RoadRunner

Arhitekturni projekat

Verzija 1.0

Pregled izmena

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

Sadržaj

1. Cilj dokumenta 5

2. Opseg dokumenta 5

3. Reference 5

4. Predstavljanje arhitekture 5

5. Ciljevi i ograničenja arhitekture 5

6. Pogled na slučajeve korišćenja 5

6.1 Dijagrami slučajeva korišćenja 6

6.2 Kratak opis slučajeva korišćenja 8

6.2.1 Izbor jezika 8

6.2.2 Pregled osnovnih podataka o laboratoriji 8

6.2.3 Pregled spiska članova 8

6.2.4 Pregled podataka o određenom članu laboratorije 8

6.2.5 Pregled publikacija po autoru 8

6.2.6 Pregled publikacija po tipu 8

6.2.7 Pregled publikacija po godini 8

6.2.8 Pregled spiska projekata 8

6.2.9 Pregled podataka o određenom projektu 8

6.2.10 Prijavljivanje 9

6.2.11 Ažuriranje podataka o članu 9

6.2.12 Dodavanje nove publikacije 9

6.2.13 Brisanje postojeće publikacije 9

6.2.14 Ažuriranje osnovnih podataka o laboratoriji 9

6.2.15 Kreiranje novog člana 9

6.2.16 Brisanje postojećeg člana 9

6.2.17 Arhiviranje postojećeg člana 9

6.2.18 Kreiranje projekta i postavljanje vođe 9

6.2.19 Ažuriranje podataka o projektu 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 PHP 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 14

10.3.1 Komponente korisničkog interfejsa 14

10.3.2 Komponente aplikacione logike 14

10.3.3 Komponente za pristup podacima 15

11. Performanse 16

12. Kvalitet 16

Arhitekturni projekat

# Cilj dokumenta

Cilj ovog dokumenta je detaljni opis arhitekture RoadRunner web apliakcije.

# Opseg dokumenta

Dokument se odnosi na RoadRunner aplikacije koji će biti razvijen od strane Intalice. Ovaj sistem će omogućiti lako pronalaženje vozača i kompanija kojima su potrebne usloge kao i laksi dogovor oko cene prevoza, on takodje omogućava veci i bolji izbor kao i bolju specifikaciju zahteva to jest uslova.

# Reference

Spisak korišćene literature:

1. RoadRunner – Predlog projekta, Intalica- RoadRunner -01, V1.0, 2023, Intalica.
2. RoadRunner – Planirani raspored aktivnosti na projektu, V1.0, 2023, Intalica.
3. RoadRunner – Plan realizacije projekta, V1.0, 2023, Intalica.

4. RoadRunner – Vizija sistema, V1.0, 2023, Intalica.

5. RoadRunner – Specifikacija zahteva, V1.0,2023, Intalica.

# 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. RoadRunner će biti implementiran kao Web aplikacija zasnovana na React.js na front endu , .NET Core-u na back endu , SQL Server-u .
2. Klijentski deo RoadRunner aplikacija će biti optimizovan za sledeće Web čitače: Internet Explorer 11.0 i noviji, Opera 31.0 i noviji, Google Chrome 47.0 i noviji, kao i Mozilla Firefox 36.0 i noviji
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 RoadRunner aplikacije su:

* Registovanje
* Izbor jezika
* Prijavljivanje
* Ažuriranje podataka o sebi
* Biranje ture
* Dodavanje vozila
* Dodavanje prikolica
* Komunikacija sa kompanijom/vozačem
* Biranje vozača
* Dodavanje ture
* Ažuriranje ture
* Ocenjivanje vozača
* Favorizacija vozača
* Slanje tura vozačima
* Brisanje kompanija
* Brisanje vozača

Ove slučajevi korišćenja mogu da iniciraju posetilac aplikacije, vozač, dispečer i kompanija.

