RaketaDoLeta

Arhitekturni projekat

Verzija 1.1

Pregled izmena

|  |  |  |  |
| --- | --- | --- | --- |
| **Datum** | **Verzija** | **Opis** | **Autor** |
| 29.04.2022. | 1.0 | Inicijalna verzija | Nada Jovanović  Milica Jovanović  Nastasija Stanković  Katarina Stanojković |
| 14.05.2022. | 1.1 | Finalna verzija | Nada Jovanović  Milica Jovanović  Nastasija Stanković  Katarina Stanojković |
|  |  |  |  |
|  |  |  |  |

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 6

6.2 Kratak opis slučajeva korišćenja 8

6.2.1 Izbor jezika 8

6.2.2 Prijavljivanje 8

6.2.3 Pregled osnovnih podataka o fitnes centru 8

6.2.4 Registrovanje 8

6.2.5 Ažuriranje lozinke 8

6.2.6 Validacija korisnika 8

6.2.7 Dodavanje profila trenera 8

6.2.8 Brisanje profila 8

6.2.9 Dodavanje ili ažuriranje postojećih usluga 8

6.2.10 Uvid u zakazane termine 8

6.2.11 Upravljanje korisničkih zahteva 9

6.2.12 Uvid u kalendar slobodnih termina 9

6.2.13 Online zakazivanje personalnih treninga 9

6.2.14 Uvid u informacije o zakazanim usluama 9

6.2.15 Prikaz svih informacija o nalogu 9

6.2.16 Uvid u napredak korisnika 9

7. Pogled na logičku arhitekturu sistema 10

7.1 Pregled arhitekture – organizacija paketa i podsistema u slojeve 10

7.1.1 Korisnički interfejs 10

7.1.2 Aplikaciona logika 11

7.1.3 Pristup podacima 11

7.1.4 HTML 11

7.1.5 React.js 11

7.1.6 Node.js 11

7.1.7 Mongodb 11

8. Pogled na procese 12

8.1 Procesi 12

8.1.1 Web čitač 12

8.1.2 Web server 12

8.1.3 Mongo Server 12

9. Pogled na raspoređivanje sistema 13

9.1 Klijent 13

9.2 Web server 13

9.3 DBMS server 13

10. Pogled na implementaciju sistema 14

10.1 Model domena 14

10.2 Šema baze podataka 15

10.3 Komponente sistema 15

10.3.1 Komponente korisničkog interfejsa 15

10.3.2 Komponente aplikacione logike 19

10.3.3 Komponente za pristup podacima 19

11. Performanse 20

12. Kvalitet 20

Arhitekturni projekat

# Cilj dokumenta

Cilj ovog dokumenta je detaljni opis arhitekture RaketaDoLeta portala.

# Opseg dokumenta

Dokument se odnosi na RaketaDoLeta portal koji će biti razvijen od strane RDLTeam-a RaketaDoLeta predstavlja često korišćenu frazu od strane vežbača koji se spremaju za letnju sezonu. Namena sistema je efikasno prezentovanje rada fitnes centra, online zakazivanja usluga i lakša komunikacija između korisnika, trenera i uprave u fitnes centru.

# Reference

Spisak korišćene literature:

1. RaketaDoLeta – Predlog projekta, RDL-RaketaDoLeta-01, V1.0, 2022, RDLTeam.
2. RaketaDoLeta – Planirani raspored aktivnosti na projektu, V1.0, 2022, RDLTeam.
3. RaketaDoLeta – Plan realizacije projekta, V1.0, 2022, RDLTeam.
4. RaketaDoLeta – Vizija sistema, V1.1, 2022, RDLTeam
5. RaketaDoLeta – Specifikacija zahteva, V1.0, 2022, RDLTeam.

# 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. RaketaDoLeta portal će biti implementiran kao Web aplikacija zasnovana na React.js, Node.js i MongoDB bazi podataka [4].
2. Klijentski deo RaketaDoLeta portala će biti optimizovan za sledeće Web čitače: Internet Explorer 11 i noviji, Opera 43 i noviji, Google Chrome 57 i noviji, kao i Firefox (Mozilla) [4].
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

