VILNIAUS UNIVERSITETAS
MATEMATIKOS IR INFORMATIKOS FAKULTETAS
INFORMACINIŲ SISTEMŲ INŽINERIJOS STUDIJŲ PROGRAMA

Elektromobilių optimalaus maršruto sudarymo informacinė sistema atsižvelgiant į oro ir kelio sąlygas

Optimal Route Planning Information System for Electric Vehicles Using Weather and Road Conditions

Baigiamasis bakalauro darbas

Atliko: Dovydas Marius Zapkus

VU el. p.: dovydas.zapkus@mif.stud.vu.lt

Vadovas: Docentė Asta Slotkienė

Vilnius
2023

TURINYS

SANTRAUKA

Elektromobilių populiarumui augant vis daugiau elektromobilių vairuotojų iškyla problemų sudaryti kelionės maršrutą, kuris atsižvelgtų į daugybę veiksnių, kurie veikia elektromobilių nuvažiuojamą atstumą. Darbo tikslas yra išsiaiškinti veiksnius, kurie įtakoja elektromobilių nuvažiuojamą atstumą ir sukurti informacinę sistemą, kuri atsižvelgtų į šiuos veiksnius ir sukurtų naudotojui optimalų maršrutą. Darbo metu išsiaiškinta, kad elektromobilių nuvažiuojamą atstumą įtakoja oro sąlygos, važiavimo greitis, elektromobilio baterijos būklė ir įkrova, krovimo įpročiai. Darbo metu sukurta informacinė sistema, kuri sudaro optimalų maršrutą elektromobilių vairuotojams pagal oro ir kelio sąlygas, kurios yra realiu laiku gaunamos iš atvirų duomenų šaltinių ir pateikiamos naudotojui.

Raktiniai žodžiai: elektromobiliai#1, įkrovimo#2, stotelė#3, informacinė#4, sistema#5, optimalus#6 maršrutas#7.

SUMMARY

With the increasing popularity of electric vehicles, more and more electric vehicle drivers are facing challenges in planning a travel route that takes into account numerous factors affecting the range of electric vehicles. The objective of this study is to identify the factors influencing the range of electric vehicles and develop an information system that considers these factors to create an optimal route for the user. The study has revealed that the range of electric vehicles is influenced by weather conditions, driving speed, the condition and charge level of the vehicle's battery, and charging habits. During the study, an information system was developed that generates an optimal route for electric vehicle drivers based on real-time weather and road conditions obtained from open data sources and presented to the user.

Keywords: ev#1, charging#2, station#3, information#4, system#5, optimal#6, route#7.

1. ĮVADAS

Išsivysčiusios šalys vis dažniau investuoja į ekologiškumą, žalią energiją bei išmetamųjų dujų mažinimą pasaulyje. Pagal Europos Sąjungos duomenis lengvosios transporto priemonės, tokios kaip lengvieji keleiviniai arba komerciniai automobiliai, išskiria apie 12% visų Europoje išmetamų CO_2 dujų [1]. Dėl to, valstybės vis dažniau savo gyventojams suteikia lengvatų bei paramų CO_2 transporto priemonių pakeitimui į elektra varomas transporto priemones arba į mažiau aplinką teršiantį transportą. Pavyzdžiui, Lietuvos Respublika suteikia 5 000 eurų kompensaciją juridiniams asmenims, kurie planuoja įsigyti iki 6 mėnesių senumo elektromobilį ir 2 500 eurų kompensaciją už elektromobilį iki 4 metų senumo [2]. Palyginus 2019 metų elektromobilių pardavimus Europos Sąjungoje ir 2022 metų duomenis, parduodamų elektromobilių skaičius išaugo 6.1 karto, t.y nuo 2.3 procentų iki 14.1 procentų [3]. Daugeliui elektromobilių vairuotojų iškyla problema planuojant maršrutus, nes elektromobilio veikimas yra sudėtingas ir elektromobilių nuvažiuojamas atstumas yra veikiamas tokių aplinkinių veiksnių kaip oro sąlygos, kelio reljefas, važiavimo greitis, elektromobilio būklė ir t.t [4, 5]. Visi minėti veiksniai tai yra tam tikri istoriniai ir esamo laiko duomenų rinkiniai, kurių panaudojimas realiu laiku, leistų optimizuoti sprendimų priimamumą elektromobilių vairuotojams. Vienas iš sprendimų, pasiūlyti naudotojui informacinę sistemą, kuri automatiškai pasiūlytų naudotojui tinkamiausią maršrutą elektromobiliui bei duotų tam tikrus patarimus pagal jo pasirinktus parametrus t.y. elektromobilio informaciją, vietovės taškus ir t.t.

Tikslas:

Pasiūlyti elektromobilių optimalaus maršruto sudarymo informacinę sistemą remiantis oro, kelio ir įkrovimo stotelių duomenimis.

Uždaviniai:

1. Išanalizuoti optimalaus elektromobilių vairavimo veiksnius.
2. Išanalizuoti optimalaus elektromobilių maršruto sudarymo duomenų šaltinius.
3. Suprojektuoti elektromobilių optimalaus maršruto sudarymo informacinę sistemą.
4. Sukurti elektromobilių optimalaus maršruto sudarymo informacinę sistemą.
5. Ištestuoti elektromobilių optimalaus maršruto sudarymo informacinę sistemą.

Bakalauro darbe optimalus maršrutas apibrėžiamas kaip pagal elektromobilio techninę specifikaciją (nuvažiuojamą atstumą su pilna baterijos įkrova) sudarytas maršrutas, leidžiantis nuvažiuoti nuo pradinio taško iki tikslo su neišsekusia baterija, atsižvelgiant į tuo momentu esančias oro ir kelio sąlygas.

Ataskaitos analitinėje dalyje aprašyti išanalizuoti optimalaus elektromobilių vairavimo veiksniai ir duomenų šaltiniai reikalingi šiems veiksniams įgyvendinti. Metodinėje dalyje pateikiama suprojektuota informacinės sistemos architektūra, pateikta komponentų bei sekos diagramomis. Taip pat, aprašomos papildomų duomenų bei maršruto gavimo diagramos. Įgyvendinimo dalyje pateikta kodo struktūra, informacinės sistemos testavimas ir vartotojo vadovas.

2. ANALITINĖ DALIS

2.1. Elektromobilių maršrutų sudarymo analizė

Elektromobilių naudotojams yra itin svarbu planuoti savo keliones atidžiai, kadangi šie automobiliai turi mažesnį nuvažiuojamą atstumą, lyginant su vidaus degimo varikliais, ir taip pat nėra gerai išplėtota elektromobilių įkrovimo stotelių infrastruktūra [5]. Todėl, kad elektromobilių savininkai galėtų pasiekti savo kelionės tikslą, jie turi pasirinkti tinkamą maršrutą, kuriame yra elektromobilių įkrovimo stotelės ir kuris neviršija elektromobilio nuvažiuojamo atstumo iki jos arba galutinio taško. Be to, elektromobilio nuvažiuojamas atstumas priklauso nuo tokių veiksnių kaip oro sąlygos, kelio reljefas, važiavimo greitis, elektromobilio baterijos būklė ir įkrova, pavarų dėžės tipas, krovimo įpročiai [4][5]. Todėl, norint nustatyti tikslų nuvažiuojamą atstumą ir tinkamą maršrutą, būtina atsižvelgti į šiuos veiksnius.

2.2. Maršruto skaičiavimo analizė

Elektromobilio maršrutas gali būti sudarytas iš kelių kriterijų. [4-5] Svarbiausi iš jų:

- Maršrutas turi būti sudarytas iš pradinio taško ir galutinio taško.
- Maršrutas gali turėti tarpinius taškus, stoteles.
- Maršrutas turi parodyti bendrą kelionės atstumą bei kelionės atstumą tarp tarpinių taškų.
- Maršrutas turi parodyti bendrą kelionės laiką bei kelionės laiką tarp tarpinių taškų.
- Maršrutas turi atsižvelgti į elektromobilio baterijos įkrovą ir pagal tai automatiškai apskaičiuoti elektromobilio nuvažiuojamą atstumą.
- Sudarytas maršrutas privalo turėti bent vieną elektromobilio įkrovimo stotelę mažesniu atstumu nei pasirinkto elektromobilio nuvažiuojamas atstumas.
- Sudarytas maršrutas turi atsižvelgti į oro sąlygas.
- Sudarytas maršrutas turi atsižvelgti į kelio sąlygas.

Daugelis paminėtų kriterijų yra išsprendžiami naudojantis prieiga prie trečiųjų šalių informacinių sistemų kaip pavyzdžiui Google Maps, OpenStreetMap ir t.t. Tačiau šios informacinės sistemos negali suteikti tokių funkcijų kaip atsižvelgimas į elektromobilio baterijos įkrovą, oro sąlygas, kelio sąlygas ir t.t. Taigi, sudarant maršrutą reikia atlikti papildomus skaičiavimus, kurie papildytų išsikeltus kriterijus.

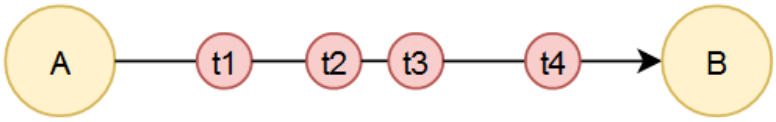
2.2.1. Maršruto kelio sudarymo analizė

Visų pirma, norint suprasti kokius skaičiavimus atlikti, reikia išanalizuoti maršruto sudarymą. Maršrutą galime suprasti kaip atvaizduotą kelią nuo taško A iki taško B.

2.1 pav. Maršrutas, kuris vaizduoja kelionę nuo taško A iki B, sudaryta autoriaus pagal šaltinius [6,7]

Šis maršruto sudarymo būdas (žr. *2.1* pav.) yra supaprastintas ir nesuteikia daug informacijos apie tarpinę kelionės informaciją. Tokiu būdu galime gauti informaciją apie tašką *A, B* ir bendrą kelionės atstumą bei tikslą, tačiau negaunama informacijos apie tarpines stoteles ir negalime pritaikyti tokių kriterijų kaip oro, kelio sąlygos.

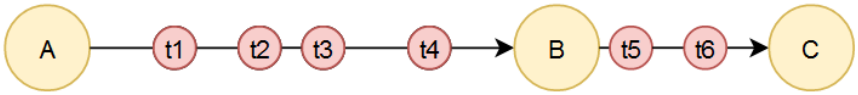
Sekantis maršruto sudarymo būdas yra *2.2* pav. atvaizduotas kelias tarp taško *A* iki taško *B*.



2.2 pav. Maršrutas, kuris vaizduoja kelionę nuo taško A iki B su tarpiniais taškais, sudarytas autoriaus pagal šaltinius [6,7]

Šis maršruto sudarymo būdas yra sudėtingesnis ir suteikia daugiau informacijos. Šiuo atveju, taškas *A* yra pradinis kelionės taškas, o taškas *B* galutinis taškas, *tn* yra tarpiniai taškai, kur susikerta keliai t.y. sankryžos, žiedai ir t.t. Tokių būdu mes galime gauti informacijos apie kiekvieną iš kelių nuo taško *A* iki taško *B* ir gauti meteorologinius duomenis iš artimiausiai keliui esančios meteorologinės stoties t.y. informaciją apie oro sąlygas. Taigi, šis maršruto sudarymo būdas suteikia daugiau informacijos, tačiau jame nėra tarpinių stotelių.

