# **KućneŽurke**

# **Sistem koji će omogućiti lakšu organizaciju i pristustvo na žurkama**

Arhitekturni projekat

Verzija 1.0

Pregled izmena

|  |  |  |  |
| --- | --- | --- | --- |
| **Datum** | **Verzija** | **Opis** | **Autor** |
| 30.04.2020. | 1.0 | Inicijalna verzija | Vidosava,  Miloš,  Aleksandar |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

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 6

6.1 Dijagrami slučajeva korišćenja 7

6.2 Kratak opis slučajeva korišćenja 8

6.2.1 Pristup osnovnim podacima sajta 8

6.2.2 Mogućnost ocenjivanja drugih korisnika 8

6.2.3 Mogućnost ocenjivanja žurki 8

6.2.4 Posedovanje profila 8

6.2.5 Registracija 8

6.2.6 Logovanje 8

6.2.7 Promena podataka o sebi 8

6.2.8 Promena lozinke 8

6.2.9 Unos podataka o kućnim žurkama 8

6.2.10 Ažuriranje podataka o kućnim žurkama 9

6.2.11 Odgovor na zahtev za posetu žurke 9

6.2.12 Prikaz liste zahteva 9

6.2.13 Prikaz podataka o konkretnim kućnim žurkama 9

6.2.14 Filtriranje prilikom pregleda 9

6.2.15 Slanje zahteva za pristustvo na žurkama 9

6.2.16 Brisanje postojećih žurki 9

6.2.17 Brisanje naloga 10

7. Pogled na logičku arhitekturu sistema 10

7.1 Pregled arhitekture – organizacija paketa i podsistema u slojeve 10

7.1.1 Korisnički interfejs 11

7.1.2 Aplikaciona logika 11

7.1.3 Pristup podacima 11

7.1.4 React 11

7.1.5 Node.js 11

7.1.6 MongoDB 11

8. Pogled na procese 11

8.1 Procesi 11

8.1.1 Web čitač 11

8.1.2 Web server 12

8.1.3 Node CGI 12

8.1.4 MongoDB Server 12

9. Pogled na raspoređivanje sistema 12

9.1 Klijent 13

9.2 Web server 13

9.3 DBMS server 13

10. Pogled na implementaciju sistema 13

10.1 Model domena 13

10.2 Komponente sistema 15

10.2.1 Komponente korisničkog interfejsa 16

10.2.2 Komponente aplikacione logike 19

10.2.3 Komponente za pristup podacima 19

11. Performanse 20

12. Kvalitet 20

Arhitekturni projekat

# Cilj dokumenta

Cilj ovog dokumenta je detaljni opis arhitekture web aplikacije KućneŽurke.

# Opseg dokumenta

Dokument se odnosi na aplikaciju KućneŽurke koji će biti razvijena od strane MARSTeam-a. KućneŽurke predstavlja skraćenicu za Sistem koji će omogućiti lakšu organizaciju i pristustvo na žurkama .

# Reference

Spisak korišćene literature:

1. SWE\_01\_Predlog\_Projekta, 2020, MARSTeam.
2. SWE\_02\_Vizija\_Sistema, V1.0, 2020, MARSTeam
3. SWE\_03\_Plan\_Realizacije, V1.0, 2020, MARSTeam.
4. SWE\_AV02\_Raspored\_aktivnosti, V1.0, 2020, MARSTeam.
5. SWE\_04\_ Spec\_Zahteva, V1.0, 2020, MARSTeam.

# 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

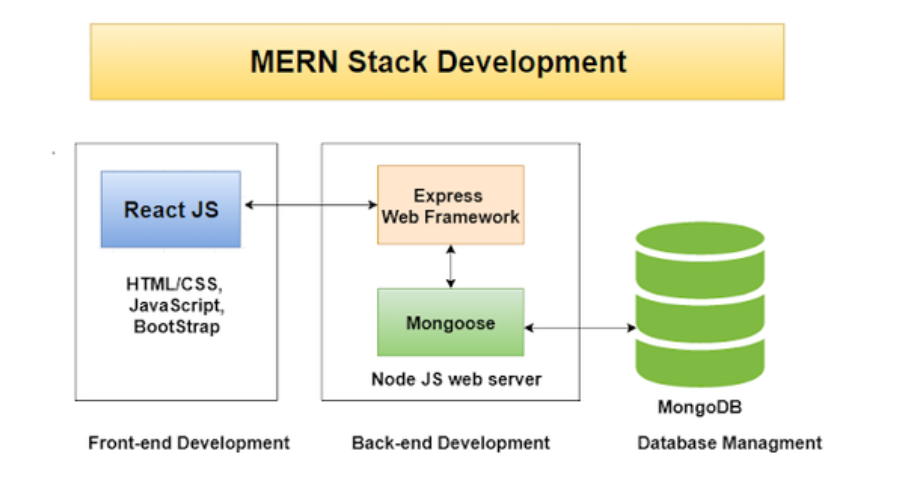
Ključni zahtevi i sistemska ograničenja koja imaju značajan uticaj na izbor arhitekture i projektovanje sistema su:

1. KućneŽurke aplikacija će biti implementirana kao Web aplikacija koja će biti zasnovana na **MERN** steku. Što podrazumeva

* **MongoDB** bazu podataka
* **Express -** framework za Node.js ,
* **React** -koji predstavlja JavaScript front-end za kreiranje korisničkog interfejsa i
* **Node.js**. Node.js je radno okruženje JavaScript programskog jezika koje se koristi na strani web servera .

1. Klijentski deo web aplikacije KućneŽurke će biti optimizovan za web čitače.
2. Svi zahtevi u pogledu performansi dati u [5] moraju biti uzeti u obzir pri izboru arhitekture i razvoju sistema.

