**StanTech**

**Arhitekturni projekat**

Verzija 1.0

Pregled izmena

|  |  |  |  |
| --- | --- | --- | --- |
| **Datum** | **Verzija** | **Opis** | **Autor** |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

Sadržaj

1. Cilj dokumenta 5

2. Opseg dokumenta 5

3. Reference 5

4. Predstavljanje arhitekture 5

5. Ciljevi i ograničenja arhitekture 5

6. Pogled na slučajeve korišćenja 5

6.1 Dijagrami slučajeva korišćenja 6

6.2 Kratak opis slučajeva korišćenja 8

6.2.1 Pregled individualnih oglasa 8

6.2.2 Pregled glavne strane 8

6.2.3 Kontaktiranje stanodavca preko broja telefona 8

6.2.4 Registrovanje 8

6.2.5 Pregled recenzija 9

6.2.6 Pregled profila ostalih korisnika 9

6.2.7 Ažuriranje ličnih podataka 9

6.2.8 Prijavljivanje 9

6.2.9 Ocenjivanje majstora 9

6.2.10 Ocenjivanje stanodavca 9

6.2.11 Komunikacija sa stanodavcem preko sajta 9

6.2.12 Obaveštenje o problemima 9

6.2.13 Prihvatanje zahteva da postane stanar 9

6.2.14 Komunikacija sa majstorom 9

6.2.15 Dodavanje oglasa za stan 9

6.2.16 Ocenjivanje stanara 9

6.2.17 Komunikacija sa stanarom 10

6.2.18 Dodavanje oglasa za majstora 10

6.2.19 Primanje obaveštenja o problemima 10

6.2.20 Organizovanje kalendara aktivnosti 10

6.2.21 Pregled oglasa za majstore 10

6.2.22 Kontaktiranje stanara u vezi popravke 10

6.2.23 Plaćanje kirije 10

6.2.24 Brisanje postojećeg oglasa 10

7. Pogled na logičku arhitekturu sistema 10

7.1 Pregled arhitekture – organizacija paketa i podsistema u slojeve 11

7.1.1 Korisnički interfejs 11

7.1.2 Aplikaciona logika 11

7.1.3 Pristup podacima 11

7.1.4 React.js 12

7.1.5 Node.js 12

7.1.6 MongoDB 12

8. Pogled na procese 12

8.1 Procesi 12

8.1.1 Web čitač 12

8.1.2 Express.js 13

8.1.3 Node.js 13

8.1.4 MongoDB 13

9. Pogled na raspoređivanje sistema 13

9.1 Klijent 13

9.2 Web server 13

9.3 MongoDB server 13

10. Pogled na implementaciju sistema 13

10.1 Model domena 14

10.2 Šema baze podataka 14

10.3 Komponente sistema 16

10.3.1 Komponente korisničkog interfejsa 16

10.3.2 Komponente aplikacione logike 17

10.3.3 Komponente za pristup podacima 17

11. Performanse 19

12. Kvalitet 19

Arhitekturni projekat

# Cilj dokumenta

Cilj ovog dokumenta je detaljni opis arhitekture StanTech web aplikacije.

# Opseg dokumenta

Dokument se odnosi na StanTech web aplikaciju koja će biti razvijena od strane StanSquad-a. StanTech je web aplikacija koja služi za lako nalaženje stanova na izdavanje, kao i za lako izdavanje stanova. Namena aplikacije je efikasno pretraživanje, filtriranje, održavanje stanova i olakšan kontakt između stanara, stanodavca i majstora.

# Reference

Spisak korišćene literature:

1. D\_01\_Predlog\_Projekta.
2. D\_02\_Vizija\_Sistema.
3. D\_03\_Plan\_Realizacije
4. D\_03\_Raspored\_Aktivnosti.
5. D\_04\_Spec\_Zahteva.

# Predstavljanje arhitekture

Arhitektura sistema u dokumentu je prikazana kao serija pogleda na sistem: pogled na slučajeve korišćenja, pogled na logičku arhitekturu sistema, pogled na procese, pogled na razmeštaj komponenti sistema i pogled na implementaciju. Ovi pogledi su predstavljeni odgovarajućim UML dijagramima.

# Ciljevi i ograničenja arhitekture

Ključni zahtevi i sistemska ograničenja koja imaju značajan uticaj na izbor arhitekture i projektovanje sistema su:

1. StanTech će biti implementiran kao Web aplikacija zasnovana na MERN-u. [4].
2. Klijentski deo StanTech web aplikacije će biti optimiyovan za sledeće Web čitače: web čitači zasnovani na Chromium-u i Firefox (Mozilla).
3. Svi zahtevi u pogledu performansi dati u [5] moraju biti uzeti u obzir pri izboru arhitekture i razvoju sistema.
4. Klasifikaciju publikacija treba standardizovati sa postojećom tipologijom propisanom od strane Ministarstva za nauku.

# Pogled na slučajeve korišćenja

U ovom odeljku je dat pogled na slučajeve korišćenja definisane u specifikaciji zahteva [5].