## Dijagrami slučajeva korišćenja

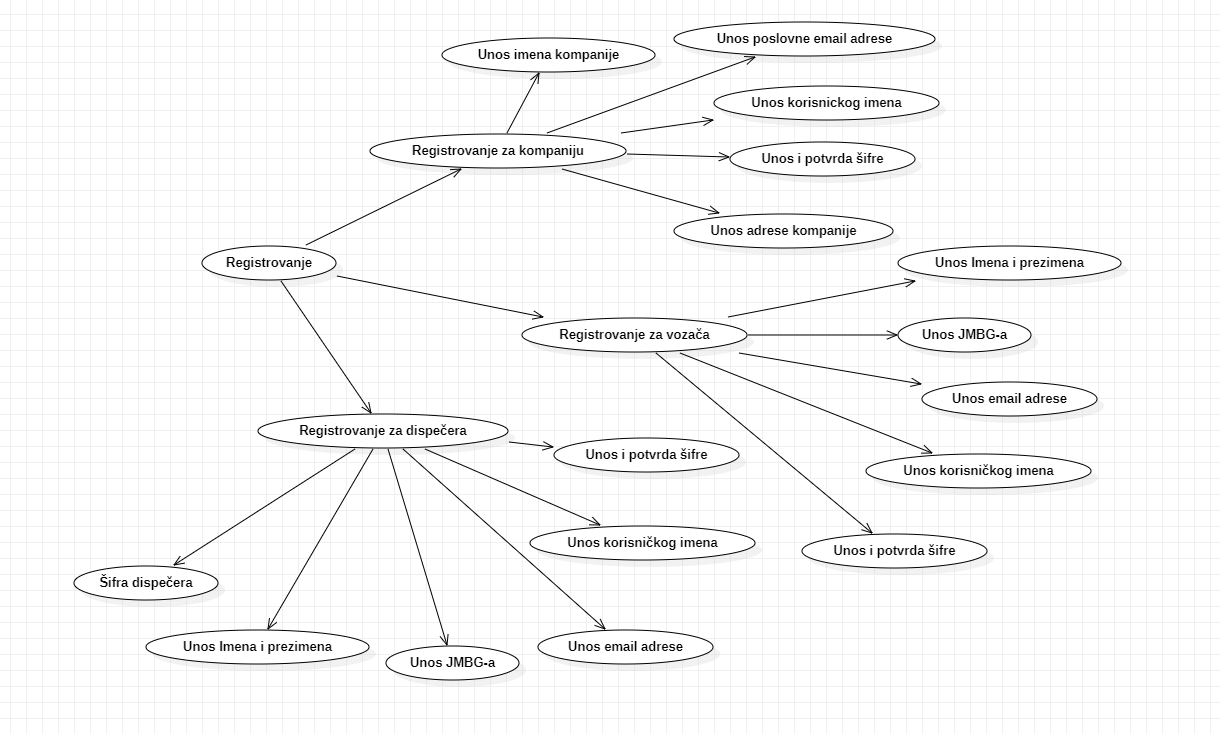
Osnovni UML dijagram koji prikazuje korisnike i slučajeve korišćenja RoadRunner portala prikazan je na sledećoj slici:

Diagram

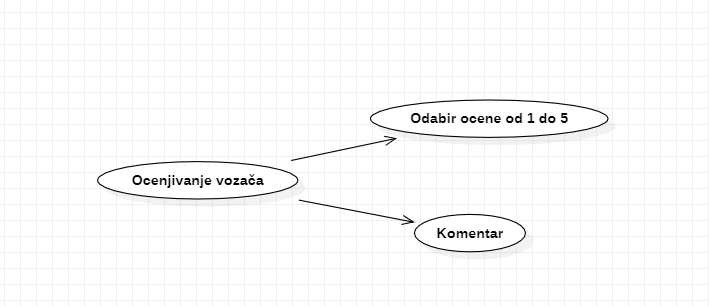
Description automatically generated

Slučajevi Prijavljivanja,Ocenjivanje vozača,Kreiranje liste vozača za ture obuhvataju složenije radnje koje se mogu razložiti dalje razložiti na pojedinačne slučajeve korišćenja.