MERN stack predstavljen naslikovit način:



# Pogled na slučajeve korišćenja

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

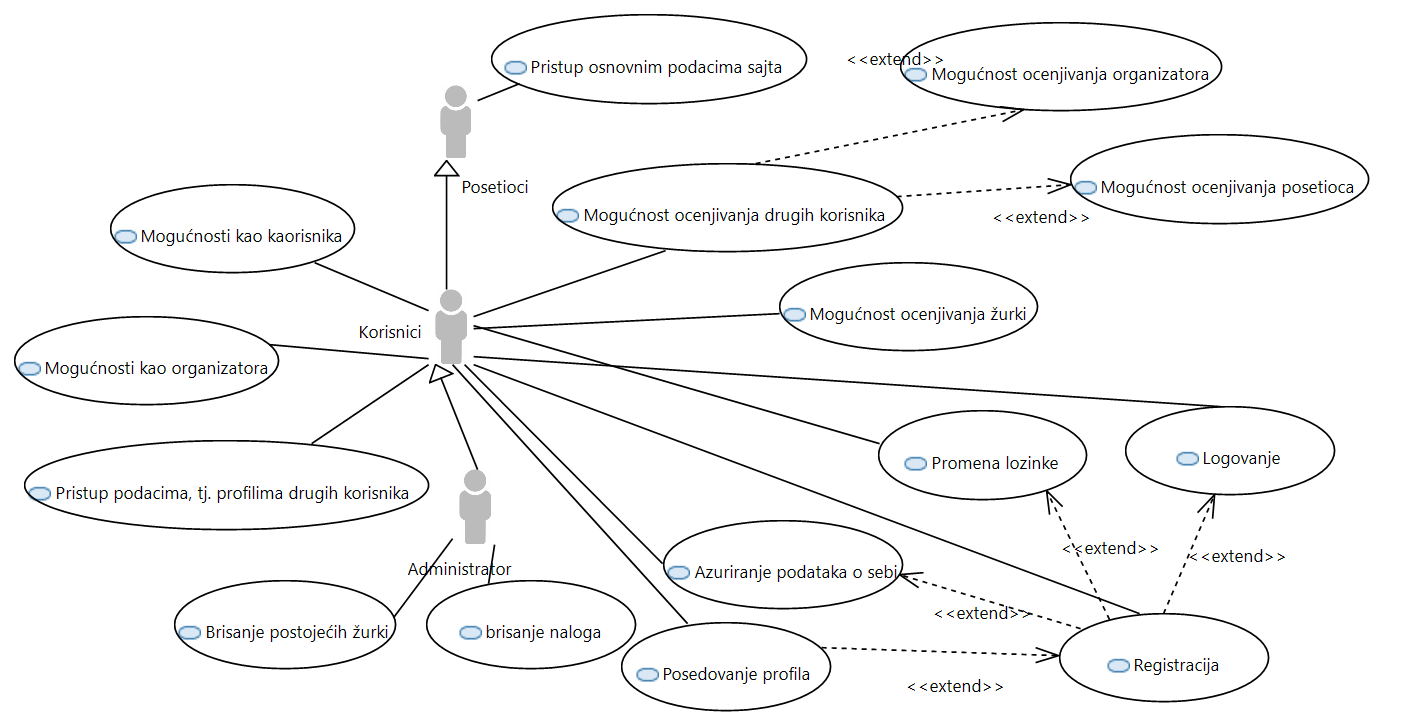
Slučajevi korišćenja web aplikacije KućneŽurke su:

* Pristup osnovnim podacima sajta
* Mogućnost ocenjivanja drugih korisnika
  + Mogućnost ocenjivanja organizatora
  + Mogućnost ocenjivanja posetioca
* Mogućnost ocenjivanja žurki
* Mogućnosti kao organizatora
  + Organizovanje žurki
  + Unos podataka o kućnim žurkama
  + Ažuriranje podataka o kućnim žurkama
  + Odgovor na zahtev za odlazak na žurku
  + Odbiti zahtev
  + Odobriti dolazak na žurku
  + Prikaz liste zahteva
* Mogućnosti kao korisnika
  + Prikaz podataka o kreiranim kućnim žurkama
  + Filtriranje prilikom pregleda
  + Slanje zahteva za prisustvo na žurkama
  + Prikaz liste svih kućnih žurki
* Pristup podacima,tj. profilima drugih korisnika
* Registracija
* Posedovanje profila
* Ažuriranje podataka o sebi
* Promena lozinke
* Logovanje
* Brisanje postojećih žurki
* Brisanje naloga

