RentASpot

Veb aplikacija za prodaju i iznajmljivanje lokala

Detaljni arhitekturni projekat

Verzija 1.0

Pregled izmena

|  |  |  |  |
| --- | --- | --- | --- |
| **Datum** | **Verzija** | **Opis** | **Autor** |
| 10.12.2022. | 1.0 | Inicijalna verzija | Boško Amanović |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

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 Izbor teme aplikacije 8

6.2.2 Brisanje objava 8

6.2.3 Registracija 8

6.2.4 Prijavljivanje 8

6.2.5 Obnova lozinke 8

6.2.6 Izmena podataka 8

6.2.7 Dodavanje novih objava 8

6.2.8 Ažuriranje postojećih objava 8

6.2.9 Pretraživanje objava 8

6.2.10 Komentarisanje objava 9

6.2.11 Ocenjivanje objava 9

6.2.12 Prikaz objava 9

6.2.13 Pregled profila 9

6.2.14 Brisanje komentara 9

6.2.15 Brisanje objava drugih korisnika 9

6.2.16 Brisanje komentara drugih korisnika 9

6.2.17 Brisanje drugih korisnika 9

6.2.18 Opomena drugih 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 React.js 10

7.1.5 Node.js 11

7.1.6 MySQL 11

8. Pogled na procese 11

8.1 Procesi 11

8.1.1 Web čitač 11

8.1.2 Web server 11

8.1.3 PHP-CGI 11

8.1.4 MySQL 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 12

10.1 Model domena 12

10.2 Šema baze podataka 13

10.3 Komponente sistema 19

10.3.1 Komponente korisničkog interfejsa 19

10.3.2 Komponente aplikacione logike 19

10.3.3 Komponente za pristup podacima 20

11. Performanse 21

12. Kvalitet 21

Detaljni arhitekturni projekat

# Cilj dokumenta

Cilj ovog dokumenta je detaljni opis arhitekture “RentASpot” veb aplikacije.

# Opseg dokumenta

Dokument se odnosi na “RentASpot” veb aplikaciju koja će biti razvijena od strane BARTeam-a. “RentASpot” predstavlja skraćenicu za Portal za prodaju i iznajmljivanje lokala. Namena sistema je efikasno prezentovanje, kreiranje i održavanje sadržaja vezanih za rad jedne aplikacije za deljenje oglasa.

# Reference

Spisak korišćene literature:

1. BARTeam-01-RentASpot- Predlog projekta, V1.0, 2022, BARTeam.
2. BARTeam-02-RentASpot- Raspored aktivnosti, V1.0, 2022, BARTeam.
3. BARTeam-03-RentASpot- Plan realizacije, V1.0, 2022, BARTeam.
4. BARTeam-04-RentASpot- Vizija sistema, V1.0, 2022, BARTeam.
5. BARTeam-05-RentASpot- Specifikacija zahteva, V1.0, 2022, BARTeam.

# Predstavljanje arhitekture

Arhitektura sistema u dokumentu je prikazana kao serija pogleda na sistem: pogled na slučajeve korišćenja, 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. “RentASpot” aplikacija će biti implementirana kao Web aplikacija zasnovana na REACT-u i MongoDB bazi podataka [4].
2. Klijentski deo “RentASpot” portala će biti optimizovan za sledeće Web čitače: Internet Explorer 6.0 i noviji, Opera 8.0 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.

# Pogled na slučajeve korišćenja

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

Slučajevi korišćenja “RentASpot” portala su:

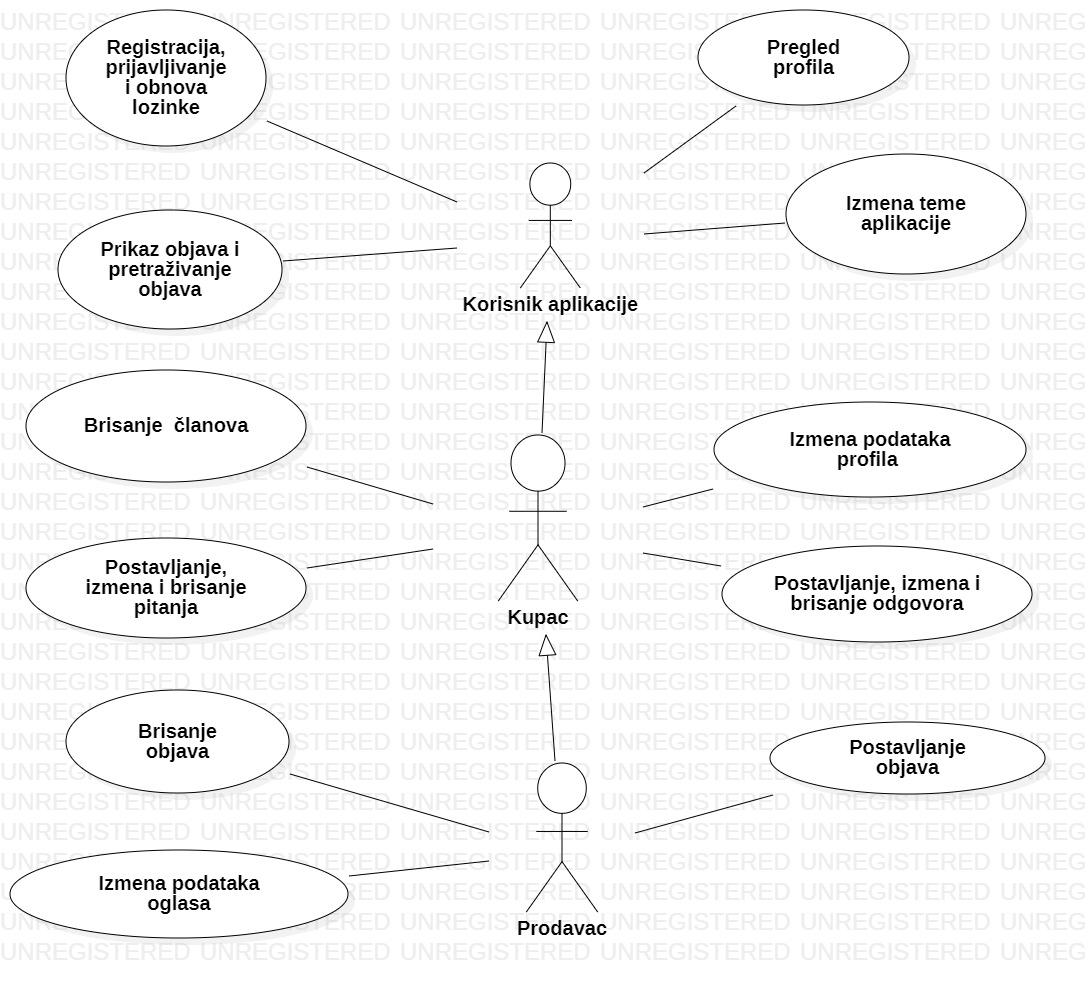
* Izbor teme
* *Registracija, prijavljivanje i obnova lozinke*
  + Registracija
  + Prijavljivanje
  + Obnova lozinke
* *Postavljanje, izmena i brisanje pitanja*
  + - Postavljanje komentara
    - Izmena komentara
    - Brisanje komentara

* Pregled profila
* Pretraživanje objava
* *Postavljanje, brisanje i izmena odgovora*
  + Postavljanje odgovora
  + Izmena odgovora
  + Brisanje odgovora
* Prikaz objava
* Brisanje članova
* Brisanje objava
* Izmena podataka oglasa
* Izmena podataka profila

Ove slučajevi korišćenja mogu da iniciraju posetilac portala(korisnik) ili registrovani korisnik.

## Dijagrami slučajeva korišćenja

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



Slučajevi korišćenja *„Registracija, prijavljivanje i obnova lozinke“, „Prikaz objava i pregled objava“ , „Postavljanje, izmena i brisanje odgovora“ , „Postavljanje, izmena i brisanje pitanja“* obuhvataju složenije radnje koje se mogu razložiti dalje razložiti na pojedinačne slučajeve korišćenja.

Detaljni UML dijagram za slučaj korišćenja *„Registracija, prijavljivanje i obnova lozinke“* je prikazan na sledećoj slici:



Detaljni UML dijagram za slučaj korišćenja *„Postavljanje, izmena i brisanje pitanja“* je prikazan na sledećoj slici:



Detaljni UML dijagram za slučaj korišćenja *„Postavljanje, izmena i brisanje odgovora“* je prikazan na sledećoj slici:



Detaljni UML dijagram za slučaj korišćenja *„Prikaz objava i pregled objava“* je prikazan na sledećoj slici:



## Kratak opis slučajeva korišćenja

### Izbor teme

Kratak opis: Izbor između ponuđenih tema za izgled aplikacije.

