        ime: firstnameForm,
        prezime: lastnameForm,
        poruka: messageForm,
        ocena: gradeForm
      }
      return this.http.post<Komentar>(`${this.uri}/komentar`, data);
    }
```

Исечак кода 5.1.3 (TypeScript) – Функција *dodajKomentar() komentarService*

На серверској страни за чување података о коментрима имплементиран је следећи код и ту је приказано прихватање *POST* захтева за коментар:

```
  @PostMapping(path = "/komentar")
    public Komentar saveKomentar(@Valid @RequestBody Komentar komentar){
        return komentarService.saveKomentar(komentar);
    }
```

Исечак кода 5.1.3 (Java) – Функција *saveKomentar() controller*

Можемо видети и како је имплементирана функција *saveKomentae* у фајлу *komentarService.* Овим смо успешно сачували коментар који је корисник додао.

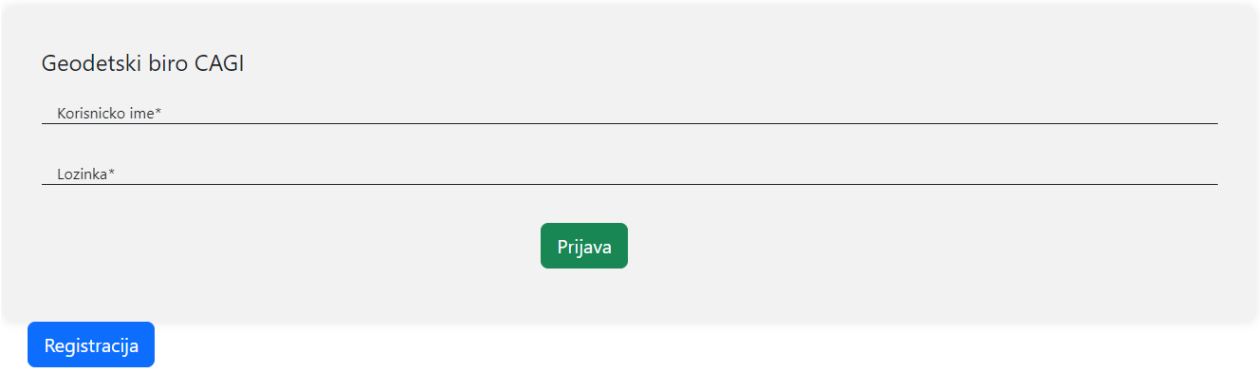
```
    public Komentar saveKomentar(Komentar komentar) {
        komentar.setDatum(Date.valueOf(LocalDate.now()));
        return komentarRepository.save(komentar);
    }
```

Исечак кода 5.1.3 (Java) – Функција *saveKomentar() service*

5.2 Радници

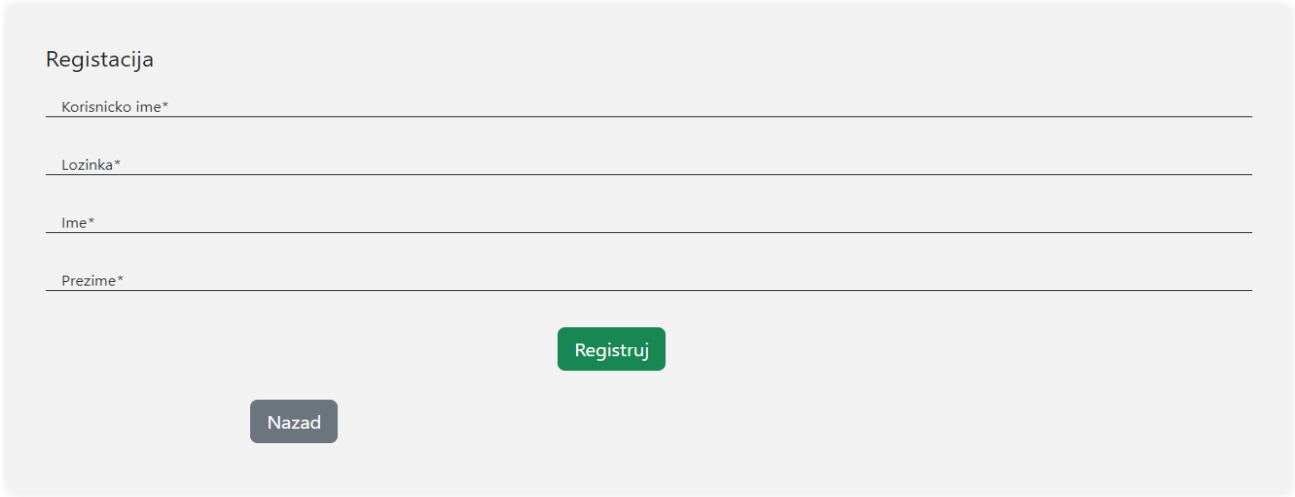
Овде ћемо приказате неке основне функционалности које имају радници фирме, а то су пријављивање затим измена налога, као и регистрацију новог радника.

- Пријава - Особа која је радник у фирми моћи ће да се пријави преко форме која је приказана на следећој слици . Уколико је пријава била успешна , налог постоји са том шифром, онда ће дати радник бити успешно пријављен, у супротном ће бити приказана порука да није могуће пријавити датог радника. Пријава се врши кликом на дугме „*Prijava*“.

