VENTSPILS AUGSTSKOLA  
INFORMĀCIJAS TEHNOLOĢIJU FAKULTĀTE

DATU BĀZU TEHNOLOĢIJAS

PATSTĀVĪGAIS DABS

**Transporta pakalpojumu datu bāzes sistēma**

Ventspils Augstskolas

Informācijas tehnoloģiju fakultātes

Bakalaura studiju programmas

“Datorzinātnes”

3. kursa students

Valters Ādmīdiņš

Kursa pasniedzējs:

Gints Neimanis

Ventspils Augstskola

2018

Saturs

[**1.** **IEVADS** 3](#_Toc532588491)

[**2.** **ERD DIAGRAMMA** 4](#_Toc532588492)

[**3.** **TABULAS AR DATIEM** 5](#_Toc532588493)

[**4.** **APLIKĀCIJA** 11](#_Toc532588494)

[**5.** **SKATI** 22](#_Toc532588495)

[**6.** **PROCEDŪRAS** 24](#_Toc532588496)

[**7.** **FUNKCIJAS** 27](#_Toc532588497)

[**8.** **TRIGERI** 30](#_Toc532588498)

[**9.** **SECINĀJUMI** 31](#_Toc532588499)

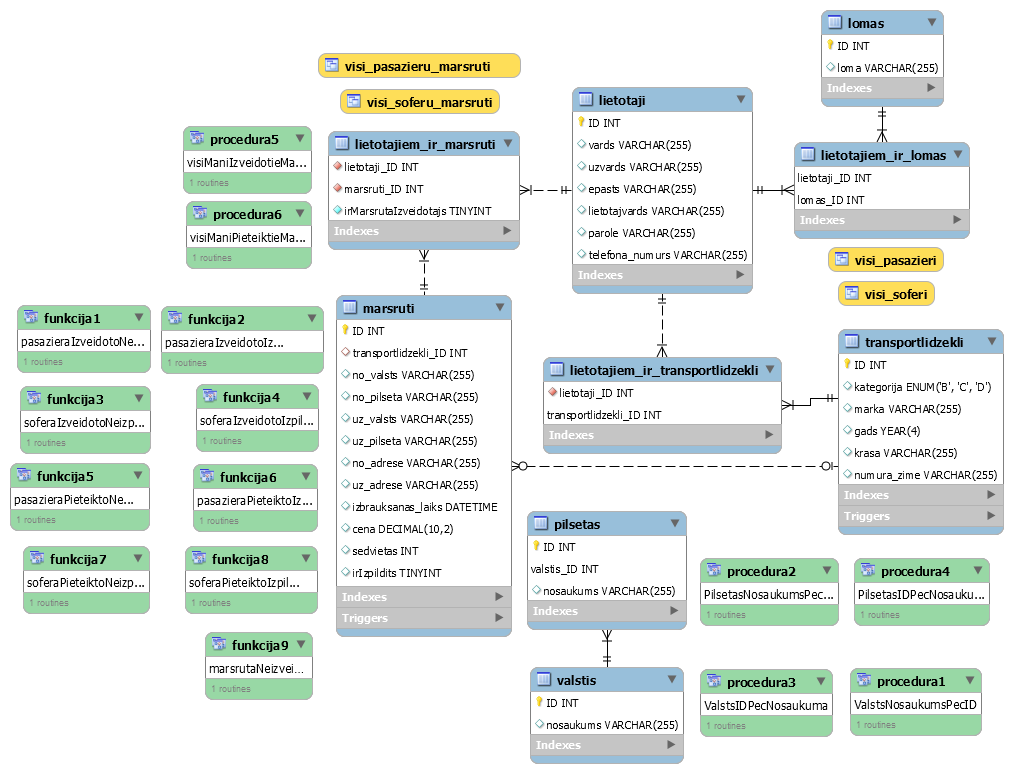
# **IEVADS**

Transporta pakalpojumu sistēma ir lietotne, kurā lietotājiem ir iespējams reģistrēties kā pasažierim vai kā šoferim. Tā ir paredzēta ātrai un vienkāršai personas un/vai kravas transportēšanai un ļauj lietotājiem izveidot pilsētu, starppilsētu un pat starpvalstu maršrutus, vispirms norādot sākuma punktu, gala punktu, datumu, laiku, vai nepieciešams pārvedāt kravu, papildus piezīmes un piedāvāto samaksu.

Pirmām kārtām lai lietotājs varētu reģistrēties kā šoferis, tam nepieciešams ievadīt savas automašīnas datus – numura zīme, automašīnas tips, marka, sēdvietu skaits, krāsa. Turklāt abām lietotāju pusēm ir jānorāda kontaktinformācija. Lietotnē gan pasažieru, gan arī šoferu lietotājiem ir iespējams izveidot maršrutu uz kuru var atsaukties pretējā partija. Abām lietotāju grupām ir redzams pretējās partijas maršrutu saraksts un, atsaucoties uz kādu no tiem, attiecīgās grupas lietotājs saņem apstiprināšanas pieprasījumu un var akceptēt vai noraidīt to. Jāatzīmē tas, ka maršrutā var piedalīties vairāki pasažieri. Kad abas puses ir vienojušās un maršruts ir izpildīts, tiek veikta samaksa un pasažieriem tiek lūgts sniegt atsauksmi par šoferi. Šoferim ir iespēja skatīt savu profilu, kurā redzama viņa maksājumu vēsture jeb gūtā peļņa kādā laika periodā.

Šī sistēma nodrošina abpusēju guvumu pasažierim un šoferim. Pasažieris nokļūst vietā, kur tam nepieciešams, šoferis gūst samaksu par sniegto pakalpojumu.