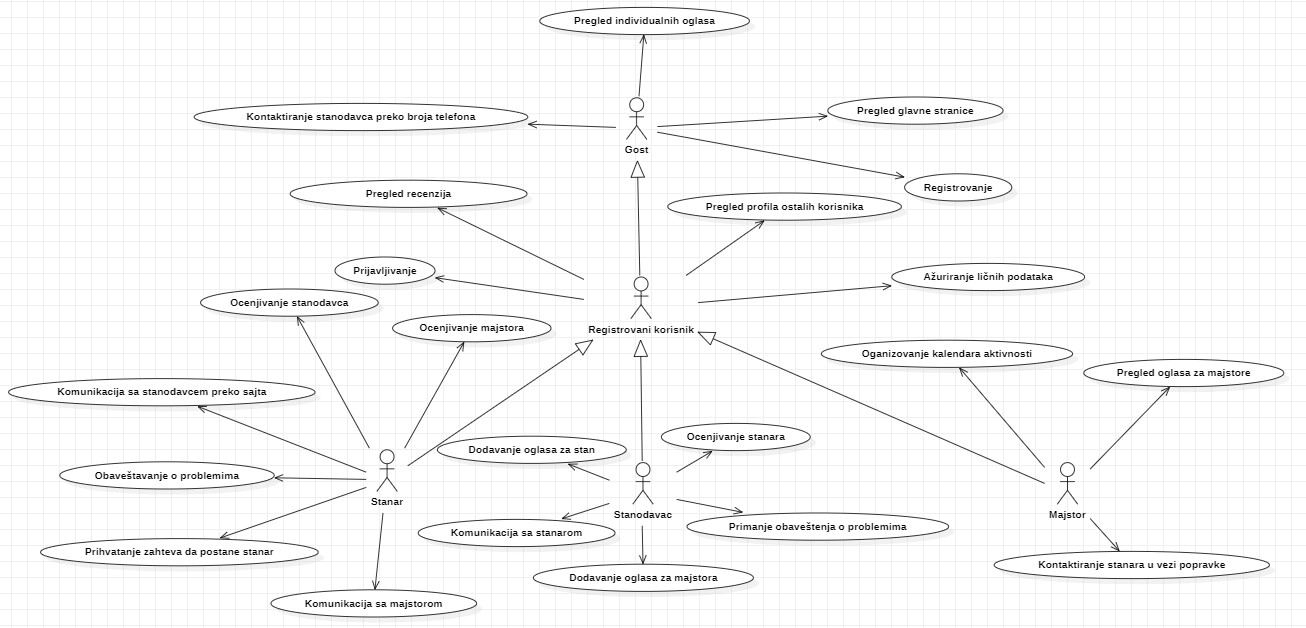
Slučajevi korišćenja StanTech portala su:

* Pregled individualnih oglasa
* Pregled glavne stranice
* Kontaktiranje stanodavca preko broja telefona
* Registrovanje
* Pregled recenzija
* Pregled profila ostalih korisnika
* Ažuriranje ličnih podataka
* Prijavljivanje
* Ocenjivanje majstora
* Ocenjivanje stanodavca
* Komunikacija sa stanodavcem preko sajta
* Obaveštenje o problemima
* Prihvatanje zahteva da postane stanar
* Komunikacija sa majstorom
* Dodavanje oglasa za stan
* Ocenjivanje stanara
* Komunikacija sa stanarom
* Dodavanje oglasa za majstora
* Primanje obaveštenja o problemima
* Organizovanje kalendara aktivnosti
* Pregled oglasa za majstore
* Kontaktiranje stanara u vezi popravke
* Plaćanje kirije
* Brisanje postojećeg oglasa

Ove slučajevi korišćenja mogu da iniciraju posetilac web aplikacije, Stanar, Stanodavac Majstor.

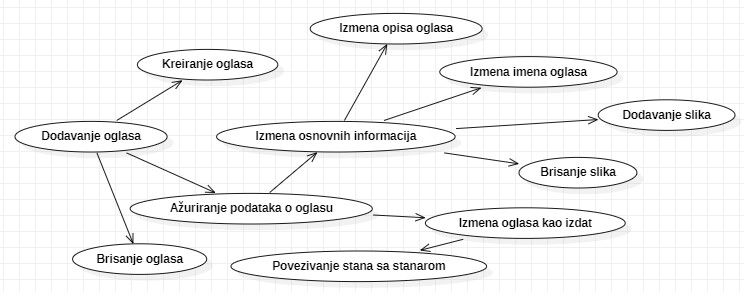
## Dijagrami slučajeva korišćenja

Osnovni UML dijagram koji prikazuje korisnike i slučajeve korišćenja StanTech web aplikacije prikazan je na sledećoj slici:

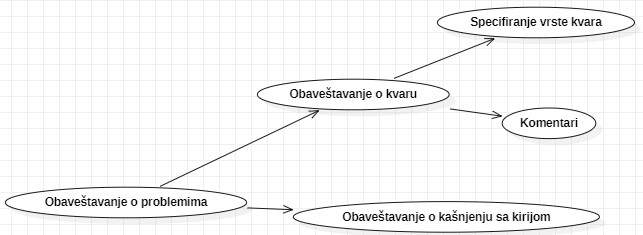


Slučajevi korišćenja *dodavanje oglasa, obaveštavanje o problemima, organizovanje kalendara aktivnosti, ocenjivanje majstora, ocenjivanje stanodavca, ocenjivanje stanara* obuhvataju složenije radnje koje se mogu razložiti dalje razložiti na pojedinačne slučajeve korišćenja.