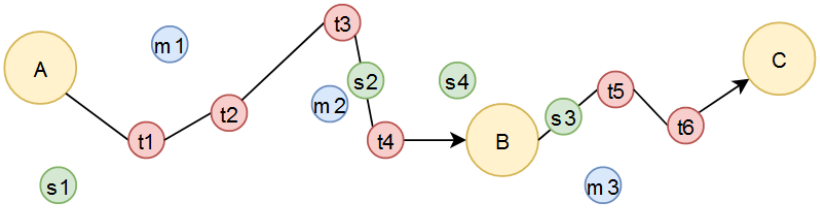
Žemiau (žr. *2.3* pav.) atvaizduojamas maršruto sudarymo būdas su tarpinėmis stotelėmis.



2.3 pav. Maršrutas, kuris vaizduoja kelionę nuo taško A iki C su tarpine stotele B ir tarpiniais taškais, sudarytas autoriaus pagal šaltinius [6,7]

Šiuo atveju, gali būti išpildyti visi aukščiau paminėti kriterijai. Iš tarpinių taškų *tn* galime gauti informaciją apie kelią bei kelio oro sąlygas, yra galimybė sudaryti maršrutą su tarpinėmis stotelėmis kaip šiuo atveju taškas *B*, gauti informacijos apie tarpinį bei bendrą kelionės atstumą ir laiką. Taigi, *2.3* pav. maršruto sudarymo principas tenkina aukščiau išsikeltus kriterijus.

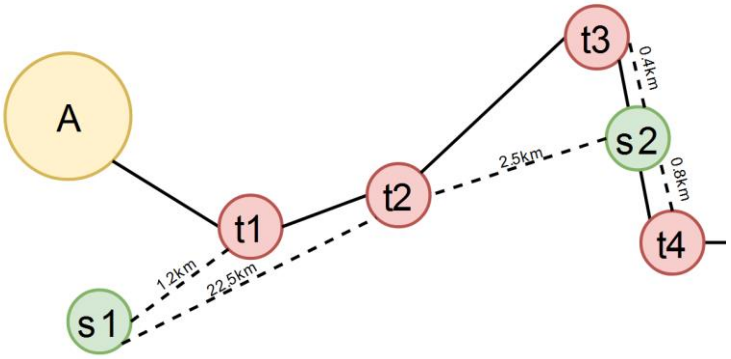
Žemiau pateiktas *2.4* pav., kuriame pateikiamas maršruto sudarymo pavyzdys, naudojasi elektromobilių įkrovimo stotelių, meteorologinių stočių bei kelio sąlygų duomenimis. Taškas *A* yra pradinis maršruto taškas, taškas *B* tarpinė stotelė, taškas *C* galutinis taškas, *sn* yra elektromobilių įkrovimo stotelių taškai, *tn* yra tarpiniai taškai (žr. *2.2* pav.), *mn* yra meteorologinių stočių taškai.



2.4 pav. Maršruto realizacija su elektromobilių įkrovimo stotelėmis ir meteorologijos stotimis, sudarytas autoriaus.

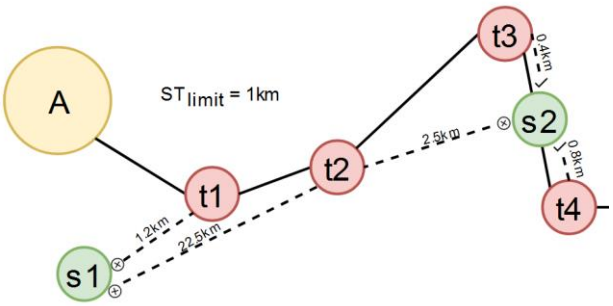
Artimiausių elektromobilių stotelių paieška

Norėdami surasti maršrutui artimiausias elektromobilių įkrovimo stoteles, galime naudotis Haversino atstumo formule [8-11], į ją įrašydami tarpinių taškų ir elektromobilio įkrovimo taškų koordinates. Tokiu būdu galime gauti atstumą tarp tarpinių taškų ir elektromoblių įkrovimo stotelių taškų.



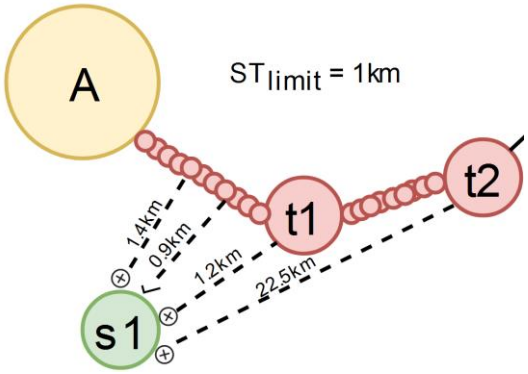
2.5 pav. Stotelių paieška naudojantis Haversino atstumu sudaryta autoriaus pagal šaltinį [12]

Išanalizavus aukščiau pateiktą iliustraciją (žr. 2.5 pav.) reikėtų atsižvelti į tai, kad Haversino formulės naudojimas atstumui tarp elektromobilių stotelių ir tarpinių taškų apskaičiuoti parodo tik artimiausią atstumą iki taško ir neatsižvelgia į kelių ilgio atstumą, dėl šios priežasties, rezultatas gali būti netikslus, dėl to, Haversino formulės rezultatą galima naudoti, tik kaip indikatorių artimiausioms elektromobilių įkrovimo stotelėms nustatyti, tačiau ne kaip tikram atstumui iki jų [10]. Taip pat, reikėtų nustatyti limitą *STlimit* (žr. 2.6 pav.), kuris nurodytų, koks atstumas nuo tarpinių taškų iki elektromobilių įkrovimo stotelių yra leidžiamas, kad į maršruto rezultatą būtų įtrauktos tik stotelės, kurios yra apibrėžtu atstumu nuo tarpinių taškų.



2.6 pav. Stotelių paieška naudojantis Haversino atstumu su ST_{limit} reikšmės pritaikymu.

Kitas sprendimas yra suskaidyti kelią tarp tarpinių taškų į smulkesnius taškus (žr. 2.7 pav.) Tokiu būdu atstumas skaičiuojamas ne tik nuo tarpinių taškų, tačiau ir taškų tarp jų. Šis sprendimas yra tikslesnis nei aukščiau esantis sprendimas, tačiau taip pat turi trūkumų, nes atstumas neatsižvelgia į kelią iki artimiausios elektromobilių stotelės ir paskaičiuoja tik artimiausią atstumą.



2.7 pav. Stotelių paieška naudojantis Haversino atstumu su kelio išskaidymu

Aukščiau pavaizduotas sprendimas yra tiksliausias iš visų siūlytų sprendimų, kitas sprendimas būtų gauti artimiausio kelio informaciją iki elektromobilio įkrovimo stotelės nuo tarpinės stotelės. Tokiu būdu būtų gautas tiksliausias atstumas iki įkrovimo stotelės, tačiau šis sprendimas reikalauja daug resursų, nes kiekvienai tarpinei stotelei reikia rasti kelią bei gauti duomenis apie jį iki kiekvienos įkrovimo stotelės, dėl to pasirinktas paskutinis sprendimas (žr. 2.7 pav.).

2.3. Elektromobilių įkrovimo stotelių ir maršruto sudarymo programų analizė

Žemiau aprašomos išanalizuotos elektromobilių maršruto sudarymo informacinės sistemos. Sistemos analizuojamos pagal tokius kriterijus kaip maršrutą (pvz. Vilnius – Klaipėda), duomenų kiekį, duomenų išsamumą, nuvažiuojamo atstumo skaičiavimo principą.

PlugShare

PlugShare yra informacinė sistema, kuri teikia paslaugas elektromobilių naudotojams. Informacinėje sistemoje galima peržiūrėti naudotojų pridėtas įkrovimo stoteles bei informaciją apie jas. PlugShare informacinėje sistemoje yra galimybė sudaryti maršrutą, tačiau sudarytas maršrutas neatsižvelgia į oro bei kelio sąlygas. Maršrutą sudaro atstumas bei laikas tarp atskirų maršruto taškų, bendras atsutmas bei laikas, šalia maršruto esančios įkrovimo stotelės. Įkrovimo stotelės aprašomos labai detaliai, t.y. stotelės pavadinimas, adresas, adapterio tipai, galia, naudotojų pastabos ir t.t.

EVTripPlanner

EVTripPlanner yra informacinė sistema, kuri leidžia sudaryti maršrutą tarp kelių taškų. Ši sistema neturi didelės įkrovimo stotelių duomenų bazės, t.y. maršrutas Vilnius-Klaipėda, parodė tik 1 įkrovimo stotelę.

ABetterRoutePlanner

ABetterRoutePlanner informacinė sistema leidžia sudaryti maršrutą tarp kelių taškų nurodžius pasirinktą elektromobilį. Pagal pasirinktą elektromobilį yra automatiškai gaunamas nuvažiuojamas atstumas. Tačiau nėra atsižvelgiama į oro bei kelio sąlygas. Taip pat, nėra galimybės nurodyti automobilio duomenis, jei jo nėra duomenų bazėje. Pateikiama informacija apie įkrovimo stoteles nėra detali. Yra atvaizduojamas tik elektromobilio įkrovimo stotelės pavadinimas bei adapterio tipas.

Elektromobilių maršruto sudarymo informacinės sistemų palyginimas

2.1 lentelė. Maršruto sudarymo IS palyginimas pagal stotelių skaičių ir elektromobilių modelių skaičių.

	Duomenų kiekis	
	Įkrovimo stotelių skaičius Lietuvoje	**Elektromobilių modelių skaičius**
PlugShare	Virš 250	Visi modeliai iki 2024 m.
EVTripPlanner	1	33
ABetterRoutePlanner	Virš 250	Visi modeliai iki 2024 m. Nuolatos atnaujinama

2.2 lentelė. Maršruto sudarymo IS palyginimas atsižvelgiant į oro, kelio sąlygas, elektromobilio specifikacijas.

	Atsižvelgiama į oro sąlygas	**Atsižvelgiama į baterijos įkrovą**	**Nuvažiuojamas atstumas gaunamas pagal automobilio modelį**	**Galimybė išvengti mokamų kelių, greitkelių ir keltų**
PlugShare			+	+
EVTripPlanner		+	+	
ABetterRoutePlanner	+ Už 5 € per mėnesį	+	+	

2.3 lentelė. Maršruto sudarymo IS palyginimas pagal tarpinių stotelių informaciją, stotelių užimtumo funkcionalumą ir pastabas

	Tarpinis maršruto laikas	**Tarpinis maršruto atstumas**	**Tiesioginiai įkrovimo stotelių užimtumas ir užimtumo prognozės**	**Pastabos**
PlugShare	+	+		
EVTripPlanner				
ABetterRoutePlanner	+	+	+ Už 5 € per mėnesį	Tiesioginis užimtumas gali būti netikslus

Išanalizavus sistemas, PlugShare ir ABetterRoutePlanner turi daugiau nei 250 įkrovimo stotelių Lietuvoje, o EVTripPlanner turi tik vieną stotelę. Be to, PlugShare ir ABetterRoutePlanner siūlo visus elektromobilių modelius, kurie yra sukurti iki 2023 metų, o EVTripPlanner siūlo tik 33 modelius. Į oro, kelio sąlygas, baterijos įkrovą ir nuvažiuojamą atstumą atsižvelgia, tik viena platforma – ABetterRoutePlanner, tačiau ši funkcija yra mokama. Taip pat tik viena informacinė sistema t.y. PlugShare, leidžia vairuotojams išvengti mokamų kelių, greitkelių ir keltų.