# **ERD DIAGRAMMA**

****

# **TABULAS AR DATIEM**

Tabula 3.1. ir lietotāju tabula kurā tiek uzglabāta pamatinformācija par reģistrēto lietotāju, tas ir, lietotāja ID, vārds, uzvārds, ē-pasts, lietotājvārds, telefona numurs un parole, kura, pirms ievietošanas datu bāzē, tiek aplikācijas līmenī, izmantojot PHP iebūvēto kriptogrāfijas funkciju, šifrēta.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| ID | vards | uzvards | epasts | lietotajvards | parole | telefona\_numurs |
| 1 | Ieva | ÅŖdrÄ“na | skenna0@usa.gov | dmartinot0 | 1111a3628d7dc3e15d7e5961e83022cf | 703-810-3816 |
| 2 | Zigurds | Zeimans | vverty1@phpbb.com | gdomek1 | 0cc175b9c0f1b6a831c399e269772661 | 956-958-6117 |
| 3 | Inita | Vellaine | rjeeves2@ibm.com | bwoodburn2 | 92eb5ffee6ae2fec3ad71c777531578f | 616-298-0151 |
| 4 | Ä€dams | BuiÄ·is | krainton3@sbwire.com | fpither3 | 4a8a08f09d37b73795649038408b5f33 | 427-870-5425 |
| 5 | Inga | LobaÅ†ova | aawmack4@infoseek.co.jp | aklemensiewicz4 | 8277e0910d750195b448797616e091ad | 301-770-3091 |
| 6 | TÄlrÄ«ts | Brahmanis | afarbrother5@nationalgeographic.com | kwynett5 | e1671797c52e15f763380b45e841ec32 | 158-893-1818 |
| 7 | Spulga | Aldare | rwollrauch6@nyu.edu | jhanhart6 | 8fa14cdd754f91cc6554c9e71929cce7 | 580-278-1680 |
| 8 | KonrÄds | MiÄ·elsons | edownes7@yolasite.com | fgianinotti7 | b2f5ff47436671b6e533d8dc3614845d | 981-173-3183 |
| 9 | Vladislava | Prole | adavinet8@fc2.com | atipling8 | 2510c39011c5be704182423e3a695e91 | 251-555-3305 |
| 10 | Ginters | LiekniÅ†Å | sscamal9@networksolutions.com | mhackinge9 | 865c0c0b4ab0e063e5caa3387c1a8741 | 855-926-7897 |

Tabula 3.1. lietotāji

|  |  |
| --- | --- |
| lietotaji\_ID | lomas\_ID |
| 5 | 2 |
| 4 | 2 |
| 3 | 2 |
| 2 | 2 |
| 1 | 2 |
| 10 | 1 |
| 9 | 1 |
| 8 | 1 |
| 7 | 1 |
| 6 | 1 |

Tabula 3.2. atzīmē visas iespējamās lietotāju lomas, turklāt, vienam lietotājam var arī eksistēt vairākas lomas, tādēļ, lietotāju tabula un lomu tabula (tabula 3.3.) tiek saistīta izmantojot saiti daudzi pret daudzi un starp tabulu, kurā pēc lietotāja ID un lomas ID tiek atdalītas konkrētā lietotāja lomas.

|  |  |
| --- | --- |
| ID | loma |
| 1 | pasazieris |
| 2 | soferis |
| 3 | administrators |

Tabula 3.3.

Tabula 3.2.

Tabula 3.4. satur visus šofera tipa lietotāju transportlīdzekļus, kuriem glabājamās vērtības ir ID, kategorija, marka, gads, krāsa, numurzīme. Lai nodrošinātu to, ka lietotāji var izmantot vairāk kā vienu transporta līdzekli kādā izveidotā maršrutā, nepieciešama starp tabula (tabula

|  |  |
| --- | --- |
| lietotaji\_ID | transportlidzekli\_ID |
| 1 | 1 |
| 1 | 2 |
| 2 | 3 |
| 3 | 4 |
| 4 | 5 |
| 5 | 6 |

3.5.), ar kuras palīdzību sasaista lietotāja ID ar transportlīdzekļa ID.

Tabula 3.5.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ID | kategorija | marka | gads | krasa | numura\_zime |
| 1 | B | Opel | 1995 | Sarkana | AA-1000 |
| 2 | B | Cadillac | 2009 | Zila | BB-2000 |
| 3 | B | Audi | 2001 | Zaļa | CC-3000 |
| 4 | C | Audi | 2002 | Sarkana | DD-4000 |
| 5 | C | Opel | 2005 | Melna | EE-5000 |
| 6 | B | BMW | 2009 | Melna | FF-6000 |

Tabula 3.4.

Tabula 3.6. paredzēta visu lietotāju gan pasažieru, gan šoferu maršrutu glabāšanai. Maršrutu tabula sastāv no ID, transportlīdzekļa ID, kurš var pieņemt vērtību NULL, gadījumā, ja maršrutu izveidojis pasažieris, no valsts, no pilsēta, uz valsts, uz pilsēta, no adrese, uz adrese, izbraukšanas laiks, cena, sēdvietas (pasažiera gadījumā nepieciešamās sēdvietas, šofera gadījumā pieejamās jeb brīvās sēdvietas) un, atzīmēšanas nolūkiem, ieraksts par to vai maršruta status ir izpildīts, vai nav.

Līdzīgi kā iepriekšējās tabulās, nepieciešams sasaistīt lietotāju ar maršrutu tāpēc izveidota starp tabulā 3.7., taču šoreiz papildus nepieciešams zināt, kurš lietotājs ir konkrētā maršruta izveidotājs, jo pretējā gadījumā, kad vienam maršrutam piesakās vairāki lietotāji, tiek zaudēts oriģinālais lietotājs, kurš sākotnēji izveidoja maršrutu.

|  |  |  |
| --- | --- | --- |
| lietotaji\_ID | marsruti\_ID | irMarsrutaIzveidotajs |
| 1 | 1 | 1 |
| 1 | 2 | 1 |
| 3 | 4 | 1 |
| 4 | 3 | 0 |
| 6 | 3 | 1 |
| 7 | 5 | 1 |
| 8 | 1 | 0 |
| 9 | 1 | 0 |

Tabula 3.7.

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| ID | transportlidzekli\_ID | no\_valsts | no\_pilseta | uz\_valsts | uz\_pilseta | no\_adrese | uz\_adrese | izbrauksanas\_laiks | cena | sedvietas | irIzpildits |
| 1 | 1 | Latvia | Liepaja | Latvia | Riga | Tirgus iela 19 | Brīvības iela 8 | 2018-12-16 17:00:00 | 6.99 | 2 | 1 |
| 2 | 1 | Latvia | Riga | Latvia | Liepaja | Brīvības iela 8 | Tirgus iela 19 | 2018-12-18 17:00:00 | 6.99 | 2 | 0 |
| 3 | NULL | Lithuania | Tallinn | Lithuania | Tartu | NULL | NULL | 2018-12-22 12:00:00 | 15.00 | 1 | 0 |
| 4 | 3 | Russia | Moscow | Latvia | Riga | Gazetny Ln., 17/9/1 | Brīvības iela 8 | 2018-12-25 06:00:00 | 45.99 | 3 | 0 |
| 5 | NULL | Poland | Kraków | Poland | Warsaw | NULL | NULL | 2019-01-15 07:00:00 | 3.99 | 1 | 0 |

Tabula 3.6.