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

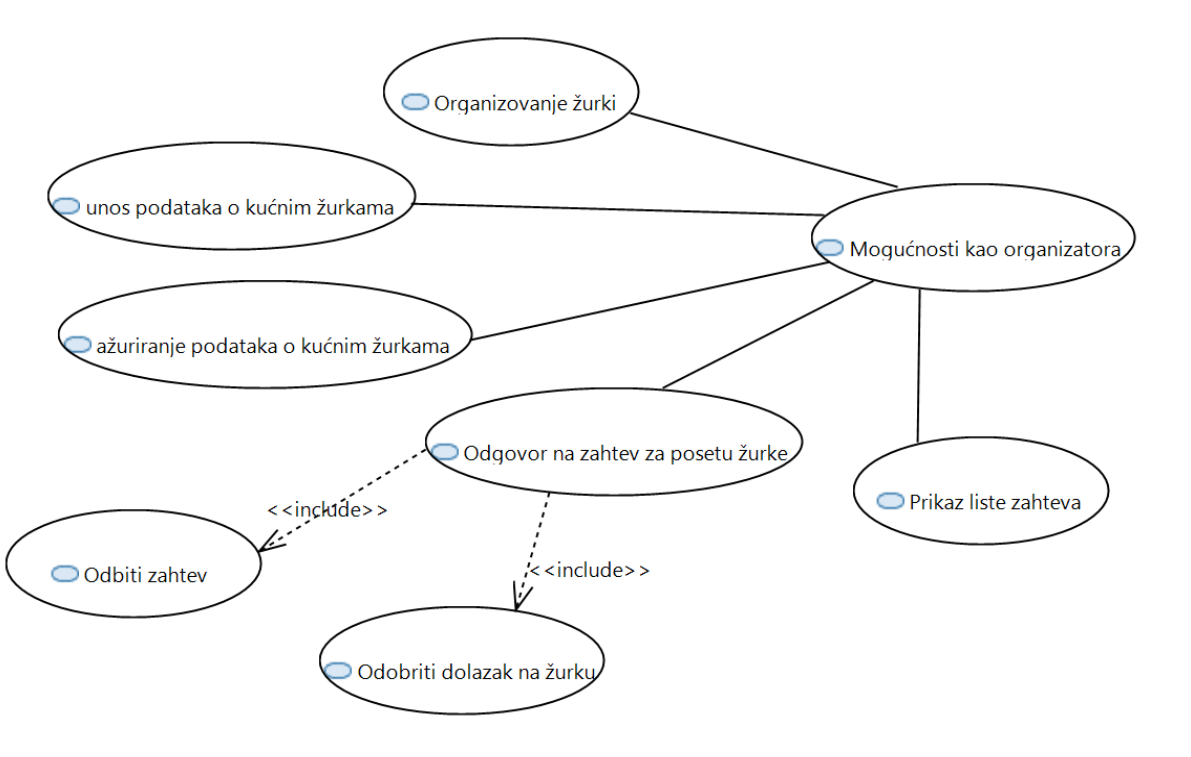
## Dijagrami slučajeva korišćenja

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

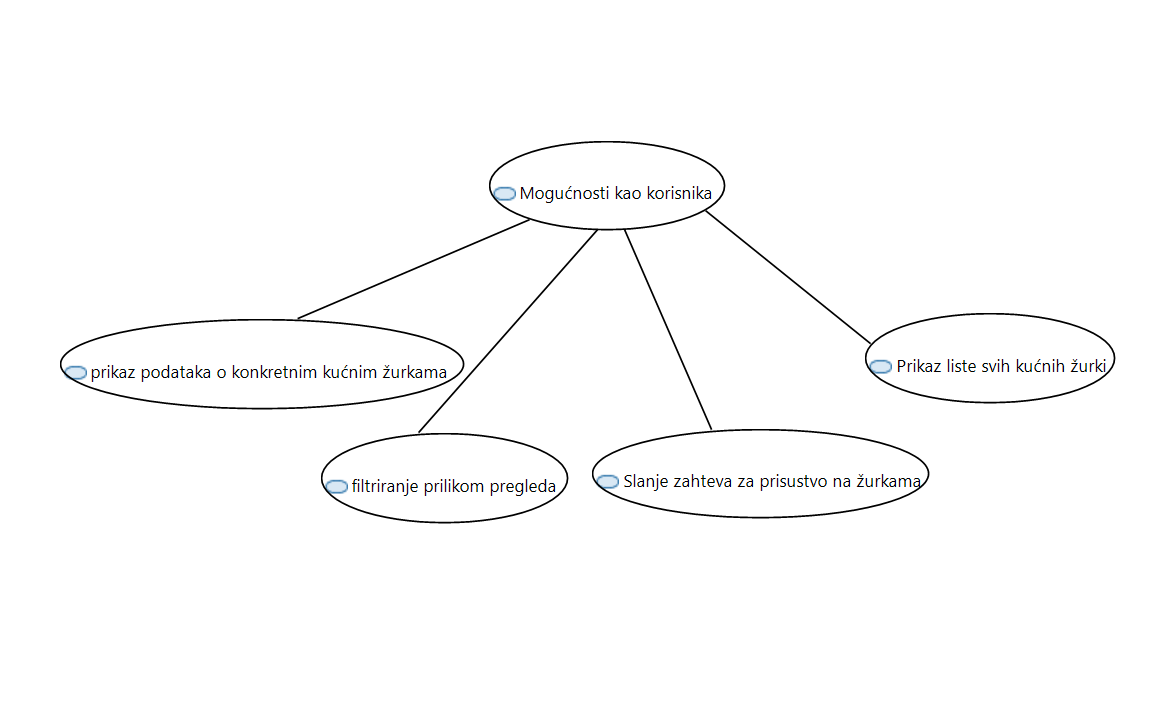


Slučajevi korišćenja  *mogućnosti kao organizatora i mogućnosti kao korisnika* obuhvataju složenije radnje koje se mogu dalje razložiti na pojedinačne slučajeve korišćenja.

Detaljni UML dijagram za slučaj korišćenja *mogućnosti kao organizatora* je prikazan na sledećoj slici:



Detaljni UML dijagram za slučaj korišćenja *mogućnosti kao korisnika* je prikazan na sledećoj slici:



## Kratak opis slučajeva korišćenja

* + 1. Pristup osnovnim podacima sajta

Prikaz početne stranice sajta. Gde se nalaze najosnovnije informacije. Cilj je da se posetiocima, dakle korisnicima koji nisu logovani ne daje mnogo informacija.

* + 1. Mogućnost ocenjivanja drugih korisnika

Ovaj slučaj korišćenja ima razgranatu strukturu, u pogledu ocenjivanja drugih korisnika(posetioci žurke) i ocenjivanja organizatora.

* + 1. Mogućnost ocenjivanja žurki

Posetioci žurke moći će da ocene žurku kojoj su prisustvovali.

* + 1. Posedovanje profila

Nakon registracije korisnika kreira se jedinstveni profil sa nekim od od podataka.

