MeinMeister

Arhitekturni projekat

Verzija 1.0

Pregled izmena

|  |  |  |  |
| --- | --- | --- | --- |
| **Datum** | **Verzija** | **Opis** | **Autor** |
| 21.04.2023. | 1.0 | Inicijalna verzija | Darko Gligorijevic,  Darko Gligorijevic |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

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 majstora 8

6.2.2 Registrovanje 8

6.2.3 Prihvatanje zahteva 8

6.2.4 Pregled usluga 8

6.2.5 Odabir usluga 8

6.2.6 Odabir majstora 8

6.2.7 Dodavanje usluga 8

6.2.8 Brisanje usluga 8

6.2.9 Azuriranje informacija o uslugama 8

6.2.10 Prijavljivanje 8

6.2.11 Ocenjivanje majstora 8

6.2.12 Upravljanje nepozeljnim sadrzajem 9

6.2.13 Brisanje korisnika 9

6.2.14 Azuriranje korisnika 9

7. Pogled na logičku arhitekturu sistema 9

7.1 Pregled arhitekture – organizacija paketa i podsistema u slojeve 10

7.1.1 Korisnički interfejs 10

7.1.2 Aplikaciona logika 10

7.1.3 Pristup podacima 10

7.1.4 HTML 10

7.1.5 ReactJS 11

7.1.6 NodeJS 11

7.1.7 MongoDB 11

8. Pogled na procese 11

8.1 Procesi 11

8.1.1 Web čitač 12

8.1.2 Web server 12

8.1.3 NodeJS 12

8.1.4 MongoDB NoSQL Server 12

9. Pogled na raspoređivanje sistema 12

9.1 Klijent 12

9.2 Web server 12

9.3 DBMS server 12

10. Pogled na implementaciju sistema 13

10.1 Model domena 13

10.2 Šema baze podataka 14

10.3 Komponente sistema 14

10.3.1 Komponente korisničkog interfejsa 14

10.3.2 Komponente aplikacione logike 19

10.3.3 Komponente za pristup podacima 20

11. Performanse 21

12. Kvalitet 21

Arhitekturni projekat

# Cilj dokumenta

Cilj ovog dokumenta je detaljni opis arhitekture MeinMeister portala.

# Opseg dokumenta

Dokument se odnosi na MeinMeister portal koji će biti razvijen od strane tima DoubleD-a. Namena sistema je pretrazivanje majstora u cilju zadovoljavanja potreba korisnika.

# Reference

Spisak korišćene literature:

1. MeinMeister – Predlog projekta, V1.1, 2023, DoubleD.
2. MeinMeister – Planirani raspored aktivnosti na projektu, V1.1, 2023, DoubleD.
3. MeinMeister – Plan realizacije projekta, V1.0, 2023, DoubleD.
4. MeinMeister – Vizija sistema, V1.1, 2023, DoubleD.
5. MeinMeister – Specifikacija zahteva, V1.1, 2023, DoubleD.

# 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. MeinMeister aplikacija će biti implementirana kao Web aplikacija zasnovana na React-u i NodeJS-u.
2. Klijentski deo MeinMeister aplikacije će biti optimizovan za sledeće Web čitače: Opera 8.0 i noviji, kao i Firefox (Mozilla), i na Google Chrome.
3. Svi zahtevi u pogledu performansi dati u 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].

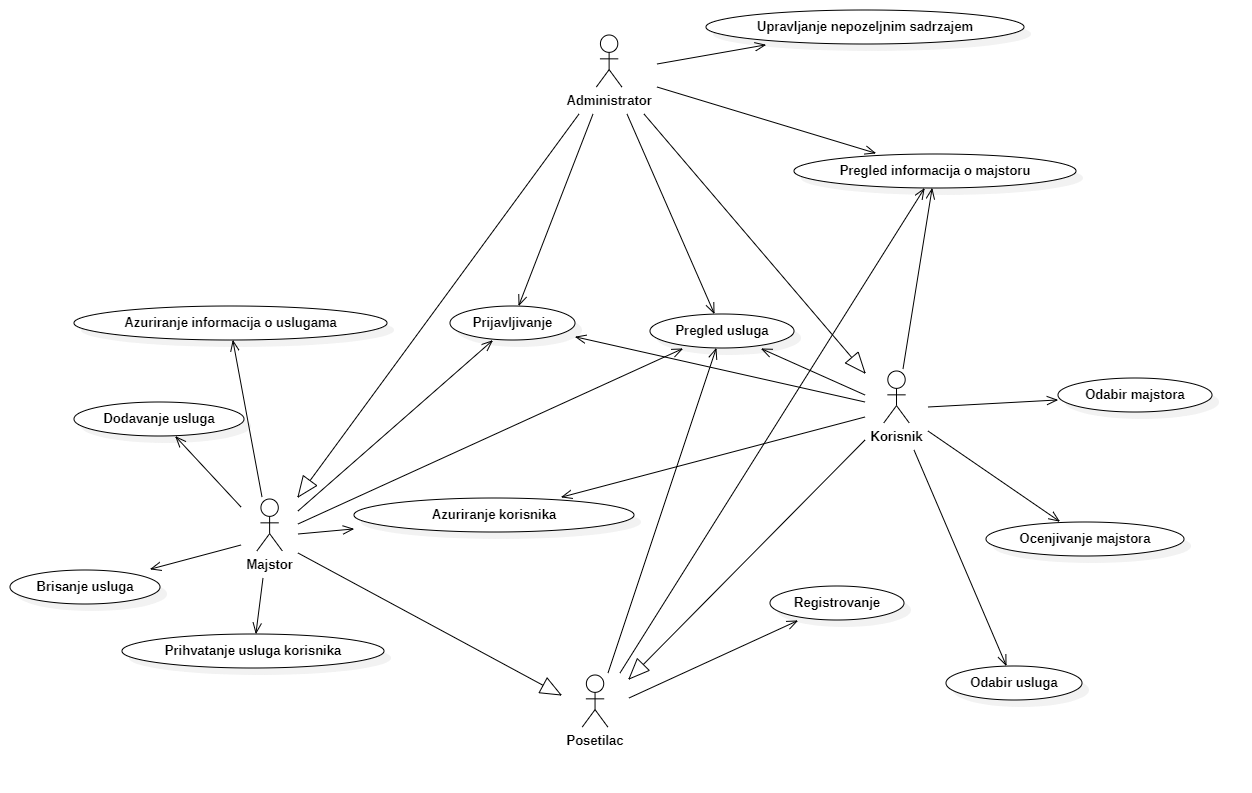
Slučajevi korišćenja MeinMeister aplikacije su:

* Prijavljivanje
* Odabir majstora
* Dodavanje usluga
* Brisanje usluga
* Prihvatanje usluga korisnika
* Azuriranje informacija o uslugama
* Ocenjivanje majstora
* Upravljanje nepozeljnim sadrzajem
* Azuriranje korisnika i majstora
* *Pregled informacija o uslugama*
  + *Pregled informacija*
  + *Pregled recenzija*
  + *Pregled ocena*
* *Upravljanje registracijom*
  + *Unos imena*
  + *Unos prezimena*
  + *Unos email-a*
  + *Unos korisnicke lozinke*
  + *Unos slike*
* *Upravljanje uslugama*
  + *Pregled usluga*
  + *Odabir usluga*

Ove slučajevi korišćenja mogu da iniciraju posetilac portala, korisnik, majstor i administrator.

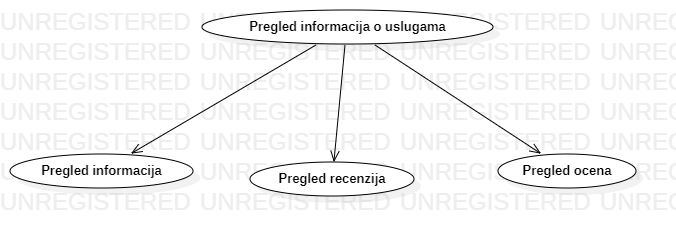
## Dijagrami slučajeva korišćenja

Osnovni UML dijagram koji prikazuje korisnike i slučajeve korišćenja MeinMeister portala prikazan je na sledećoj slici:



Slučajevi korišćenja *pregled informacija o uslugama, registrovanje i pregled i odabir zahteva* obuhvataju složenije radnje koje se mogu razložiti dalje razložiti na pojedinačne slučajeve korišćenja.

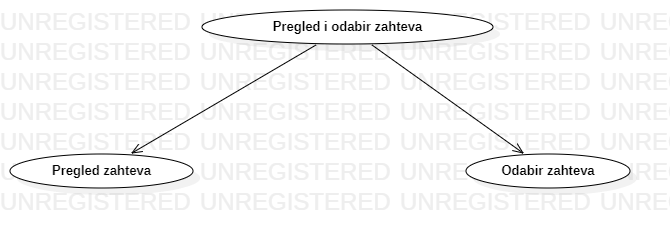
Detaljni UML dijagram za slučaj korišćenja *pregled informacija o uslugama* je prikazan na sledećoj slici:



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



Detaljni UML dijagram za slučaj korišćenja *pregled i odabir zahteva* je prikazan na sledećoj slici:



## Kratak opis slučajeva korišćenja

### Registrovanje

Kratak opis: Prikaz stranice sa formom za registrovanje gde korisnik unosi odredjene informacije da bi se registrovao.

Akteri koji iniciraju slučaj korišćenja: Posetilac portala.

### Prihvatanje zahteva

Kratak opis: Prikaz stranice sa pristiglim zahtevima za majstora i mogucnost odabira zahteva.

Akteri koji iniciraju slučaj korišćenja: Majstor.

### Pregled zahteva

Kratak opis: Prikaz stranice portala sa svim informacijama o korisniku koji je poslao zahtev.

Akteri koji iniciraju slučaj korišćenja: Majstor.

### Pregled usluga

Kratak opis: Prikaz stranice portala sa svim pristiglim zahtevima od strane majstora

Akteri koji iniciraju slučaj korišćenja: Korisnik.

### Odabir usluga

Kratak opis: Prikaz stranice sa informacijama o svim pristiglim zahtevima i odabir odgovarajucih usluga.

Akteri koji iniciraju slučaj korišćenja: Korisnik.

### Dodavanje usluga

Kratak opis: Prikaz stranice sa formom za dodavanje usluga.