2.4. Informacinės sistemos sprendimo technologijų analizė

Informacinėje sistemoje naudojamos technologijos, kurios padeda įgyvendinti elektromobilių įkrovimo stotelių informacinę sistemą. Iš šių technologijų gauti duomenys leidžia pasiūlyti naudotojui tinkamiausią maršrutą, pagal naudotojo nurodytus vietovės taškus ir pasirinktus kriterijus, tokius kaip elektromobilio modelis, baterijos įkrova, nuvažiuojamas atstumas, ar automatiškai maršrutui pritaikytus kriterijus, tokius kaip oro sąlygos, greitis, kelyje susidariusios kliūtys ir t.t.

2.4.1. Informacinės sistemos žemėlapių sistema

Elektromobilių įkrovimo stotelių informacinėje sistemoje turi būti integruota žemėlapių sistema. Svarbiausi žemėlapio sistemos kriterijai informacinei sistemai:

- Įkrovimo stotelių atvaizdavimas bei interaktyvumas žemėlapyje.
- Maršruto atvaizdavimas žemėlapyje.
- Vietovių paieška ir atvaizdavimas žemėlapyje.
- Galimybė sudaryti maršrutą su tarpinėmis stotelėmis.
- Žemėlapių sistema turi išsamią dokumentaciją.
- Žemėlapių sistemos naudojimosi kaštai.

Google Maps

Google žemėlapiai plačiausiai pasaulyje naudojamas žemėlapių sistemos sprendimas. Google žemėlapiai suteikia prieigą prie API, kuriuo galima naudotis norint pridėti papildomą funkcionalumą žemėlapio komponentui. Google žemėlapiai turi nuolatos atnaujinamą ir koreguojamą dokumentaciją, dėl to yra patogūs naudoti ir nėra sudėtinga integruoti į aplikaciją. Google Maps API siūlo tokius funkcionalumus, kaip geokodavimas, maršrutų planavimas, gatvės vaizdai ir kt.

Google Maps API, turi išsamią vietovių duomenų bazę, kuri yra nuolat atnaujinama, užtikrinant, kad programuotojai turėtų prieigą prie naujausios informacijos. Be to, Google Maps API suteikia patikimą geokodavimo paslaugą, kuri leidžia programuotojams konvertuoti adresus į platumos ir ilgumos koordinates.

Taip pat, Google maps API turi patogią vartotojo sąsają, dėl to programuotojams yra nesudėtinga su ja dirbti. Programuotojai gali pasirinkti įvairius žemėlapių, žymeklių, sluoksnių tipus, kad atitiktų jų specifinius poreikius, reikalavimus. Be to, Google Maps API suteikia stiliaus parinkčių, kad būtų galima pakeisti žemėlapio išvaizdą.

Google Maps API sistema yra nemokama, jei per mėnesį nėra viršijama 28 500 užklausų. Už papildomas 1 000 užklausų mokama 7 JAV dolerius.

OpenStreetMap

OpenStreetMap atviro kodo, žemėlapių teikėjas, kuri yra populiarus tarp atviro kodo kūrėjų. OSM duomenys yra nemokami ir prieinami visiems, kad jie galėtų būti naudojami bet kokiame

projekte. Tačiau palyginti su Google Maps, OSM duomenų bazė yra mažesnė, o geokodavimo ir maršrutų planavimo funkcijos yra ribotos. OpenStreetMap gali atvaizduoti maršrutus, tačiau negali atvaizduoti maršruto su keliomis sustojimo vietomis, šiam uždaviniui įgyvendinti reikia diegti papildomus papildinius.

OpenStreetMap naudojimasis ir integravimas į aplikaciją yra nemokamas asmeniniais ir/ar komerciniais tikslais [13].

Mapbox

Mapbox yra mokamas žemėlapių teikėjas, kuri teikia įvairias funkcijas, įskaitant geokodavimą, maršrutų planavimą ir žemėlapio išvaizdos pritaikymą. Mapbox žemėlapių sistema kūrėjams suteikia daugiau funkcionalumo nei Google Maps API, tačiau duomenų bazė yra mažesnė už Google Maps.

Mapbox yra pigesnė sistema nei Google Maps API, ypač jei reikia didesnio kiekio užklausų arba didesnio lankytojų srauto. Mapbox žemėlapis yra nemokamas iki 50 000 užklausų. Už kiekvieną papildomą 1 000 užklausų mokama 5 JAV dolerius.

Žemėlapių sistemų palyginimas

Žemiau pateikta lentelė su skyriaus pradžioje išsikeltais kriterijais. Jei sistema tenkina kriterijų, lentelės langelis pažymimas „+“, jei netenkina yra paliekamas tuščias langelis, išskyrus, langelyje „Naudojimosi kaštai“, nurodomos skaitmeninės reikšmės.

2.4 lentelė. IS žemėlapių sistemos palyginimas pagal dokumentacijos išsamumą, naudojimosi kaštus ir funkcionalumą.

	Galimybė sudaryti maršrutą su tarpinėmis stotelėmis	**Išsami dokumentacija**	**Naudojimosi kaštai**	**Pastabos**
Google Maps	+	+	7 $ už 1000 užklausų	Iki 28500 užklausų nemokama
OpenStreetMap		+	Nemokamas	
Mapbox	+	+	5 $ už 1000 užklausų	Iki 50000 užklausų nemokama

Iš žemėlapių IS palyginimo (žr. 2.4 lentelė) galime pastebėti, kad mokami sprendimai tenkina daugiau kriterijų, tačiau visi paminėti sprendimai turėjo tokius kriterijus kaip: objektų atvaizdavimas, maršruto atvaizdavimas, vietovių paieška bei išsami dokumentacija.

2.4.2. Informacinės sistemos elektromobilių įkrovimo stotelių informacija

Elektromobilių įkrovimo stotelių informacinė sistema turi talpinti įkrovimo stotelių duomenis, kurie aplikacijoje būtų prieinami naudotojui. Svarbiausi kriterijai renkantis elektromobilių įkrovimo stotelių duomenų šaltinius:

- Įkrovimo stotelės duomenų išsamumas.
- Įkrovimo stotelės duomenų aktualumas.
- Įkrovimo stotelės duomenų kiekis.

Elektromobilių įkrovimo stotelėms gauti reikės naudotis įšoriniais API, kurie prižiūri ir talpina duomenis apie įkrovimo stotelių informaciją.

Lietuvos automobilių kelių direkcija. Elektromobilių įkrovos stotelės

Lietuvos automobilių kelių direkcijos atviri duomenys apie šalyje esančias elektromobilių įkrovimo stoteles.

Duomenų šaltinio duomenys labai išsamūs. Duomenyse yra tokia informacija kaip stotelei suteiktas identifikacinis kodas, vietos koordinatės, už stotelę atsakingų įmonių/asmenų kontaktai, stotelės įkrovimo adapterių tipai, stotelių įkrovimo adapterių užimtumas realiu laiku, įkrovimo kaina (eur/kWh).

Duomenų šaltinis nuolatos prižiūrimas ir atnaujinamas Lietuvos automobilių kelių direkcijos. Taip pat, yra galimybė tiesiogiai peržiūrėti įkrovimo stotelės užimtumą ir integruoti į aplikaciją, tačiau iš 275 įkrovos stoteles tiesioginius duomenis galima gauti tik iš 25 t.y. tik iš dešimtadalio visų elektromoblių stotelių.

Įkrovimo stotelių duomenų šaltinis iš viso turi 275 įkrovos stoteles.

PlugShare

PlugShare internetinė platforma talpina visame pasaulyje esančių elektromobilių įkrovimo stotelių duomenis.

Pateikiama tokia informacija, kaip įkrovimo stotelės adresas, vietos koordinatės, darbo laikas, įkrovimo adapterio tipai, stotelė mokama/nemokama, „Plugshare“ naudotojų komentarai apie stotelę, stotelės nuotraukos.

PlugShare duomenys yra nuolatos atnaujinami tiek platformos moderatorių, tiek pačių naudotojų. Dėl to, mažesnė rizika, kad pateikta informacija bus neteisinga.

Prieiga prie platformos yra uždara ir nekomerciniais tikslais duomenų gauti negalima. Dėl to pasakyti tikslaus skaičiaus duomenų negalime. Tačiau patikrinus PlugShare platformos žemėlapyje Lietuvoje esančias stoteles vizualiai, galime teigti, kad stotelių skaičius žymiai nesiskiria nuo LAKD pateiktų duomenų apie Lietuvoje esančias elektromobilių įkrovimo stoteles.

Įkrovimo stotelių šaltinių išsamumo informacinei sistemai palyginimas

Įkrovimo stotelių šaltinių išsamumas ir tinkamumas vertinimas pagal šiuos informacinei sistemai išrinktus svarbiausius kriterijus:

- Vietos koordinatės
- Vietos adresas
- Įkrovimo adapterių tipai
- Įkrovimo stotelių užimtumas
- Įkrovimo kaina (Eur/kWh)

Žemiau pateikta lentelė, kurioje palyginami aukščiau aprašyti įkrovimo stotelių informacijos duomenų šaltiniai. Lentelės langeliai pažymėti „+“ simboliu, jei atitinka iškeltą kriterijų,

2.5 lentelė. IS įkrovimo stotelių šaltinių palyginimas pagal duomenų išsamumą.

	Vietos koordinatės	**Vietos adresas**	**Įkrovimo adapterių tipai**	**Įkrovimo stotelių užimtumas**	**Įkrovimo kaina (Eur/kWh)**
LAKD	+		+	+	+
PlugShare	+	+	+		

Įkrovimo stotelių šaltinių aktualumo informacinei sistemai palyginimas

Įkrovimo stotelių šaltinių aktualumas vertinimas pagal duomenų atnaujinimo dažnumą.

2.6 lentelė. IS įkrovimo stotelių šaltinių palyginimas pagal duomenų atnaujinimo dažnumą.

	Duomenų atnaujimo dažnumas
LAKD	Kas 2-3 mėnesius.
PlugShare	Nuolatos platformos naudotojų.

Lietuvos įkrovimo stotelių esančių šaltinyje, duomenų kiekio palyginimas

2.7 lentelė. IS įkrovimo stotelių šaltinių palyginimas pagal duomenų skaičių.

	Duomenų skaičius
Lietuvos automobilių kelių direkcija. Elektromobilių įkrovos stotelės	275
PlugShare	Apie 275 ir daugiau

Pastebima, kad aukščiau (žr. 2.7 lentelės) pateiktoje lentelėje PlugShare duomenų skaičius buvo įvertintas netiksliai, nes norint tiksliai įvertinti duomenų skaičių reikia licenzijos, kuri yra suteikiama tik komerciniais tikslais. Dėl to, PlugShare duomenų skaičius buvo įvertintas vizualiai ištyrinėjus PlugShare platformos žemėlapį ir buvo pastebėta, kad Lietuvoje esančiu

elektromobilių stotelių skaičius nesiskiria nuo LAKD duomenų, tačiau tiksliai to įvertinti nepavyko.