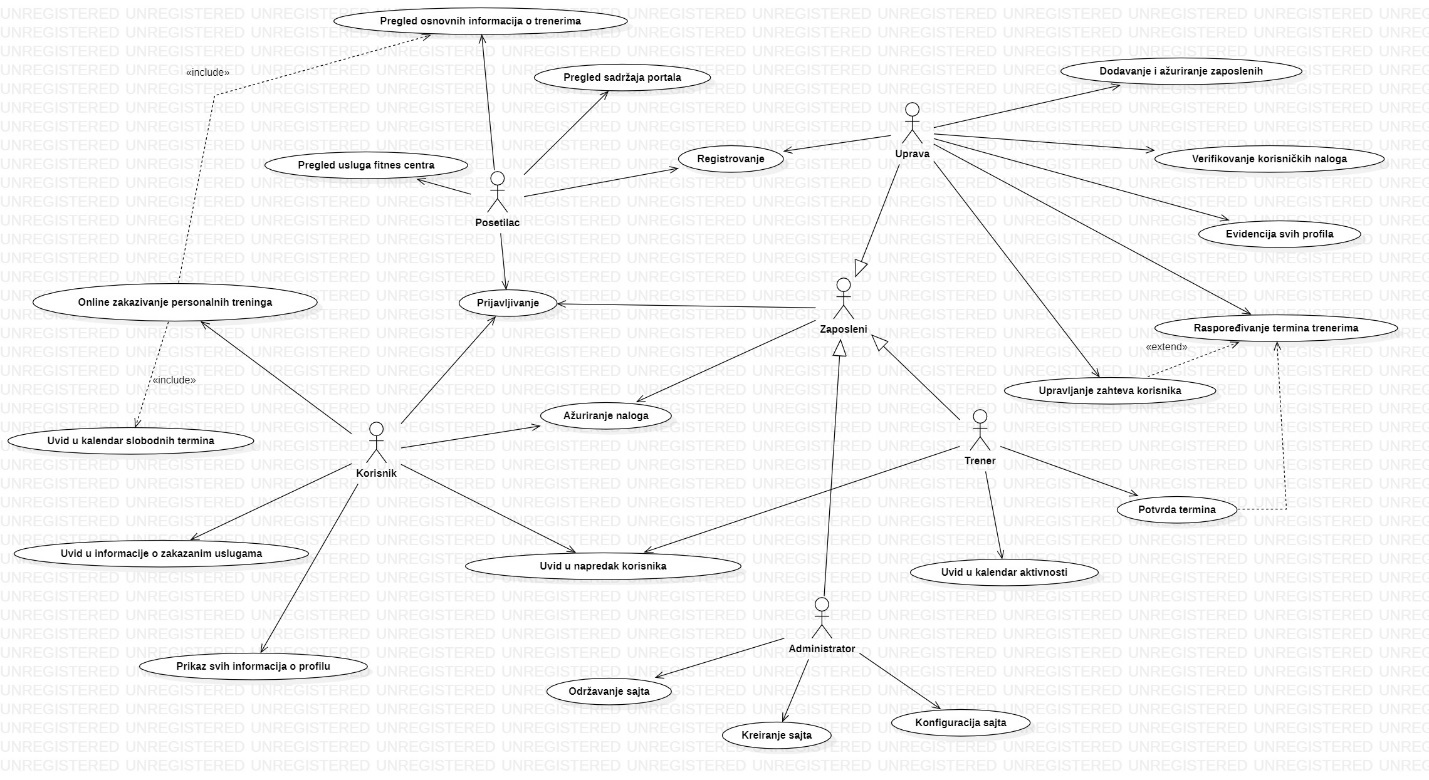
U ovom odeljku je dat pogled na slučajeve korišćenja definisane u specifikaciji zahteva [5].

Slučajevi korišćenja RaketaDoLeta portala su:

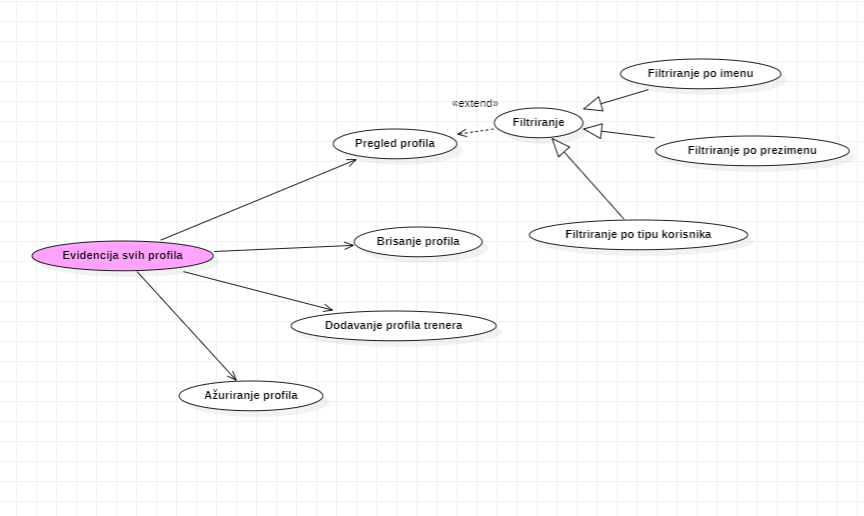
* Izbor jezika
* Prijavljivanje
* Registrovanje
* Ažuriranje lozinke
* Pregled osnovnih podataka o fitnes centru
* *Koriscničke funkcionalnosti* 
  + Online zakazivanje personalnih treninga
  + Uvid u informacije o zakazanim uslugama
  + Uvid u kalendar slobodnih tremina
  + Prikaz svih informacija o nalogu
  + Uvid u napredak korisnika
* *Funkcinalnosti uprave*
  + Validacija korisnika
  + Dodavanje profila trenera
  + Brisanje profila
  + Dodavanje ili ažuriranje postojećih usluga
  + Upravljanje korisničkim zahtevima
* *Funkcionalnosti trenera*
  + Uvid u zakazane termine
  + Upravljanje korisničkih zahteva
  + Uvid u napredak korisnika