Akteri koji iniciraju slučaj korišćenja: Majstor.

### Brisanje usluga

Kratak opis: Prikaz stranice sa opcijom za brisanje usluga kojeg je majstor prethodno dodao na portal

Akteri koji iniciraju slučaj korišćenja: Majstor, Administator

### Azuriranje informacija o uslugama

Kratak opis: Prikaz stranice sa sa opcijom za azuriranje podataka o uslugama, gde masjtor bira kojim uslugama ce izmeniti podatke.

Akteri koji iniciraju slučaj korišćenja: Majstor

### Prijavljivanje

Kratak opis: Prijavljivanje korisnika na portal u cilju pristupa specifičnim funkcijama koje zahtevaju autorizaciju

Akteri koji iniciraju slučaj korišćenja: Korisnik, Majstor, Administrator

### Ocenjivanje majstora

Kratak opis: Kada majstor zavrsi sa uslugama, korisnik moze da oceni i komentarise majstora.

Akteri koji iniciraju slučaj korišćenja: Korisnik.

### Upravljanje nepozeljnim sadrzajem

Kratak opis: Ukoliko dodje do postavljanja nepozeljnog sadrzaja na portalu, administrator je duzan da reaguje .

Akteri koji iniciraju slučaj korišćenja: Administrator.

### Brisanje korisnika

Kratak opis: Korisnik ima mogucnost brisanja naloga.

Akteri koji iniciraju slučaj korišćenja: Korisnik.

### Azuriranje korisnika

Kratak opis: Ukoliko korisnik želi da izmeni neke podatke o sebi ima mogućnost ažuriranja svojih podataka.

Akteri koji iniciraju slučaj korišćenja: Korisnik

# 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.

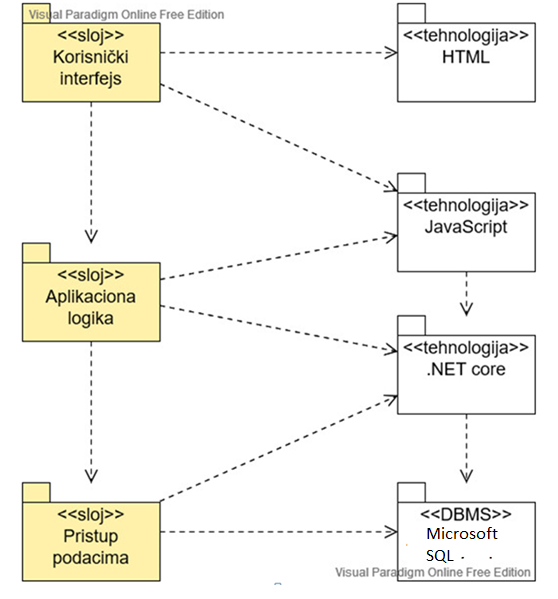
Logički pogled na MeinMeister portal obuhvata 3 glavna paketa: Korisnički interfejs, Aplikaciona logika, Pristup podacima.

Paket *Korisnički interfejs* sadrži Web stranice, JavaScript(ReactJS) 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 JavaScript skripte zadužene za realizaciju funkcionalnosti specifičnih za domen sistema koji se razvija.

Paket *Pristup podacima* sadrži NodeJS CRUD operacije koje predstavljaja interfejs za pristup, dodavanje i ažuriranje podataka koji se čuvaju u bazi podataka.

## Pregled arhitekture – organizacija paketa i podsistema u slojeve



NodeJS

mongoDB

### Korisnički interfejs

sloj

Ovaj sloj realizuje korisnički interfejs portala. U njemu su sadržane sve HTML, multimedijalni sadržaji i ReactJS skripte koje generišu HTML stranice preko kojih korisnici komuniciraju sa sistemom.

Sloj korisničkog interfejsa zavisi od sloja aplikacione logike, kao i paketa HTML i ReactJS.

### Aplikaciona logika

sloj

Sloj aplikacione logike je srednji sloj u troslojnoj arhitekturi MeinMeister portala. Sadrži JavaScript skripte koje realizuju funkcionalnost karakterističnu za domen primene portala i uspostavljaju vezu između korisničkog interfejsa i sloja za pristup podacima.

Ovaj sloj zavisi od sloja za pristup podacima i JavaScript paketa.

### Pristup podacima

sloj

Sloj za pristup podacima se nalazi na dnu troslojne arhitekture i sadrži NodeJS skripta zadužene za pribavljanje, dodavanje i ažuriranje podataka koji se čuvaju u mongoDB.

Ovaj sloj ne zavisi od drugih slojeva, ali je zavisan od JavaScript paketa i mongoDB-a.

### HTML

tehnologija

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

### ReactJS

Tehnologija

Tehnologija ReactJS-a obezbeđuje mehanizam za pisanje skripti koje će se izvršavati na klijentskoj strani, koje će dinamički generisati HTML elemente i povezati događaje za HTML elemente.

### NodeJS

tehnologija

Tehnologija NodeJS obezbeđuje mehanizam za pisanje i izvršavanje koda na strani servera. Ovaj kod može generisati zahteve koji pristupaju bazi podataka u cilju pribavljanja, unosa i ažuriranja podataka.