Tabula 3.8. uzglabā datus ar visām valstīm (ID, nosaukums), kuras iespējams izvēlēties lietotājiem, kā arī tabula 3.9. atrodas visu pilsētu ID, nosaukums un papildus tiek glabāta valsts ID ārējā atslēga. Tā ļauj atrast visas pilsētas kādai konkrētai valstij.

|  |  |
| --- | --- |
| ID | nosaukums |
| 1 | Estonia |
| 2 | Latvia |
| 3 | Lithuania |
| 4 | Russia |
| 5 | Poland |

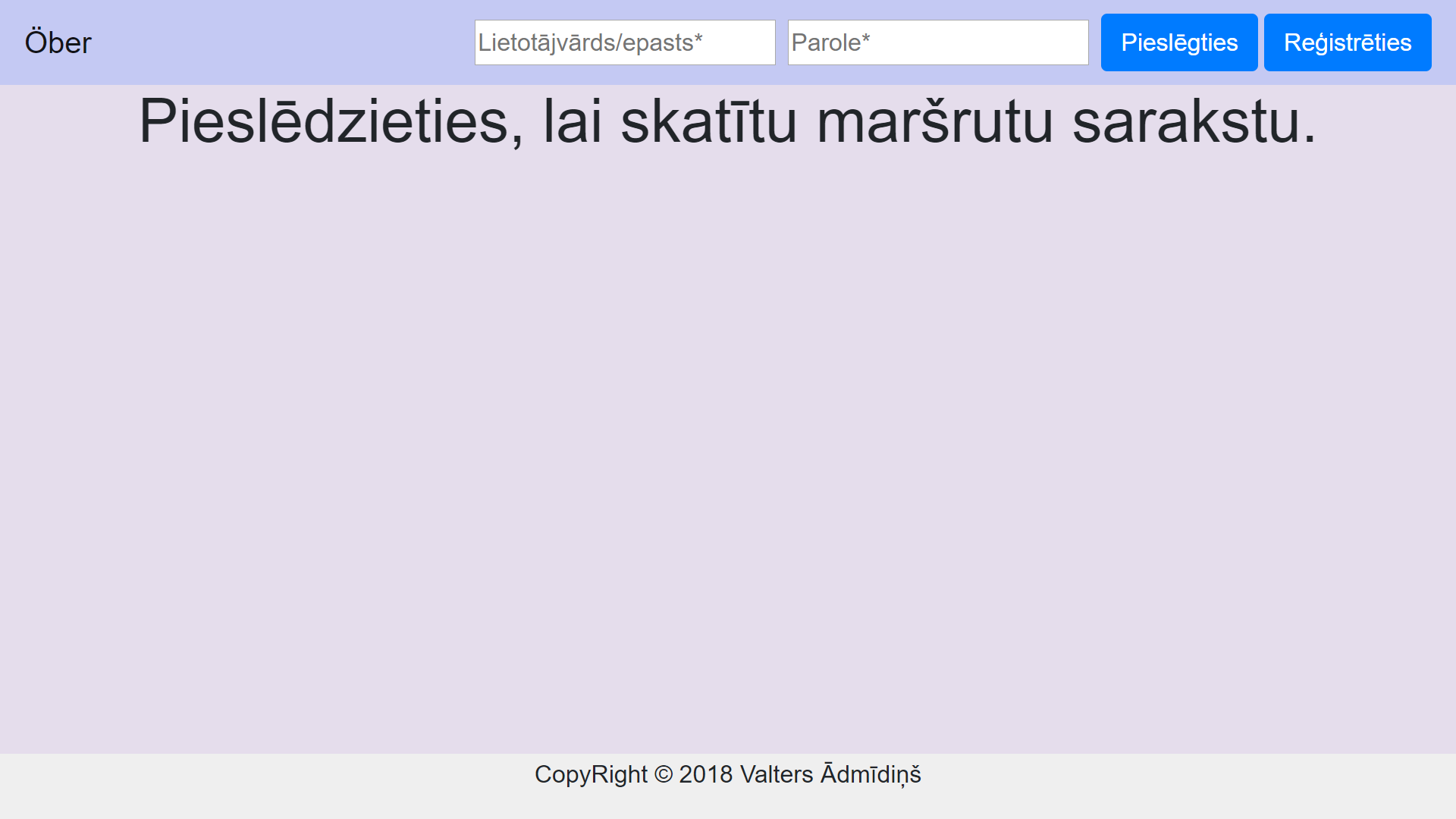
Tabula 3.8.

|  |  |  |
| --- | --- | --- |
| ID | valstis\_ID | nosaukums |
| 1 | 1 | Tallinn |
| 2 | 1 | Tartu |
| 3 | 2 | Ventspils |
| 4 | 2 | Riga |
| 5 | 2 | Liepaja |
| 6 | 3 | Vilnius |
| 7 | 3 | Kaunas |
| 8 | 4 | Moscow |
| 9 | 4 | Saint Petersburg |
| 10 | 4 | Novosibirsk |
| 11 | 5 | Warsaw |
| 12 | 5 | Kraków |
| 13 | 5 | Łódź |

Tabula 3.9.

# **APLIKĀCIJA**

1. Pirmo reizi atverot tīmekļa aplikāciju lietotājs tiek uzskatīts par viesi, jeb bez lomas, tādēļ tiek izvadīts paziņojums, par to, ka nepieciešams “Pieslēgties”, lai redzētu jebkādu informāciju par šoferu vai pasažieru maršrutiem.



1. Nospiežot pogu “Reģistrēties” lietotājam ir nepieciešams ievadīt datus par sevi, un izvēlēties vai viņš vēlas būt pasažiera vai šofera tipa lietotājs.