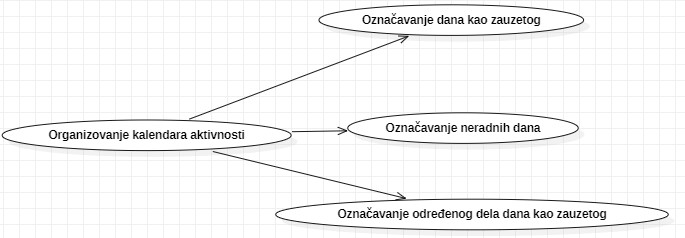
Detaljni UML dijagram za slučaj korišćenja *dodavanje oglasa* je prikazan na sledećoj slici:



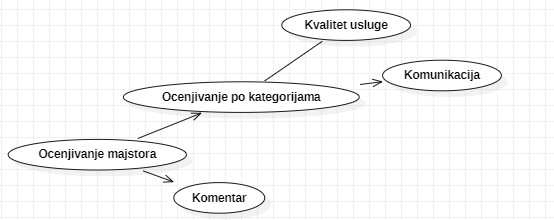
Detaljni UML dijagram za slučaj korišćenja *obaveštavanje o problemima* je prikazan na sledećoj slici:



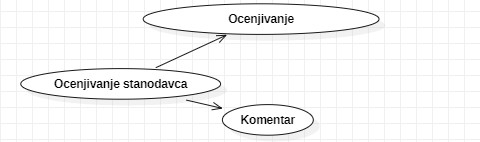
Detaljni UML dijagram za slučaj korišćenja *organizovanje kalendara aktivnosti* je prikazan na sledećoj slici:



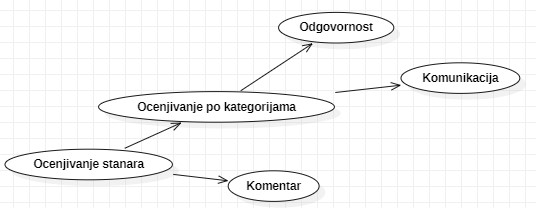
Detaljni UML dijagram za slučaj korišćenja *ocenjivanje majstora* je prikazan na sledećoj slici:



Detaljni UML dijagram za slučaj korišćenja *ocenjivanje stanodavca* je prikazan na sledećoj slici:



Detaljni UML dijagram za slučaj korišćenja *ocenjivanje stanara* je prikazan na sledećoj slici:



## Kratak opis slučajeva korišćenja

### Pregled individualnih oglasa

Kratak opis: Prikaz stranice oglasa za određeni stan. Korisnik bira oglas koji želi detaljnije da pogleda i klikne na njega.

Akteri koji iniciraju slučaj korišćenja: Gost Stanar, Stanodavac, Majstor.

### Pregled glavne strane

Kratak opis: Prikaz stranice sajta sa oglasima i odvojenim delom za pretraživanje.

Akteri koji iniciraju slučaj korišćenja: Gost Stanar, Stanodavac, Majstor.

### Kontaktiranje stanodavca preko broja telefona

Kratak opis: Otvaranje aplikacije za poziv ili prikaz broja telefona ako je korisnik na računaru.

Akteri koji iniciraju slučaj korišćenja: Gost, Stanar, Stanodavac, Majstor.

### Registrovanje

Kratak opis: Korisnik se registruje na web aplikaciju aktivacijom komande *Registruj se* nakon čega mu se prikaže forma za registraciju u kojoj on bira vrstu naloga (majstor, stanar, stanodavac).

Akteri koji iniciraju slučaj korišćenja su posetioci web aplikacije

### Pregled recenzija

Kratak opis: Prikaz stranice sa svim recenzijama za nekog korisnika.

Akteri koji iniciraju slučaj korišćenja: Gost, Stanar, Stanodavac, Majstor.

### Pregled profila ostalih korisnika

Kratak opis: Prikaz stranice profila određenog korisnika.

Akteri koji iniciraju slučaj korišćenja: Stanar, Stanodavac, Majstor.

### Ažuriranje ličnih podataka

Kratak opis: Ažuriranje podataka o sebi od strane prijavljenog člana StanTech-a.

Akteri koji iniciraju slučaj korišćenja: Stanar, Stanodavac, Majstor.

### Prijavljivanje

Kratak opis: Prikaz stranice za prijavljivanje koja se sastoji od dva text boxa za unos korisničkog imena i šifre.

Akteri koji iniciraju slučaj korišćenja: Stanodavac, Stanar, Majstor.

### Ocenjivanje majstora

Kratak opis: Dodavanje recenzije majstoru nakon izvršenog posla.

Akteri koji iniciraju slučaj korišćenja: Stanar.

### Ocenjivanje stanodavca

Kratak opis: Dodavanje recenzije stanodavcu nakon izvrešnog posla.

Akteri koji iniciraju slučaj korišćenja: Stanar.

### Komunikacija sa stanodavcem preko sajta

Kratak opis: Prikaz stranice koja služi za komunikaciju sa stanodavcem.

Akteri koji iniciraju slučaj korišćenja: Stanar.

### Obaveštenje o problemima

Kratak opis: Slanje obaveštenja o problemima stanodavcu.

Akteri koji iniciraju slučaj korišćenja: Stanar*.*

### Prihvatanje zahteva da postane stanar

Kratak opis: Zahtev koji stiže stanaru od stanodavca.

Akteri koji iniciraju slučaj korišćenja: Stanar.

### Komunikacija sa majstorom

Kratak opis: Prikaz stranice koja služi za komunikaciju sa majstorom.

Akteri koji iniciraju slučaj korišćenja: Stanar.

### Dodavanje oglasa za stan

Kratak opis: Dodavanje oglasa za stan na izdavanje od strane stanodavca.

Akteri koji iniciraju slučaj korišćenja: Stanodavac.