### MongoDB

DBMS

mongoDB predstavlja sistem za upravljanje bazama podataka koji će se koristiti za realizaciju MeinMeister portala.

# Pogled na procese

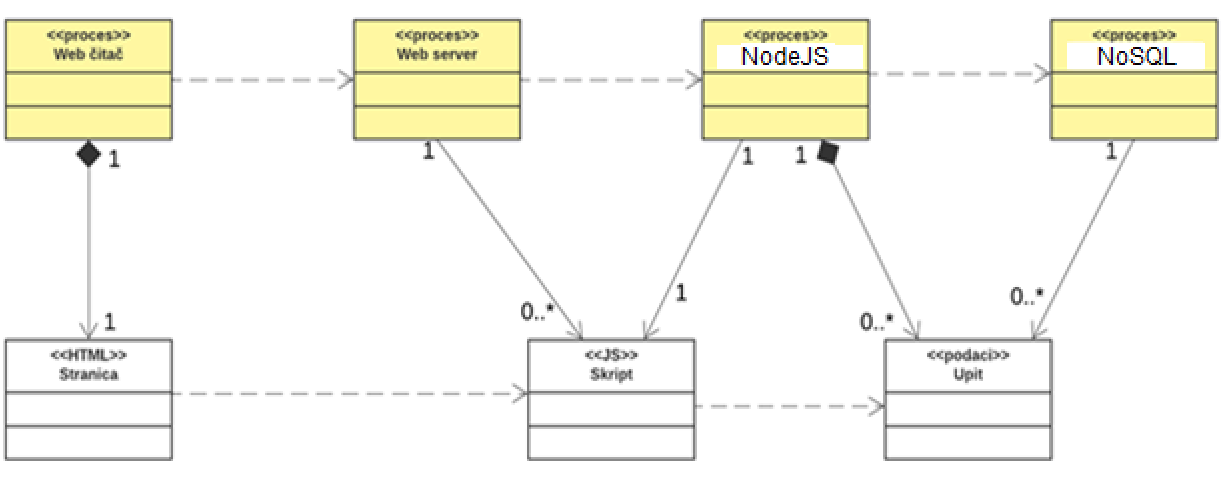
U ovom odeljku je sadržan pogled na procesnu arhitekturu sistema. Ovaj opis treba da sadrži specifikaciju različitih zadataka (procesa i niti) uključenih u rad sistema. Takođe je potrebno dati dijagrame koji pokazuju njihovu interakciju i konfiguraciju. Dodela objekata i klasa na određene zadatke takođe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

Web aplikacije zasnovane na NodeJS-u imaju relativno jednostavan procesni model koji je u potpunosti pod kontrolom Web servera. Sa stanovišta projektanta NodeJS Web aplikacije nije potrebno voditi računa o načinu rada Web servera i načinu izvršavanja skripti.

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

## Procesi

Na sledećem UML dijagramu klasa prikazani su procesi koji učestvuju u izvršenju MeinMeister portala. Dijagram je opšteg tipa i može se primeniti na bilo koju Web aplikaciju zasnovanu na mongoDB 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.

### Web server

Web server je proces koji izvršava funkcionalnost opsluživanja zahteva prispelih sa više Web čitača. Ukoliko je zahtevana stranica JavaScript skript, Web server inicira izvršenje NodeJS procesa koji obrađuje odgovarajući skript i generiše sadržaj koji se vraća čitaču. Web server može paralelno da inicira veći broj NodeJS procesa.

### NodeJS

*NodeJS*  proces obavlja posao obrade zadatog JavaScript skripta i generiše odgovarajući tekstualni sadržaj koji Web server šalje Web čitaču. Za izvršenje JavaScript skripta ovaj proces može da zahteva usluge NoSQL servera-a. Komunikacija između JavaScript i NoSQL servera se obavlja preko prosleđivanja upita i vraćanja rezultat.

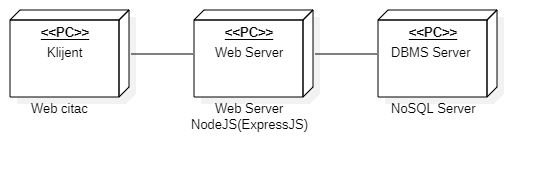
### mongoDB NoSQL Server

mongoDB NoSQL Server 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 MeinMeister portala.



## Klijent

Pristup MeinMeister portala 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 NodeJS-a koji vrše obradu zadatih JavaScript skripti. U najopštioj konfiguraciji DBMS se izvršava na cloud-u.

## DBMS server

DBMS server je računar na kome se izvršava AWS 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 cija je ip adresa odobrena na AWS serveru.