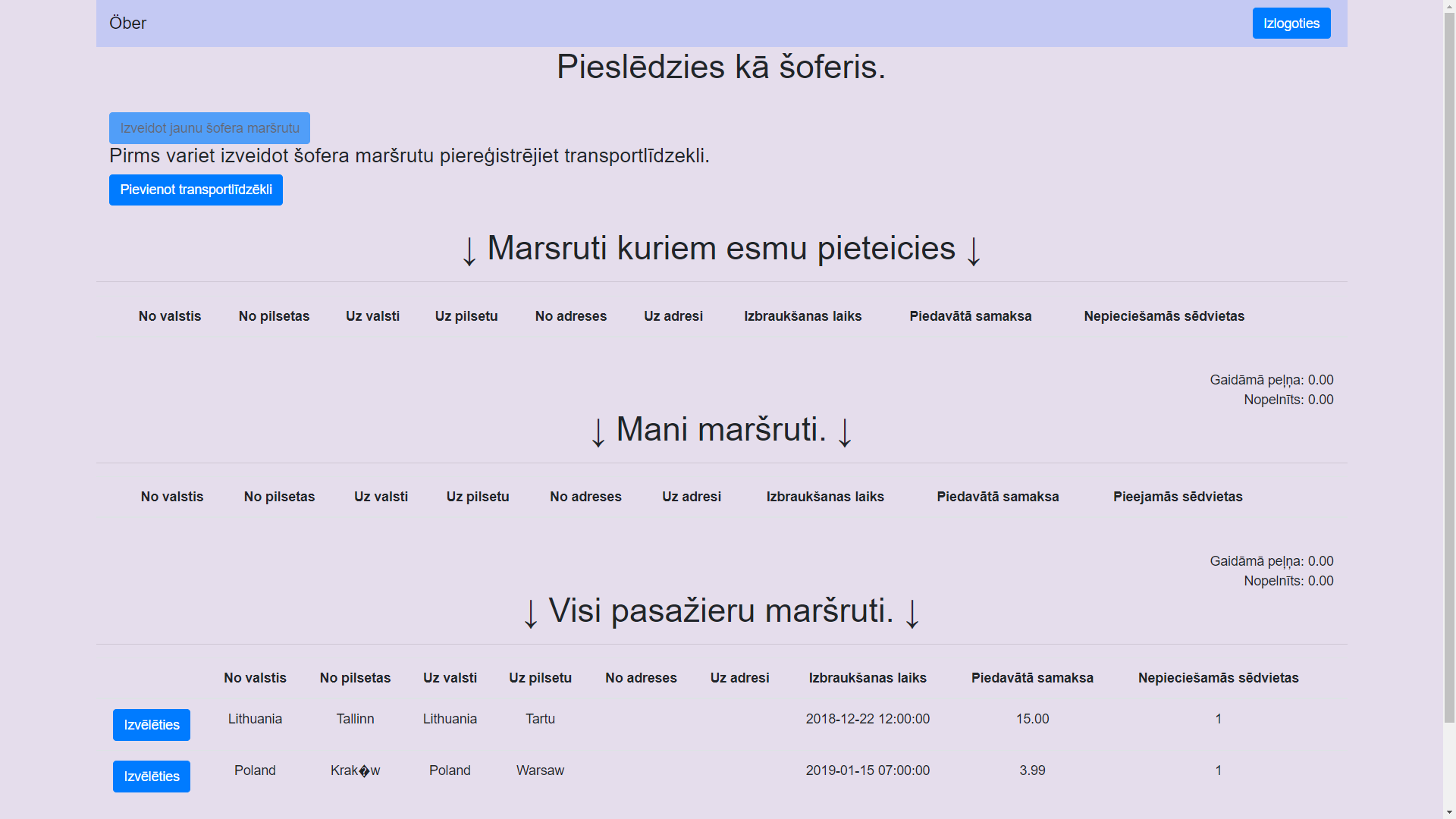


Pēc pogas “Reģistrēties!” piespiešanas lietotāja dati tiek ievadītu datu bāzes tabulā lietotāji.

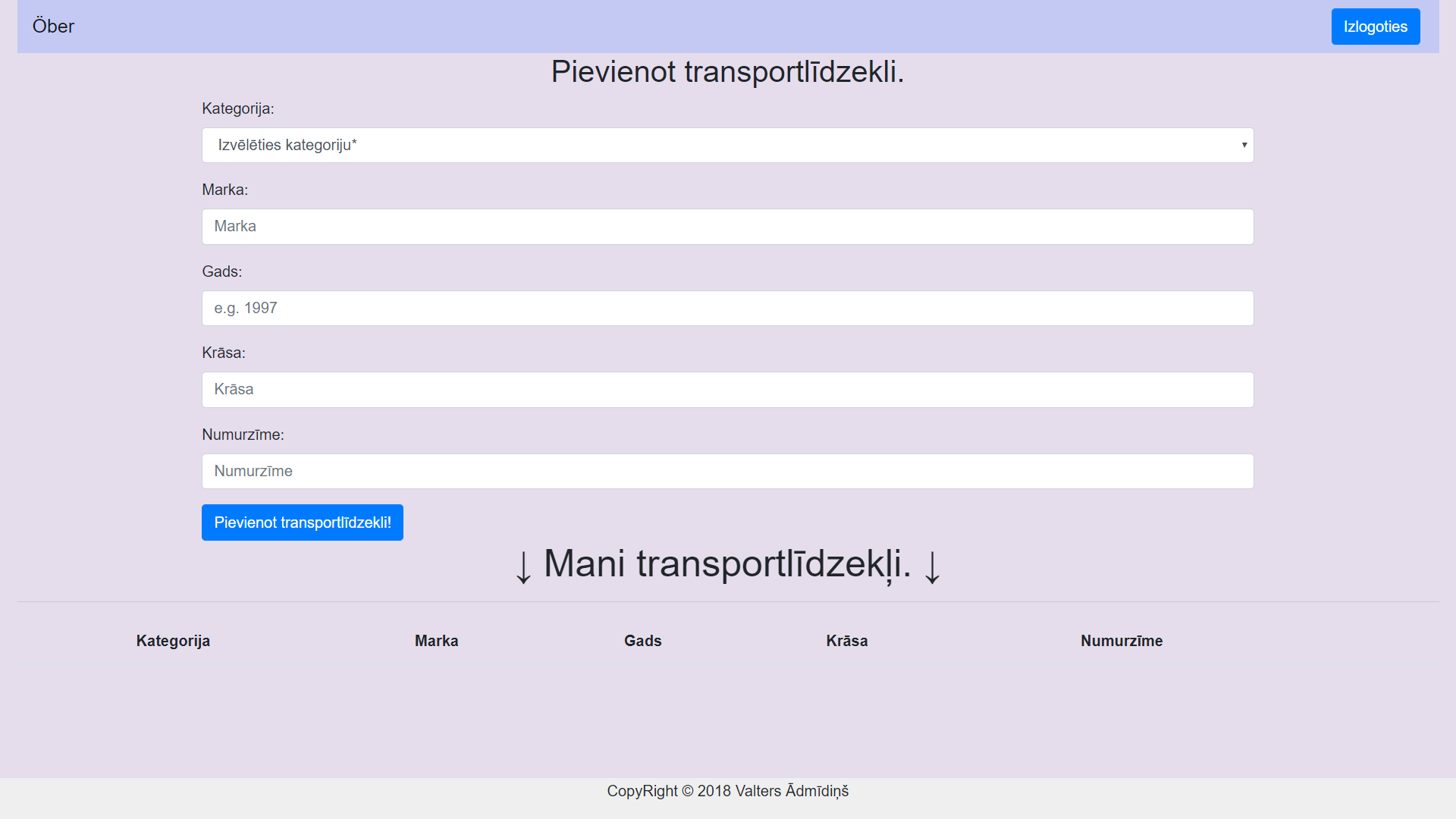
1. Šajā gadījumā lietotājs ir reģistrējies kā šoferis, un, pēc veiksmīgas pieslēgšanās, tiek novests pie galvenā loga, kurā ir apskatāmi visi maršruti kuriem lietotājs ir pieteicies kā šoferis un ir gatavs, pēc pasažiera lietotāja nosacījumiem, izpildīt izveidoto maršrutu.