2.4.3. Informacinės sistemos elektromobilių informacijos duomenys

Elektromobilių įkrovimo stotelių informacinėje sistemoje naudotojas turi galimybę pasirinkti automobilį iš sąraše jau esančių elektromobilių ir pagal pasirinktą elektromobilį gauti techninius elektromobilio duomenis, tokius kaip:

- Automobilio markė (Toyota, BMW, Mercedes-Benz ir t.t.)
- Automobilio modelis (Corolla, EQC ir t.t.)
- Elektromobilio pagaminimo metai (2021 ir t.t.)
- Nuvažiuojamas atstumas su 100 % baterijos įkrova (matavimo vienetas: km)
- Įkrovimo adapterio tipas (ccs, chademo ir t.t.)
- Maksimali AC įkrovimo galia (matavimo vienetas: kw)
- Maksimali DC įkrovimo galia (matavimo vienetas: kw)
- Baterijos talpa (matavimo vienetas: kwh)

Aukščiau nurodytiems duomenims gauti, informacinei sistemai reikia naudoti išorinius prieigos taškus (angl. API), kurie talpintų šiuos duomenis ir leistų prie jų prieiti. Žemiau yra išvardintos ir palygintos informacinės sistemos, kurios suteikia prieigą prie šių duomenų ir talpina elektromobilių specifikacijos informaciją kartu su informacinės sistemos naudojimosi taisyklėmis ir naudojimosi rizikomis.

ChargePrice API

ChargePrice talpina elektromobilių įkrovimo stotelių bei elektromobilių specifikacijos informaciją. Elektromobilių specifikacijos duomenyse nurodoma elektromobilio markė, modelis, pagaminimo data, DC įkrovimo adapterio tipas, AC maksimali įkrovimo galia, DC maksimali įkrovimo galia, baterijos talpa. Duomenys nėra atviri ir reikalinga licenzija norint gauti visą informaciją. ChargePrice nėra populiari platforma ir yra abejotina, jei ateityje, nusipirkus licenzija, visi duomenys bus prižiūrimi ir atnaujinami.

EV Database API

EV Database (nuoroda: http://www.ev-database.org) talpina elektromobilių pagamintų Europos Sąjungoje duomenis. Duomenyse nurodoma elektromobilio markė, modelis, pagaminimo metai, nuvažiuojamas bendras atstumas kilometrais, nuvažiuojamas atstumas mieste ir greitkelyje kilometrais, įkrovimo adapterio tipas, baterijos talpa, įkrovimo greitis, elektrinio variklio galia. Informacinės sistemos duomenys nėra atviri. Norint gauti prieigą prie duomenų reikia įsigyti licenziją.

US Environmental Protection Agency Database API

US Environmental Protection Agency (angl. Jungtinių Amerikos Valstijų Aplinkos apsaugos ministerija) talpina visų automobilių bei elektromobilių eksploatuojamų Jungtinėse Amerikos Valstijose specifikacijos duomenis. Duomenys yra atviri ir pateikiami CSV arba XML formatu, taip pat galima naudotis REST prieigos taškais. Duomenyse nurodoma elektromobilio markė, modelis, pagaminimo metai, bendras nuvažiuojamas atstumas myliomis, nuvažiuojamas

atstumas mieste ir greitkelyje myliomis, elektrinio variklio galia. Duomenyse yra informacijos apie dar neparduodamų, t.y. išleidžiamų kitais metais modelius ir šie duomenys yra nuolatos atnaujinami. Naudojantis šį API, reikia atsižvelgti, kad duomenys yra Jungtinių Amerikos Valstijų rinkoje parduodamų elektromobilių ir nebus tikslūs Europos Sąjungoje parduodamiems elektromobiliams. Taip pat, pateikiamas atstumas yra pateiktas myliomis, dėl to reikalinga konvertacija į metrinės matavimo sistemos vienetus.

European Environment Agency. CO_2 emissions from new passengers

Europos aplinkos apsaugos agentūra talpina Europoje registruojamų automobilių išmetamųjų dujų CO_2 kiekį. Kartu su CO_2 duomenimis talpinami ir tokie duomenys kaip automobilio markė, modelis, svoris, dydis, bendras nuvažiuojamas atstumas, registracijos data. Duomenys yra atviri ir pateikiami CSV formatu.

Lietuvoje įregistruotų kelių transporto priemonių duomenys

Lietuvoje įregistruotų automobilių transporto priemonių duomenys prižiūrimi valstybės įmonės „Regitros“. Duomenyse nurodoma tokia informacija kaip automobilio kuro tipas („Benzinas“, „Elektra“ ir t.t.), gamintojo markė, modelis, kėbulo pavadinimas („Sedanas“ , „Universalas“ ir t.t.), elektrinė rida, pavarų dėžės tipas. Duomenų šaltinis turi informacijos apie 16 elektromobilių, t.y. kurie yra papildyti informacija apie elektromobilio elektrinę ridą. Duomenys atnaujinami kas 3 mėnesius.

Elektromobilių specifikacijos informacijos šaltinių palyginimas

Šioje skiltyje palyginami aukščiau paminėti elektromobilių specifikacijos duomenų šaltiniai pagal skyriaus pradžioje išsikeltus vertinimo kriterijus. Jei duomenų šaltinis suteikia informacijos apie kriterijų, lentelės langelis yra pažymimas „+“, kitu atveju, yra paliekamas tuščias langelis. Taip pat, pridėtas langelis „Duomenys atviri“, kuris indikuoja ar duomenys yra atviri ir langelis „Duomenų kiekis“, kuris nurodo, kiek skirtingų elektromobilių specifikacijų yra.

2.8 lentelė. IS elektromobilių techninės specifikacijos šaltinių palyginimas pagal duomenų išsamumą.

	Markė	**Modelis**	**Pagaminimo metai**	**Nuvažiuojamas atstumas**	**Įkrovimo adapterio tipas**
ChargePrice	+	+	+		+
EV Database	+	+	+	+	+
US EPA	+	+	+	+	+
EEA	+	+	+	+	
Lietuvoje įregistruotų kelių transporto priemonių duomenys	+	+		+	

2.9 lentelė. IS elektromobilių techninės specifikacijos šaltinių palyginimas pagal duomenų išsamumą ir skaičių.

	AC įkrovimo galia	**DC įkrovimo galia**	**Baterijos talpa**	**Duomenys atviri**	**Duomenų kiekis**
ChargePrice	+	+	+		
EV Database	+	+	+		
US EPA			+	+	
EEA				+	
Lietuvoje įregistruotų kelių transporto priemonių duomenys					16

Iš aukščiau pateiktos lentelės pastebime, kad uždari duomenų šaltiniai yra daug išsamesni t.y. talpina daugiau informacijos apie išsikeltus kriterijus nei atviri šaltiniai. Taip pat, uždari duomenys yra turi didesnį kiekį išsamių duomenų nei atviri duomenys.

2.4.4. Informacinės sistemos oro sąlygų duomenys

Informacinei sistemai įgyvendinti reikalingi oro sąlygų bei prognozių duomenys. Šiuos duomenis galima gauti naudotis viešai prieinamais API, kurie talpina meteorologinių stočių duomenis.

Meteo

Meteo yra Lietuvos hidrometeorologijos tarnybos informacinė sistema, kuri talpina Lietuvos meteorologijos stočių duomenys. Duomenys atnaujinami kas valandą. Duomenyse teikiama informacija apie oro temperatūrą, vėjo greitį, vėjo kryptį, debesuotumą, drėgmę ir trumpą aprašymas kaip pvz. „cloudy“, „cloudy-with-sunny-intervals“, „sunny“, „snow“ ir t.t. Oru prognozes galima gauti 6 dienas į priekį.

AccuWeather

AccuWeather yra informacinė sistema, kuri talpina bei leidžia prieiti prie oro sąlygų visame pasaulyje. Naudojantis informacine sistema orų sąlygas galima sužinoti koordinačių tikslumu. AccuWeather nemokamas iki 50 užklausų per dieną. Galima gauti tokius duomenis kaip oro temperatūra, oro drėgmė, kritulių kiekis per paskutines 24 valandų, debesuotumas, vėjo greitis bei kryptis. Taip pat galima gauti prognozuojamus duomenis iki 120 minučių.

2.5. Analitinės dalies išvados

Išanalizavus maršruto sudarymą paaiškėjo, kad tinkamiausias sprendimas maršrutui priskirti elektromobilių įkrovimo taškus yra pritaikyti Haversino atstumą maršruto sustojimo taškams ir įkrovimo stotelėms. Taip pat, deklaruoti papildomą kintamąjį, kuris priklausomai nuo Haversino atstumo reikšmės patikrintų ar kelias iki taško neviršija šios nustatytos reikšmės. Išanalizuota, kad Haversino atstumas gali taip pat būti taikomas ir maršruto dalies

meteorologinėms stotims identifikuoti, naudojantis Haversino atstumu ir apsibrėžtu papildomu kintamuoju, kuris patikrintų meteorologinės stoties tinkamumą maršruto daliai.

Atlikus kūrimo technologijų analizę buvo priimti sprendimai, kurie įtakoja informacinės sistemos funkcionalumą, kokybę, patvarumą ir t.t. Informacinei sistemai intergruoti pasirinkta Google Maps žemėlapių sistema dėl plačios dokumentacijos aprašymo, populiarumo, patvarumo ir tikslumo. Kitas sprendimas - elektromoblių įkrovimo stotelių duomenų šaltinio pasirinkimas. Pasirinktas LAKD duomenų šaltinis, dėl atvirų bei detaliai aprašytų duomenų. Be to, LAKD suteikia informacijos apie visas Lietuvoje esančias elektromobilių įkrovimo stoteles. Elektromobilių modelių techninių duomenų gauti nepavyko, nes visi minėti šaltiniai gali būti naudojami tik komerciniais tikslais ir yra mokami arba šaltiniai nėra pakankamai išsamūs, kad galėtų būti panaudoti informacinėje sistemoje. Informacinei sistemai buvo pasirinkti Meteo duomenys oro sąlygoms gauti, nes talpina ne tik momentines oro sąlygas, bet ir prognozuojamas, dėl to, galima planuoti maršrutus 6 dienas į priekį, taip pat, duomenys yra atviri.

3. METODINĖ DALIS

3.1. IS Funkciniai ir nefunkciniai reikalavimai

Elektromobilių įkrovimo stotelių ir maršruto sudarymo informacinė sistema yra sudaryta iš daugybės komponentų ir duomenų šaltinių, kurie turi atlikti savo funkcijas. Norint, kad informacinė sistema būtų naudinga ir veiktų kaip numatyta, reikia apibrėžti funkcinius ir nefunkcinius reikalavimus.

3.10 lentelė. Informacinės sistemos funkciniai reikalavimai

Funkcinio reikalavimo Nr.	**Funkcinis reikalavimas**
FR1	Naudotojas turi turėti galimybę gauti rekomendacijas apie artimiausias elektromobilių įkrovimo stoteles savo maršrute
FR2	Naudotojas turi turėti galimybę pridėti tarpines stoteles prie maršruto
FR3	Naudotojas turi turėti galimybę pridėti elektromobilių įkrovimo stotelę prie pasirinkto maršruto
FR4	Naudotojas turi turėti galimybę peržiūrėti informaciją apie pasirinktą elektromobilių įkrovimo stotelę
FR5	Naudotojas turi turėti galimybę matyti pasirinktoje maršruto dalyje elektromobilio įkrovą ir likusį atstumą
FR6	Naudotojas turi turėti galimybę išsaugoti pasirinktą maršrutą
FR7	Sistemos administratorius turi turėti galimybę peržiūrėti, kada buvo atnaujinti oro ir įkrovimo stotelių duomenys
FR8	Sistemos administratorius turi turėti galimybę peržiūrėti, kokios užklausos buvo vykdomos aplikacijoje

Lentelėje pateikti funkciniai reikalavimai (žr. *3.10* lentelė) atsižvelgia į maršruto ir papildomų duomenų funkcionalumą ir teisingą šio funkcionalumo veikimą. Taip pat, į naudotojo autentifikacijos bei maršruto išsaugojimo funkcionalumus.