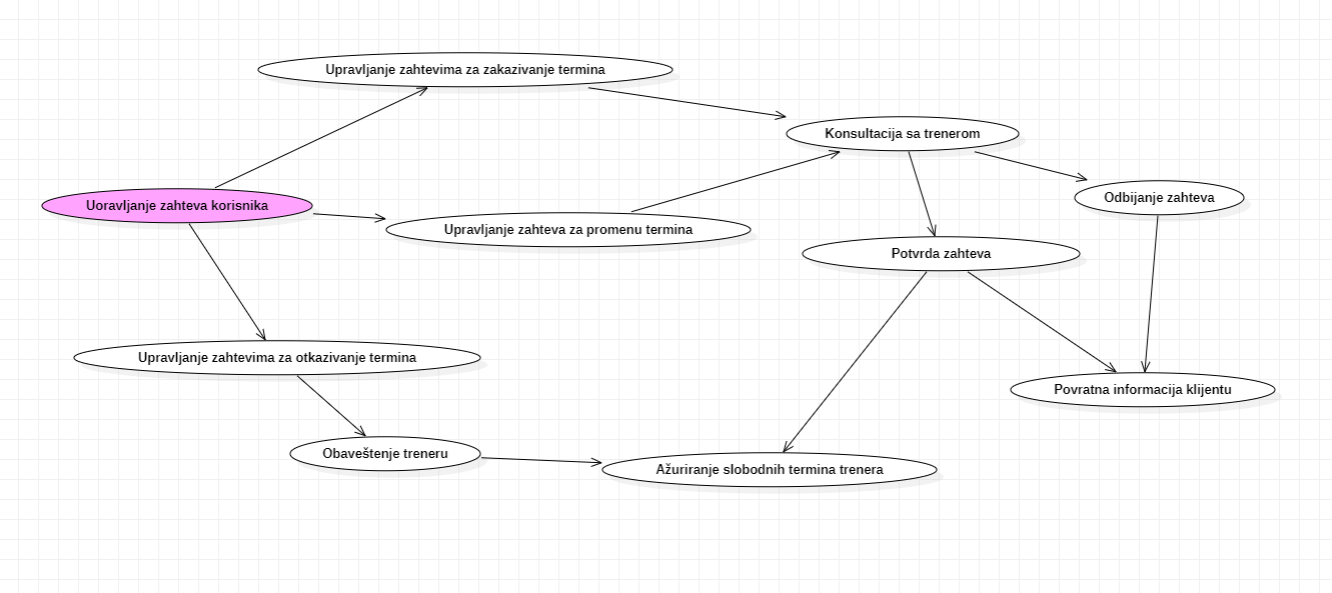
Ovi slučajevi korišćenja mogu da iniciraju posetilac portala, korisnik, trener, uprava ili administrator.

## Dijagrami slučajeva korišćenja

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

Slučajevi korišćenja *evidencija svih profila i upravljanje zahteva korisnika* obuhvataju složenije radnje koje se mogu razložiti dalje razložiti na pojedinačne slučajeve korišćenja.

Detaljni UML dijagram za slučaj korišćenja *evidencija svih profila* je prikazan na sledećoj slici:

Detaljni UML dijagram za slučaj korišćenja *upravljanje zahteva korisnika* je prikazan na sledećoj slici:

## Kratak opis slučajeva korišćenja

### Izbor jezika

Kratak opis: Izbor između srpskog i engleskog jezika za prikaz informacija na portalu.

Akteri koji iniciraju slučaj korišćenja: Posetilac portala, Korisnik, Trener, Uprava, Administrator.

### Prijavljivanje

Kratak opis: Logovanje na portal koristeći mejl ili lozinku.

Akteri koji iniciraju slučaj korišćenja: Korisnik, Trener, Uprava, Administrator.

### Pregled osnovnih podataka o fitnes centru

Kratak opis: Prikaz stranice portala sa osnovnim podacima o fitnes centru.

Akteri koji iniciraju slučaj korišćenja: Posetilac portala, Korisnik, Trener, Uprava, Administrator.

### Registrovanje

Kratak opis: Prikaz stranice portala sa formom za unos osnovnih ličnih podataka i mogućnost kreiranja koriničkog naloga koji nudi dodatne servise.

Akteri koji iniciraju slučaj korišćenja: Posetilac portala.

### Ažuriranje lozinke

Kratak opis: Ažuriranje korisničkog naloga.

Akteri koji iniciraju slučaj korišćenja: Korisnik.

### Validacija korisnika

Kratak opis: Validacija naloga za novog korisnika.

Akteri koji iniciraju slučaj korišćenja: Uprava.

### Dodavanje profila trenera

Kratak opis: Kreiranje korisničkog naloga za novog trenera.

Akteri koji iniciraju slučaj korišćenja: Uprava.

### Brisanje profila

Kratak opis: Brisanje korisničkog naloga.

Akteri koji iniciraju slučaj korišćenja: Uprava.

### Dodavanje ili ažuriranje postojećih usluga

Kratak opis: Uprava ima mogućnost dodavanja ili ažuriranja postojećih usluga fitnes centra.

Akteri koji iniciraju slučaj korišćenja: Uprava.

### Uvid u zakazane termine

Kratak opis: Trener ima uvid u angažovanja koja su mu dodeljena od Uprave fitnes centra.

Akteri koji iniciraju slučaj korišćenja: Trener.

### Upravljanje korisničkih zahteva

Kratak opis: Uprava ima mogućnost prihvatanja ili odbijanja korisničkih zahteva.

Akteri koji iniciraju slučaj korišćenja: Uprava, Trener.

### Uvid u kalendar slobodnih termina

Kratak opis: Korisnik ima uvid u kalendar slobodnih termina izabranog trenera.

Akteri koji iniciraju slučaj korišćenja: Korisnik.

### Online zakazivanje personalnih treninga

Kratak opis: Prijavljeni korisnik ima mogućnost slanja zahteva upravi za zakazivanje personalnog treninga uz prethodni uvid u slobodne termine izabranog trenera.

Akteri koji iniciraju slučaj korišćenja: Korisnik.

### Uvid u informacije o zakazanim usluama

Kratak opis: Prikaz stranica sa listom zakazanih usluga.