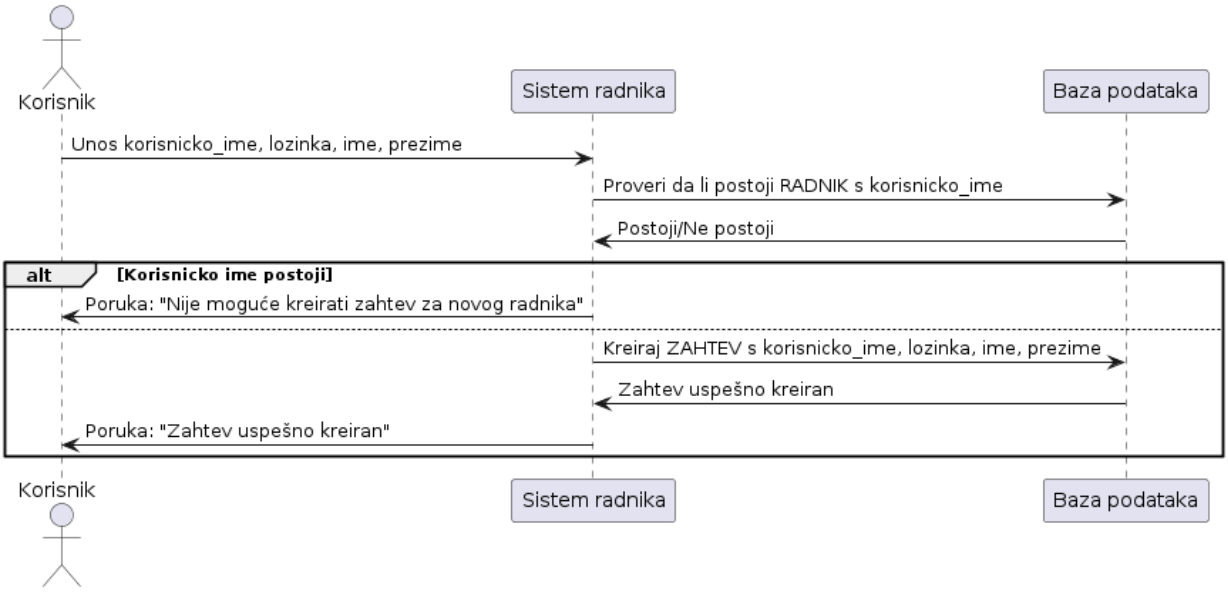


Слика 5.2 Форма за пријављивање

- Радници - Особа која нема налог а жели да има може кликом на дугме „*Registracija*“ (Слика 5.2 форма за пријављивање) да попуни форму и пошаље захтев који админ треба да потврди. Да би регистрација била успешна потребно је да налог са тим корисничким именом не постоји .



Слика 5.2 Форма за регистрацију



Слика 5.2 УМЛ дијаграм за регистрацију радника

```
    public Radnik getRadnikByKorisnickoImeILozinka(String korisnicko_ime,
String lozinka) {
        Radnik r =
radnikRepository.findRadnikByKorisnickoImeILozinka(korisnicko_ime, lozinka);
        if (r == null) {
            return null;
        } else if (r.isFired()) {
            return null;
        } else {
            return r;
        }
    }
```

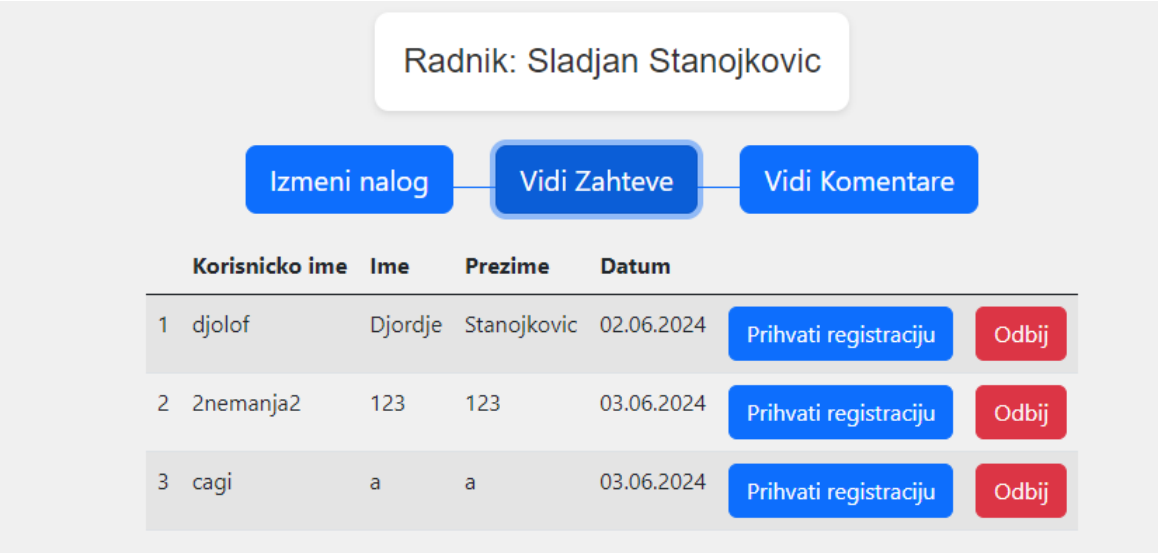
Исечак кода 5.2 (Java) – Функција *getRadnikByKorisnickoImeILozinka*

5.3 Admin

Admin je radnik koji ima veće privilegije. On se u sistemu razlikuje od običnog radnika tako što njegovo polje *admin* ima vrednost *true* . On može da dodaje nove radnike, zatim da otpusti određenog radnika, a takođe ima mogućnost da vidi sve prihode i troškove za određeni mesec , kao i da doda određeni trošak.

5.3.1 Prihvatanje zahteva

Na sledećoj slici se vidi tabela zahtева koji admin može da prihvati ili odbije .