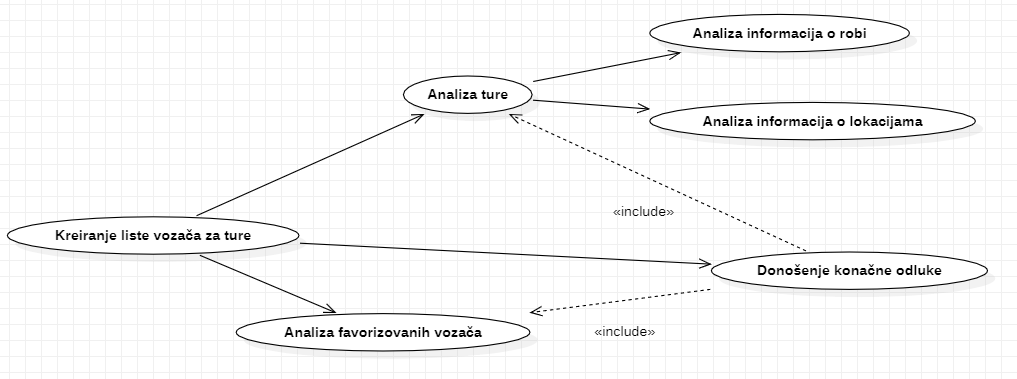
Detaljni UML dijagram za slučaj Registrovanja je prikazan na sledećoj slici:



Detaljni UML dijagram za slučaj Ocenjivanja vozača je prikazan na sledećoj slici:



Detaljni UML dijagram za slučaj Kreiranja liste vozača za ture je prikazan na sledećoj slici:



## Kratak opis slučajeva korišćenja

### Registrovanje

Kratak opis: Registrovanje korisnika na aplikaciju.

Akteri koji iniciraju slučaj korišćenja: Posetilac stranice

### Izbor jezika

Kratak opis: Izbor između srpskog i engleskog jezika za prikaz informacija u aplikaciji

Akteri koji iniciraju slučaj korišćenja: Posetilac stranice, Kompanija, Vozač, Dispečer.

### Prijavljivanje

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

Akteri koji iniciraju slučaj korišćenja: Kompanija, Vozač, Dispečer.

### Ažuriranje podataka o sebi

Kratak opis: Korisnik u zavisnosti od svog naloga ima različite lične podatke kojima ovde može pristupiti i promeniti.

Akteri koji iniciraju slučaj korišćenja: Kompanija, Vozač, Dispečer.

### Biranje ture

Kratak opis: Prikaz stranice sa listom tura koje je vozač dobio i od kojih on bira one koje njemu odgovaraju.

Akteri koji iniciraju slučaj korišćenja: Vozač.

### Dodavanje vozila

Kratak opis: Prikaz stranice sa listom vozila koje vozač poseduje.

Akteri koji iniciraju slučaj korišćenja: Vozač.

### Dodavanje prikolica

Kratak opis: Prikaz stranice sa listom prikolica koje vozač poseduje.

Akteri koji iniciraju slučaj korišćenja: Vozač.

### Komunikacija sa kompanijom/vozačem

Kratak opis: Uspostavanje komunikacije vozača i kompanije za čiju turu je on izabran.

Akteri koji iniciraju slučaj korišćenja: Kompanija, Vozač.

### Biranje vozača

Kratak opis: Prikaz stranice sa listom vozača koje je kompanija dobila i od kojih on bira onog koji odgovara turi.

Akteri koji iniciraju slučaj korišćenja: Kompanija.

### Dodavanje ture

Kratak opis: Dodavanje ture od strane kompanije.

Akteri koji iniciraju slučaj korišćenja: Kompanija.

### Ažuriranje ture

Kratak opis: Ažuriranje ture od strane kompanije koja ju je dodala.

Akteri koji iniciraju slučaj korišćenja: Kompanija.

### Ocenjivanje vozača

Kratak opis: Kompanija dodeljuje ocenu vozaču koji je izvršio neku turu.