### Ocenjivanje stanara

Kratak opis: Dodavanje recenzije stanaru.

Akteri koji iniciraju slučaj korišćenja: Stanodavac.

### Komunikacija sa stanarom

Kratak opis: Prikaz stranice koja služi za komunikaciju sa stanarom.

Akteri koji iniciraju slučaj korišćenja: Stanodavac.

### Dodavanje oglasa za majstora

Kratak opis: Dodavanje novog oglasa za majstora.

Akteri koji iniciraju slučaj korišćenja: Stanodavac.

### Primanje obaveštenja o problemima

Kratak opis: Notifikacije stanodavca koje stanar šalje.

Akteri koji iniciraju slučaj korišćenja: Stanodavac.

### Organizovanje kalendara aktivnosti

Kratak opis: Organizovanje kalendara aktivnosti majstora. Dodavanje novih poslova u kalendar, ili izmena postojećih.

Akteri koji iniciraju slučaj korišćenja: Majstor.

### Pregled oglasa za majstore

Kratak opis: Prikaz stranice sa oglasima za majstore.

Akteri koji iniciraju slučaj korišćenja: Majstor.

### Kontaktiranje stanara u vezi popravke

Kratak opis: Kontaktiranje stanara u vezi popravke određenog stana. Majstor pretražuje oglase i preko oglasa stupa u kontakt sa stanarima.

Akteri koji iniciraju slučaj korišćenja: Majstor.

### Plaćanje kirije

Kratak opis: Način plaćanja kirije.

Akteri koji iniciraju slučaj korišćenja: Stanar.

### Brisanje postojećeg oglasa

Kratak opis: Brisanje postojećeg oglasa od strane stanodavca koji je taj oglas kreirao.

Akteri koji iniciraju slučaj korišćenja: Stanodavac.

# Pogled na logičku arhitekturu sistema

U ovom odeljku je dat pregled logičke arhitekture sistema. Ovaj pogled sadrži opis najznačajnijih klasa, njihove organizacije u pakete i podsisteme, i organizacija podsistema u slojeve. U cilju opisivanja dinamičkih aspekata arhitekture, ovaj odeljak može da uključi opise realizacije najznačajnijih slučajeva korišćenja. Da bi se ilustrovala veza između arhitekturno značajnih klasa, podsistema, paketa ili slojeva moguće je uključiti i odgovarajuće dijagrame klasa.

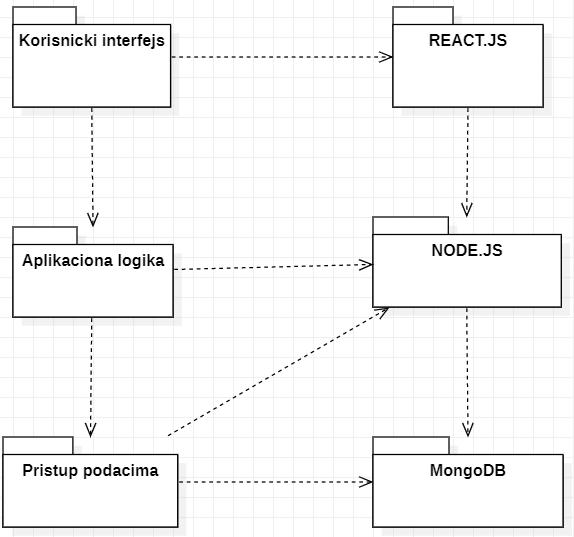
Logički pogled na StanTech obuhvata 3 glavna paketa: Korisnički interfejs, Aplikaciona logika, Pristup podacima.

Paket *Korisnički interfejs* sadrži Web stranice, React skripte i multimedijalni sadržaj koji realizuju grafički dizajn i forme preko kojih korisnici sistema komuniciraju sa sistemom.

Paket *Aplikaciona logika* predstavlja srednji sloj sistema koji sadrži Node.js i Express.js fajlove zadužene za realizaciju funkcionalnosti specifičnih za domen sistema koji se razvija.

Paket *Pristup podacima* sadrži Node.js i Express.js skripte koje predstavljaju interfejs za pristup, dodavanje i ažuriranje podataka koji se čuvaju u bazi podataka.

## Pregled arhitekture – organizacija paketa i podsistema u slojeve



### Korisnički interfejs

sloj

Ovaj sloj realizuje korisnički interfejs portala. U njemu su sadržani svi JSX kodovi, multimedijalni sadržaji i React komponente koje služe da generišu virtuelnu reprezentaciju interfejsa u memoriji, i zatim se koriste za nadogradnju prave HTML stranice preko kojih korisnici komuniciraju sa sistemom.

Sloj korisničkog interfejsa zavisi od sloja aplikacione logike, kao i JSX koda i React.js.

### Aplikaciona logika

sloj

Sloj aplikacione logike je srednji sloj u troslojnoj arhitekturi StanTech-a. Sadrži Express.js i Node.js fajlove koji realizuju funkcionalnost karakterističnu za domen primene i uspostavljaju vezu između korisničkog interfejsa i sloja za pristup podacima.

Ovaj sloj zavisi od sloja za pristup podacima, Express.js i Node.js paketa.

### Pristup podacima

sloj

Sloj za pristup podacima se nalazi na dnu troslojne arhitekture i sadrži Node.js i Express.js fajlove zadužene za pribavljanje, dodavanje i ažuriranje podataka koji se čuvaju u MongoDB bazi podataka.