Akteri koji iniciraju slučaj korišćenja: Korisnik.

### Prikaz svih informacija o nalogu

Kratak opis: Prikaz stranice sa informacijama o određenom nalogu.

Akteri koji iniciraju slučaj korišćenja: Korisnik.

### Uvid u napredak korisnika

Kratak opis: Trener i korisnik imaju uvid u celokupni napredak korisnka.

Akteri koji iniciraju slučaj korišćenja: Trener, 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.

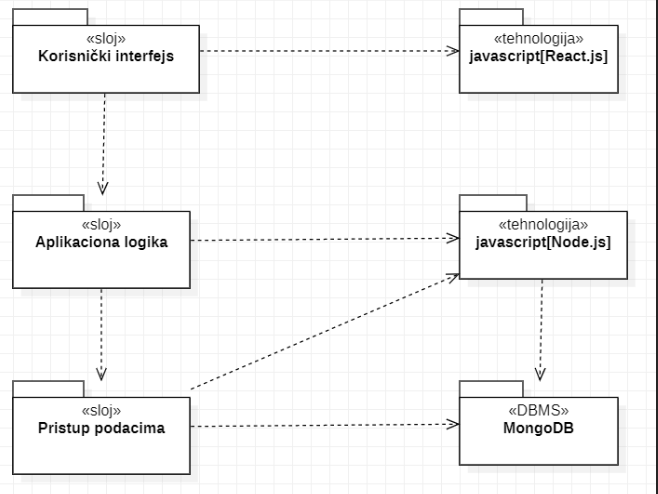
Logički pogled na RaketaDoLeta portal obuhvata 3 glavna paketa: Korisnički interfejs, Aplikaciona logika, Pristup podacima.

Paket *Korisnički interfejs* sadrži Web stranice, React.js 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 JS skripte zadužene za realizaciju funkcionalnosti specifičnih za domen sistema koji se razvija.

Paket *Pristup podacima* sadrži 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 sadržane sve HTML, multimedijalni sadržaji, CSS i React.js koje generišu HTML stranice preko kojih korisnici komuniciraju sa sistemom.

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

### Aplikaciona logika

sloj

Sloj aplikacione logike je srednji sloj u troslojnoj arhitekturi RaketaDoLeta portala. Sadrži JS 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 JS skripte 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 .NET Core i MongoDB baze podataka.

### 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.

### React.js

tehnologija

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

### Node.js

tehnologija

Tehnologija Node.js se izvršava na serverskoj strani, obezbeđuje mehanizam za pisanje i izvršavanje skripti na strani servera. Ove skripte mogu da pristupaju bazi podataka u cilju pribavljanja, unosa i ažuriranja podataka.

### Mongodb

DBMS

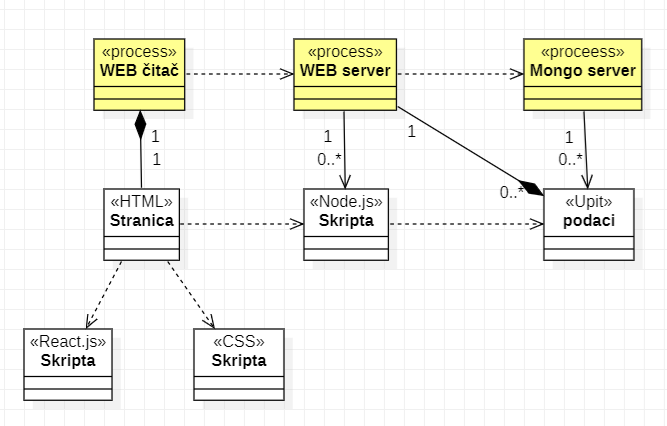
Mongodb predstavlja sistem za upravljanje bazama podataka koji će se koristiti za realizaciju RaketaDoLeta portala.

# 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.

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

## Procesi

Na sledećem UML dijagramu klasa prikazani su procesi koji učestvuju u izvršenju RaketaDoLeta portala. Dijagram je opšteg tipa i može se primeniti na bilo koju Web aplikaciju zasnovanu na React.js, Node.js-u i 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 pisana u JavaScriptu-u, Web server inicira izvršenje procesa koji obrađuje odgovarajući skript i generiše sadržaj koji se vraća čitaču. Web server obavlja posao obrade zadate skripte i generiše odgovarajući sadržaj koji šalje Web čitaču. Za izvršenje skripte ovaj proces može da zahteva usluge Mongo servera-a.

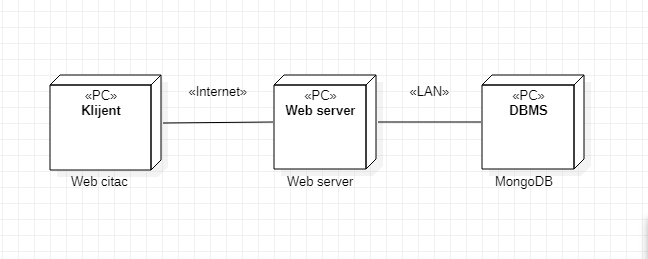
### Mongo Server

Mongo Server je proces koji izvršava funkcionalnost Mongo 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 RaketaDoLeta portala.



## Klijent

Pristup RaketaDoLeta portalu 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. U najopštijoj 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 Mongo 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).