Akteri koji iniciraju slučaj korišćenja: Kompanija.

### Favorizovanje vozača

Kratak opis: Kompanija dodaje vozača u svoju listu favorizovanih vozača.

Akteri koji iniciraju slučaj korišćenja: Kompanija.

### Slanje tura vozačima

Kratak opis: Dispečer u zavisnosti o podacima koje vozači poseduju šalje ture koje njima odgovaraju.

Akteri koji iniciraju slučaj korišćenja: Dispečer.

### Brisanje kompanija

Kratak opis: Dispečer briše kompanije.

Akteri koji iniciraju slučaj korišćenja: Dispečer.

### Brisanje vozača

Kratak opis: Dispečer briše vozača.

Akteri koji iniciraju slučaj korišćenja: Dispečer.

# 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.

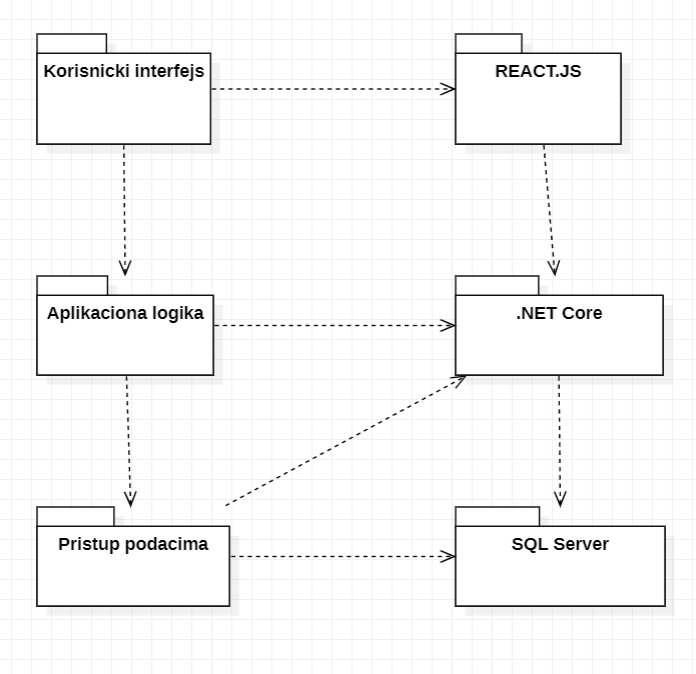
Logički pogled na RoadRunner obuhvata 3 glavna paketa: Korisnički interfejs, Aplikaciona logika, Pristup podacima.

Paket *Korisnički interfejs* sadrži Web stranice, React.js scripte 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 scripte na front endu i .NET Core na back endu koji su zaduženi za realizaciju funkcionalnosti specifičnih za domen sistema koji se razvija.

Paket *Pristup podacima* sadrži React.js scripte 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-a.

### Aplikaciona logika

sloj

Sloj aplikacione logike je srednji sloj u troslojnoj arhitekturi RoadRunner aplikacije. 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 .NET Core-a.

### Pristup podacima

sloj

Sloj za pristup podacima se nalazi na dnu troslojne arhitekture i sadrži JavaScript skripte zadužene za pribavljanje, dodavanje i ažuriranje podataka koji se čuvaju u SQL Server bazi podataka.

Ovaj sloj ne zavisi od drugih slojeva, ali je zavisan od paketa .NET Core i SQL Server bazi 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.

### .NET Core

tehnologija

Tehnologija .NET Core obezbeđuje mehanizam za pisanje i izvršavanje skripti na strani servera. Ove skripte mogu da generišu HTML kod koji realizuje korisnički interfejs i pristupaju bazi podataka u cilju pribavljanja, unosa i ažuriranja podataka.

### SQL server

DBMS

SQL server predstavlja sistem za upravljanje bazama podataka koji će se koristiti za realizaciju RoadRunner aplikacije.