3.11 lentelė. Informacinės sistemos nefunkciniai reikalavimai

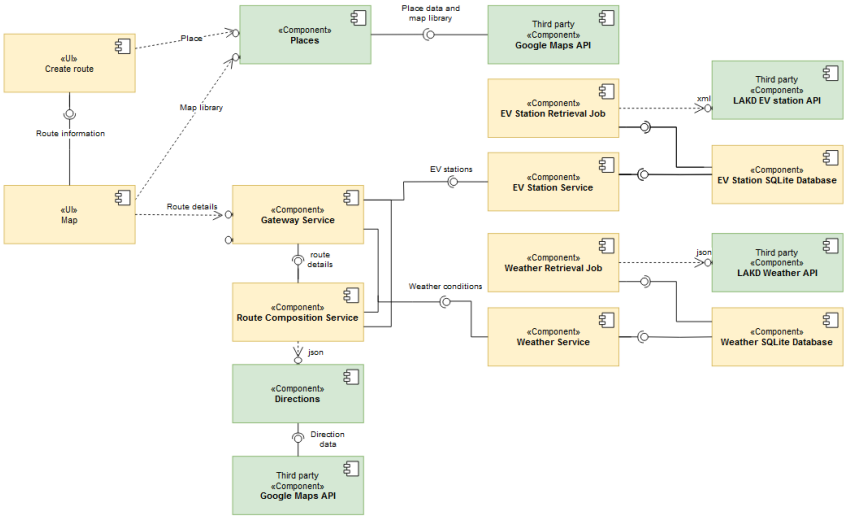
Nefunkcinio reikalavimo Nr.	Nefunkcinis reikalavimas
NFR1	Maršrutas su pradiniu, galutiniu tašku bei trimis tarpinėmis stotelėmis turi būti grąžinamas naudotojui per mažiau nei 7 sekundes.
NFR2	Vartotojo sąsaja naudotojo naršyklėje su trimis tarpinėmis stotelėmis negali užimti daugiau nei 150 MB atminties
NFR3	Informacinės sistemos vartotojo sąsaja turi būti lietuvių kalba, išskyrus Google Maps žemėlapio elementus
NFR4	Naudotojui gali būti pasiekiamas tik *GatewayService* sisteminis komponentas
NFR5	Elektromobilių įkrovimo stotelių komponentas turi pradėti įkrovimo stotelės duomenų bazės atnaujinimo darbą, kiekvieną naktį nuo 01:00 valandos (GMT+3)
NFR6	Oro salygų komponentas turi atnaujinti oro sąlygų duomenų bazę kiekvieną naktį nuo 02:00 iki 04:00 valandos (GMT+3)

Nefunkciniai reikalavimai (žr. *3.11* lentelė) atsižvelgia į sistemos reikalavimus, tokius kaip vartotojo sąsajos pasikrovimo laikas, atminties naudojimas naudotojo naršyklėje. Taip pat, atsižvelgiama, kad informacinė sistemos vartotojo sąsaja turi būti parašyta lietuvių kalba.

3.2. Informacinės sistemos architektūra

Elektromobilių įkrovimo stotelių ir maršruto sudarymo informacinė sistema sudaryta iš daugybės komponentų ir duomenų šaltinių. Informacinė sistema sukurta taikant mikroservisų struktūrą. Mikroservisų struktūra taikoma informacinei sistemai dėl lankstumo, nes pritaikius šią infrastruktūrą nesudėtinga prijungti papildomų komponentų ar duomenų šaltinių [22]. Taip pat, mikroservisų infrastruktūra yra patikimesnė, nes jei vienas iš komponentų veikia neteisingai arba yra išjungtas, sistema gali veikti be šio komponento [23].

Žemiau pateikiama elektromobilių įkrovimo stotelių IS komponentų diagrama.



3.1 pav. Sistemos architektūros komponentų diagrama

Diagramoje pastebimi skirtingų spalvų žymėjimai. Žalia spalva pažymėti trečiųjų šalių duomenų šaltinių priegos komponentai (angl. API), geltona spalva pažymėti informacinės sistemos vidiniai komponentai. Informacinėje sistemoje yra 2 duomenų bazės: elektromobilių įkrovimo stotelių bei oro sąlygų. Informacinėje sistemoje yra 4 atskiri aptarnaujantys komponentai (angl. services), kurie yra atsakingi už atskiras informacinės sistemos funkcijas tokias kaip:

- Gateway Service. Atlieka tarpininko funkciją. Aptarnauja kliento užklausas, yra atsakingas už užklausų persiuntimą tolimesniems komponentams. Taip pat, atlieka saugumo funkciją, nes naudotojui nesuteikiama informacija apie kitus, vidinius komponentus ir yra galimybė sukurti papildomas saugumo taisykles, kurios patikrintų siunčiamas užklausas.
- EV Station Service. Suteikia galimybę gauti bei talpinti elektromobilių įkrovimo stotelių duomenis iš LAKD duomenų šaltinio. Komponentas sudarytas iš duomenų prieigos taško (angl. API), foninių darbų, kurie atsakingi už periodišką elektromobilių stotelių duomenų atnaujinimą bei duomenų bazės, kurioje talpinami elektromobilių stotelių duomenys.
- Weather Service. Suteikia galimybę gauti bei talpinti oro sąlygų duomenis iš LAKD duomenų šaltinio. Komponentas sudarytas iš duomenų prieigos taško (angl. API), foninių darbų, kurie atsakingi už periodišką oro sąlygų bei prognozių atnaujinimą bei duomenų bazės, kurioje talpinami oro sąlygų duomenys.
- Route Composition Service. Atlieka informacijos apie maršrutą formavimo funkciją. Kreipiasi į trečiųjų šalių Google Maps API prieigos tašką ir gauna sudarytą maršrutą, taip pat, kreipiasi į elektromobilių stotelių *EV Station Service* ir oro sąlygų komponentą *Weather Service* ir gautą informaciją iš šių komponentų suformuoja į vieną atsakomąją užklausą, kuri yra išsiunčiama klientui.

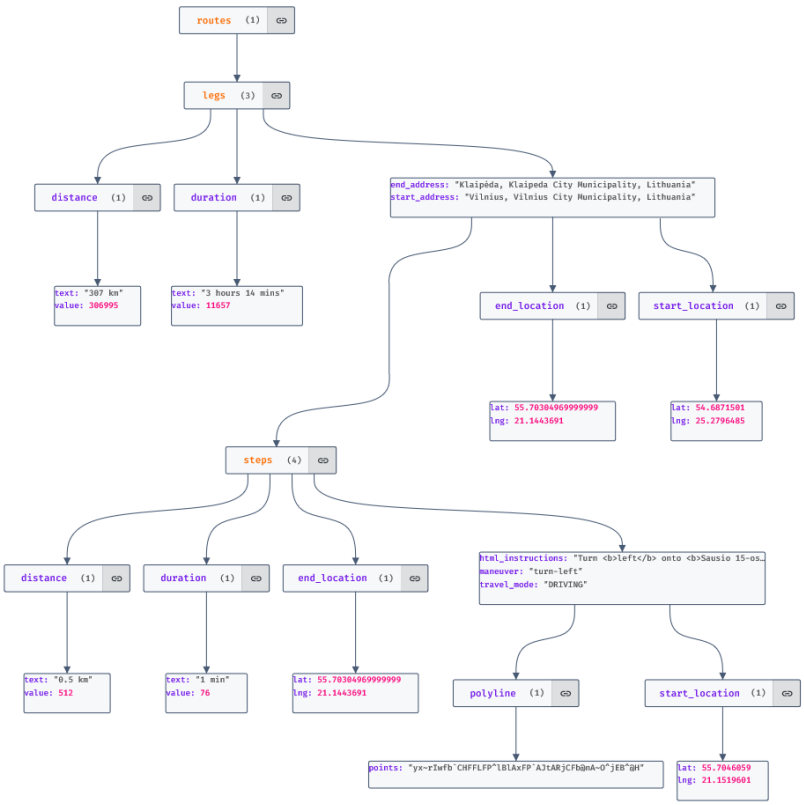
Taigi, elektromobilių įkrovimo stotelių ir maršruto sudarymo informacinė sistema yra sukurta naudojantis mikroservisų struktūra. Mikroservisų struktūra padeda užtikrinti lankstumą ir patikimumą, nes sistema gali veikti net jei kuri nors dalis yra neveikianti. Darbe pateikta informacinės sistemos komponentų diagrama, kurioje matyti dvi skirtingos duomenų bazės ir keturi atskiri aptarnaujantys komponentai, atsakingi už skirtingas informacinės sistemos funkcijas.

3.3. Maršruto sudarymas naudojantis Google Maps

Informacinėje sistemoje pasirinkta naudoti Google Maps žemėlapių sistemą, kuri turi galimybę sudaryti ir atvaizduoti maršrutą. Maršruto sudarymas naudojantis Google Maps yra įgyvendinamas *Google Maps Javascript API* biblioteka, kurią galima importuoti į Javascript aplikacijas, tokias kaip React, Vue js ir t.t. Google Maps gali sudaryti maršrutą, jei yra pateikiama tokia informacija kaip:

- Pradžios, galutinis bei tarpiniai taškai/vietovės.
- Papildomos nuostatos: ar vengti greitkelių, mokamų kelių, keltų.

Pateikus informaciją Google Maps bibliotekai gaunami tokie duomenys apie maršrutą:



3.2 pav. Google maps maršruto duomenų struktūra

Iš diagramos pastebime, kad duomenyse yra tokia informacija kaip bendras visos kelionės laikas, atstumas, pradžios taško koordinatės, galutinio taško koordinatės, informacija apie kiekvienos iš tarpinių stotelių atstumą, laiką iki sekančios tarpinės stotelės bei jų koordinates. Taip pat, pastebima, kad *polyline* parametras talpina *points* tekstinį kintamąjį, kuris yra atsakingas už kiekvienos tarpinės stotelės iki sekančios tarpinės stotelės kelio atkarpų seką. Šį tekstinį kintamąjį

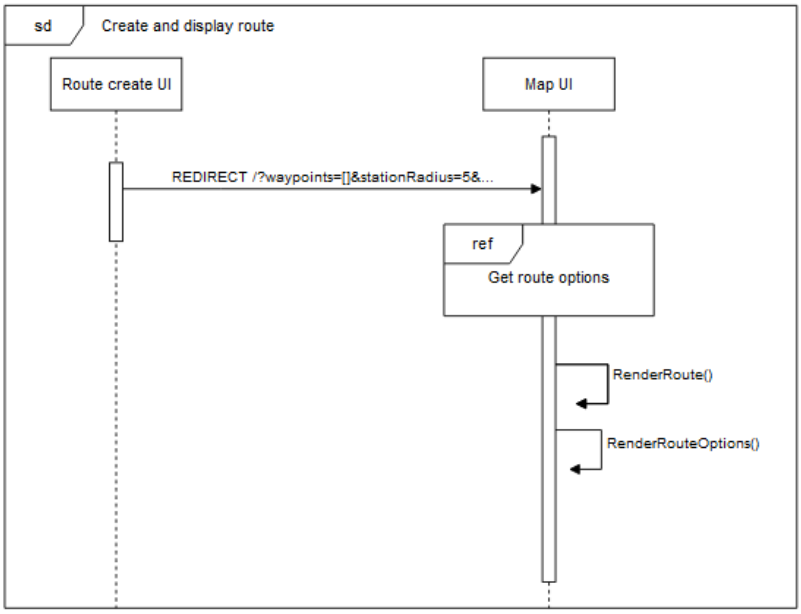
galima iššifruoti naudojantis Google Maps *decodePath* metodu ir paversti į platumos bei ilgumos koordinates.