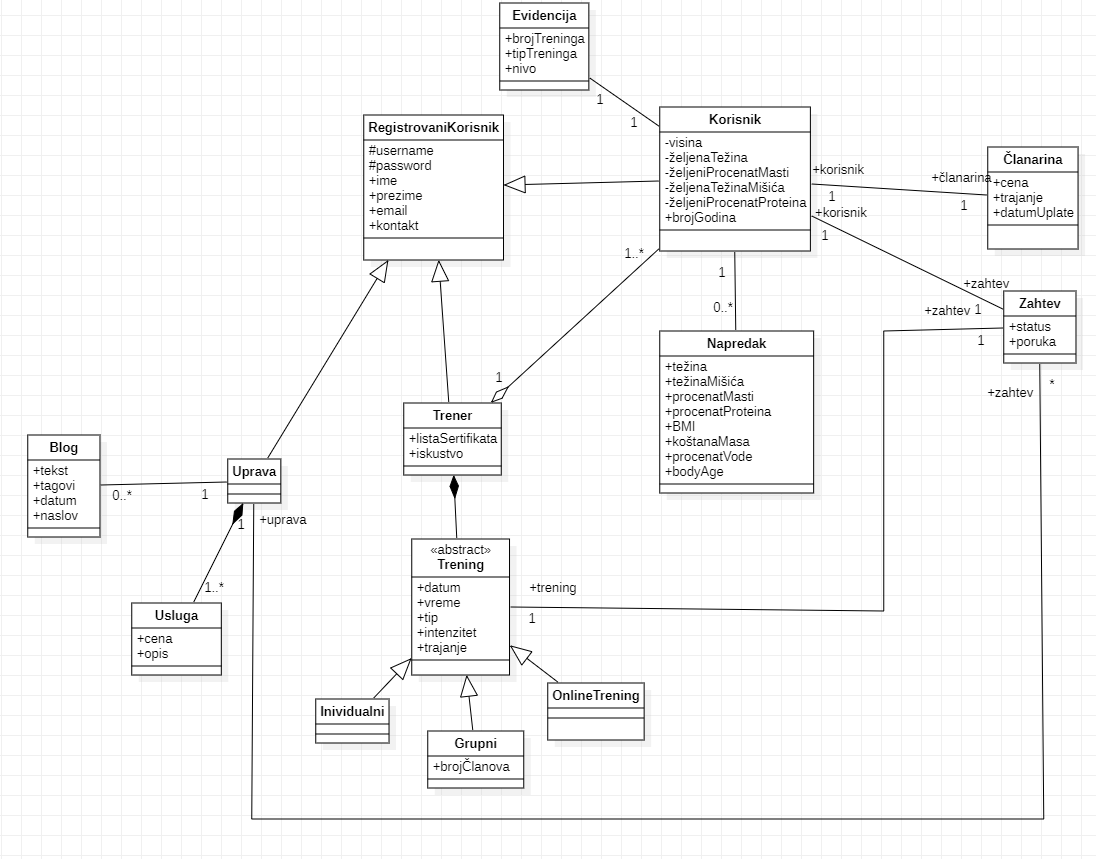
# Pogled na implementaciju sistema

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

## Model domena

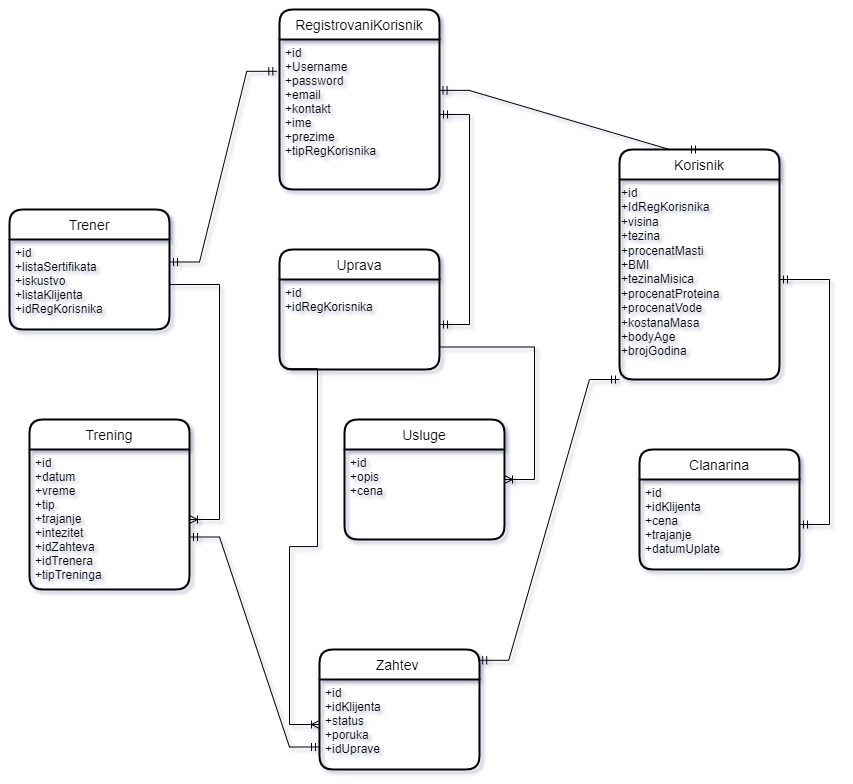
Model domena za koji se RaketaDoLeta portal projektuje je ilustrovan UML dijagramom klasa. U njemu su prikazane domenske klase, neki od njihovih atributa, kao i veze koje se mogu identifikovati između njih.

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



## Šema baze podataka

Obzirom da ce aplikacija koristiti MongoDB kao bazu podataka koja ima fleksibilnu šemu, u nastavku će biti predstavljena okvirna šema baze podataka. Dijagram je kreiran korišćenjem alata draw.io.