# Pogled na procese

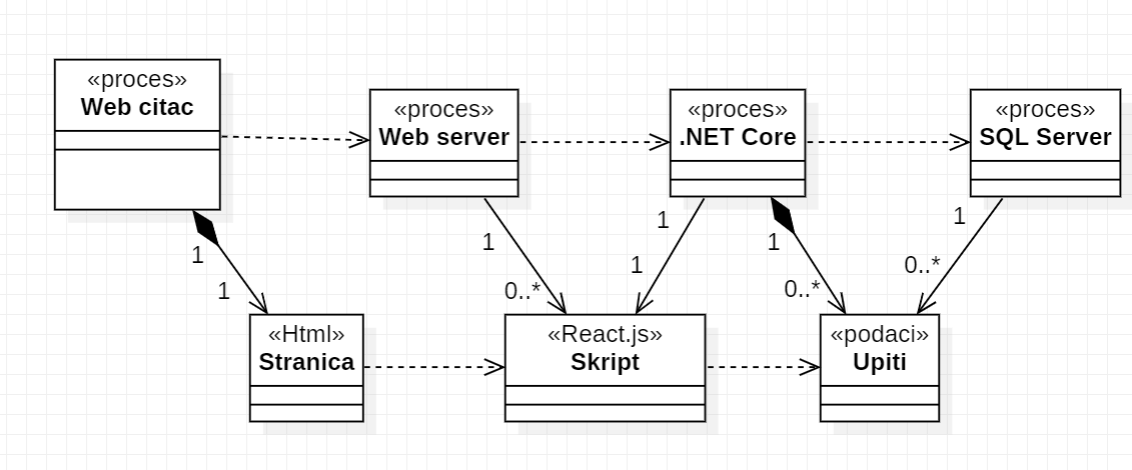
U ovom odeljku je sadržan pogled na procesnu arhitekturu sistema. Ovaj opis treba da sadrži specifikaciju različitih zadataka (procesa i niti) uključenih u rad sistema. Takođe je potrebno dati dijagrame koji pokazuju njihovu interakciju i konfiguraciju. Dodela objekata i klasa na određene zadatke takođe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

Web aplikacije zasnovane na .NET Core-u imaju relativno jednostavan procesni model koji je u potpunosti pod kontrolom Web servera. Sa stanovišta projektanta .NET Core 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 RoadRunner kao Web aplikacije.

## Procesi

Na sledećem UML dijagramu klasa prikazani su procesi koji učestvuju u izvršenju RoadRunner. Dijagram je opšteg tipa i može se primeniti na bilo koju Web aplikaciju zasnovanu na .NET Core i SQL server 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 React.js skript, Web server inicira izvršenje .NET Core procesa koji obrađuje odgovarajući skript i generiše sadržaj koji se vraća čitaču. Web server može paralelno da inicira veći broj .NET Core procesa.

### .NET Core

.NET Core proces obavlja posao obrade zadatog React.js skripta i generiše odgovarajući tekstualni sadržaj koji Web server šalje Web čitaču. Za izvršenje React.js skripta ovaj proces može da zahteva usluge SQL servera-a. Komunikacija između .NET Core procesa i SQL servera se obavlja preko prosleđivanja upita i vraćanja rezultat.