Tačiau Google Maps bibliotekai nėra galimybės sudaryti maršruto pagal elektromobilio specifikacijas, oro sąlygas bei kitus parametrus, dėl to, yra naudojami papildomi duomenys bei atskiri vidiniai komponentai (žr. 3.2 skyrius), kurie Google Maps sudarytą maršrutą papildo elektromobilių specifikacijos, oro bei kelio sąlygų duomenimis ir grąžina šią apdorotą informaciją apie maršrutą naudotojui.

3.4. Maršruto ir papildomų maršruto duomenų gavimas

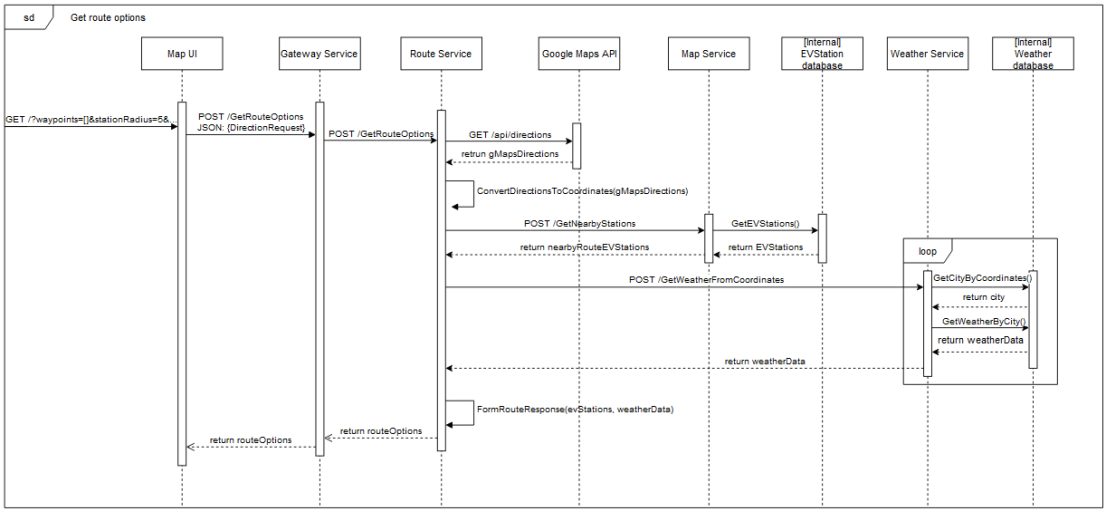
Maršruto ir papildomų maršruto duomenų gavimas yra labai svarbus elektromobilių įkrovimo stotelių IS. Maršrutą sudaro tokie parametrai kaip atstumas, laikas, tarpinių stotelių taškų kordinatės ir t.t. Papildomus maršruto duomenis sudaro iš skirtingų komponentų paminėtų 3.2 skyriuje gaunami duomenys, tokie kaip: elektromobilių stotelių vietovės bei specifikacijos duomenys, oro sąlygų duomenys, vietovės duomenys ir t.t.

Žemiau pateikta maršruto sukūrimo ir atvaizdavimo sekos diagrama.



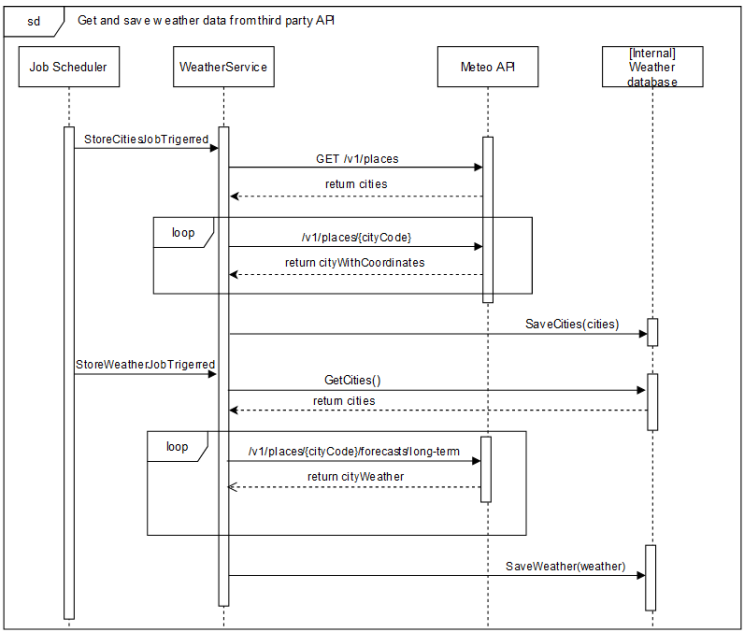
3.3 pav. Maršruto sukūrimo ir atvaizdavimo sekos diagrama

Diagramoje (žr. *3.3* pav.) pastebima, kad maršruto duomenys yra nurodomi skirtingoje vartotojo sąsajoje t.y. *Route Create UI*. Diagramoje yra sąlygojama, kad *Route Create UI* komponente yra įvesta tokia informacija kaip pradinis kelionės taškas, galutinis kelionės taškas, tarpiniai taškai bei kiti parametrai. Diagramoje pastebima, kad diagrama nukreipia į papildomą srauto diagramą esančią žemiau.



3.4 pav. Maršruto sukūrimo sekos diagrama GetRouteOptions

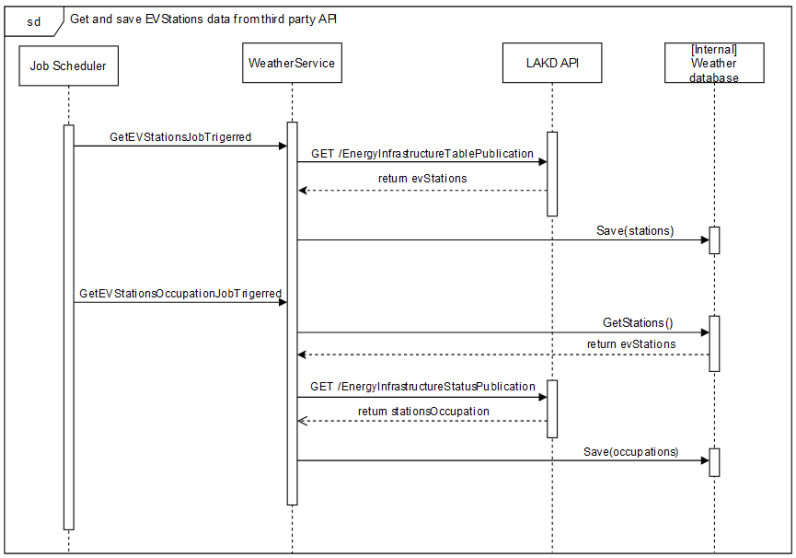
Papildoma srauto diagrama reikalauja tokių parametrų kaip naudotojo įvestos pradinės informacijos *Route Create UI* vartotojo sąsajoje apie maršruto pradinį bei galutinį taškus ir tarpines stoteles bei kitus parametrus tokius kaip: atstumas nuo įkrovimo stotelių ir maršruto, ar vengti greitkelių, keltų ir mokamų kelių. Duomenys gauti iš Google Maps API apie maršrutą ir papildomi mašruto duomenys gauti iš *Get route options* sekos diagramos yra atvaizduojami Google Maps žemėlapyje.



3.5 pav. Oro sąlygų gavimo ir išsaugojimo foninių darbų sekų diagrama

Aukščiau pateiktoje iliustracijoje (žr. *3.5* pav.) pateikiama oro sąlygų duomenų gavimo ir išsaugojimo į vidinę duomenų bazę foninių darbų sekų diagrama. Sekų diagramoje atliekami kreipimaisi į *Meteo API* /v1/places adresą visų vietovių gavimui, tuomet adresu /v1/places/{cityCode} siunčiamos užklausos kiekvienai iš vietovių platumai ir ilgumai gauti, nes tik tokiu būdu jas galima gauti. Atlikus vietovių gavimą į Meteo prieigos tašką kreipiamasi

/v1/places/{cityCode}/forecasts/long-term adresu ir gaunama kiekvienos vietovės orų prognozė. Gavus visas orų prognozes, jos yra įrašomos į duomenų bazę.



3.6 pav. Įkrovimo stotelių ir jų būsenų gavimo ir išsaugojimo foninių darbų sekų diagrama

Aukščiau pateiktoje iliustracijoje (žr. *3.6* pav.) pateikiama įkrovimo stotelių ir jų būsenų gavimo ir išsaugojimo foninių darbų sekų diagrama. Diagramoje, *GetEVStationsJob* darbas, atlieka įkrovimo stotelių duomenų gavimo ir išsaugojimo iš LAKD prieigos taško darbus. Sekantis darbas GetEVStationsOccupationJob iš LAKD prieigos taško gauna duomenis apie įkrovimo stotelių tiesioginį užimtumą ir juos išsaugo į vidinę duomenų bazę.

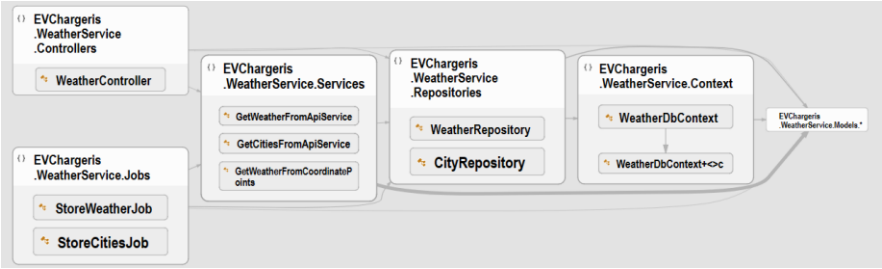
4. ĮGYVENDINIMO DALIS

4.1. Kodo struktūra

Informacinės sistemos kodas sudarytas iš dviejų programinių kalbų: C# ir Javascript. C# programavimo kalba naudojama kuriant serverio pusę (angl. backend), Javascript programavimo kalba naudojama kuriant naudotojo sąsają pasitelkiant Vue JS karkasą.

Serverio pusė kurta remiantis saugyklų-servisų (angl. repository-service) tipo kodo struktūra. Serverio pusėje aprašomi tokių tipų kodo dalys:

- Kontroleriai (angl. Controllers). Apibrėžia valdymo logiką ir nustato, kaip serveris turi reaguoti į užklausas iš klientų. Kontroleriai priima užklausas, apdoroja jas ir grąžina atsakymą.
- Repozitorijos (angl. Repositories). Kodo dalys, susijusios su duomenų baze ir duomenų valdymu. Repozitorijos aprašo, kaip programinė įranga turi bendrauti su duomenų baze ir teikia funkcijas, skirtas vykdyti CRUD operacijas [24].
- Sąsaja (angl. Interface). Taisyklių rinkinys suteikiantis papildomą abstrakcijos lygį servisų arba repozitorijos kodo dalims [25].
- Servisai (angl. Services). Atlieka specifinius veiksmus, susijusius su serverio pusės funkcionalumu. Servisai gali apimti duomenų apdorojimą, el. pašto siuntimą, duomenų tvarkymą ir kt.
- Foniniai darbai (angl. Jobs). Atlieka tam tikrus veiksmus informacinės sistemos fone. Foniniai darbai atlieka reguliarius duomenų atnaujinimus [26].
- Modeliai (angl. Models). Apibrėžia, kaip duomenys turi būti saugomi duomenų bazėje ir kokios struktūros turi laikytis.