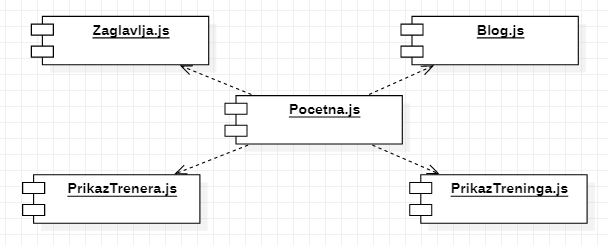


## Komponente sistema

Komponente sistema RaketaDoLeta portala su JS 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 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 za *posetioca* RaketaDoLeta portala je obuhvaćen tipom komponenti *.js* skripte.



Komponenta **Pocetna.js** implementira početnu stranu RaketaDoLeta portala sa koje korisnik ima pristup multimedijalnom sadržaju stranice i osnovnim podacima fitnes centra, sa mogućnošću logovanja na sajt.

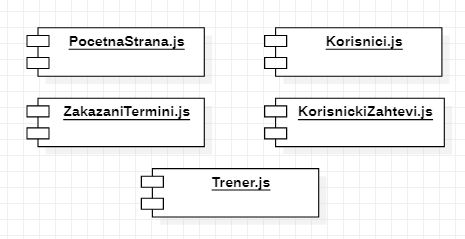
Komponenta **Zaglavlja.js** implementira navigacioni panel sa opcijom log-ovanja i registrovanja na sajt i footer bar koji sadrži osnovne informacjie o fitnes centru.

Komponenta **PrikazTrenera.js** prikazuje stranicu sa listom svih trenera koji rade u fitnes centru.

Komponenta **PrikazTreninga.js** prikazuje trenutnu ponudu treninga koji postoje u fitnes centru.

Komponenta **Blog.js** prikazuje stranicu sa korisnim tesktovima o fitnesu, ishrani...

Dizajn korisničkog interfejsa *Trener* je obuhvaćen tipom komponenti *.js* skripte.



Komponente forme:

* **KorisnickiZahtevi.js** omogućava odobravanje zahtevanog treninga od strane trenera
* **ZakazaniTremini.js** prikazuje obaveze sortirane po vremenu i datumu.

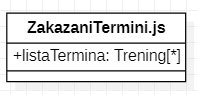
Komponente zadužene za renderovanje:

* **Korisnici.js** prikazuje listu svih klijenta jednog trenera.

Komponente kontejneri:

* **Trener.js** je glavna komponenta odnosno kontejner komponenta za sve funkcionalnosti dostupne treneru.

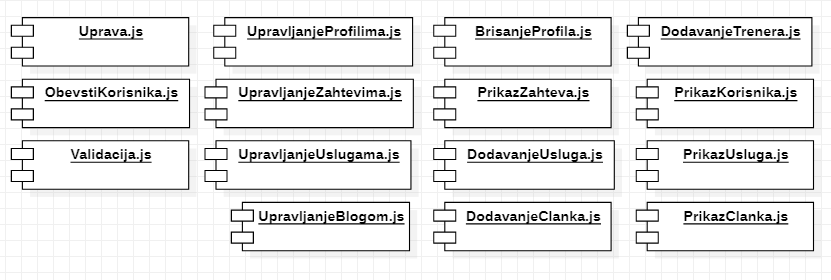
Parametri koji utiču na sadržaj komponenata koje se prikazuju su ilustrovani sledećim dijagramom klasa.



Značenje atributa je sledeće:

* listaTermina– lista svih termina treninga trenera.

Dizajn korisničkog interfejsa *Uprava* je obuhvaćen tipom komponenti *.js* skripti.



Komponente forme:

* **DodavanjeTrenera.js** komponenta koja služi za dodavanje novog trenera u fitnes centru i podešavanje njegovog naloga.
* **DodavanjeUsluga.js** komponenta koja služi za dodavanje novih usluga od strane uprave.
* **ObavestiKorisnika.js** komponenta koja služi za slanje poruke korisniku od strane uprave.
* **Validacija.js** komponenta koja služi za validaciju korisnika od strane uprave.
* **DodavanjeClanka.js** komponenta koja služi za dodaovanje novog članka.

Komponente zadužene za renderovanja:

* **PrikazKorisnika.js** prikazuje listu korisnika fitnes centra.
* **PrikazUsluga.js** prikazuje usluge koje fitnes centra trenurno pruza.
* **PrikazZahteva.js** komponenta koja prikazuje listu svih zahteva pristiglih od strane korisnika.
* **BrisanjeProfila.js** komponenta koja služi za brisanje naloga treneru ili korisniku od strane uprave.
* **PrikazClanka.js** komponenta koja služi za prikaz postojećih objava na blogu.

Komponente kontejneri:

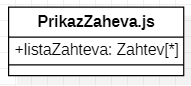
* **UpravljanjeProfilima.js** je komponenta koja je zadužena za upravljanje nalozima korisnika i trenera u fitnes centru.
* **UpravljanjeZahtevima.js** komponenta koja služi za prikaz pristiglih zahteva i upravljanje zahtevima korisnika.
* **UpravljanjeUsluga.js** komponenta koja služi za upravljanje svim uslugama koje fitnes centra pruza.
* **UpravljanjeBlogom.js** komponenta koja služi za zadužena za upravljenje blogom.
* **Uprava.js** je glavna komponenta odnosno kontejner komponenta za sve funkcionalnosti dostupne upravi.