Tālāk ir redzami visi pašreizējā lietotāja izveidotie maršruti, un visi pasažieru maršruti, kurus, piespiežot pogu “Izvēlēties” var skatīt un pieteikties tiem.

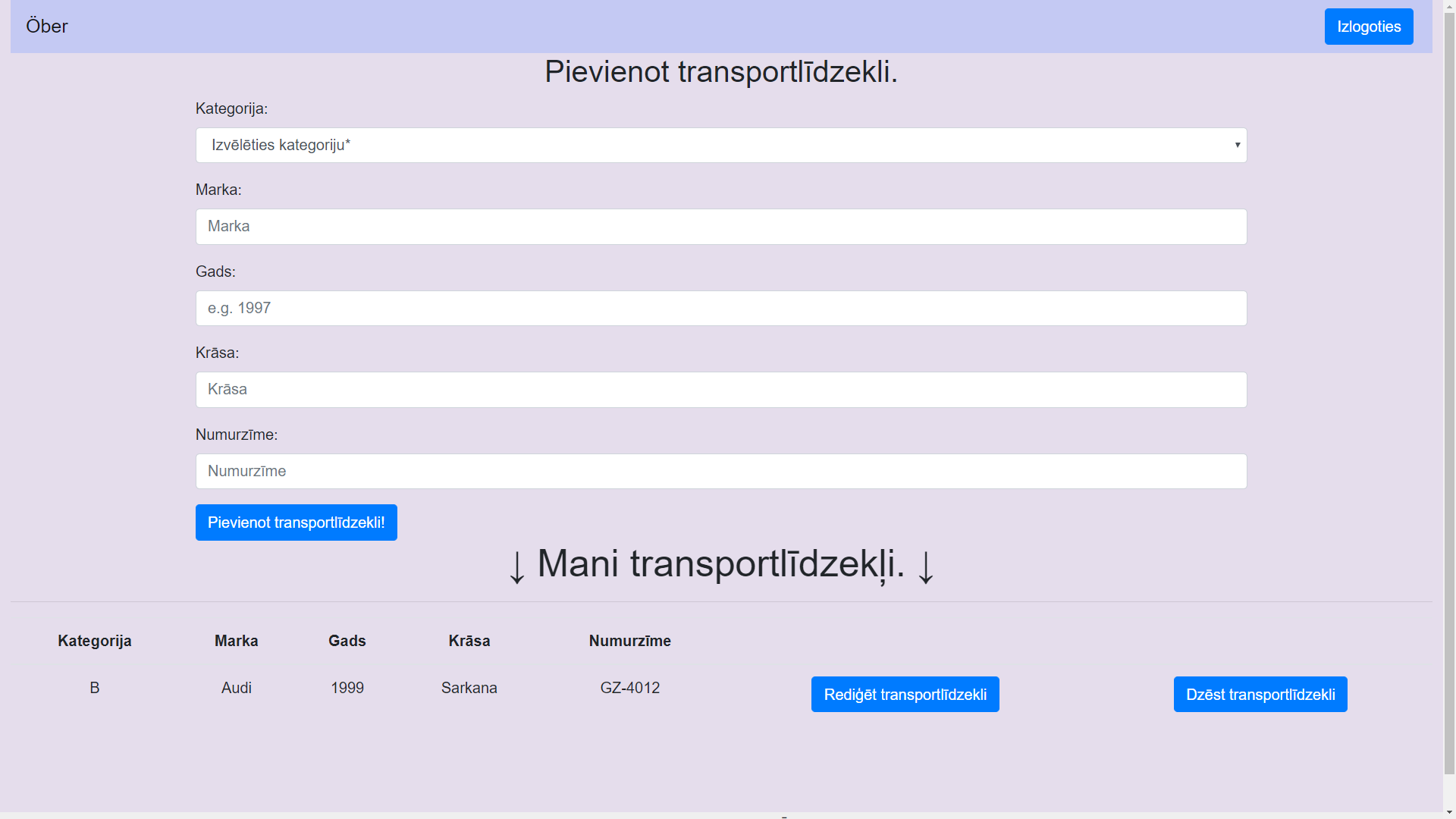
Kā šofera lietotājam maršrutu nav iespējams izveidot, pirms nav pievienots šofera lietotāja transportlīdzeklis.



1. Piespiežot pogu “Pievienot transportlīdzekli” lietotājam ir jānorāda sava transportlīdzekļa dati.



Pēc transporta līdzekļa pievienošanas parādās 2 iespējas “Rediģēt transportlīdzekli” un “Dzēst transportlīdzekli”.



1. Pēc transportlīdzekļa pievienošanas šofera lietotājam ir iespējams izveidot jaunu šofera maršrutu.



1. Kad maršruts izveidots tas tiek uzrādīts sadaļā “Mani maršruti”. Tiek uzrādīti visi attiecīgie, lietotāja ievietotie dati. Šofera lietotājam norādot “Piedāvāto samaksu” tiek aprēķināta viņa gaidāmā samaksa, ja visas maršruta sēdvietas tiks aizpildītas. Šajā gadījumā ja lietotājs norāda, ka vēlas saņemt 5.99 un maršruta pieejamo sēdvietu skaits ir 2, tad viņa gaidāmā peļņa ir 11.98.



1. Pieslēdzoties pasažierim, sadaļā “Visi šoferu maršruti” tam ir redzams nupat izveidotā lietotāja maršruts.



Uz maršruta nospiežot pogu “Izvēlēties”, lietotājam tiek atvērts maršruts ar visu tam svarīgo informāciju, kā arī ir iespēja pieteikties uz maršrutu.