* + 1. Registracija

Registracija je neophodna radi kreiranja naloga.

* + 1. Logovanje

Nakon registrovanja, postoji mogućnost logovanja, korisnik se loguje uz pomoć korisničkog imena i šifre. Logovanjem se otvra nova stranica odnosno profil korisnika..

* + 1. Ažuriranje podataka o sebi

Podatke koje korisnik unese prilikom registrovanja može kasnije izmeniti.

* + 1. Promena lozinke

Lozinka koja je prvobitno postavljena se može promeniti unošenjem nove lozinke i postojeće.

* + 1. Unos podataka o kućnim žurkama

Unosom podataka se kreira žurka.

* + 1. Ažuriranje podataka o kućnim žurkama

Podaci koji se unesu kao podaci vezani za žurku nisu fiksni, mogu da uslede promene.

* + 1. Odgovor na zahtev za posetu žurke

Zahtev koji pristigne može biti odbijen ili odobren od strane organizatora žurke.

* + 1. Prikaz liste zahteva

Zahtevi koji su poslati od strane korisnika moraju biti prikazani radi lakšeg preglada od strane organizatora žurke.

* + 1. Prikaz podataka o konkretnim kućnim žurkama

Podaci o žurkama mogu biti prikazani.

* + 1. Filtriranje prilikom pregleda

Prikaz podataka o žurkama.

* + 1. Slanje zahteva za pristustvo na žurkama

Za sve kreirane žurke postoji opcija slanja zahteva.

* + 1. Brisanje postojećih žurki

Iako su organizatori kreirali žurku, administrator je taj koji može da kreiranu žurku iz određenih razloga ukloni.

* + 1. Brisanje naloga

Administrator ima tu privilegiju da određene naloge ukloni.

# 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.

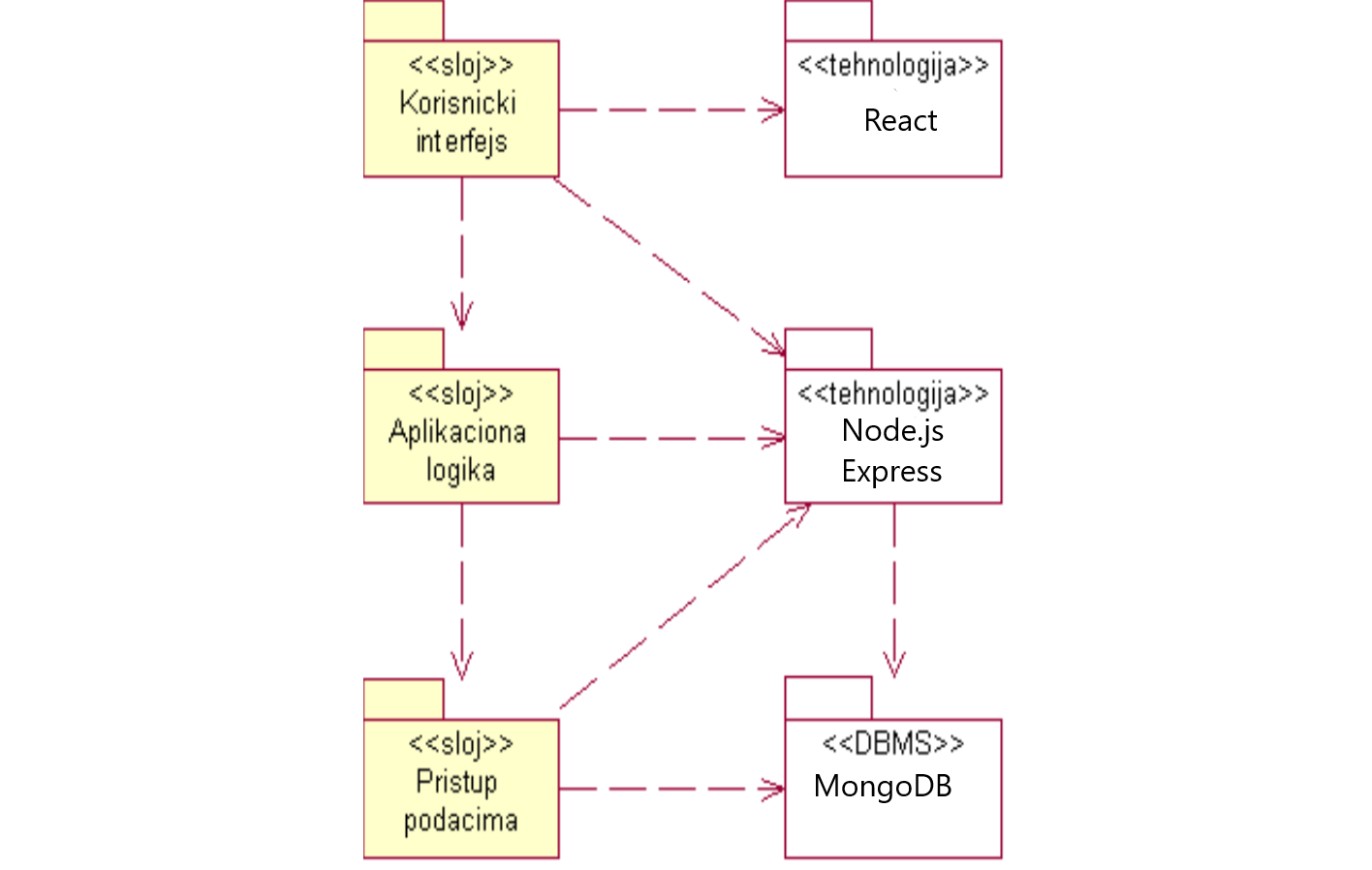
Logički pogled na KućneŽurke portal obuhvata 3 glavna paketa: Korisnički interfejs, Aplikaciona logika, Pristup podacima.