Ovaj sloj ne zavisi od drugih slojeva, ali je zavisan od paketa Express.js, Node.js i MongoDB baza podataka.

### React.js

tehnologija

Tehnologija React.js definiše gradivne elemente stranica koje se prikazuju u Web čitaču i koje omogućavaju prikaz formatiranih informacija i realizaciju formi za unos i ažuriranje podataka.

### Node.js

tehnologija

Tehnologija Node.js omogućuje podršku za back-end deo projekta. Ona pruža podršku za pisanje modela i kontrolera projekta, dok u sklopu sa Express tehnologijom (server) pruža mogućnost rutiranja, korišćenja statičkih i json fajlova itd.

### MongoDB

NoSQL

MongoDB predstavlja sistem za upravljanje bazom podataka koja će se koristiti za realizaciju StanTech-a.

# Pogled na procese

U ovom odeljku je sadržan pogled na procesnu arhitekturu sistema. Ovaj opis treba da sadrži specifikaciju različitih zadataka (procesa i niti) uključenih u rad sistema. Takođe je potrebno dati dijagrame koji pokazuju njihovu interakciju i konfiguraciju. Dodela objekata i klasa na određene zadatke takođe spada u opis procesne arhitekture.

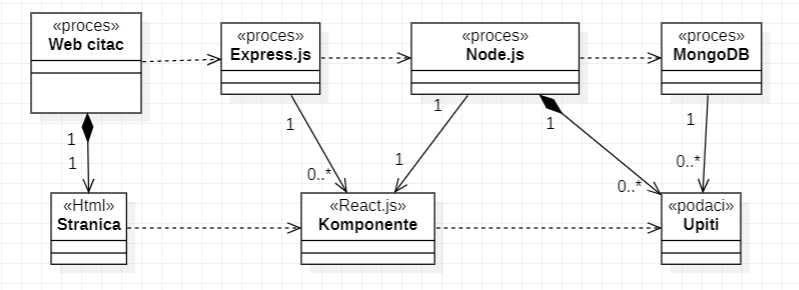
MERN stack dozvoljava kreiranje dinamičkih i interaktivnih web aplikacija kombinacijom MongoDB-a za čuvanje podataka, Express.js za serversku stranu logike,

Node.js za vreme rada na strani servera i React.js za korisnički interfejs na klijentskoj strani. Front i back komponente komuniciraju preko API-a, što dozvoljava neprimetnu interakciju između korisničkog interfejsa i back-end logike.

Ilustracije radi u nastavku je dat opis procesa uključenih u izvršenje StanTech kao Web aplikacije.

## Procesi

Na sledećem UML dijagramu klasa prikazani su procesi koji učestvuju u izvršenju StanTech. Dijagram je opšteg tipa i može se primeniti na bilo koju Web aplikaciju zasnovanu na MERN-u i Mongo bazi podataka.



### Web čitač

Web čitač je proces koji izvršava funkcionalnost aplikacije za prikaz HTML stranica dobijenih od nekog Web servera. U najopštijem slučaju Web čitač u jednom trenutku može da prikazuje samo jednu HTML stranicu.

Web čitač zavisi od Web servera koji generiše i vraća odgovarajuću HTML stranicu na zahtev.

### Express.js

Express.js je proces koji izvršava funkcionalnost opsluživanja zahteva prispelih sa više Web čitača. Ukoliko je zahtevana stranica, Express.js izvršava rutiranje i inicira izvršenje Node.js koji obrađujue odgovarajuće zahteve i generiše sadržaj koji se prosleđuje React.js sloju.

### Node.js

Node.js proces obavlja posao obrade zahteva koji dobije od React.js-a i generiše odgovarajući JSON sadržaj kao odgovor na dati zahtev. Node.js je taj koji komunicira sa Mongo bazom podataka preko moongose paketa. Moongose paket omogućuje lakšu interakciju Node.js-a sa Mongo bazom podataka.

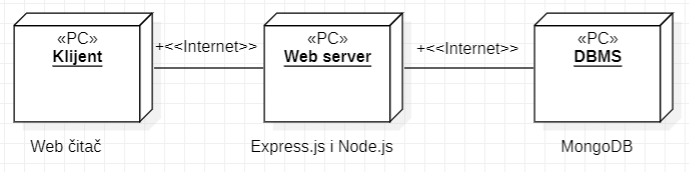
### MongoDB

MongoDB je proces koji izvršava funkcionalnost MongoDB sistema za upravljanje bazama podataka. Ovaj proces može konkurentno da prihvati određen broj upita, izvrši ih nad bazom podataka i vrati rezultate procesu koji je upite postavio.

# Pogled na raspoređivanje sistema

Pogled na raspoređivanje sistema prikazuje različite fizičke čvorove za najopštiju konfiguraciju sistema. Fizičkim čvorovima koji predstavljaju procesore vrši se dodeljivanje identifikovanih procesa.

Na sledećoj slici dat je UML dijagram raspoređivanja StanTech web aplikacije.



## Klijent

Pristup StanTech web aplikaciji se obavlja preko klijentskih računara na kojima se izvršava Web čitač. Za povezivanje između klijenta i Web servera koristi se Internet infrastruktura tako da nema ograničenja u pogledu lokacije klijenta.

## Web server

Računar na kome se izvršava Web server opslužuje više klijenata koji pristupaju preko Interneta. Pored osnovnog procesa koji realizuje funkcionalnost Web servera, na ovom računaru mogu da se izvršavaju i procesi Node.js-a koji vrše obradu zadatih React.js zahteva. U najopštioj konfiguraciji DBMS se izvršava na posebnoj mašini koja je sa Web serverom preko interneta.