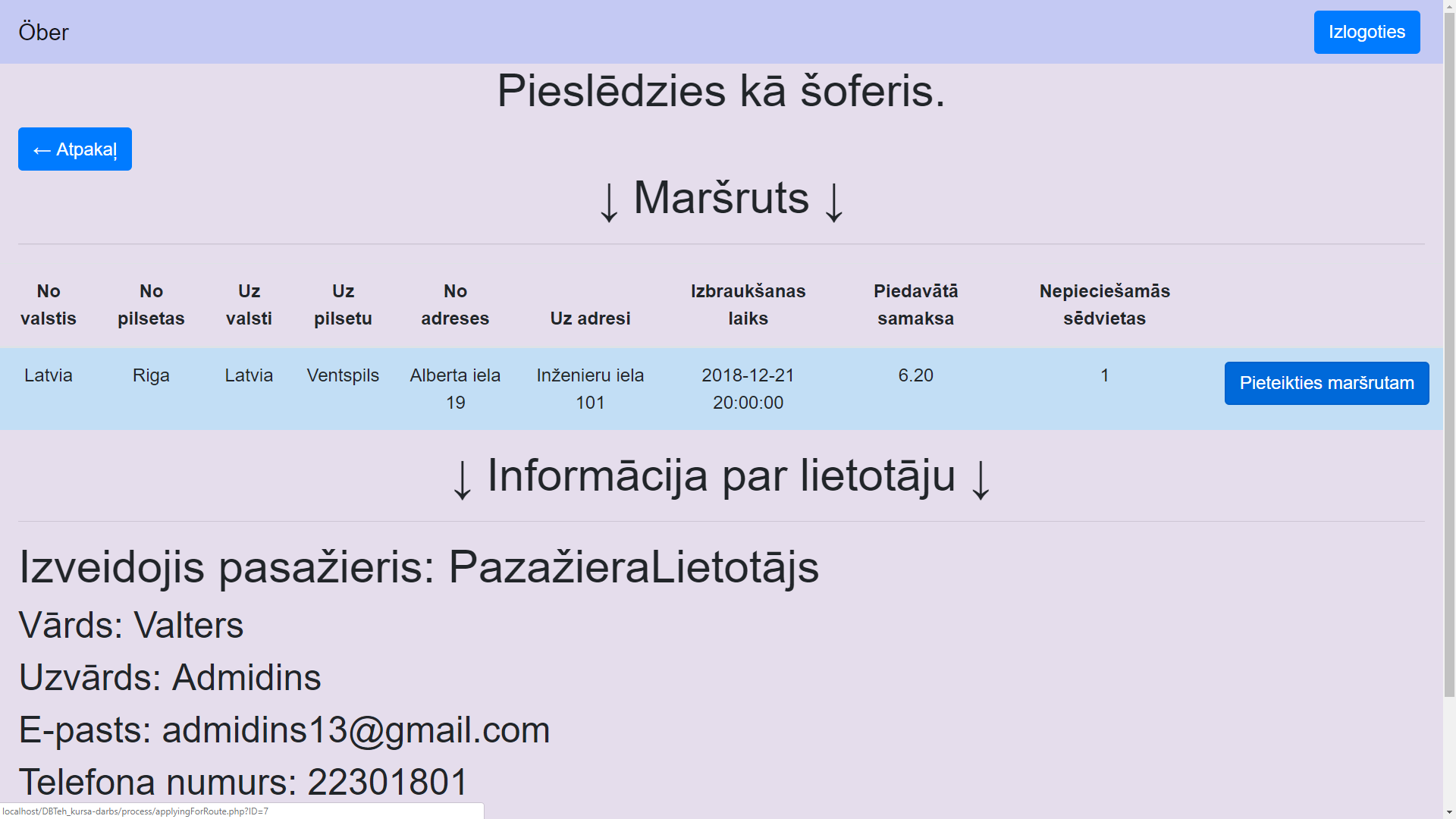
Piesakoties maršrutam, tagadējam lietotājam tiek atspējots pieteikties šim maršrutam atkārtoti. Arī tiek atjaunināts Pieejamo sēdvietu skaits. Šajā gadījumā lietotājs ir aizņēmis vienu sēdvietu, un ir atstāta viena brīva sēdvieta.



1. Pieslēdzoties atpakaļ šoferim, tam ir redzams, ka viņa maršrutam ir pieteicies lietotājs. Tālāk maršrutu atzīmējot kā izpildītu, tas kļūst neaktīvs un šofera nopelnītā nauda tiek atjaunināta atkarībā pēc tā, cik lietotāji kopā bija pieteikušies uz maršrutu.



1. Apgrieztais gadījums ir, kad šoferis var pieteikties uz kādu no pasažieru maršrutiem. Šeit redzams, ka pasažiera lietotājam ir nepieciešama 1 sēdvieta un viņš ir gatavs samaksāt šoferim 6.20.



1. Kad kāds šoferis ir pieteicies uz maršrutu, pasažiera lietotājam ir iespēja to atzīmēt kā izpildītu.





# **SKATI**

1. Kopā tiek pielietoti 2 skati ar kuru palīdzību tiek atlasīta konkrētā lietotāju grupa, un vēl 2 skati ar kuru palīdzību tiek atrasta tās konkrētās grupas visi maršruti.

Pasažieru skats:

DROP TABLE IF EXISTS `DBTeh-kursa\_darbs`.`visi\_pasazieri`;

USE `DBTeh-kursa\_darbs`;

CREATE OR REPLACE VIEW `visi\_pasazieri` AS

SELECT ID, vards, uzvards, epasts, lietotajvards, telefona\_numurs FROM `lietotaji`

JOIN `lietotajiem\_ir\_lomas` ON ID=lietotaji\_ID WHERE lomas\_ID=1;

Šoferu skats:

DROP TABLE IF EXISTS `DBTeh-kursa\_darbs`.`visi\_soferi`;

USE `DBTeh-kursa\_darbs`;

CREATE OR REPLACE VIEW `visi\_soferi` AS

SELECT ID, vards, uzvards, epasts, lietotajvards, telefona\_numurs FROM `lietotaji`

JOIN `lietotajiem\_ir\_lomas` ON ID=lietotaji\_ID WHERE lomas\_ID=2;

Visu pasažieru maršruti:

DROP TABLE IF EXISTS `DBTeh-kursa\_darbs`.`visi\_pasazieru\_marsruti`;

USE `DBTeh-kursa\_darbs`;