Slika 5.3.1 Zahtevi

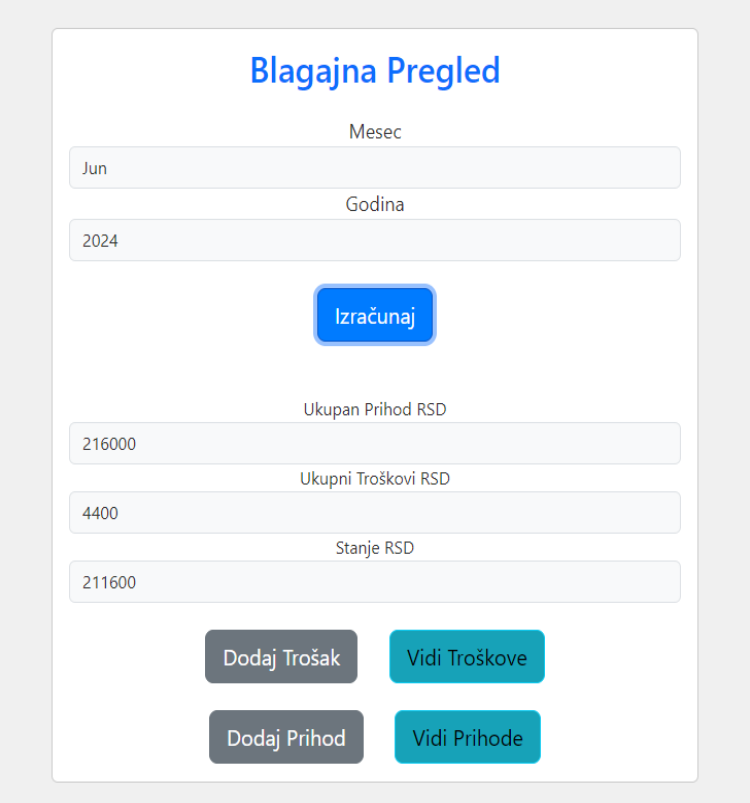
5.3.2 Odbijanje komentara

Na sledećnj se može videti i tabela komentara koje samo admin može da uklони kликom na dugme „*Izbrisi komentar*“.

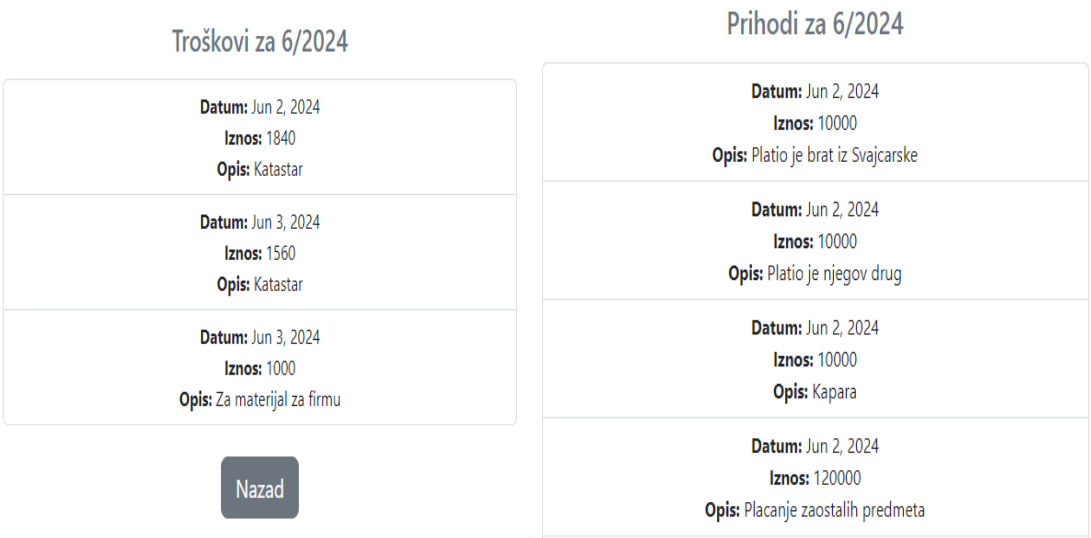
Slika 5.3.2 Komentari

5.3.3 Благајна

На слици (Слика 5.3.3.[1]) се може видети форма која админу пружа увид у приходе и трошкове за одређени месец. Постоји и могућност да види детаље о трошковима за тај месец , као и приходе . Осим ових функционалности овде можемо и да додамо одређени приход или трошак.



Слика 5.3.3[1] Благајна преглед



Слика 5.3.3 Трошкови и приходи

```java
public List<Trosak> getTroskoviMesecno(int godina, int mesec) {
    YearMonth mesecIGodina = YearMonth.of(godina, mesec);
    LocalDate prviDanMeseca = mesecIGodina.atDay(1);
    LocalDate poslednjiDanMeseca = mesecIGodina.atEndOfMonth();
    Date prviDan = Date.valueOf(prviDanMeseca);
    Date poslednjiDan = Date.valueOf(poslednjiDanMeseca);
return trosakRepository.findByDatumBetween(prviDan, poslednjiDan);
 }
```

Исечак кода 5.3.3 (Java) – Функција *getTroskoviMesecno*

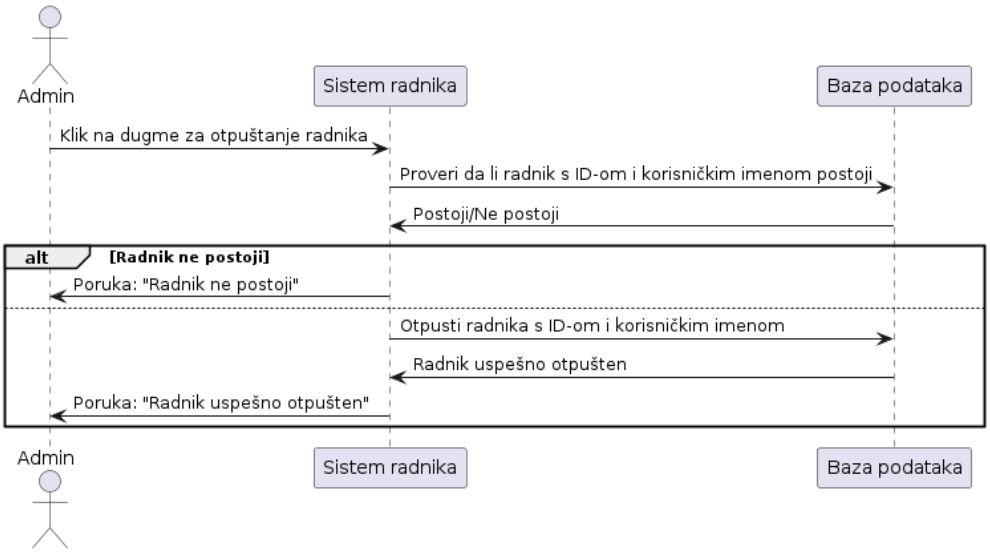
5.3.4 Отпуштање радника

Админ може да види табелу радника као и сваки други радник али једино он има привилегију да отпусти неког радника и то кликом на дугме „*Otpusti*“. Такође на слици се види да је колона где је админ обојена зеленом бојом.