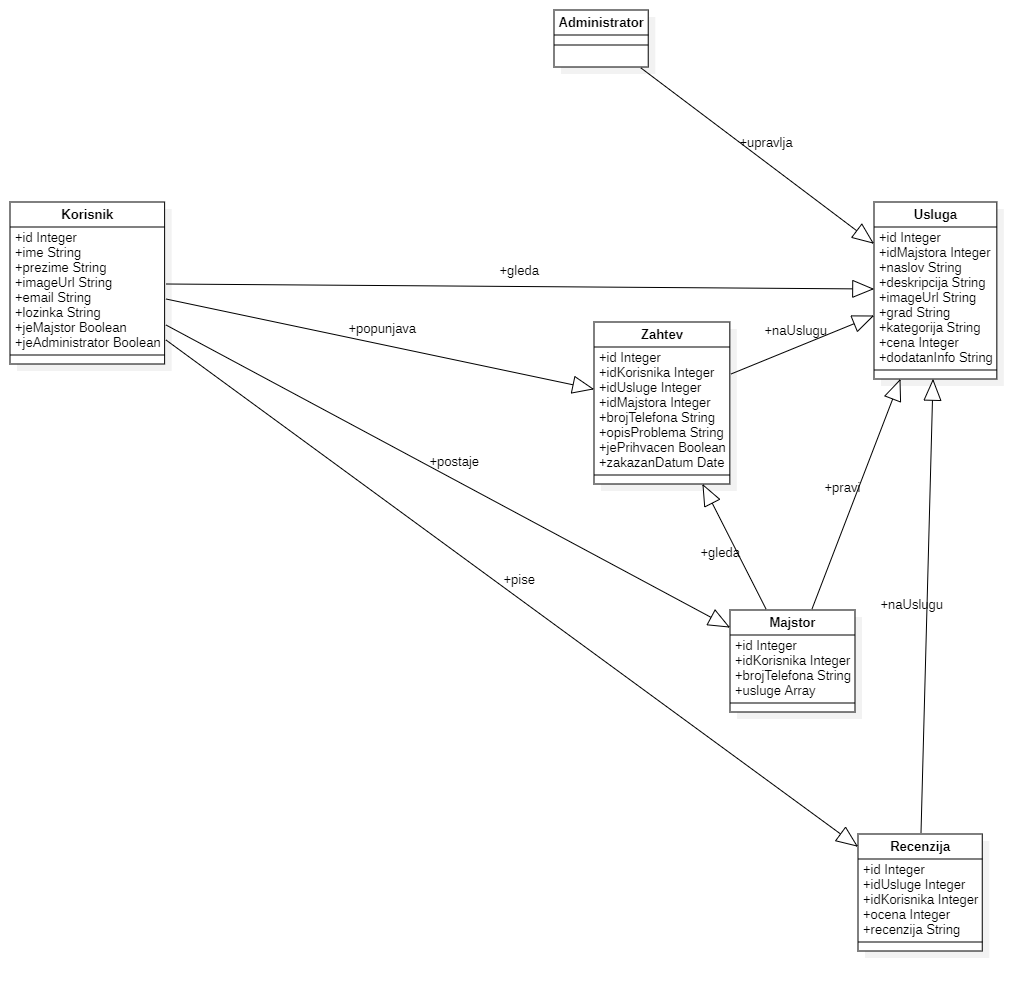
# Pogled na implementaciju sistema

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

## Model domena

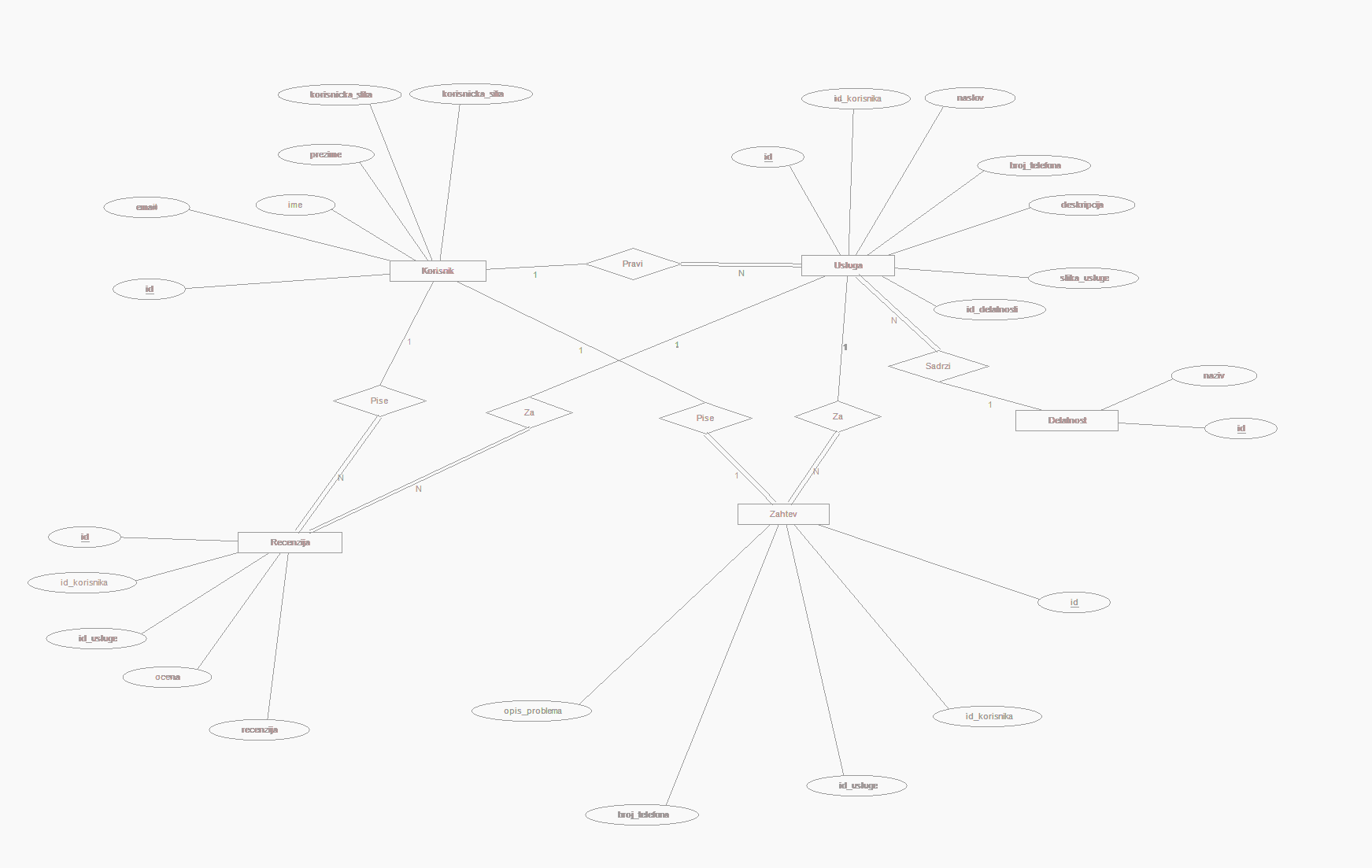
Model domena za koji se MeinMeister portal projektuje je ilustrovan UML dijagramom klasa. U njemu su prikazane domenske klase, neki od njihovih atributa, kao i veze koje se mogu identifikovati između njih.