Paket *Korisnički interfejs* sadrži Web stranice HTML/CSS , Bootstrap, JavaScript za koji je zadužen React.

Paket *Aplikaciona logika* predstavlja srednji sloj sistema koji sadrži JavaScript komponente zadužene za realizaciju funkcionalnosti specifičnih za domen sistema koji se razvija.

Paket *Pristup podacima* sadrži JavaScript komponente koje predstavljaju interfejs za pristup , dodavanje i ažuriranje podataka koji se čuvaju u bazi podataka.Podaci se prenose i cuvaju u MongoDB kao json fajlovi.

## Pregled arhitekture – organizacija paketa i podsistema u slojeve



### Korisnički interfejs

sloj

Ovaj sloj realizuje korisnički interfejs portala. U njemu su sadržane stranice preko kojih korisnici komuniciraju sa sistemom. Stranice su realizovane uz pomoć HTML/CSS-a,Bootstrap-a, JavaScript-a.

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

### Aplikaciona logika

sloj

Sloj aplikacione logike je srednji sloj u troslojnoj arhitekturi web aplikacije KućneŽurke. Sadrži komponente 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.

### Pristup podacima

Sloj

Sloj za pristup podacima se nalazi na dnu troslojne arhitekture sadrzi .js fajlove zadužene za pribavljanjne , dodavanje , ažuriranje podataka koji se čuvaju u MongDB bazi podataka.

### React

tehnologija

React (poznat i као React.js ili ReactJS) је JavaScript biblioteka otvorenog koda koja оbezbeđuје pregled podataka zapisanih preko HTML-a. React koristi komponente , koje su definisane kao HTML stranice,kako bi obezbedila izgled ekrana, odnosno izgled same aplikacije.

### Node.js

tehnologija

Node.js je višeplatformsko Script okruženje otvorenog koda za izvršavanje JavaScripta na serverskoj strani.

### MongoDB

DBMS

MongoDB predstavlja sistem za upravljanje bazama podataka koji će se koristiti za realizaciju KućneŽurke

web aplikacije.

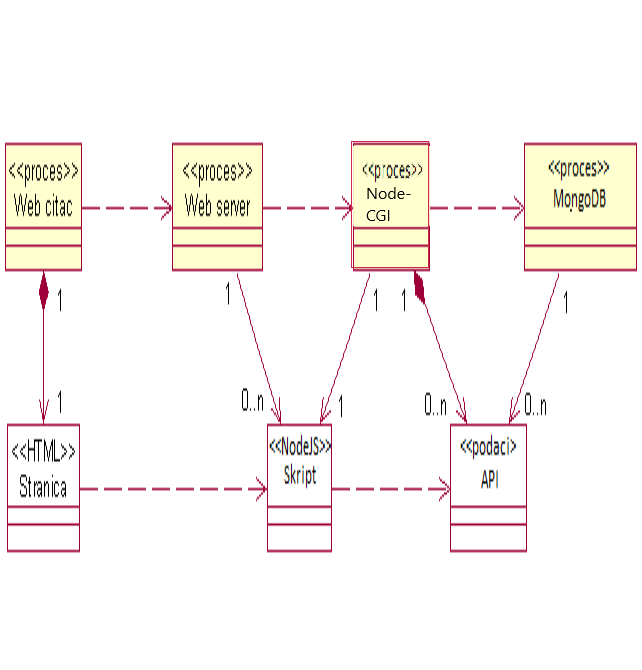
# Pogled na procese

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.

Opis procesa uključenih u izvršenje KućneŽurke Web aplikacije.

## Procesi

Na sledećem UML dijagramu klasa prikazani su procesi koji učestvuju u izvršenju Web aplikacije KućneŽurke.



### 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 NodeJS skript, Web server inicira izvršenje Node-CGI procesa koji obrađuje odgovarajući skript i generiše sadržaj koji se vraća čitaču. Web server može paralelno da inicira veći broj Node-CGI procesa.

### Node-CGI

Node-CGI proces obavlja posao obrade zadatog Node JS skripta i generiše odgovarajući tekstualni sadržaj

koji Web server šalje Web čitaču. Za izvršenje Node JS skripta ovaj proces može da zahteva usluge

MongoDB servera-a. Komunikacija između Node-CGI procesa i MongoDB servera se obavlja preko

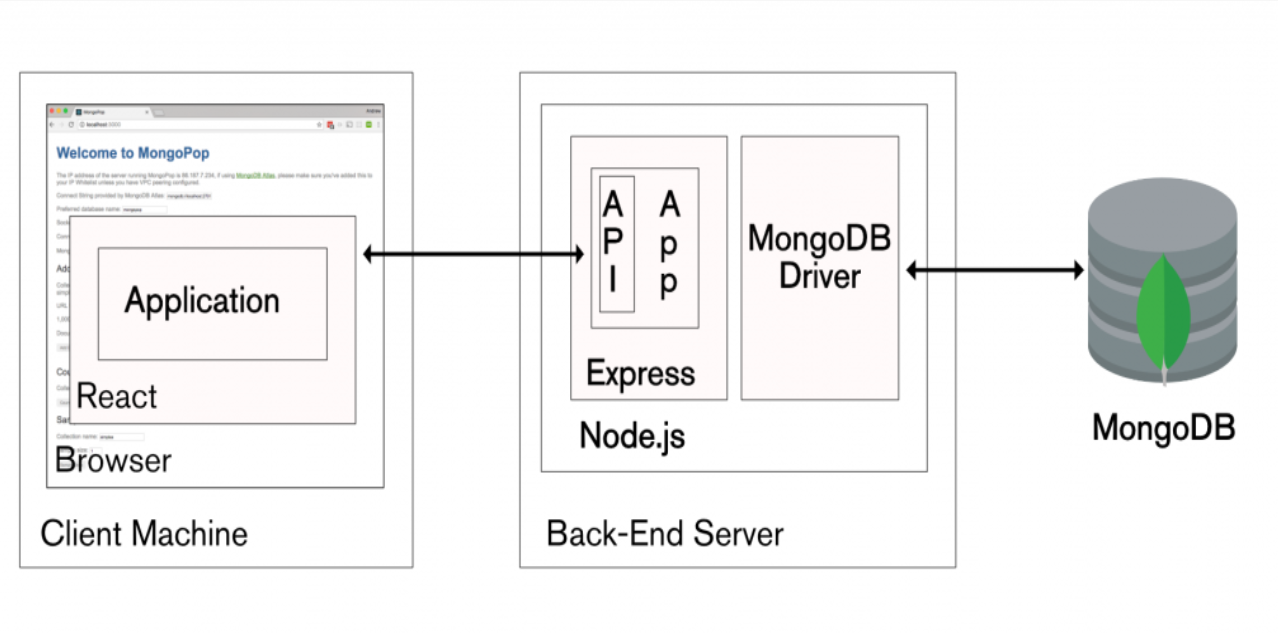
API-ja.