Radnici

Ime	Prezime	Korisnicko ime	Lozinka		
Petar	Stanojkovic	petar2	*********	Otpusti	Ne radi
Sladjan	Stanojkovic	cagi	********	Otpusti	Admin
Aleksandar	Stankovic	aca	*******	Otpusti	
Fatmir	Kadriu	fatmir	**********	Otpusti	

Слика 5.3.4 Радници табела



Слика 5.3.4 УМЛ дијаграм отпуштања радника

5.4 Предмети

У овој апликацији главни подаци су предмети. Они представљају основу ове аплиакације. Ова апаликација нуди табеларно приказивање свих пројеката на веома леп визуалан начин . Могуће је додати нови предмет слика (Слика 5.4.5), изменити и обрисати дати пројекат. Такође у овом пројекту имамо посебно табеларно приказивање клијената и парцела као и њихово додавање и придружовање података из катастра одређеном предмету али и додавање терена одређеном предмету.

5.4.1 Табеларно приказивање

На следећој слици се може видети табеларно приказивање свих предмета, Овде је омогућено и сортирање по датуму, клијенту парцели и по томе да ли је предмет завршен. Као што се види на слици предмет може да се заврши и онда је цела колона зелене боје. Предмет се завршава кликом на дугме „*Zavrsi*“. Осим свих предмета ми табеларно можемо да видимо и све клијенте слика(Слика 5.4.1[1]) и све парцеле слика(Слика 5.4.1 [2]). Колона која је обојена црвено за клијенте означава да је клијент дужан и тачно пише колико је дужан. Клијент не мора да буде везан за предмет. Колона која је зелена за парцелу означава да се парцела налази у два или више предмета. Парцела не мора да буде везана за предмет.

Predmeti

Dodaj novi predmet

Posao	Datum	Klijent	Parcela	Cena	Dug						
Snimanje objekta	02.06.2024.	Šefkett (Taibe) Salihu	611/1 Alidjerce	16000 RSD	Plaćeno	Plati	Katastar	Teren	Zavrsi	Promeni	Izbriši
Realizacija projekta preparcelacije	02.06.2024.	Mustafa (Mustafa) Jusein	1575 Preševo	25000 RSD	Dug: 25000	Plati	Katastar	Teren	Zavrsi	Promeni	Izbriši
Snimanje objekta	05.05.2024.	Asan (Ibraim) Huseni	811/1,811/2 Norča	15000 RSD	Plaćeno	Plati	Katastar	Teren	Zavrsi	Promeni	Izbriši
Izrada KTP-a	02.06.2024.	Vesil (Lorik) Musliu	1735 Žujice	18000 RSD	Dug: 18000	Plati	Katastar	Teren	Zavrsi	Promeni	Izbriši
Snimanje objekta	08.05.2024.	Fadil (Afrim) Habibi	386 Rajince	27000 RSD	Plaćeno	Plati	Katastar	Teren	Zavrsi	Promeni	Izbriši

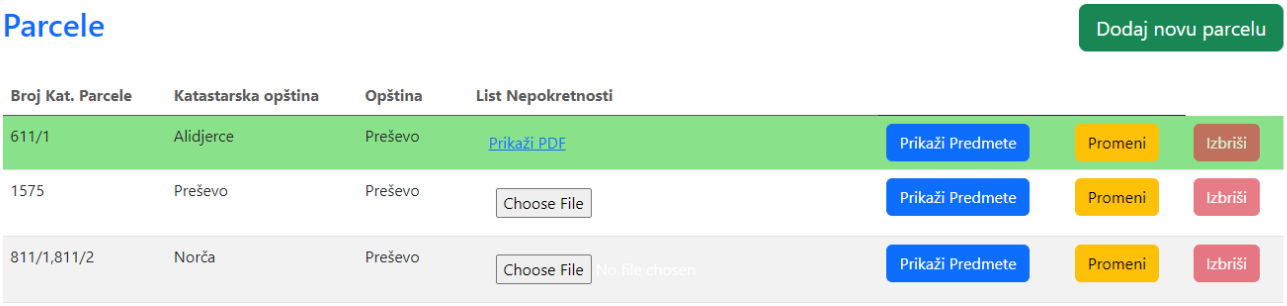
Слика 5.4.1 Предмети табела

Klijenti

Dodaj novog klijenta

Ime	Prezime	Ime oca	Prebivalište	Telefon	JMBG	BRLK	DUG				
Šefkett	Salihu	Taibe	Alidjerce	063/415063	1007978747912	005533085		Predmeti	Placanja	Promeni	Izbriši
Mustafa	Jusein	Mustafa	Preševo, ul. Seljami Halači 15A	062/8206805	0105961742915	006463292	25000RSD	Predmeti	Placanja	Promeni	Izbriši
Asan	Huseni	Ibraim	Norča	063/8542115	1210944742911	008118993		Predmeti	Placanja	Promeni	Izbriši

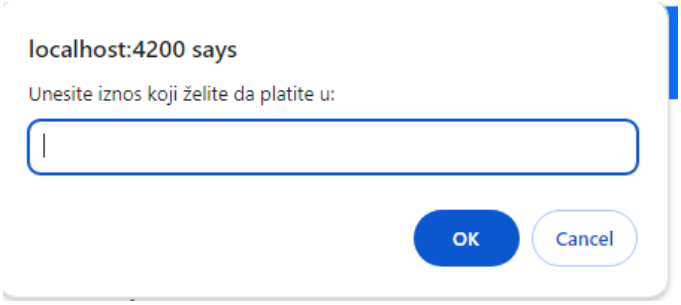
Слика 5.4.1 [1] Клијенти табела



Слика 5.4.1 [2] Парцеле табела

5.4.2 Плаћање

Као што се види на слици (Слика 5.4.1), предмет може да се плати кликом на дугме „*Plati*“ и том приликом ми уносимо новац у динарима колико желимо да платимо .