4.1 pav. Oro sąlygų komponento priklausomybių diagrama

Aukščiau (žr. *4.1* pav.) pateikta oro sąlygų komponento priklausomybių diagrama (angl. dependency diagram), kurioje pažymėtos visos aprašytos kodo dalys.

4.2. Artimiausių įkrovimo stotelių paieška

Analitinėje dalyje (žr. 2.1 skyrius) artimiausioms įkrovimo stotelėms surasti buvo pasiūlyta naudoti Haversino formulę. Šiame skyriuje aprašomas įgyvendintas Haversino formulės pritaikymas artimiausių elektromobilių įkrovimo stotelių paieškai maršruto atžvilgiu.

Artimiausių įkrovimo stotelių gavimas atliekamas serverio pusėje (angl. backend). Žemiau pateiktas metodas *DistanceBetweenCoordinates*, kuris suskaičiuoja atstumą tarp dviejų koordinačių naudojantis Haversino formule.

```
DistanceBetweenCoordinates(coord1, coord2, radius)
lat1 ← ToRadians(coord1.Latitude)
lon1 ← ToRadians(coord1.Longitude)
lat2 ← ToRadians(coord2.Latitude)
lon2 ← ToRadians(coord2.Longitude)

deltaLat ← lat2 - lat1
deltaLon ← lon2 - lon1

a ← Sin(deltaLat / 2) * Sin(deltaLat / 2) +
Cos(lat1) * Cos(lat2) *
Sin(deltaLon / 2) * Sin(deltaLon / 2)

c ← 2 * Atan2(Sqrt(a), Sqrt(1 - a))

distance ← radius * c
return(distance)
```

4.2 pav. DistanceBetweenCoordinates metodo pseudokodas, kuris suskaičiuoja atstumą tarp dviejų koordinačių naudojantis Haversino formule

Metodui *4.2* pav. reikalingi tokie parametrai kaip: lyginamųjų taškų koordinatės t.y. jų plokštumos ir aukštumos, taip pat ir žemės spindulio reikšmė, kuri gali būti pateikta kaip konstanta (6371 km). Grąžinamas *double* reikšmės kintamasis *distance*, kuris nurodo atstumą tarp dviejų taškų.

Metodo tikslumas buvo ištestuotas, gautos reikšmės buvo patikrintos su *Google Maps Measure Distance* [27] įrankiu, kuris parodo atstumą tarp dviejų taškų žemėlapyje. Žemiau pateikiama lentelė, kurioje atvaizduojamos reikšmės gautos nuo maršruto artimiausio taško iki įkrovimo stotelės su *DistanceBetweenCoordinates* metodu ir *Google Maps Measure Distance* įrankiu.

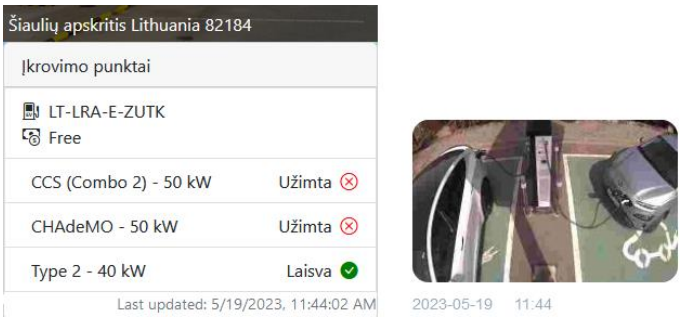
4.12 lentelė. Artimiausių įkrovimo stotelių paieškos metodo testavimas.

Įkrovimo stotelės adresas	**Artimiausiai įkrovimo stotelės esanti gatvė, kuri įtraukta į maršrutą.**	**Pasirinktas atstumas nuo maršruto ir įkrovimo stotelės**	**Atstumas su DistanceBetweenCoordinates metodu, km**	**Atstumas su Google Maps Measure Distance įrankiu, km**
Parodų g. 1, Vilnius (Ignitis-ON)	Oslo g., Vilnius	5 km	0.7 km	0.673 km
Titnago gatvė 217, Vilnius (Ignitis-ON)	Gariūnų g., Vilnius	5 km	2.9 km	2.90 km
Vydūno g. 27, Vilnius (Ignitis-ON)	Oslo g., Vilnius	5 km	4.1 km.	4.14 km
Vieškeliuko g. 8, Kaunas (Inbalance)	Greitkelis A1	5 km	4 km	4.01 km
J. Basanavičiaus gatvė 2A, Varėna (Elinta)	Kelias A4	5 km	4.7 km	4.95 km
Vilniaus gatvė 1B, Lazdijai (Ignitis-ON)	Kelias 2509	10 km	7.8 km	7.84 km

Iš lentelės (žr. *4.12* lentelė) pastebėta, kad atstumai apskaičiuoti *DistanceBetweenCoordinates metodu ir Google Maps Measure Distance* įrankiu skiriasi labai nežymiai, vidutiniškai 0.052 km arba 52 metrais. Toks skirtumas galėjo iškilti dėl *DistanceBetweenCoordinates* metodo kintamųjų tipų bei *distance* kintamojo reikšmės apvalinimo, taip pat, dėl *Google Maps Measure Distance* įrankio netikslaus taškų padėjimo.

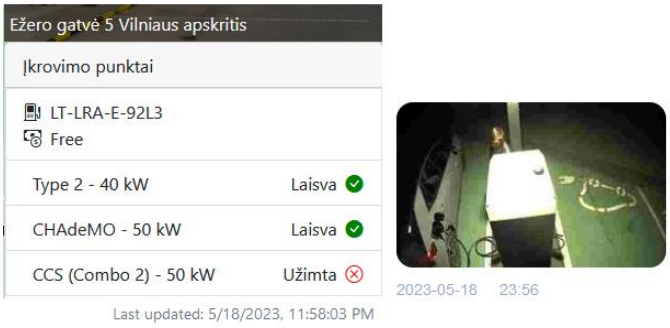
4.3. Įkrovimo stotelių užimtumo testavimas

Įkrovimo stotelių užimtumo testavimas atliktas pasitelkus Lietuvos automobilių kelių direkcijos svetainę eismoinfo.lt. Svetainė talpina nuotraukas iš elektromobilių įkrovimo stotelių stebėjimo kamerų kas 4 minutes ir šie duomenys yra prieinami viešai. Testavimas vykdomas suradus įkrovimo stotelę, kurioje kraunasi elektromobilis/iai ir patikrinus šią informaciją su vaizdo kamerų duomenimis. Tikrinamos buvo tik LAKD įrengtos elektromobilių įkrovimo stotelės, nes iš kitų tiekėjų duomenų apie įkrovimo stotelių užimtumą gauti nepavyko. Žemiau pateikiamos testavimo iliustracijos.



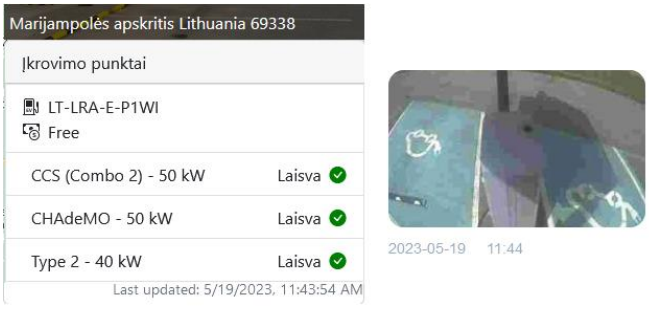
4.3 pav. Įkrovimo stotelės užimtumo duomenų testavimo rezultatas su dvejomis užimtomis vietomis

Pirmojo testavimo metu buvo pasirinkta testuoti įkrovimo stotelę su dvejomis užimtomis įkrovimo vietomis (žr. *4.3* pav.). Informacinė sistema parodė teisingą rezultatą, t.y. kad užimtos dvi įkrovimo jungtys, tai matoma ir per vaizdo kamerų duomenis.



4.4 pav. Įkrovimo stotelės užimtumo duomenų testavimo rezultatas su viena užimta vieta

Sekančio testavimo metu (žr. *4.4* pav.) buvo pasirinkta testuoti vieną užimtą įkrovimo vietą. Informacinė sistema parodė teisingą rezultatą iš iliustracijos matoma, kad užimta CCS (Combo 2) jungtis, tai patvirtina vaizdo kamerų duomenys.



4.5 pav. Įkrovimo stotelės užimtumo duomenų testavimo rezultatas su laisvomis įkrovimo vietomis

4.5 pav. atliktas testavimas su neužimta elektromobilių įkrovimo stotele. Informacinė sistema ir šiuo atveju parodė teisingą rezultatą, pastebima, kad vaizdo kamerų duomenys parodė, kad įkrovimo stotelė yra laisva ir informacinė sistema grąžino tokius pačius rezultatus.

Iš testavimo rezultatų pastebima, kad įkrovimo stotelių užimtumo funkcionalumas veikia teisingai. Visi atlikti testavimo atvejai parodė teisingą rezultatą.

4.4. Informacinės sistemos testavimas

Informacinės sistemos testavimas sudarytas iš vienetų, funkcinio, nefunkcinio ir integracinio testavimo.

4.4.1. Vienetų testavimas

Vienetų testavimas buvo atliktas naudojantis .NET XUnit biblioteka. Naudojantis biblioteka yra galimybė sukurti automatizuotus vienetinius testus. Žemiau pateiktas vienetinio testo kodo pavyzdys iš informacinės sistemos.

```
[Fact]
0 references
public void DistanceBetweenCoordinatesTest1()
{
    List<Coordinate> coordinates = new();

    // Distance between points is 5.706 km
    coordinates.Add(new Coordinate() { Latitude = 40.758701, Longitude = -73.985641 });
    coordinates.Add(new Coordinate() { Latitude = 40.805919, Longitude = -73.959112 });

    double result = StationDistanceCalculationHelper
        .DistanceBetweenCoordinates(coordinates[0], coordinates[1], EarthRadius);

    output.WriteLine(Math.Round(result, 3).ToString());

    Assert.True(Math.Round(result, 3) == 5.706);
}
```

4.6 pav. Atliktas vienetų testavimas DistanceBetweenCoordinates metodui.

4.6 pav. testo tikslas patikrinti *DistanceBetweenCoordinates* metodo tikslumą. Apibrėžtos dvi koordinatės, iš anksto žinoma, kad atstumas tarp jų yra 5.706 km. Šios koordinatės yra įrašomos į *DistanceBetweenCoordinates* metodo parametrus kartu su Žemės spindulio ilgiu, nurodytu kilometrais. Iš metodo gaunamas rezultatas, atstumas tarp dviejų koordinačių kilometrais. Šis atstumas yra palyginamas su iš anksto žinomu atstumu ir jei reikšmės sutampa, testas sėkmingas.

4.4.2. Funkcinis testavimas

Funkcinis testavimas atliktas naudojantis Selenium web driver [31] programine įranga. Funkciniame testavime testuojami 3.1 skyriuje išsikelti funkciniai reikalavimai. Prieš testavimą yra daromos prielaidos, kad informacinė sistema yra pasiekiama ir maršruto sudarymo metu įvesti taškai yra egzistuojantys.