Model domena predstavlja osnovu za projektovanje baze podataka, ali i identifikaciju nekih od komponenti (ReactJS skripti) koje će biti implementirane.



## Šema baze podataka

Detaljna šema baze podataka je prikazana na sledećem dijagramu. Baza podataka i dijagram su kreirani korišćenjem *SharpEER programa*.



## Komponente sistema

Komponente sistema MeinMeister portala su ReactJS skripte čiji će pregled biti dat po arhitekturnim slojevima. Za ilustraciju će biti korišćeni UML dijagrami komponenti, ali i dijagrami klasa. U slučajevima gde je ReactJS skripta prikazana kao klasa atributa koji predstavljaju ulazne podatke koji se uzimaju iz GET ili POST dela HTTP poruke, dok metodi predstavljaju funkcije definisane u okviru skripta.

### Komponente korisničkog interfejsa

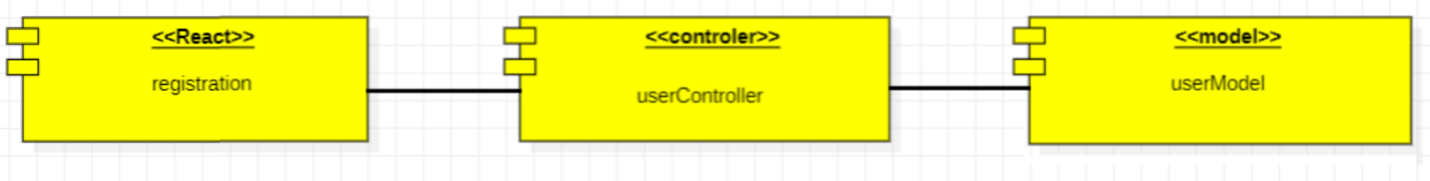
Dizajn korisničkog interfejsa je obuhvaćen dvema komponentama:



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

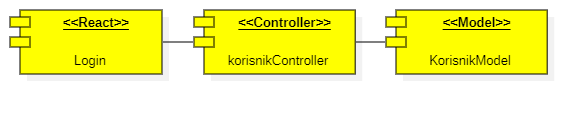
Komponenta **style.css** predstavlja opis stilova za pojedine HTML elemente koji se javljaju na različitim stranicama.

**Registracija:**



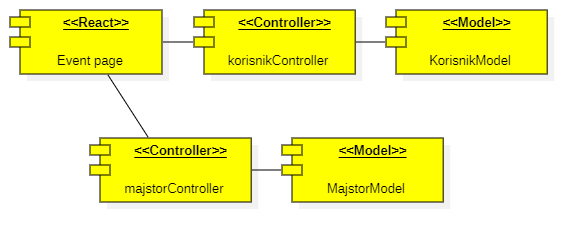
Potencijalni korisnik portala na stranici za kreiranje naloga unosi svoje podatke. ReactJS skripta prosleđuje te podatke User Controlleru za registraciju koji ih prosleđuje User Modelu koji vrši njihovu proveru i vraća korisnika na stranicu za prijavljivanje ili ga obaveštava o nevalidnosti određenih podataka. Te podatke pokuplja ReactJS i prosleđuje ih korisniku. Ukoliko je registracija uspešna korisnik se prosleđuje na portal kao ulogovani korisnik.