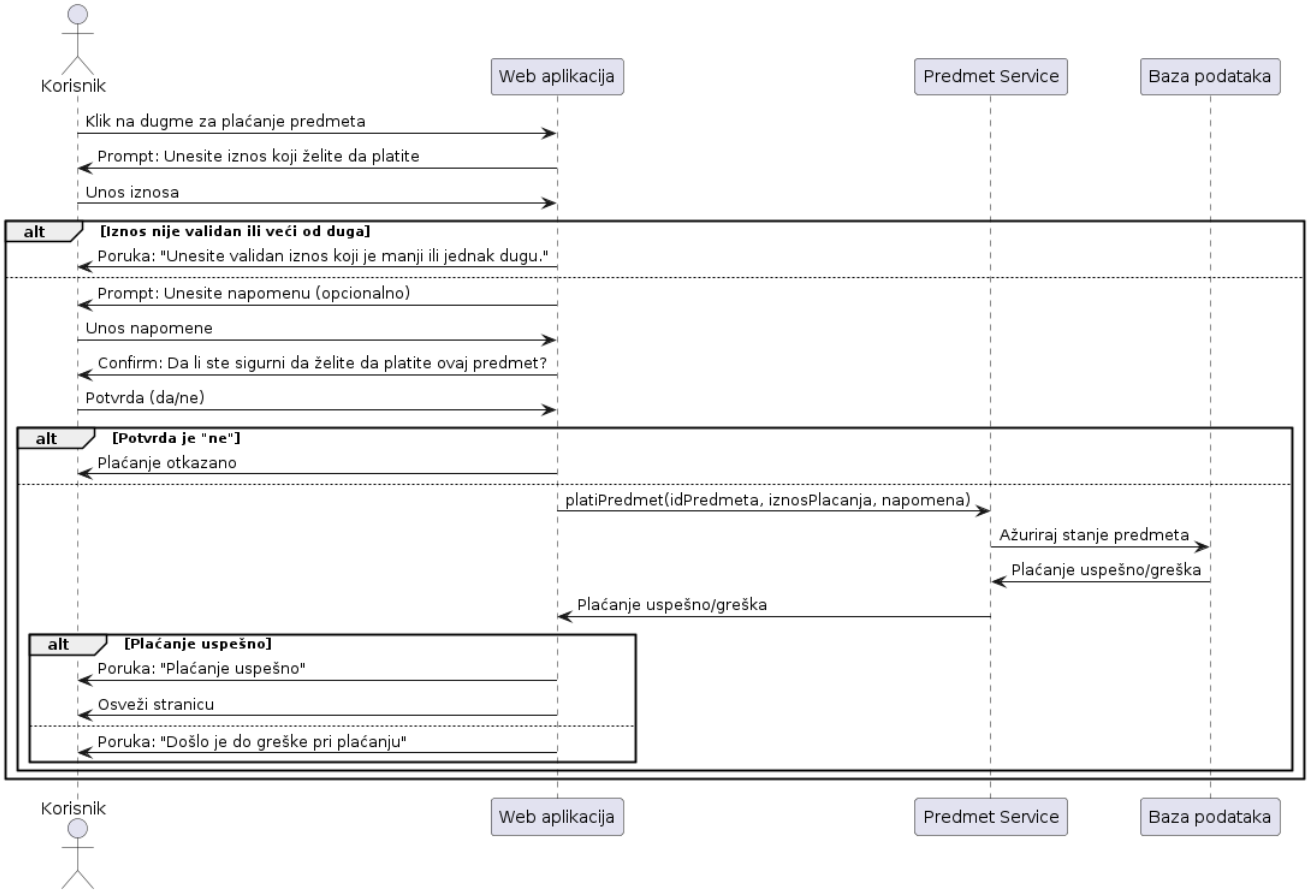
Слика 5.4.2 Форма за плаћање

Плаћање можемо само да извршимо уколико није исплаћен цео предмет и том приликом се аутоамтски смањује дуг или скроз уклања ако смо исплатили цео дуг и у колони „*Dug*“ онда пише „*Placeno*“,

```
 platiPredmet(idPredmeta: number, dug: number) {
   const iznos = prompt("Unesite iznos koji želite da platite:");
   if (iznos !== null) {
     const iznosPlacanja = parseFloat(iznos);
  if (!isNaN(iznosPlacanja) && iznosPlacanja > 0 && iznosPlacanja <= dug) {
     const napomena = prompt("Unesite napomenu (opcionalno):");
   const potvrda = confirm("Da li ste sigurni da želite da platite ovaj
predmet?");
           if (potvrda) {
this.predmetService.platiPredmet(idPredmeta, iznosPlacanja, napomena || '')
               .subscribe(data => {
                 console.log("Plaćanje uspješno", data);
                 window.location.reload();
               }, error => {
                 console.error("Greška pri plaćanju", error);
                 alert("Došlo je do greške pri plaćanju.");
               });
```

```
        }
      } else {
        alert("Unesite validan iznos koji je manji ili jednak dugu.");
      }
    }
  }
```

Исечак кода 5.4.2 (TypeScript) – Функција *platiPredmet*

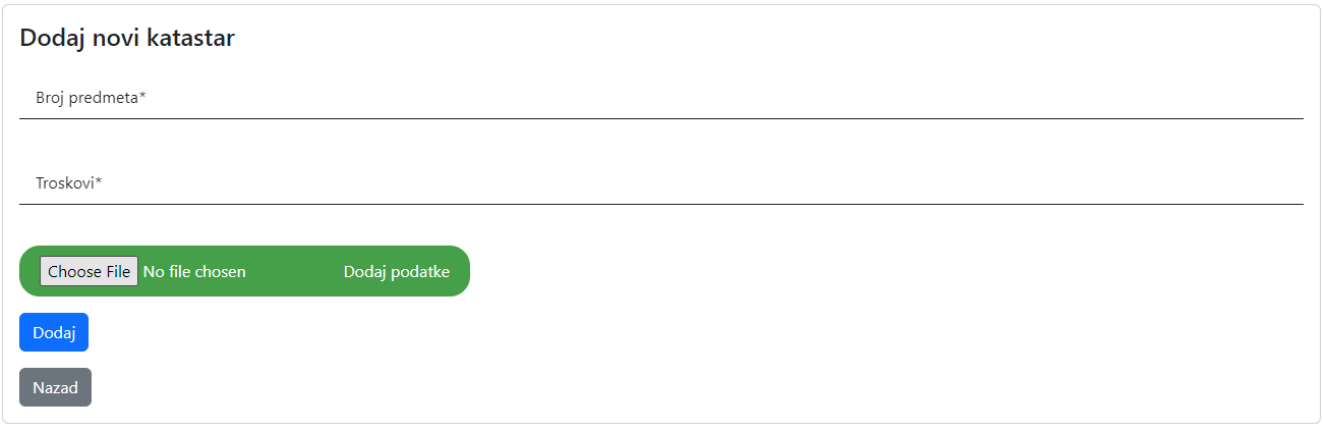


Слика 5.4.2 УМЛ дијаграм за плаћање

5.4.3 Катастар

Одређени предмет може да има и податке који су преузети из катастра. Ако желимо да придружимо одређене података из катастра предмету, то можемо да урадимо тако што ћемо кликнути дугме „*Katastar*“ и онда ће се отвориће форма за додавање тих података из катастра . Такође може се видети и подаци из катастра који су већ придружени том предмету уколико постоје .