### *MongoDB Server*

MongoDB Server je proces koji izvršava funkcionalnost MongoDB sistema za upravljanje bazama podataka

# 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.



## Klijent

Pristup KućneŽurke 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 Node-CGI koji vrše obradu NodeJS .U najopštioj konfiguraciji DBMS se izvršava na posebnoj mašini koja je sa Web serverom u lokalnoj mreži (LAN).

## DBMS server

DBMS 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 iz lokalne mreže (LAN).

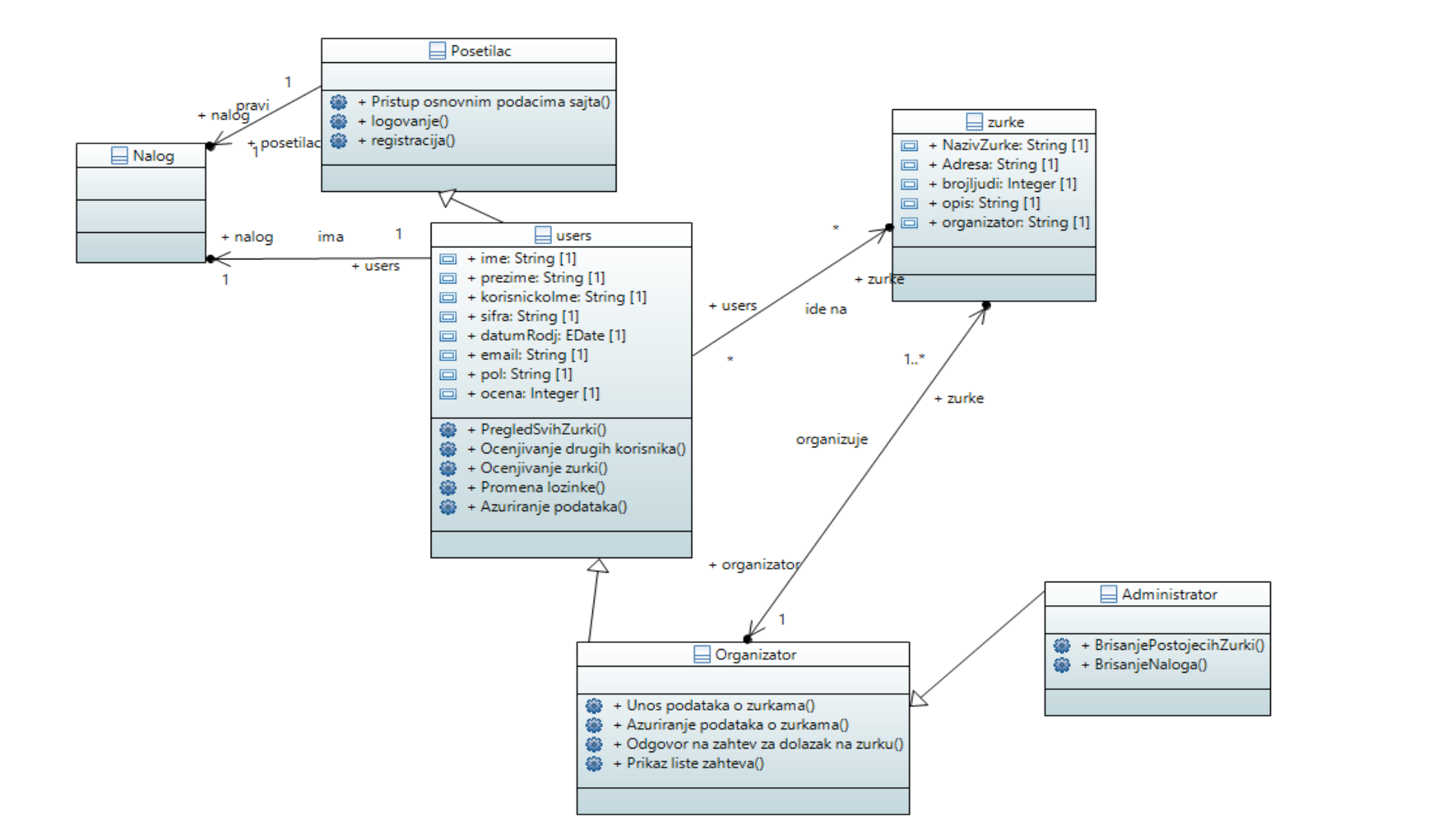
# Pogled na implementaciju sistema

Pogled na implementaciju prikazuje različite aspekte bitne za implementaciju sistema. U slučaju KućneŽurke 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 KućneŽurke Web aplikacija projektuje je ilustrovan UML dijagramom klasa. U

su prikazane domenske klase, neki od njihovih atributa, kao i veze koje se mogu identifikovati između njih.