**Prijavljivanje:**



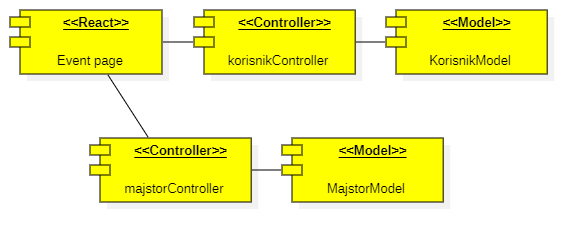
Korisnik na Login stranici popunjava formu sa svojim podacima za prijavu. ReactJS te podatke prihvata i prosleđuje Korisnik Controlleru koji ih dalje prosleđuje Korisnik Modelu koji vrši proveru heširanih podataka kao što su korisnicko ime i password i ukoliko je prijava uspešna, tu informaciju pokuplja ReactJS koji prosleđuje korisnika na portal kao ulogovanog korisnika. Ukoliko je prijava neuspešna, ReactJS skripta prikazuje tu informaciju korisniku.

**Ocenjivanje majstora:**



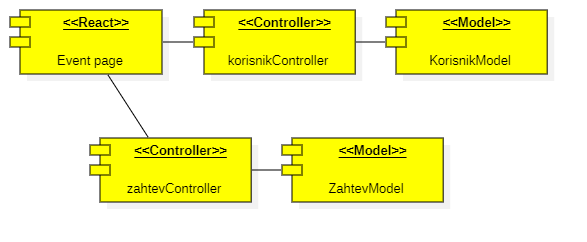
Nakon što se svaka uspesno pruzena usluga obavi, biće omogućeno da korisnik oceni majstora.. Cilj ocenjivanja je da se postigne određeni nivo poverenja o majstoru i da se poboljša korisničko iskustvo. Ocene korisnika biće dostupne i javne na stranici korisničkog profila, da bi ih pored korisnika, videli i naivni korisnici, tj. posetioci portala.

**Odabir majstora:**



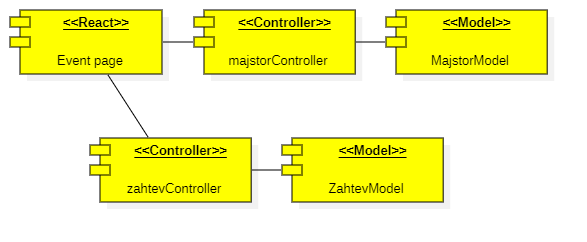
Korisnik ima mogucnost odabira majstora na stranici web portala. Ne postoji neki, unapred definisani sablon prilikom odabira majstora, vec ce se odabir majstora vrsiti na osnovu potreba korisnika.

**Slanje zahteva:**

****

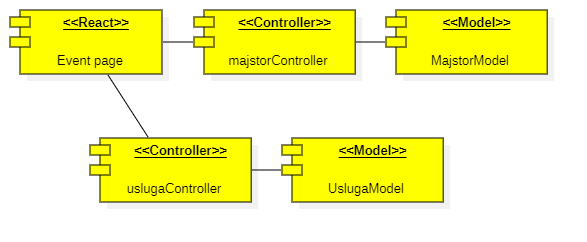
Korisnik klikom na dugme Zaposli, salje korisnika na drugoj stranici, gde popunjuje formu o problemu, gde ce majstoru detaljnije opisati problem, i ta forma ce biti poslata zahtevControlleru, koja ce informaciju ubaciti u bazi nakon cega se ceka odgovor majstora.

**Odabir zahteva:**



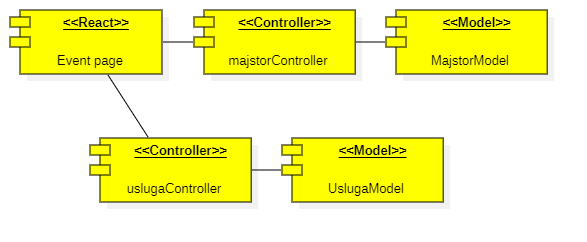
Kada majstor dobije razlicit broj zahteva, on ce na osnovu razlicith stvari birati onaj zahtev koji njemu najvise odgovara. Nakon ovoga bice inicirana komunikacija izmedju korisnika i majstora.

**Dodavanja usluga:**



Kontroler uz pomoć Modela kreira funkciju koja ce biti zaduzena za to da sa ReactJS u formatu JSON-a primi podatke i smesti u bazi podataka. Nakon smestanja salje odgovor ReactJS-u da li je smestanje uspesno obavljeno ili je doslo do greske priliko smestanja.

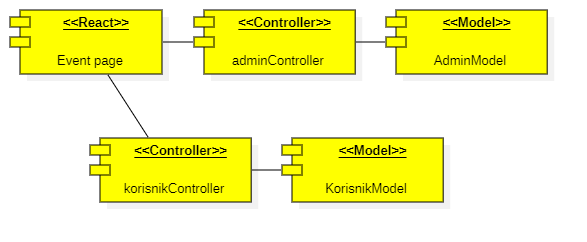
**Brisanje usluga:**



.

Ukoliko majstor iz nekog razloga odluci da obrise uslugu sa web portala, to moze da uradi klikom na dugme Obrisi. Nakon ovoga, rezultat ove akcije se prosledjuje uslugaControlleru i sve informacije o obrisanom usluzi se uklanjaju.