Слика 5.4.3 Катастар приказивање



Слика 5.4.3 Катастар додавање

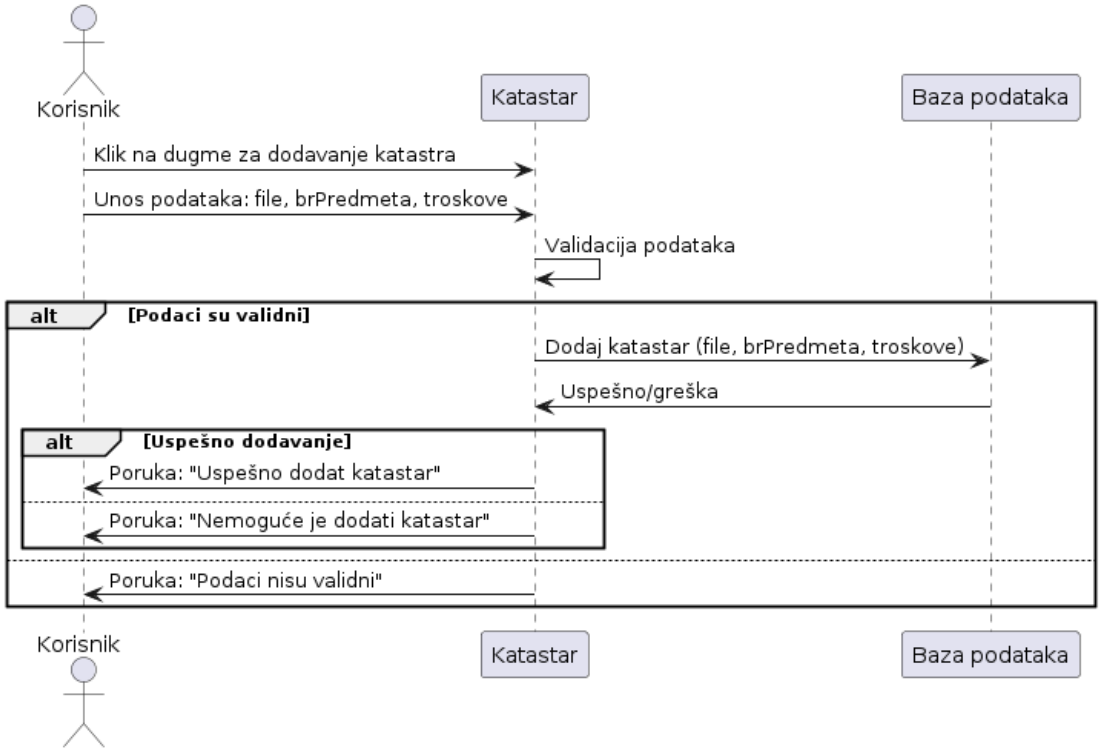
```
sacuvajKatastar() {
  this.katastar.predmet = this.predmet;

  if (this.file) {
    this.convertMultipartFileToNumberArray(this.file).pipe(
      catchError(error => {
        console.error("Greška pri konverziji fajla u niz brojeva.",error);
        return throwError("Greška pri konverziji fajla u niz brojeva.");
      }),
      flatMap((numberArray: number[]) => {
        this.katastar.podaci = numberArray;
        return this.katastarService.dodajKatastar(this.katastar);
      })
    ).subscribe(
      (data: Katastar) => {
            alert("Uspešno dodat katastar");
            window.location.reload();
          },
          (error) => {
            console.error("Nemoguće je dodati katastar.", error);
```

```
          alert("Nemoguće je dodati katastar");
        }
      );
    } else {
      this.katastarService.dodajKatastar(this.katastar).subscribe(
        (data: Katastar) => {
          alert("Uspešno dodat katastar");
          window.location.reload();
        },
        (error) => {
          console.error("Nemoguće je dodati katastar.", error);
          alert("Nemoguće je dodati katastar");
        }
      );
    }

    this.file = null;
  }
```

Исечак кода 5.4.3 (TypeScript) – Функција *sacuvajKatastar*



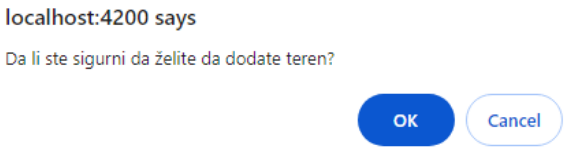
Слика 5.4.3 УМЛ дијаграм додавање катастра

5.4.4 Терен

За одређени предмет ми можемо да извршимо терен и том приликом радник који је тренутно пријављен на систем извршава терен ако кликнемо на дугме „*Teren*“ (Слика 5.4.1).