CREATE OR REPLACE VIEW `visi\_pasazieru\_marsruti` AS

SELECT lietotaji\_ID, marsruti\_ID, irMarsrtutaIzveidotajs FROM lietotajiem\_ir\_marsruti WHERE irMarsrutaIzveidotajs=1 AND lietotaji\_ID IN (SELECT lietotaji\_ID FROM visi\_pasazieri ORDER BY lietotaji\_ID DESC)

Visu šoferu maršruti:

DROP TABLE IF EXISTS `DBTeh-kursa\_darbs`.`visi\_soferu\_marsruti`;

USE `DBTeh-kursa\_darbs`;

CREATE OR REPLACE VIEW `visi\_soferu\_marsruti` AS

SELECT lietotaji\_ID, marsruti\_ID, irMarsrtutaIzveidotajs FROM lietotajiem\_ir\_marsruti WHERE irMarsrutaIzveidotajs=1 AND lietotaji\_ID IN (SELECT lietotaji\_ID FROM visi\_soferi ORDER BY lietotaji\_ID DESC);

Skatos visu\_pasažieru\_maršruti un visu\_šoferu\_maršruti tiek izmantoti skati visi\_pasažieri un visi\_šoferi, pielietojot IN izteikumu, lai atrastu visas sakrītošās vērtības no starp tabulas lietotajiem\_ir\_maršruti. Pielietojot šos skatus, ievērojami tiek samazināta tālāko MYSQL izsaukumu sarežģītība, un uzlabota veiktspēja.

# **PROCEDŪRAS**

1. Aplikācijas optimizēšanas nolūkiem tiek lietotas vairākas procedūras. Par cik pastāvīgi nepieciešams atgūt informāciju par patreizējā, ielogotā lietotāja maršrutiem, ir izveidots skats.

DELIMITER $$

USE `DBTeh-kursa\_darbs`$$

CREATE PROCEDURE visiManiIzveidotieMarsruti(lietotaja\_ID INT)

    BEGIN

    SELECT marsruti.ID, no\_valsts, no\_pilseta, uz\_valsts, uz\_pilseta, no\_adrese, uz\_adrese, izbrauksanas\_laiks, cena, sedvietas, irIzpildits FROM marsruti

    JOIN lietotajiem\_ir\_marsruti ON marsruti\_ID=ID

    WHERE lietotaji\_ID=lietotaja\_ID AND irMarsrutaIzveidotajs=1

    ORDER BY irIzpildits, Izbrauksanas\_laiks;

    END$$

DELIMITER ;

Līdzīgā veidā tiek pielietots skats, kas atrod visus patreizējā lietotāja maršrutus kuriem tas ir pieteicies.

DELIMITER $$

USE `DBTeh-kursa\_darbs`$$

CREATE PROCEDURE visiManiPieteiktieMarsruti(lietotaja\_ID INT)

    BEGIN

    SELECT marsruti.ID, no\_valsts, no\_pilseta, uz\_valsts, uz\_pilseta, no\_adrese, uz\_adrese, izbrauksanas\_laiks, cena, sedvietas, irIzpildits FROM marsruti

    JOIN lietotajiem\_ir\_marsruti ON marsruti\_ID=ID

    WHERE lietotaji\_ID=lietotaja\_ID AND irMarsrutaIzveidotajs=0

    ORDER BY irIzpildits, Izbrauksanas\_laiks;

    END$$

DELIMITER ;

1. Par cik aplikācijā tiek lietots AJAX Javascript, lai nodrošinātu to, ka maršruta izveidē, lietotājam izvēloties kādu valstu automātiski tiktu atlasītas visas tās konkrētās valstis pilsētas nepārlādējot interneta lapas skatu, bija nepieciešamas papildus procedūras. To mērķis ir atrast valsts un pilsētas nosaukumu pēc to ID un arī pretēji – valsts un pilsētas nosaukumu pēc ID. Šis process ir nepieciešams, lai ar AJAX palīdzību valstu un pilsētu nolaižamajos sarakstos varētu uzstatīt pareizos nosaukumus, par cik tiem ir vajadzīgs būt indeksētiem pēc ID. Pretējā gadījumā datu bāzē notiktu ļoti liels skaits salīdzinājumu ar varchar vērtībām, kas būtiski palēninātu datu bāzes darbību pie liela valstu un pilsētu skaita.

DELIMITER $$

USE `DBTeh-kursa\_darbs`$$

CREATE PROCEDURE ValstsNosaukumsPecID(valsts\_ID INT)

    BEGIN

    SELECT nosaukums FROM valstis WHERE ID=valsts\_ID ;

    END$$

DELIMITER ;

DELIMITER $$

USE `DBTeh-kursa\_darbs`$$

CREATE PROCEDURE PilsetasNosaukumsPecID(pilsetas\_ID INT)

    BEGIN

    SELECT nosaukums FROM pilsetas WHERE ID=pilsetas\_ID ;

    END$$

DELIMITER ;

DELIMITER $$

USE `DBTeh-kursa\_darbs`$$

CREATE PROCEDURE ValstsIDPecNosaukuma(valsts\_nosaukums INT)

    BEGIN

    SELECT ID FROM valstis WHERE nosaukums=valsts\_nosaukums;

    END$$

DELIMITER ;

DELIMITER $$

USE `DBTeh-kursa\_darbs`$$

CREATE PROCEDURE PilsetasIDPecNosaukuma(pilsetas\_nosaukums INT)

    BEGIN

    SELECT ID FROM pilsetas WHERE nosaukums=pilsetas\_nosaukums;

    END$$

DELIMITER ;

# **FUNKCIJAS**

1. Aprēķinu veikšanai nepieciešamas vairākas funkcijas, jo atkarībā no lietotāja veida (pasažieris vai šoferis), no tā vai maršruts ir paša lietotāja izveidots, vai nav, un no tā, vai šis maršruts ir izpildīts, vai nav, aprēķini ir atšķirīgi. Šim nolūkam kopā tiek lietotas 8 dažādas funkcijas un viena funkcija, kas darbojās kā palīgfunkcija, jo tā atlasa to lietotāju skaitu, kuri ir maršruta dalībnieki, bet ne maršruta izveidotāji. Palīgfunkcija nepieciešama, lai atskaitītu aizņemtās un brīvās sēdvietas šofera tipa maršrutiem.