## MongoDB server

MongoDB server je računar na kome se izvršava MongoDB Server proces koji realizuje funkcionalnost sistema za upravljanje bazama podataka. Zbog sigurnosti podataka koji se na ovom računaru čuvaju pristup bazi je ograničen samo na računare koji su na ,,beloj listi” u CORS-u.

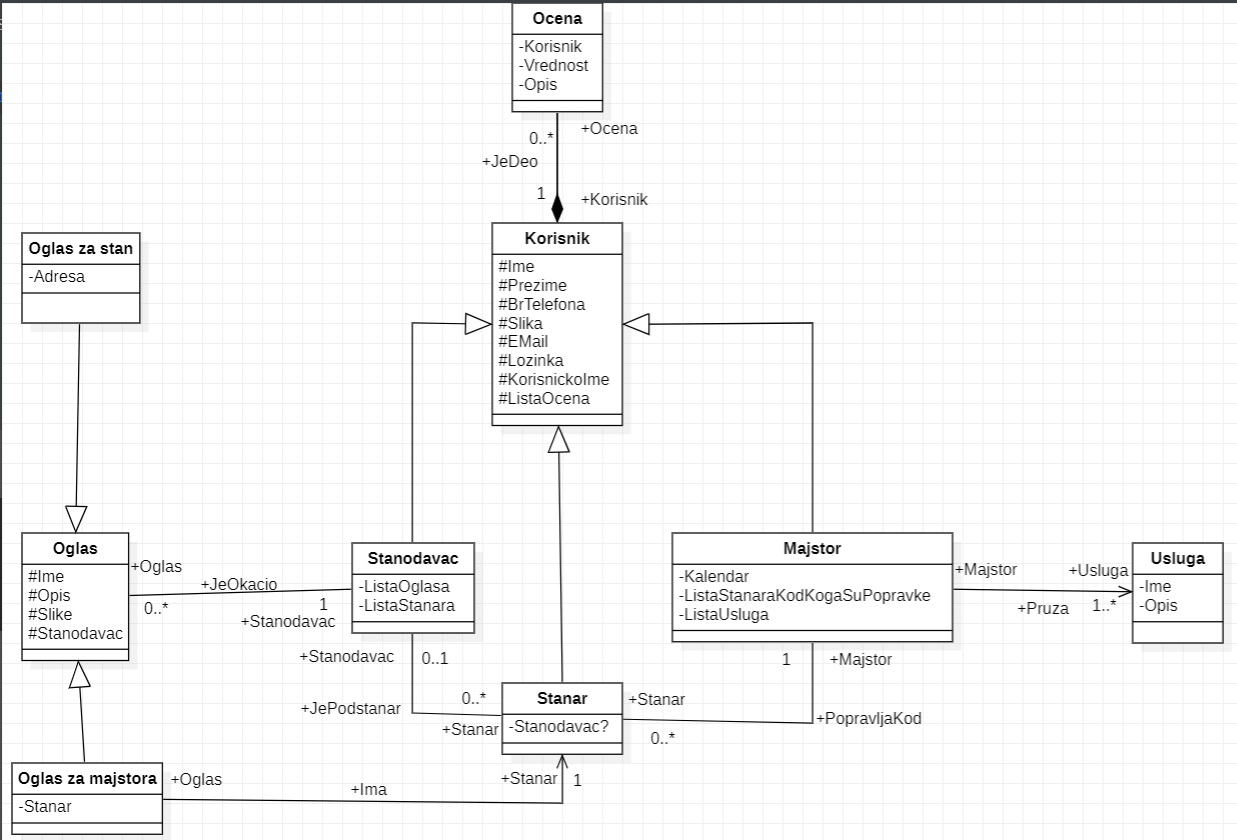
# Pogled na implementaciju sistema

Pogled na implementaciju prikazuje različite aspekte bitne za implementaciju sistema. U slučaju StanTech web aplikacije, ovaj odeljak sadrži model domena, šemu baze podataka i prikaz komponenti sistema razvrstanih u ranije identifikovane pakete.

## Model domena

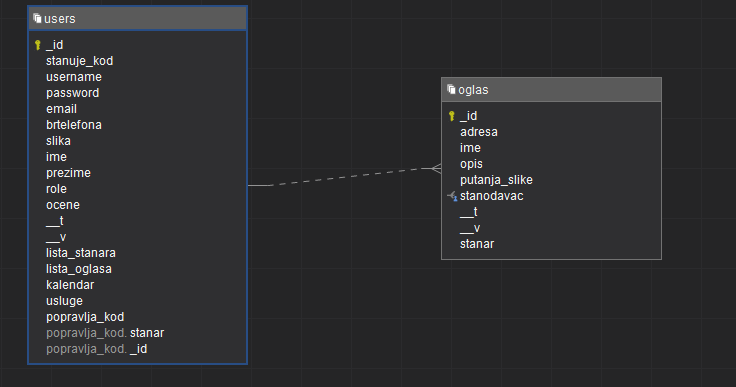
Model domena za koji se StanTech web aplikacija projektuje je ilustrovan UML dijagramom klasa. U njemu su prikazane domenske klase, neki od njihovih atributa, kao i veze koje se mogu identifikovati između njih.

Model domena predstavlja osnovu za projektovanje baze podataka.