За један предмет терен се може обавити само једном и ако је терен урађен онда ће дугме „*Teren*“ бити плаве боје и моћи ћемо да видимо терен (Слика 5.4.1), у супротном дугме ће бити жуте боје и моћи ћемо да додамо терен .

Слика 5.4.4 Терен приказивање

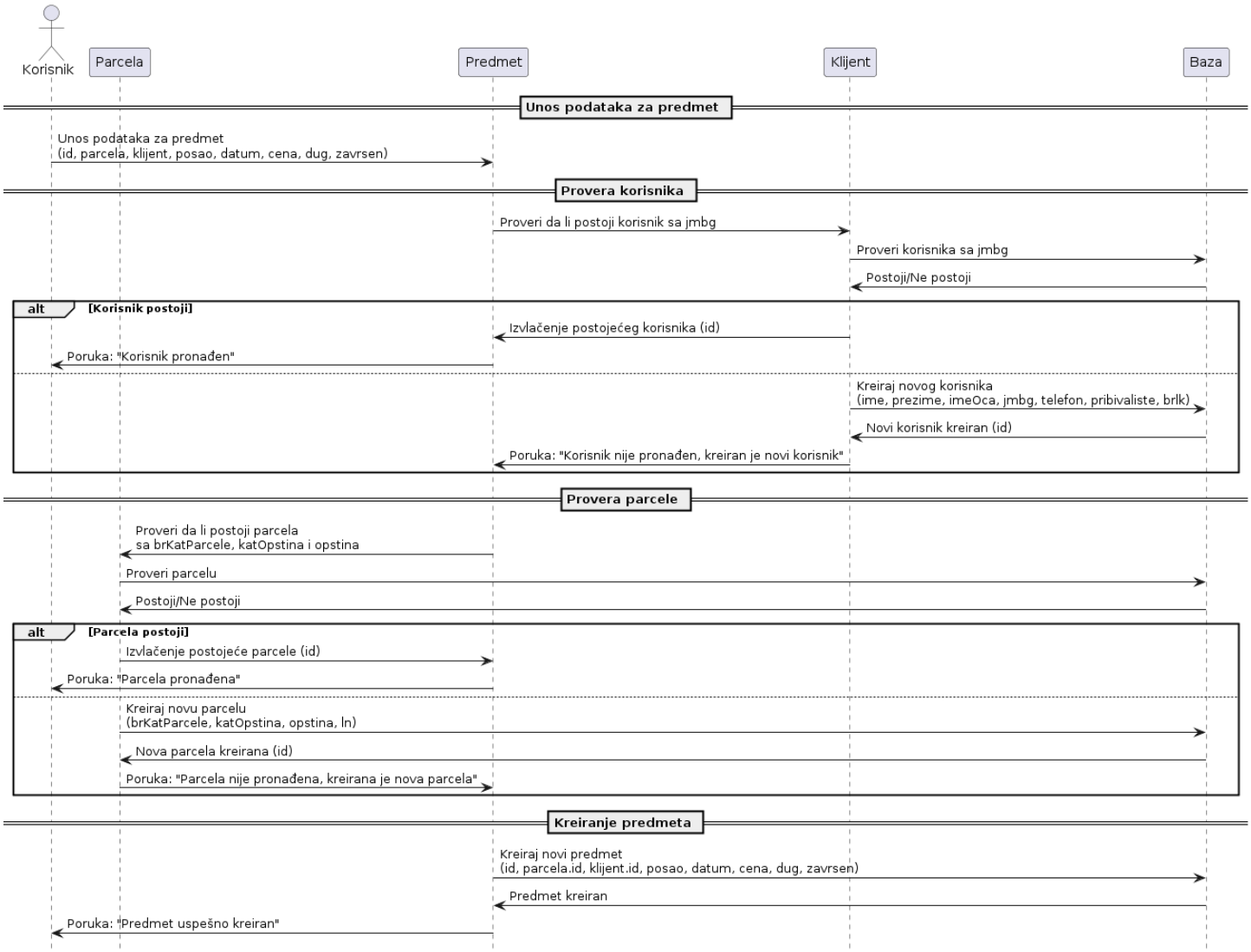


Слика 5.4.4 Терен потврда додавања

5.4.5 Додавање предмета

Пројекат мора да има клијента и парцелу и приликом додавање новог пројекта ми додајемо и клијента и парцелу ако не постоје, или једноставно повежемо пројекат са клијентом односно парцелом који постоје. Форма за додавање новог пројекта изгледа као на слици испод.