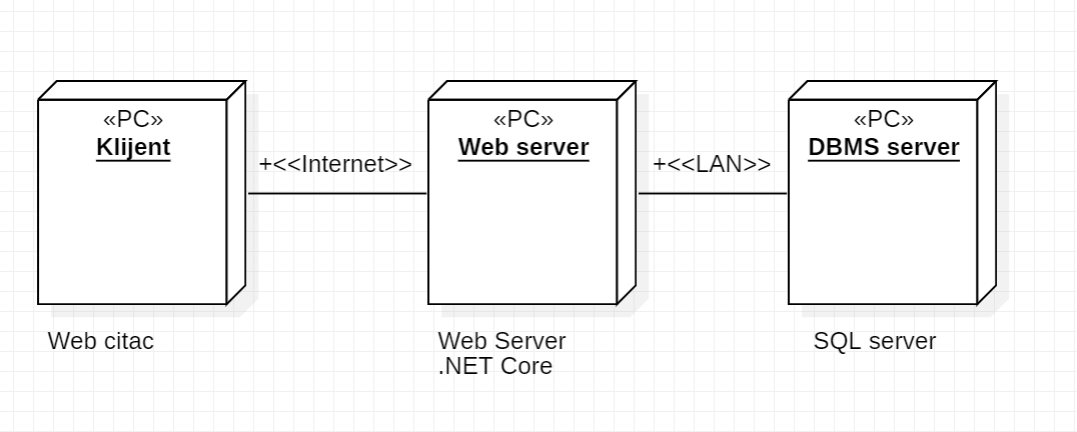
### SQL Server

SQL Server je proces koji izvršava funkcionalnost MySQL 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 RoadRunner.



## Klijent

Pristup RoadRunner 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 .NET Core koji vrše obradu zadatih React.js skripti. 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 SQL 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).