## Komponente sistema

## KućneŽurke sadrže JavaScript fajlove i .json fajlove. Za ilustraciju će biti korišćeni UML dijagrami komponenti i dijagrami klasa. Komponente aplikacije KućneŽurke koje su zadužene za backend i frontend aplikacije su izdvojene u zasebnim fajlovima, radi lakšeg pregleda i pristupa prilikom kreiranja aplikacije. U okviru foldera components, koji se nalazi u folderu src, nalaze se .js fajlovi zaduženi za korisnički interfejs aplikacije. Backend aplikacije nalazi se u okviru foldera backend.

## Razlikujemo dve vrste fajlova, u okviru components foldera, sa ekstenzijom .js. Postoje oni ( koji započinju sa – import axios from “axios“;) koji sadže implementaciju funkcija koje se koriste u drugom tipu JavaScript fajlova.Taj drugi tip .js fajlova predstavlja klase koje su zadužene za korisnički interfejs aplikacije.

Startni JavaScript fajl je index.js koji renderuje App.js fajl. Glavna mogućnost, odnosno prednost

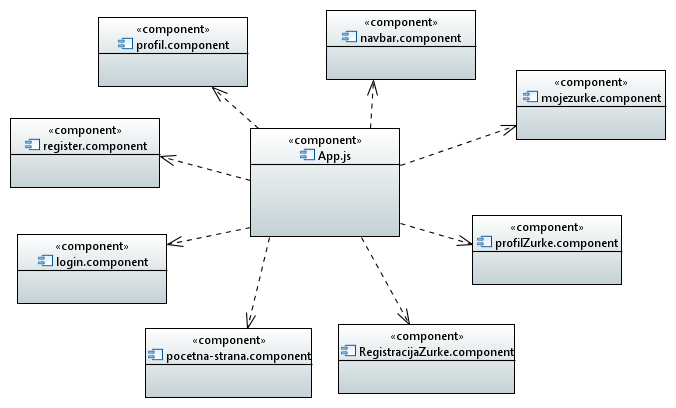
react aplikacija je način za rutiranje komponenti , koji je omogućen instalacijom i primenom

react-rouer-dom-a , u sklopu App.js fajla vrši se importovanje komponenti i kreiranje ruta odgo

varajućih komponenti.

U sledećem dijagramu komponenti prikazane su neke od komponenti kloje App.js komponenta

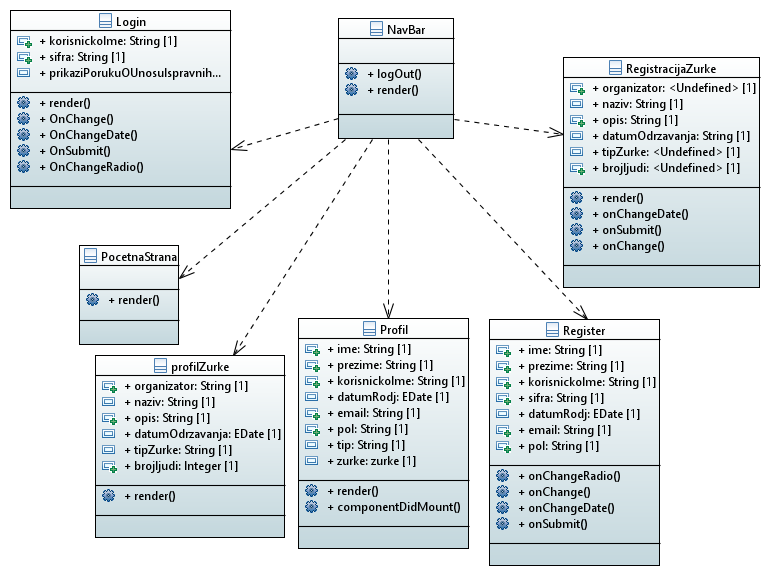
sadrži.



### Komponente korisničkog interfejsa

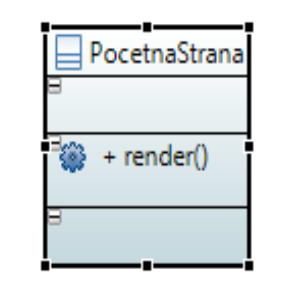
Neke od komponenti korisničkog interfejsa aplikacije , koje se konretno odnose na logovanje i

registraciju date su u sledećem klasnom dijagramu:

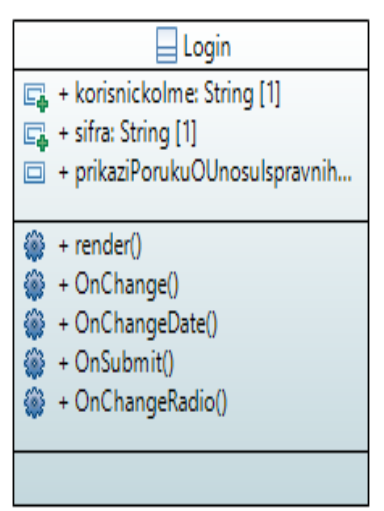


NavBar klasa obezbeđuje nam lak pregled ostalih komponenti.