DELIMITER $$

USE `DBTeh-kursa\_darbs`$$

CREATE FUNCTION pasazieraPieteiktoNeizpilditoMarsrutuSamaksa(lietotaja\_ID INT)

    RETURNS DECIMAL(10,2)

BEGIN

DECLARE kopsumma DECIMAL(10,2);

SELECT SUM(cena) FROM marsruti

    JOIN lietotajiem\_ir\_marsruti ON marsruti\_ID=ID

    WHERE lietotaji\_ID=lietotaja\_ID AND irIzpildits=0 AND irMarsrutaIzveidotajs=0 INTO kopsumma;

IF kopsumma > 0 THEN

        RETURN kopsumma;

    ELSE

        RETURN 0;

    END IF;

END$$

DELIMITER ;

DELIMITER $$

USE `DBTeh-kursa\_darbs`$$

CREATE FUNCTION soferaPieteiktoNeizpilditoMarsrutuSamaksa(lietotaja\_ID INT)

    RETURNS DECIMAL(10,2)

BEGIN

DECLARE kopsumma DECIMAL(10,2);

SELECT SUM(cena \* sedvietas) FROM marsruti

    JOIN lietotajiem\_ir\_marsruti ON marsruti\_ID=ID

    WHERE lietotaji\_ID=lietotaja\_ID AND irIzpildits=0 AND irMarsrutaIzveidotajs=0 INTO kopsumma;

IF kopsumma > 0 THEN

        RETURN kopsumma;

    ELSE

        RETURN 0;

    END IF;

END$$

DELIMITER ;

Tālāk tiek atrasti pārējo kombināciju varianti mainot vērtības pie irIzpildīts un irMaršrutaIzveidotājs, taču aprēķini pamatā ir līdzīgi. Atšķirības ir meklējamas soferaIzveidotoIzpilditoMarsrutuSamaksa funkcijā, jo tās gadijumā ir nepieciešams aprēķināt nopelnīto samaksu, un tiek atkārtoti atrasts patiesais pieteikto pasažieru skaits.

Palīgfunkcija:

DELIMITER $$

USE `DBTeh-kursa\_darbs`$$

CREATE FUNCTION marsrutaNeizveidotajiDalibnieki(marsruta\_ID INT)

    RETURNS INT

BEGIN

DECLARE dalibniekuSkaits INT;

SELECT COUNT(\*) FROM lietotajiem\_ir\_marsruti

    WHERE marsruti\_ID=marsruta\_ID AND irMarsrutaIzveidotajs=0 INTO dalibniekuSkaits;

RETURN dalibniekuSkaits;

END$$

DELIMITER ;

# **TRIGERI**

1. Trigeris pirms maršruta ievietošanas un arī rediģēšanas pārliecinās vai maršruta izbraukšanas laiks nav vecāks par patreizējo laiku. Tāpat ir ievietota pārbaude uz to, lai nebūtu iespējams izveidot ierakstu, kura samaksa ir mazāka par nulli. Tādos gadījumos tiek izvadīts kļūdas paziņojums, ka ierakstu nav bijis iespējams ievietot datu bāzē.

USE `DBTeh-kursa\_darbs`$$

CREATE DEFINER = CURRENT\_USER TRIGGER `DBTeh-kursa\_darbs`.`marsruti\_BEFORE\_INSERT` BEFORE INSERT ON `marsruti` FOR EACH ROW

BEGIN

    IF NEW.izbrauksanas\_laiks < NOW() THEN

        SIGNAL SQLSTATE '45001' SET MESSAGE\_TEXT = "Datums nedrīkst būt vecāks par šodienas datumu";

    END IF;

    IF NEW.cena < 0 THEN

        SIGNAL SQLSTATE '45001' SET MESSAGE\_TEXT = "Cena nedrīkst būt mazāka par 0";

    END IF;

END$$

1. Transporta līdzekļiem pastāv līdzīga tipa pārbaudes pie ievietošanas un rediģēšanas. Tiek pārbaudīts vai transportlīdzekļa gads ir intervālā no 1901 līdz patreizējam laikam.

DELIMITER $$

USE `DBTeh-kursa\_darbs`$$

CREATE DEFINER = CURRENT\_USER TRIGGER `DBTeh-kursa\_darbs`.`transportlidzekli\_BEFORE\_UPDATE` BEFORE UPDATE ON `transportlidzekli` FOR EACH ROW

BEGIN

    IF NEW.gads < 1901 THEN

        SIGNAL SQLSTATE '45001' SET MESSAGE\_TEXT = "Transportlīdzekļa gads nedrīkst būt mazāks par 1901";

    END IF;

IF NEW.gads > NOW() THEN

        SIGNAL SQLSTATE '45001' SET MESSAGE\_TEXT = "Transportlīdzekļa gads nedrīkst būt lielāks par tagadējo laiku";

    END IF;

END$$

1. Citur ierakstu dzēšanas gadījumos tiek CASCADE, lai nepieļautu ierakstus, kuriem nav nekādu saišu.

ON DELETE CASCADE

# **SECINĀJUMI**

Praktisko darbu uzsākot bija vēlme padarboties ar tīmekļa izstrādes tehnoloģijām, pamatā, Bootstrap v4.1.3, AJAX Javascript un JQuery, jo vēlējos iegūt iemaņas arī tīmekļa tehnoloģiju nozarē, tāpēc uzsāku praktiskā darba izveidi nedaudz ātrāk. Taču, kopumā, pavadīju vienlīdzīgi daudz laika arī datu bāzes izveidošanā.

No datu bāzu viedokļa, ir iegūtas vairākas atziņas, kas noteikti nākotnē palīdzēs uzsākt datu bāzes izveidi mazliet efektīvāk un izvairīties no kļūdām, jo darba sākumā nācās vairākkārt pārskatīt un mainīt datu bāzes saites un loģiku. Zināšanas par procedūrām, funkcijām un trigeriem noteikti optimizēs, paātrinās un atvieglos datu bāzes izstrādi tālākos projektos.

Galvenās grūtības praktiskā darba izstrādē bija saprast datu bāzu tabulu loģiku un kā tieši aplikācijas līmenī izpaudīsies nepieciešamie, atgūstamie dati un aprēķini, un līdz ar to ir lietas kurām vairs laika neatlika, kā, piemēram, šoferu atsauksmju ievietošanu un kravas maršrutu ievietošanu datu bāzē, bet galu galā esmu apmierināts ar paveikto.

Kopumā esmu guvis arī vairāk iemaņu PHP objektorientētā paņēmiena pielietošanā, MySQL Workbench diagrammu izveidē un automatizētā koda ģenerēšanā, kā arī nedrīkst aizmirst Git versiju kontroles rīku.