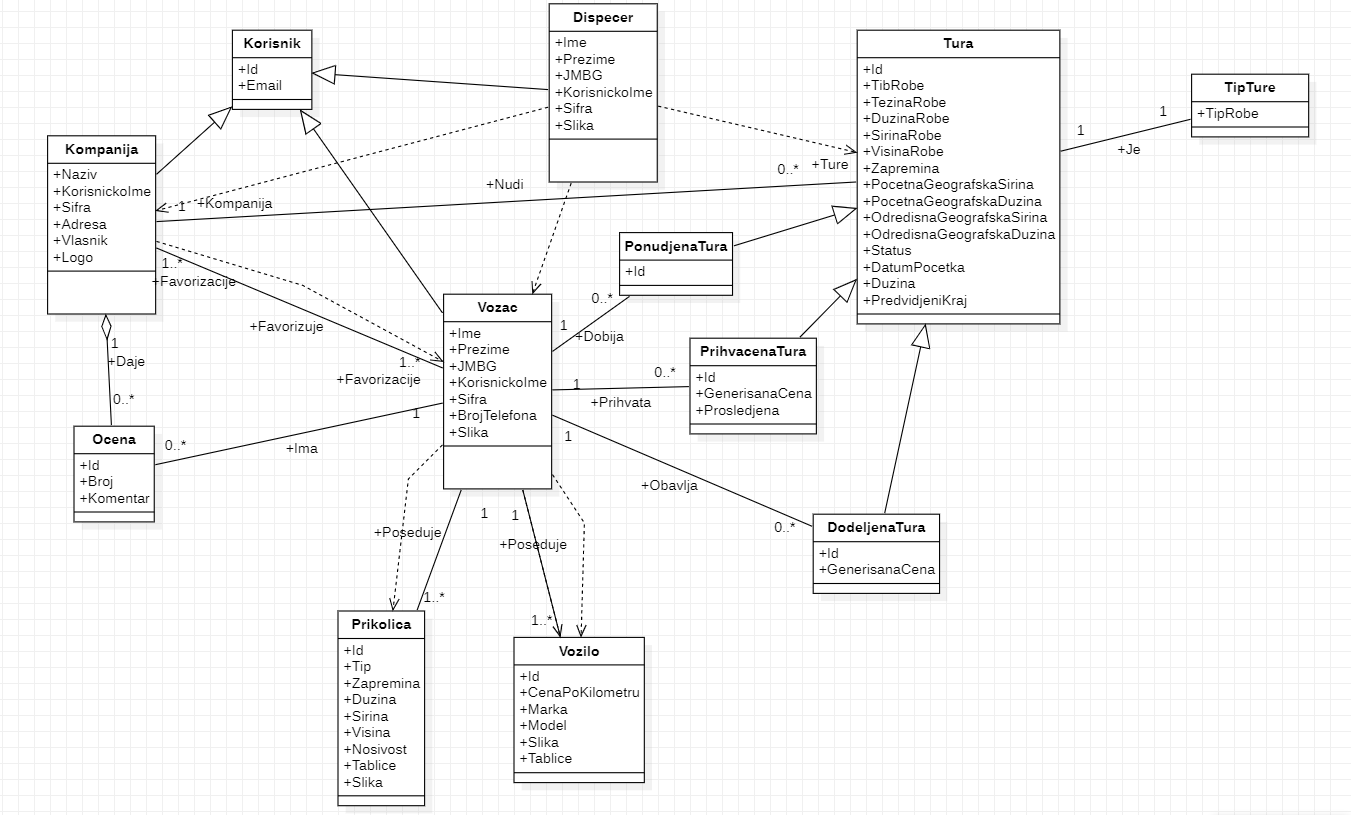
# Pogled na implementaciju sistema

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

## Model domena

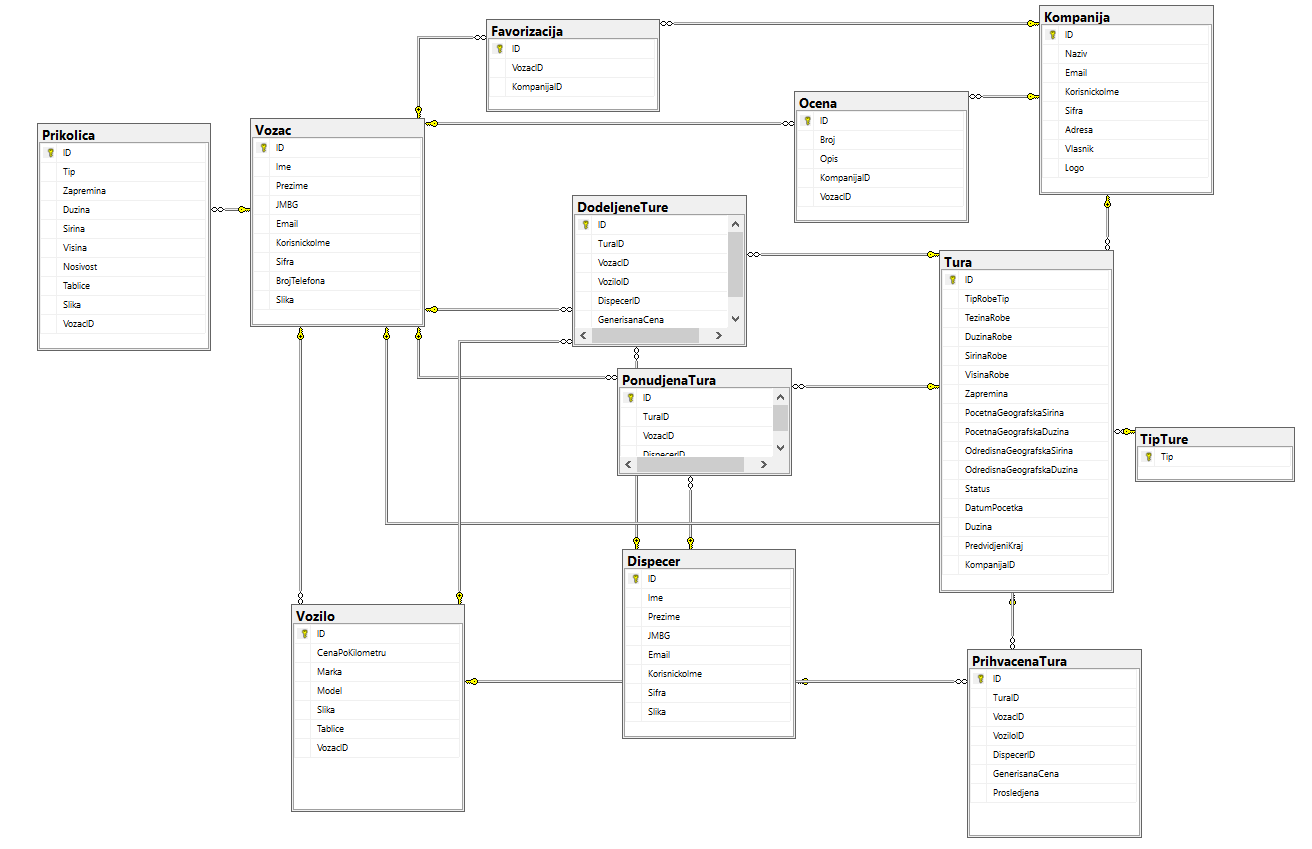
Model domena za koji se RoadRunner projektuje je ilustrovan UML dijagramom klasa. U njemu su prikazane domenske klase, neki od njihovih atributa, kao i veze koje se mogu identifikovati između njih.