Akteri koji iniciraju slučaj korišćenja: Posetilac portala, Registrovani korisnik.

### Pregled profila

Kratak opis: Pregled željenog profila.

Akteri koji iniciraju slučaj korišćenja: Posetilac portala, Registrovani korisnik.

### Registracija

Kratak opis: Kreiranje profila na portalu.

Akteri koji iniciraju slučaj korišćenja: Posetilac portala.

### Prijava

Kratak opis: Prijavljivanje na portal.

Akteri koji iniciraju slučaj korišćenja: Registrovani korisnik.

### Obnova lozinke

Kratak opis: Korisniku se šalje lozinka.

Akteri koji iniciraju slučaj korišćenja: Registrovani korisnik.

### Prikaz objava

Kratak opis: Pregled oglasa.

Akteri koji iniciraju slučaj korišćenja: Posetilac portala, Registrovani korisnik.

### Brisanje članova

Kratak opis: Brisanje sopstvenog profila.

Akteri koji iniciraju slučaj korišćenja: Registrovani korisnik.

### Brisanje objava

Kratak opis: Brisanje sopstvenih objava.

Akteri koji iniciraju slučaj korišćenja: Registrovani korisnik.

### Pretraživanje objava

Kratak opis: Pretraga postojećih objava.

Akteri koji iniciraju slučaj korišćenja: Posetilac portala, Registrovani korisnik.

### Brisanje odgovora

Kratak opis: Brisanje svojih odgovora.

Akteri koji iniciraju slučaj korišćenja: Registrovani korisnik.

### Izmena podataka oglasa

Kratak opis: Izmena podataka svog oglasa.

Akteri koji iniciraju slučaj korišćenja: Registrovani korisnik.

### Postavljanje odgovora

Kratak opis: Postavljanje odgovora na portal.

Akteri koji iniciraju slučaj korišćenja: Registrovani korisnik.

### Izmena odgovora

Kratak opis: Izmena svog odgovora.

Akteri koji iniciraju slučaj korišćenja: Registrovani korisnik.

### Brisanje pitanja

Kratak opis: Brisanje svojih pitanja.

Akteri koji iniciraju slučaj korišćenja: Registrovani korisnik.

### Izmena pitanja

Kratak opis: Izmena svojih pitanja.

Akteri koji iniciraju slučaj korišćenja: Registrovani korisnik.

### Postavljanje pitanja

Kratak opis: Postavljanje pitanja na portal.

Akteri koji iniciraju slučaj korišćenja: Registrovani 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 “RentASpot” portal obuhvata 3 glavna paketa: Korisnički interfejs, Aplikaciona logika, Pristup podacima.

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

Paket *Pristup podacima* sadrži Node.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 i React.js skripte koje generišu HTML stranice preko kojih korisnici komuniciraju sa sistemom.

Sloj korisničkog interfejsa zavisi od sloja aplikacione logike, kao i paketa HTML i React.js-a.

### Aplikaciona logika

sloj

Sloj aplikacione logike je srednji sloj u troslojnoj arhitekturi “RentASpot” portala. Sadrži Node.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 React.js paketa.

### Pristup podacima

sloj

Sloj za pristup podacima se nalazi na dnu troslojne arhitekture i sadrži Node.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 Node.js i MongoDB baza podataka.

### HTML

tehnologija

Tehnologija React.js-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.

### Node.js

tehnologija

Tehnologija Node.js-a obezbeđuje mehanizam za pisanje i izvršavanje skripti na strani servera. Ove skripte 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 “RentASpot” portala.

# Pogled na procese

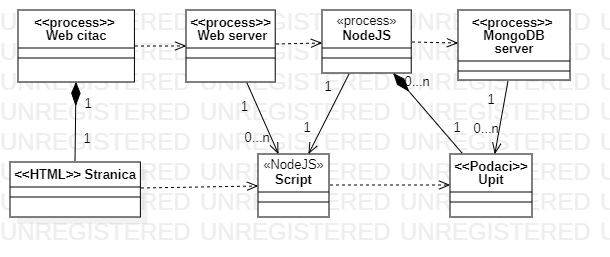
U ovom odeljku je sadržan pogled na procesnu arhitekturu sistema. Ovaj opis treba da sadrži specifikaciju različitih zadataka (procesa i niti) uključenih u rad sistema. Takođe je potrebno dati dijagrame koji pokazuju njihovu interakciju i konfiguraciju. Dodela objekata i klasa na određene zadatke takođe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

Web aplikacije zasnovane na PHP-u imaju relativno jednostavan procesni model koji je u potpunosti pod kontrolom Web servera. Sa stanovišta projektanta PHP 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 “RentASpot” portala kao Web aplikacije.