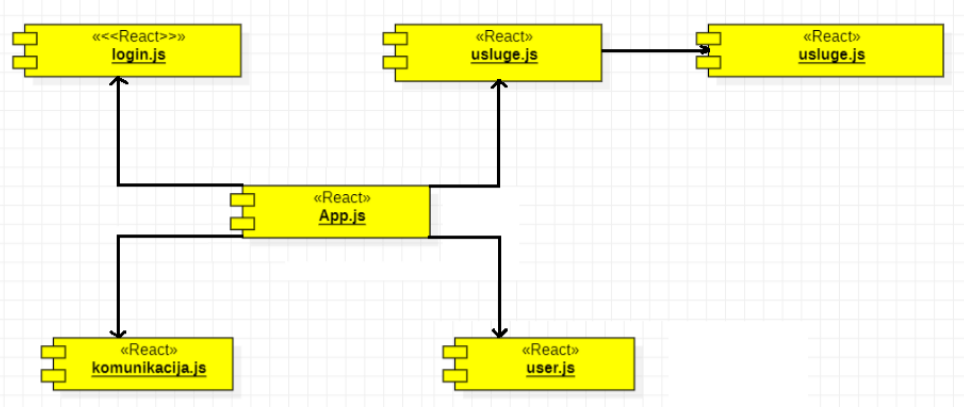
**Brisanje korisnika:**

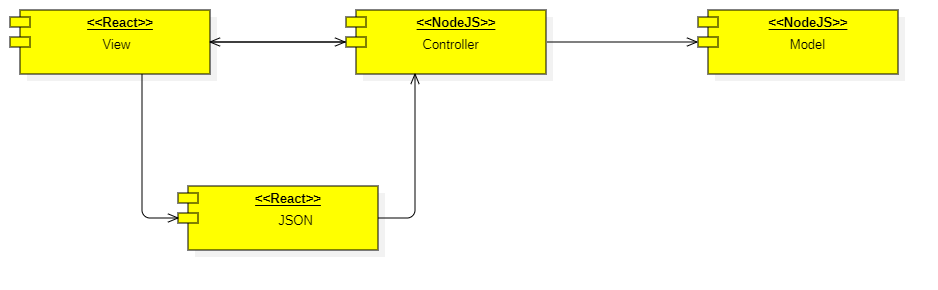


Na adminovoj stranici se nalazi kontrolna tabla sa svim prijavljenim korisnicima, koji su ranije bili dodati od strane administratora. Admin ima mogucnost brisanja korisnika ukoliko ostavlja neprimeren sadrzaj.

### Komponente aplikacione logike

Komponente koje realizuju domen problema se uključuju isključivo preko **App.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:



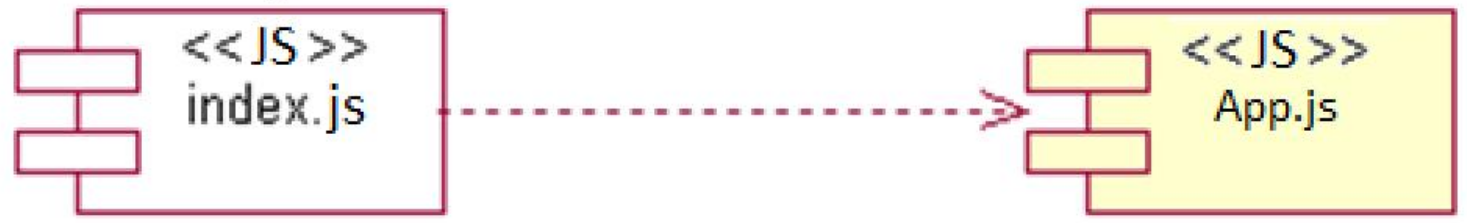


MeinMeister portal je baziran na Model-View-Controller (MVC) projektnom obrascu. MVC je softverski pristup koji razdvaja aplikacionu logiku od prezentacione.

* **Model** predstavlja strukturu podataka. Mode klase sadrži funkcije pomoću kojih preuzimamo, dodajemo i ažuriramo podatke u bazi podataka.
* **View** je skup informacija koje se predstavljaju korisniku. View može biti stranica ali isto tako može biti deo kao što je header ili footer.
* **Controller** je međusloj između modela, view-a i bilo kog drugog resursa potrebnog za obradu HTTP zahteva i generisanja web stranice.

### Komponente za pristup podacima

Komponente koje realizuju domen problema se uključuju isključivo preko **App.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:



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



Navedene funkcije obavljaju sledeće zadatke:

* postUserRegister – kreira korisnika i skladisti podatke u bazi
* postUserLogin – proverava podatke korisnika, i loguje ga ukoliko su ispravni
* getOrderById – vraca zahtev na osnovu id zahteva
* postOrderByPostId – kreira zahtev za specificnu uslugu
* getAllOrdersByWorkerId – vraca sve zahteve specificnog majstora
* getPostById– vraca uslugu na osnovu jedinstvenog ID usluge
* deletePostById – brise specificnu uslugu
* updatePostById– azurira specificnu uslugu
* ... – ostale funkcije

# 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 portalu.
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. MeinMeister portal će biti dostupan 24 časa dnevno, 7 dana u nedelji. Vreme kada portal nije dostupan ne sme da pređe 10%.
2. Srednje vreme između dva sukcesivna otkaza ne sme da padne ispod 120 sati.