Parametri koji utiču na sadržaje komponenata će biti navedeni sledećim dijagramima klasa.



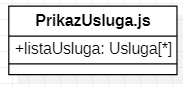
Značenje atributa je sledeće:

* listaKorisnika – lista svih registrovanih korisnika u fitnes centru.



Značenje atributa je sledeće:

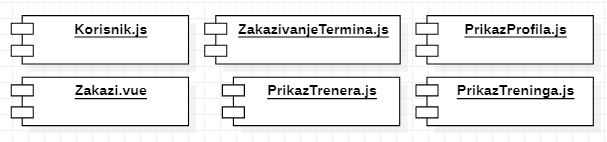
* listaZahteva – lista svih zahteva podnetih od strane korisnika a dostupnih upravi.



Značenje atributa je sledeće:

* listaUsluga – lista svih usluga napravljenih od uprave.

Dizajn korisničkog interfejsa *Korisnik* je obuhvaćen tipom komponenti *.js* skripti.



Komponente forme:

* **Zakazi.js** je komponenta služi za popunjavanje zahteva od stranje korisnika za personalni trening.

Komponente zadužene za renderovanje:

* **PrikazTrenera.js** je komponenta koja prikazuje listu svih trenera koji rade u fitnes centru.
* **PrikazTreninga.js** je komponenta koja je zadužena za prikaz trenutnih treninga koji postoje u fitnes centru.
* **PrikazProfila.js** je komponenta koja služi za prikaz svih informacija o korisniku.

Komponente kontejneri:

* **ZakazivanjeTermina.js** je komponenta koja sadrži komponente za zakazivanje treninga i prikazuje kalendar sa slobodnim i zauzetim terminima
* **Korisnik.js** je glavna komponenta odnosno kontejner komponenta za sve funkcionalnosti dostupne korisniku

Parametri koji utiču na sadržaj komponenata biće predstavljeni dijagramima klasa.

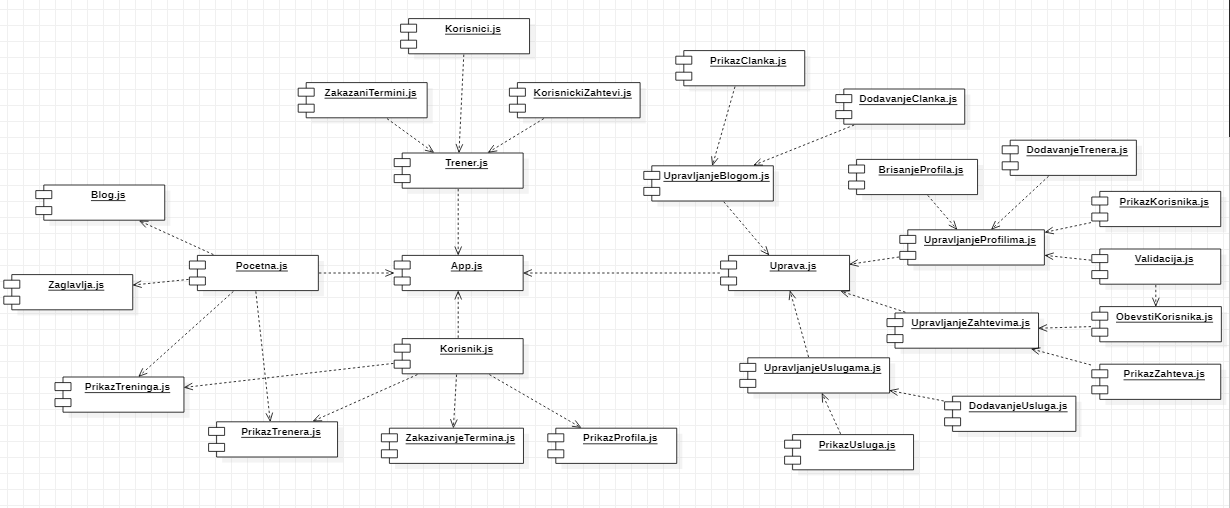


Značenje atributa je sledeće:

* listaTreninga– lista zakazanih treninga od strane korisnika.

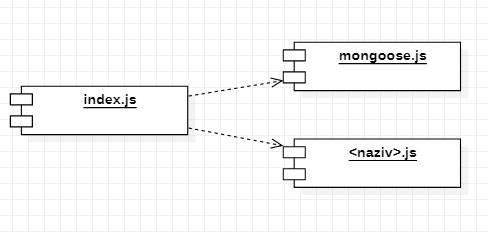
### Komponente aplikacione logike

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:

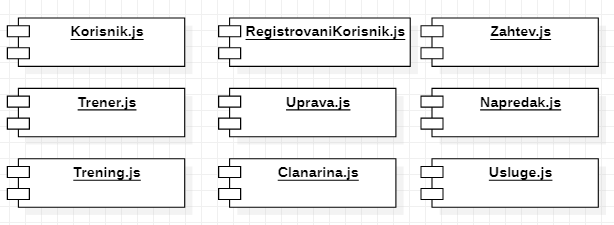


### Komponente za pristup podacima

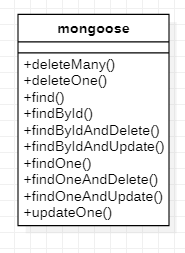
Pristup bazi podataka je omogućen funkcijama koje su definisane u okviru modula **mongoose.js**. U okviru folder *Route*s nalaze se JS fajlovi (<naziv>.js) koji sadrže sve potrebne funkcije za pristup bazi podataka. Fajlovi iz pomenutog foldera, kao i mongoose.js se uključuju u **index.js** tako da su sve funkcije za pristup i komunikaciju sa bazom dostupne svim kreiranim komponentama.