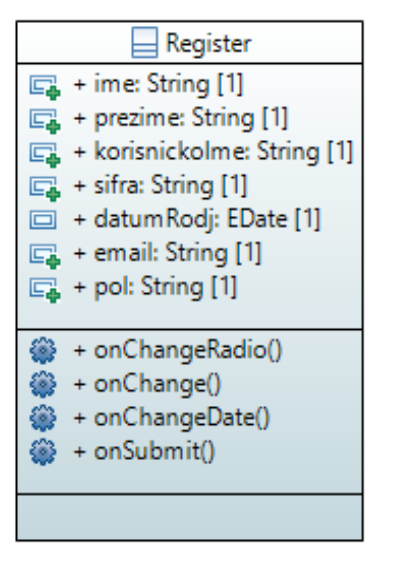
Parametri koji utiču na izgled stranice koja se prva otvara prilikom pokretanja aplikacije se nalaze u klasi PocetnaStrana, koja je prikazana klasnim dijagramom:



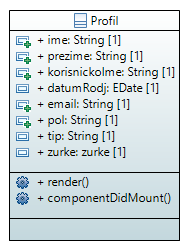
- render funkcija je ona koja vraća izgled stranice , odnosno vraća html kod, što je slučaj i kod ostalih komponenti.

Klasni dijagram komponente Login:

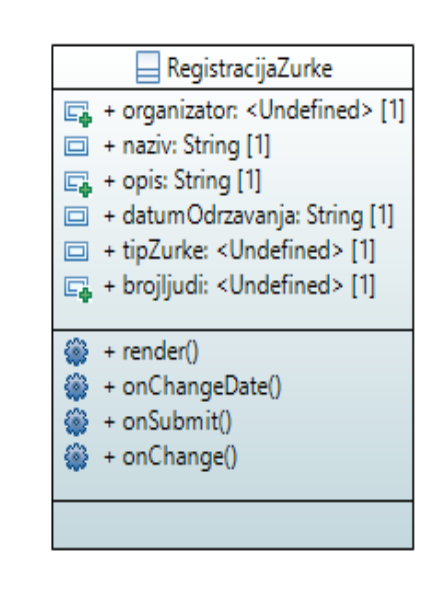
Klasni dijagram komponente Register:



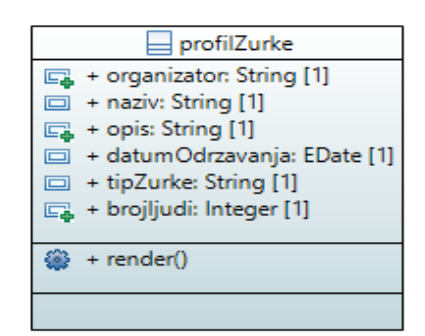
Dijagram klase komponente Profil:



Dijagram klase komponente RegistrcijaZurke:



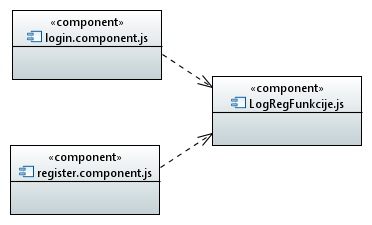
Dijagram klase komponente ProfilZurke:



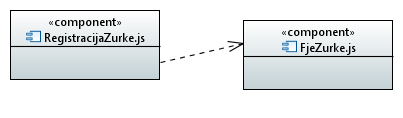
### Komponente aplikacione logike

Paket *Aplikaciona logika* predstavlja srednji sloj sistema koji sadrži JavaScript komponente zadužene za realizaciju funkcionalnosti specifičnih za domen sistema koji se razvija, ovo su samo neke od komponenti aplikacione logike neophodne za realizaciju aplikacije „KućneŽurke“.

Komponenta LogRegFunkcije.js sadrži funkciju „register“ koja se koristi u register.component.js fajlu i funkciju „login“ koja se koristi u login.component.js fajlu.



Komponenta FjeZurke.js sadrži funkciju „registracijaZurke“ koja se koristi u RegistracijaZurke.component.js fajlu.



### Komponente za pristup podacima

U okviru server.js fajla koji se nalazi u backend folderu, vrši se povezivanje sa MongoDB, odnosno prvo se kreira konstanta konekcionog stringa koji se dobija od strane MongoDB, prilikom kreiranja baze. Nakon toga se obavi konekcija.

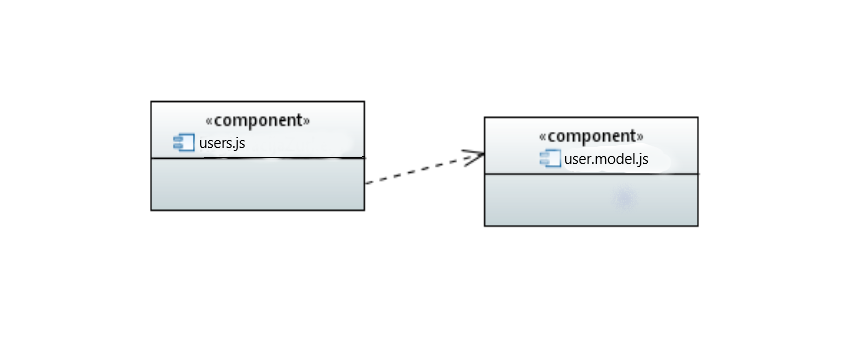
Pristup bazi podataka je u potpunosti zatvoren u funkcije koje su definisane u okviru zasebnih .js fajlova u okviru foldera routes. Same tabele kreiraju se u okviru foldera modeles, u zasebnim fajlovima, u slučaju user-a kreiranje je u fajlu user.model.js. U user.model.js fajlu kreira se Schema, koja je ekvivalent Tabeli u reliacionim bazama podataka.

Primer kako se koriste mogućnosti , odnosno neke od funkcija baze, u slučaju users.js.

Kreira se konstanta „User“ zahvaljujući user.model fajlu, a zatm u okviru post i get metde se pozivaju funkcije koje se odnose na bazu, poput funkcije create(),pomoću koje kreiramo novu stavku tabele, odnosno .json fajl.

### 

Dijagram kompeonenti koji prikazuje zavisnost users.js .



# 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. KućneŽurke 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.