## Procesi

Na sledećem UML dijagramu klasa prikazani su procesi koji učestvuju u izvršenju “RentASpot” portala. Dijagram je opšteg tipa i može se primeniti na bilo koju Web aplikaciju zasnovanu na NodeJS-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 NodeJS 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 NodeJS skripta i generiše odgovarajući tekstualni sadržaj koji Web server šalje Web čitaču. Za izvršenje NodeJS skripta ovaj proces može da zahteva usluge MongoDB servera-a. Komunikacija između NodeJS procesa i MongDB servera se obavlja preko prosleđivanja upita i vraćanja rezultat.

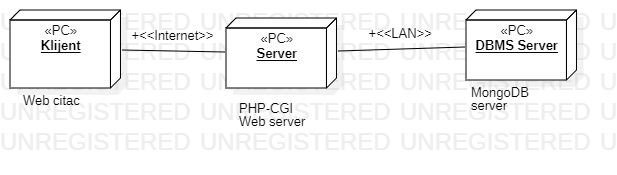
### MongoDB Server

MongoDB 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 “RentASpot” portala.



## Klijent

Pristup “RentASpot” portal 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 koji vrše obradu zadatih NodeJS skripti. U najopštijoj konfiguraciji DBMS se izvršava na posebnoj mašini koja je sa Web serverom u lokalnoj mreži (LAN).

## DBSM server

DBMDB 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 “RentASpot” portal ovaj odeljak sadrži model domena, šemu baze podataka i prikaz komponenti sistema razvrstanih u ranije identifikovane pakete.

## Model domena

Model domena za koji se “RentASpot” 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 (NodeJS skripti) koje će biti implementirane.



## Šema baze podataka

Detaljna šema baze podataka je prikazana na sledećem dijagramu. Baza podataka i dijagram su kreirani korišćenjem *MS Access*-a, dok je migracija na MongoDB obavljena pomoću *MongoDB Migration Toolkit*-a.

## Komponente sistema

Komponente sistema “RentASpot” portala su NodeJS skripti čiji će pregled biti dat po arhitekturnim slojevima. Za ilustraciju će biti korišćeni UML dijagrami komponenti, ali i dijagrami klasa. U slučajevima gde je NodeJS 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:

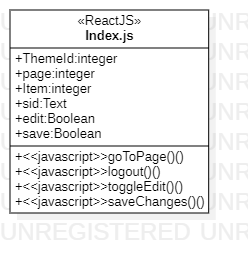


Komponenta **index.js** je implementira stranicu portala čiji sadržaj može da varira od parametra koji joj se proslede u zahtevu.

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

Komponenta **HTMLArea.js** predstavlja OpenSource HTML editor koji se koristi za ažuriranje složenijih opisa.

Parametri koji utiču na izbor i jezik za prikaz stranice ilustrovani su sledećim dijagramom klasa:



Značenje atributa je sledeće:

* ThemeID – identifikator teme
* page – interni identifikator
* item – identifikator stavke koja se detaljno prikazuje na stranici (određeni oglas ili pitanje)
* sid – identifikator sesije kada je korisnik ulogovan
* edit – definiše da li je uključeno uređivanje
* save – definiše da li treba izvršiti snimanje podataka sa tekuće stranice/stavke

### Komponente aplikacione logike

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

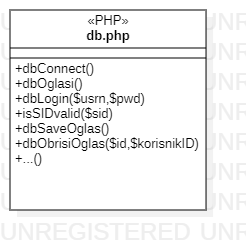


### Komponente za pristup podacima

Pristup bazi podataka je u potpunosti zatvoren u funkcije koje su definisane u okviru NodeJS 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 samom skriptu
* dbOglasi – vraća oglase
* dbLogin – loguje korisnika i vraća SID
* isSIDValid – vrši proveru validnosti SID-a
* dbSaveOglas – upisuje oglas u bazu
* dbObrisiOglas + briše oglas iz baze
* …()-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. “RentASpot” 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.