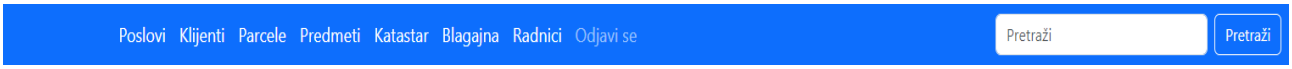
Слика 5.4.5 Форма за додавање предмета



Слика 5.4.5 УМЛ дијаграм додавања предмета

5.5 Мени

Када се радник пријави он ће на врху странице имати мени који ће му помоћи у брзом и лаком сналажењу и мењању страница. Такође на следећој се може и видети претрага која омогућава лаку и брзу претрагу за налажење одређеног предмета.



Слика 5.5 Мени

```
    pretrazi(){
        if(this.searchTerm){
       if(   JSON.parse(localStorage.getItem('Search'))){
        localStorage.removeItem('Search')
       }
        localStorage.setItem('Search',JSON.stringify(this.searchTerm));
          this.router.navigate(['predmeti']).then(() => {
            window.location.reload();
          });
        }
    }
```

Исечак кода 5.5 (TypeScript) – Функција *pretrazi*

У датом коду функција *pretrazi()* се користи за чување претраживачког појма у *localStorage*. Након што корисник унесе претраживачки појам *(this.searchTerm),* он се конвертује у *JSON* формат помоћу *JSON.stringify()* и чува у *localStorage* користећи *localStorage.setItem('Search', ...).* Ово омогућава да се претраживачки појам трајно сачува у прегледачу, док корисник не обрише податке. Постоји и провера да ли већ постоји сачуван претраживачки појам у *localStorage* помоћу *JSON.parse (localStorage.getItem ('Search')*), што омогућава уклањање старог претраживачког појма пре чувања новог, ако је потребно. Након чувања у *localStorage*, страница се преусмерава на *'predmeti'*, а затим се освежава ради ажурирања приказаа

```
 sviPredmetiSaFilterom() {
     this.predmetService.getAllPredmeti().subscribe(
       (data: Predmet[]) => {
         const searchLower = this.existingSearch.toLowerCase();
         this.predmeti = data.filter(predmet =>
           predmet.posao.naziv.toLowerCase().includes(searchLower) ||
           predmet.klijent.ime.toLowerCase().includes(searchLower)||
           predmet.klijent.prezime.toLowerCase().includes(searchLower) ||
           predmet.klijent.imeOca.toLowerCase().includes(searchLower) ||
        predmet.parcela.brKatParcele.toLowerCase().includes(searchLower)||
         predmet.parcela.katOpstina.toLowerCase().includes(searchLower) ||
             );
           },
           (error) => {
             alert("Greška prilikom uzimanja svih predmeta");
           }
         );
       }
```

Исечак кода 5.5 (TypeScript) – Функција *sviPredmetiSaFilterom*

6. ЗАКЉУЧАК

Апликација развијена за управљање геодетским бироом представља кључни алат за ефикасније управљање оперативним процесима и повећање производности у предузећу. Централна функција апликације је централизација и брз приступ информацијама, што елиминише проблеме фрагментације података и значајно убрзава свакодневни рад запослених.

Увођењем додатних функционалности као што су извештаји о месечним перформансама, прегледи зарада и трошкова, апликација омогућава власнику свеобухватан увид у пословање. Ово значајно олакшава процес доношења информисаних одлука. Имплементација система улога на корисничке налоге доприноси бољој интерној комуникацији и сарадњи између различитих делова предузећа.

У будућности, у случају ширења коришћења међу више предузећа, могуће је разматрати додавање опције претплата као додатног извора прихода. Ова стратегија подржава скалирање апликације и доприноси њеној дугорочној стабилности и конкурентности на тржишту.

Ова апликација није само инструмент за оптимизацију оперативних процеса, већ представља кључни фактор у трансформацији предузећа ка модерном и технолошки напредном приступу пословању. Напредак у ефикасности и производности који она доноси значајно подиже конкурентност на тржишту геодетских услуга.

Литература

[1] TypeScript, https://en.wikipedia.org/wiki/TypeScript, јун 2024.

[2] Angular Material, https://material.angular.io/, јун 2024.

[3] Explain the Architecture Overview of Angular, https://www.geeksforgeeks.org/explain-the-architecture-overview-of-angular/, јун 2024.

[4] Bootstrap , https://getbootstrap.com/docs/5.0/getting-started/introduction/ , јун 2024 .

[5] Bootstrap animation , https://mdbootstrap.com/docs/standard/content-styles/animations/ , јун 2024.

[6] Bootstrap wow fade in offset , https://mdbootstrap.com/docs/standard/extended/quick-docs-wow-fade/ , јун 2024

[7] Bootstrap Navbar, https://getbootstrap.com/docs/4.0/components/navbar/ , јун 2024

[8] Java Spring Boot, https://www.ibm.com/topics/java-spring-boot, јун 2024.

[9] Spring Boot Architecture, https://www.interviewbit.com/blog/spring-boot-architecture/, јун 2024.

[10] Spring framework, https://docs.spring.io/spring-framework/reference/overview.html, јун 2024.

[11] Spring initializr, https://start.spring.io/ , јун 2024.

[12] Spring boot: https://spring.io/projects/spring-boot , јун 2024.

[13] Spring Boot pdf uploading and serving, https://www.baeldung.com/java-pdf-creation, јун 2024.

[14] Геосрбија, https://a3.geosrbija.rs/, јун 2024.

[15] Републички геодетски завод, https://www.rgz.gov.rs/, јун 2024.

СПИСАК СЛИКА

СПИСАК ИСЕЧАКА КОДОВА