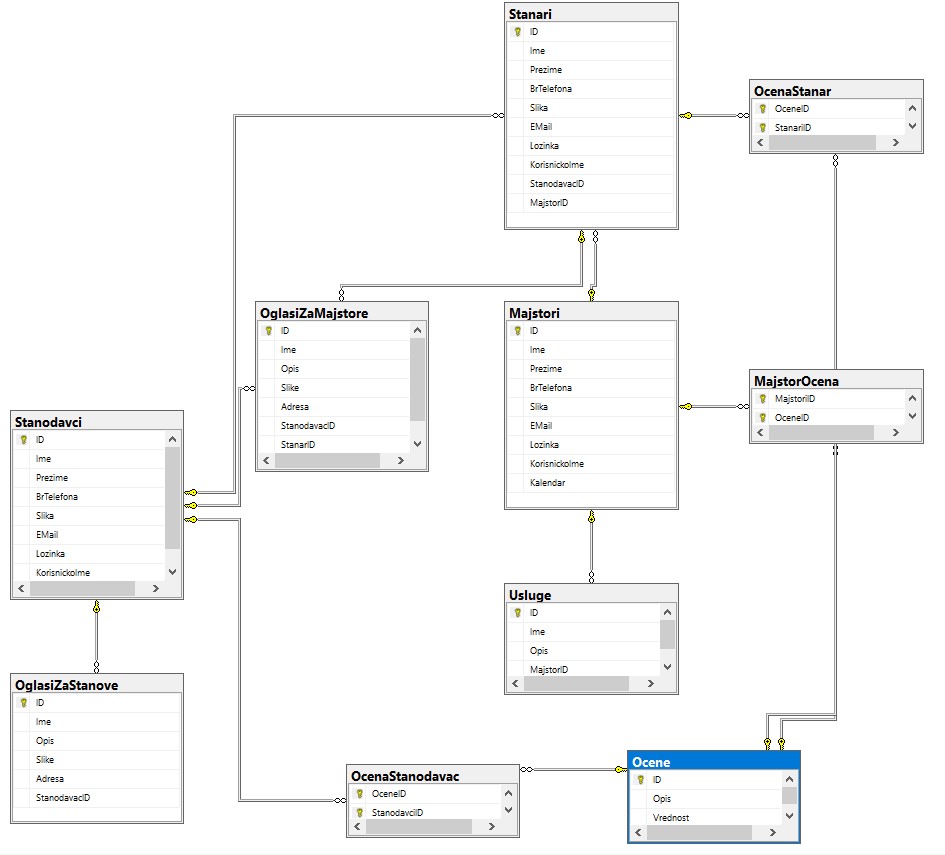


## Šema baze podataka

Detaljna šema baze podataka je prikazana na sledećem dijagramu. Baza podataka i dijagram su kreirani korišćenjem *Dataedo* i na slici se vidi NoSQL baza. Jedan od načina za nasleđivanja u MongoDB je korišćenje diskriminatora šema tako da on sve nasleđene šeme čuva u jednoj kolekciji.



Prikaz baze kada bi se koristila relaciona baza podataka kao što je MySql bi izgledala ovako

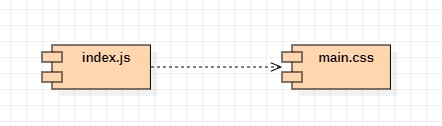


## Komponente sistema

Komponente sistema StanTech portala su React komponente čiji će pregled biti dat po arhitekturnim slojevima. Za ilustraciju će biti korišćeni UML dijagrami komponenti, ali i dijagrami klasa. U slučajevima gde je Node.js skript prikazan kao klasa atributi predstavljaju ulazne podatke koji se uzimaju iz GET ili POST dela HTTP poruke, dok metodi predstavljaju funkcije definisane u okviru skripta.

### Komponente korisničkog interfejsa

Dizajn korisničkog interfejsa je obuhvaćen dvema komponentama:

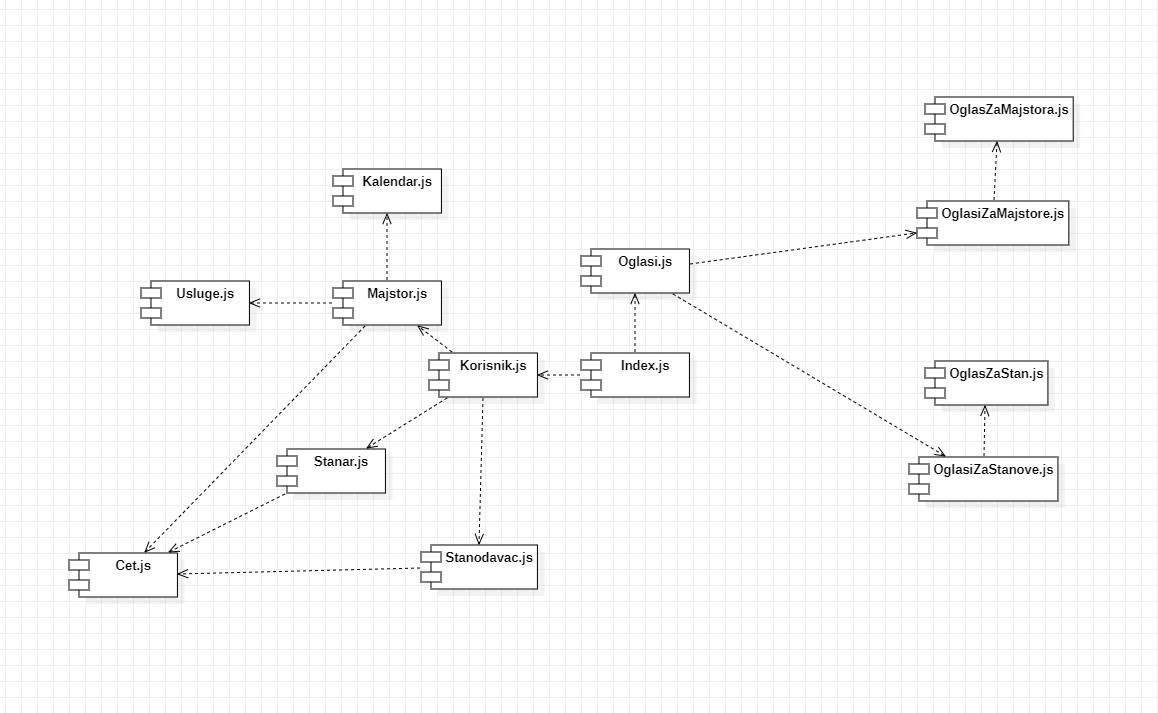


Komponenta **index.js** implementira stranicu portala čiji sadržaj može da varira od parametra koji joj se proslede u zahtevu.

Komponenta **main.css** predstavlja opis stilova za pojedine HTML elemente koji se javljaju na različitim stranicama.

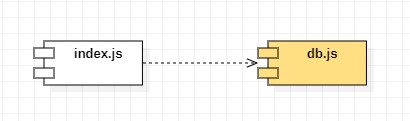
### Komponente aplikacione logike

Komponente koje realizuju domen problema se uključuju isključivo preko **index.js** komponente korisničkog interfejsa. Na taj način zadržavaju sva podešavanja stila definisana u ovom skriptu. Na sledećem dijagramu su prikazane komponente ovog sloja i njihove međusobne zavisnosti:

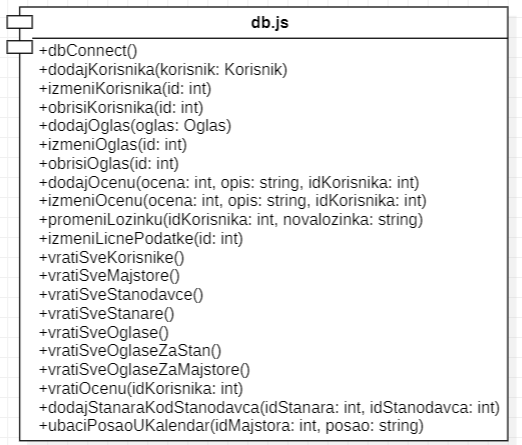


### Komponente za pristup podacima

Pristup bazi podataka je u potpunosti zatvoren u funkcije koje su definisane u okviru Node.js skripta **db.js**. Pomenuti skript se uključuje na početku index.js-a, tako da su sve funkcije za pristup podacima dostupne svim komponentama.



Na sledećem UML dijagramu klasa pobrojane su funkcije za pristup podacima iz baze:



Navedene funkcije obavljaju sledeće zadatke:

* dbConnect – povezivanje na bazu podataka, poziva se u samoj skripti
* dodajKorisnika– dodaje korisnika
* izmeniKorisnika– izmenjivanje korisnika
* obrisiKorisnika– briše korisnika
* dodajOglas– dodaje oglas
* izmeniOglas–izmenjivanje oglasa
* obrisiOglas– briše oglas
* dodajOcenu– dodaje ocenu
* izmeniOcenu – izmenjuje ocenu
* promeniLozinku– promena lozinke
* izmeniLicnePodatke– izmenjivanje ličnih podataka
* vratiSveKorisnike– vraća listu svih korisnika
* vratiSveMajstore– vraća listu svih majstora
* vratiSveStanodavce– vraća listu svih stanodavaca
* vratiSveStanare– vraća listu svih stanara
* vratiSveOglase– vraća listu svih oglasa
* vratiSveOglaseZaStan– vraća listu svih oglasa za stan
* vratiSveOglaseZaMajstore– vraća listu svih oglasa za majstore
* vratiOcenu– vraća ocenu korisnika
* dodajStanaraKodStanodavaca– dodaje stanara kod stanodavca
* ubaciPosaoUKalendar– dodaje datum u niz datuma koji označavaju kad majstor treba da odradi posao

# Performanse

Izabrana arhitektura softvera podržava zahteve u pogledu broja korisnika koji mogu simultano pristupati sistemu i vremena odziva za pristup bazi podataka specificirane u zahtevima u pogledu performansi [5]:

1. Sistem će da podrži do 1000 simultanih pristupa korisnika web aplikaciji.
2. Vreme potrebno za pristupanje bazi podataka u cilju izvršenje nekog upita ne sme da bude veće od 5 sekundi.

Zahtevane performanse su zadovoljene izborom tehnologija na kojima će sistem biti razvijen i definisane hardverske platforme [5].

# Kvalitet

Izabrana arhitektura softvera podržava zahteve u pogledu dostupnosti i srednjeg vremena između otkaza specificirane u zahtevima u pogledu pouzdanosti [5]:

1. StanTech web aplikacija će biti dostupna 24 časa dnevno, 7 dana u nedelji. Vreme kada web aplikacija nije dostupan ne sme da pređe 10%.
2. Srednje vreme između dva sukcesivna otkaza ne sme da padne ispod 120 sati.