U okviru foldera *Routes* nalaze se sledeći fajlovi:



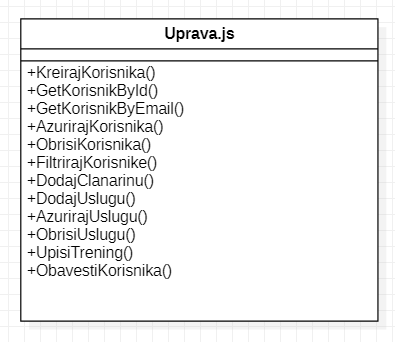
Na sledećim UML klasnom dijagramu pobrojane su definisane funkcije za pristup podacima iz baze koje pruža mongoose:

[[1]](#footnote-1)

Navedene funkcije obavljaju sledeće zadatke:

* deleteMany - briše objekte sa osobinama navedenim u objektima koji se prosleđuje kao paramater funkcije
* deleteOne – briše prvi pronađeni objekat sa osobinama navedenim u objektu koji se prosleđuje kao parametar funkcije
* find – vraća objekte sa osobinama navedenim u objektu koji se prosleđuje kao parametar funkcije
* findById – vraća objekat na osnovu Id
* findByIdAndDelete – briše objekat iz baze na osnovu Id
* findByIdAndUpdate – ažuriranje podataka o objektu sa navedenim Id-om
* findOne – vraća jedan objekat sa osobinama navedenim u objektu koji se prosleđuje kao parametar funkcije
* findOneAndDelete – briše objekat sa osobinama navedenim u objektu koji se prosleđuje kao paramater funkcije
* findOneAndUpdate – ažurira objekat sa osobinama navedenim u objektu koji se prosleđuje kao paramater funkcije
* updateOne – ažuriranje jednog objekta sa navedenim osobinama

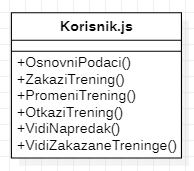
Na sledećim UML klasnom dijagramu pobrojan je deo korisničkih definisanih funkcija za pristup podacima iz baze u okviru fajla Uprava.js:



Navedene funkcije obavljaju sledeće zadatke:

* KreirajKorisnika – kreira novog korisnika
* GetKorisnikById – pronalazi korisnika čiji je Id prosleđen kao parametar f-je
* GetKorisnikByEmail – pronalazi korisnika čiji je email isti kao paramatar prosleđen f-ji
* AzurirajKorisnika – ažurira korisnika
* ObrisiKorisnika – briše korisnika
* FiltrirajKorisnike – filtrira korisnike po tipu koji je prosleđen kao paramatar f-je
* DodajClanarinu – odgovarajućem korisniku dodaje novu članarinu
* DodajUslugu – dodaje novu uslugu u listu usluga fitnes centra
* AzurirajUslugu – ažurira već postojeću uslugu koja je prosleđene kao parametar f-je
* ObrisiUslugu – briše uslugu koja je prosleđena kao parametar f-je
* UpisiTrening – dodaje novi trening u listu treninga
* ObavestiKorisnika – korisniku prosleđuje odgovarajuću poruku

Na sledećim UML klasnom dijagramu pobrojan je deo korisničkih definisanih funkcija za pristup podacima iz baze u okviru fajla Korisnik.js:



Navedene funkcije obavljaju sledeće zadatke:

* OsnovniPodaci – vraća osnovne podatke o korisniku
* ZakaziTrening – šalje zahtev za zakazivanje treninga
* PromeniTrening – menja trening čiji je Id prosleđen kao parametar f-je
* OtkaziTrening – otkazuje trening čiji je Id prosleđen kao parametar f-je
* VidiNapredak – vraća napredak korinika
* VidiZakazaneTreninge – vraća sve zakazane treninge korisnika
* DodajUslugu – dodaje novu uslugu u listu usluga fitnes centra

Na sledećim UML klasnom dijagramu pobrojan je deo korisničkih definisanih funkcija za pristup podacima iz baze u okviru fajla Trener.js:



Navedene funkcije obavljaju sledeće zadatke:

* DodajKorisnika – ubacuje novog korisnika u listu klijenata čiji je Id prosleđen kao parametar f-je
* VidiSveKorisnike – pronalazi sve korisnike koji treniraju kod trenera
* ObrisiKorisnika – izbacuje korisnika iz listu klijenata čiji je Id prosleđen kao parametar f-je
* VidiNapredakKorisnika – prikazuje se napredak korisnika čiji je Id prosleđen kao parametar f-je
* DodajNapredakKorisnika – dodaju se nova merenja za korisnika čiji je Id prosleđen kao parametar f-je
* ZakaziGrupniTrening – dodaje se novi grupni trening
* PromeniTrening – odgovarajući trening se ažurira
* VidiSveTreninge – prikazuje sve buduće treninge trenera
* PrihvatiTrening – trener odobrava zahtev za trening
* OdbijTrening – trener odbija zahtev za trening

# 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 izvršenja 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. RaketaDoLeta 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.

1. [Parametri funkcija su izostavljeni radi preglednosti klasnog dijagrama](https://mongoosejs.com/docs/queries.html), kao i kod ostalih klasnih dijagrama u ovoj sekciji [↑](#footnote-ref-1)