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



## Šema baze podataka

Detaljna šema baze podataka je prikazana na sledećem dijagramu. Baza podataka i dijagram su kreirani korišćenjem *SQL Server Menagment Studi-*a, dok je migracija na SQL Server obavljena pomoću *.NET Core-a*.



## Komponente sistema

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

### Komponente korisničkog interfejsa

Dizajn korisničkog interfejsa je obuhvaćen dvema komponentama:

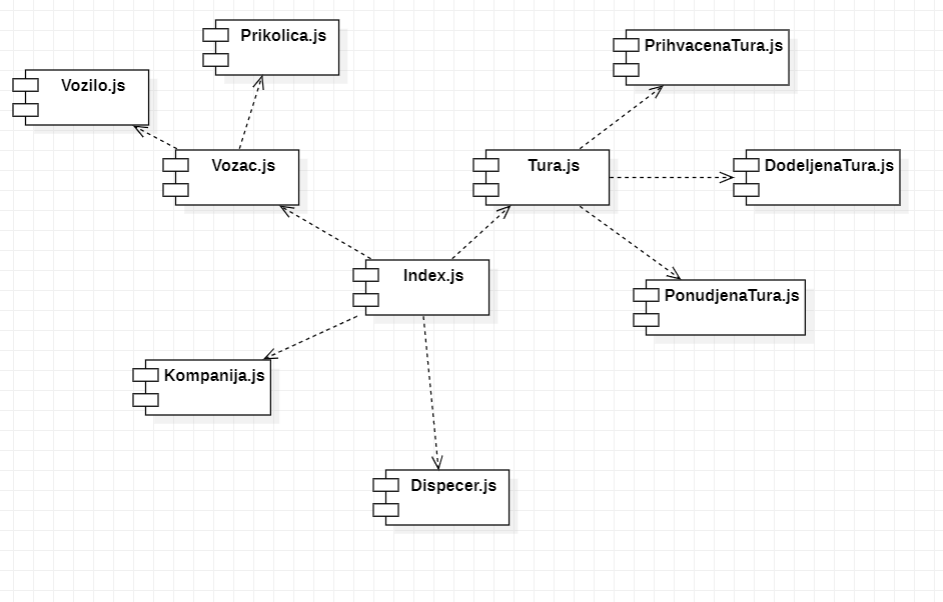


Komponenta **index.js** implementira stranicu portala čiji sadržaj može da varira od parametra koji joj se proslede u zahtevu.

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

### Komponente aplikacione logike

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



### Komponente za pristup podacima

Pristup bazi podataka je u potpunosti zatvoren u funkcije koje su definisane u okviru JavaScript 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
* dbDodajKompaniju – kreiramo kompaniju
* dbObrisiKompaniju – briše kompaniju
* dbIzmeniTuru – menjaTuru
* dbDodajOcenu – dodaje se ocena
* dbvratiDodeljeneTure – vraća dodeljene ture od strane Vozaču
* dbvratiKreiraneTure – vraća kreirane ture od strane Kompanije
* dbvratiPonudjeneTure – vraća ponudjene ture koje je dobio Vozač
* dbvratiPrihvaceneTure – vraća prihvaćene ture koje je odobrio Dispečer
* ... – 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. RoadRunner 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.