4.13 lentelė. Funkcinių reikalavimų testavimas

Nr.	Funkc. reikalavimo nr.	Prielaidos	Žingsniai	Rezultatai
T1	FR1		1. Viršutinėje tinklapio juostoje paspaudžiamas mygtukas „Naujas maršrutas“. 2. Įvedami laukai: „Pradinis taškas“, „Galutinis taškas“ 3. Paspaudžiamas mygtukas „Pridėti tarpinę stotelę“ 4. Įvedamas laukas: „Tarpinė stotelė 1“ 5. Įvedami laukai: „El. automobilio nuvažiuojamas atstumas“, „El. automobilio baterijos talpa“. 6. Pasirenkamas laukelis: „El. automobilio baterijos įkrova“. 7. Spaudžiamas mygtukas „Pateikti“. 8. Laukiama kol maršrutas užsikraus. 9. Vizualiai patikrinama, ar žemėlapyje pateiktos įkrovimo stotelės yra šalia maršruto.	• Sistema sėkmingai suformuoja maršrutą pagal naudotojo įvestus pradžios ir pabaigos taškus bei tarpines stoteles. • Įkrovimo stočių žemėlapyje atvaizduojamos tik tos stotelės, kurios yra šalia maršruto. Testas naudojantis Selenium scenarijų (žr. *PRIEDAS 1.1* pav.) sėkmingas.
T2	FR2		1. Viršutinėje tinklapio juostoje paspaudžiamas mygtukas „Naujas maršrutas“. 2. Paspaudžiamas mygtukas „Pridėti tarpinę stotelę“ 3. Įvedamas laukas: „Tarpinė stotelė 1“ 4. Įvedami likusieji laukai. 5. Spaudžiamas mygtukas pateikti. 6. Laukiama kol maršrutas užsikraus. 7. Vizualiai patikrinama, ar maršrutas žemėlapyje veda per pasirinktą tarpinę stotelę.	• Sistema sėkmingai priima ir apdoroja naudotojo įvestus duomenis apie tarpinę stotelę. • Maršrutas žemėlapyje atvaizduojamas taip, kad jis eina per pasirinktą tarpinę stotelę. Testas naudojantis Selenium scenarijų (žr. *PRIEDAS 1.1* pav.) sėkmingas.

T3	FR3	Atlikti T1 teste minėti žingsniai	1. Žemėlapyje paspaudžiama ant žalios žymos. 2. Laukiama, kol užsikraus elektromobilio įkrovimo stotelė. 3. Kairėje pusėje spaudžiamas mygtukas „Pridėti į maršrutą“	• Paspaudus ant žalios žymos žemėlapyje, elektromobilio įkrovimo stotelė sėkmingai užkraunama ir parodoma informacija apie ją. • Po elektromobilio įkrovimo stotelės užkrovimo, mygtukas "Pridėti į maršrutą" tampa aktyvus ir prieinamas naudotojui. Selenium scenarijus (žr. *PRIEDAS 3.1* pav.) sėkmingas.
T4	FR4	Atlikti T1 teste minėti žingsniai	1. Žemėlapyje paspaudžiama ant žalios žymos. 2. Laukiama, kol užsikraus elektromobilio įkrovimo stotelė. 3. Vizualiai patikrinama, ar įkrovimo stotelės informacija teisinga	• Paspaudus ant žalios žymos žemėlapyje, elektromobilio įkrovimo stotelė sėkmingai užkraunama. Selenium scenarijus (žr. *PRIEDAS 2.1* pav.) sėkmingas. • Vizualiai patikrinus įkrovimo stotelės informaciją, ji atitinka realią stotelės būseną ir yra teisinga.
T5	FR5	Atlikti T1 teste minėti žingsniai	1. Žemėlapyje pelytė nuvedama ant bet kurios maršruto atkarpos. 2. Patikrinama, ar informacija apie baterijos įkrova, nuvažiuotą atstumą ir oro sąlygas suteikiama.	• Pelytės paspaudimas ant bet kurios maršruto atkarpos žemėlapyje sėkmingai pateikė informaciją apie elektromobilio baterijos įkrovą, nuvažiuotą atstumą ir oro sąlygas. • Informacija apie elektromobilio įkrovą ir likusį atstumą atitiko pasirinktą maršruto dalį ir buvo teisinga.

T6	FR6	Atlikti T1 teste minėti žingsniai	1. Dėšinėje pusėje spaudžiamas mygtukas „Išsaugoti maršrutą Google Maps platformoje“ 2. Patikrinama, ar Google maps platformoje maršruto informacija nepakitus.	• Paspaudus ant mygtuko naudotojas perkeliamas į naują langą su sukurtu maršrutu Google Maps platformoje. • Maršruto informacija nesikeičia tiek maršruto suukūrimo platformoje, tiek Google Maps platformoje
T7	FR7		1. Prisijungiama prie serverio, kuriame vykdoma aplikacija 2. Aplikacijos kataloge peržiūrimi katalogai *logs* 3. Atidaromas naujausias log failas ir ieškoma oro ir įkrovimo stotelių darbų pradžios įrašų	• Įrašas surastas, testas sėkmingas.
T8	FR8		1. Prisijungiama prie serverio, kuriame vykdoma aplikacija 2. Aplikacijos kataloge peržiūrimi katalogai *logs* 3. Atidaromas katalogas ir peržiūrima informacija	• Įrašai apie užklausas gaunami (žr. *PRIEDAS 4.1* pav.).

4.4.3. Nefunkcinis testavimas

Nefunkcinis testavimas atliktas naudojantis tokiais įrankiais kaip *Firefox Developer Tools* [28], *.NET SDK* [29] ir *Postman* [30]. Atliekant testus daroma prielaida, kad informacinė sistema yra pasiekiama tiek lokaliame tinkle, tiek išoriniame tinkle.

4.14 lentelė. Nefunkcinių reikalavimų testavimas

Nr.	**Funkc. reikalavimo nr.**	**Įrankiai**	**Žingsniai**	**Rezultatai**
T1	NFR1	Firefox developer tools	1. Vartotojo sąsajoje spaudžiamas mygtukas „Naujas maršrutas“. 2. Įvedamas maršrutas su pradžios ir pabaigos taškais ir trimis tarpinėmis stotelėmis. 3. Spaudžiamas mygtukas „Pateikti“. 4. Naudojantis *Firefox Developer Tools*, *Network* funkcija stebimas puslapio užsikrovimo laikas.	• Sistema sėkmingai suformuoja maršrutą pagal naudotojo įvestus pradžios ir pabaigos taškus bei tarpines stoteles ir užikrovimo laikas yra mažesnis nei 7 sekundės. Testavimo rezultatas pateikiamas *PRIEDAS 5.1* pav.
T2	NFR2	Firefox developer tools	1. Sukuriami skirtingi maršrutai su trimis tarpinėmis stotelėmis, įskaitant pradinę ir galutinę stoteles. 2. Vykdoma Firefox Developer Tools, Memory -> Take snapshot funkcija. 3. Išanalizuojamas rezultatas	• Vartotojo sąsaja su trimis tarpinėmis stotelėmis vidutiniškai užima 150 mb atminties. • Rezultatas neviršija nustatytos ribos (žr. *PRIEDAS 6.1* pav.) t.y. 200 mb atminties.
T3	NFR3	Rankinis tikrinimas	1. Atidaroma maršruto kūrimo vartotojo sąsaja. 2. Patikrinama, ar visi tekstiniai laukeliai yra lietuvių kalba. 3. Atidaroma žemėlapio vartotojo sąsaja su pasirinktu maršrutu. 4. Patikrinama, ar visi tekstiniai laukeliai yra lietuvių kalba. 5. Atidaroma įkrovimo stotelės peržiūros vartotojo sąsaja. 6. Su pelyte paspaudžiama ant maršruto atkarpos. 7. Patikrinama, ar visi tekstiniai laukeliai yra lietuvių kalba.	• Visi patikrinti tekstiniai laukeliai yra lietuvių kalba.

T4	NFR4	Postman	1. Atidaroma postman programinė įranga įrenginyje, esančiame tinklo išorėje. 2. Siunčiama užklausą GET /route. 3. Tikrinama, ar užklausa pasiekia serverį. 4. Siunčiama užklausa: POST /api/Weather. 5. Siunčiama užklausa: POST /api/Route. 6. Siunčiama užklausa: POST /api/NearbyStations. 7. Tikrinama, ar užklausos nepasiekia serverių.	• Užklausa GET /route grąžino atsakymą, tačiau visos kitos užklausos grąžino klaidas t.y., kad nėra pasiekiamos (žr. PRIEDAS 10.1 pav.) Tokiu atveju, reikalavimas išpildytas ir pasiekiamas yra tik *GatewayService* komponentas.
T5	NFR5	.NET SDK	1. Naudojantis .NET SDK paleidžiamas komponentas: MapService. 2. Laukiama iki 1:00 valandos 3. Atidaromas log failas 4. Iš log failo, nustatoma, ar duomenų atnaujinimo darbas pasileido.	• Log faile yra įrašas, kuris nurodo, kad foninis darbas pradėtas vykdyti.
T6	NFR6	.NET SDK	1. Naudojantis .NET SDK paleidžiamas komponentas: WeatherService. 2. Laukiama iki 4:00 valandos 3. Atidaromas log failas 4. Iš log failo, nustatoma, ar duomenų atnaujinimo darbas pasileido.	• Log faile yra įrašas, kuris nurodo, kad foninis darbas pradėtas vykdyti.

4.4.4. Integracinis testavimas

Integracinis testavimas atliktas naudojantis Postman programine įranga. Postman programinė įranga suteikia galimybę rašyti integracinius testus Javascript kalba [32].

Ištestuoti 3 duomenų prieigos taškai (angl. API endpoints), kurie buvo testuojami pagal grąžintus atsakymo kodus, identifikacinius numerius bei duomenų validumą. Žemiau pateikiamas įkrovimo stotelės duomenų gavimo integracinis testas (daugiau žr. PRIEDAS 7.1 pav.), sukurtas su Postman programine įranga.

```
pm.test("Status code is 200", function () {
    pm.response.to.have.status(200);
});
```

4.7 pav. Grąžinto *atsakymo* kodo integracinis testas

Pateiktas integracinis testas (žr. *4.7* pav.) siunčia GET užklausą /api/Stations/{id} prieigos taškui, kuris grąžina atsakymo kodą 200, jei užklausa sėkminga ir 500, jei užklausa nepavyko. Testas yra sėkmingas, jei atsakymo kodas 200.

Žemiau pateikiamas integracinis testas, kuris patikrina, ar išsiųstas identifikacinis kodas sutampa su gautos įkrovimo stotelės identifikaciniu kodu.

```
6   const pathSegments = pm.request.url.getPath().split('/');
7   const receivedID = pathSegments[pathSegments.length - 1];
8
9   pm.test("Id matches result id", function () {
10      var jsonData = pm.response.json();
11      pm.expect(jsonData[0].id).to.eq(parseInt(receivedID));
12  });
```

4.8 pav. Grąžinto identifikacinio kodo integracinis testas

Integracinis testas esantis *4.8* pav. patikrina, ar grąžintos įkrovimo stotelės identifikacinis kodas sutampa su išsiųstu. Prieš identifikacinio kodo validavimą, atliekamas išsiųsto identifikacinio kodo gavimas iš užklausos adreso ir atsakymo, tuomet vykdomas gautų kodų palyginimas. Jei kodai sutampa, testas sėkmingas.

Taip pat pateikiamas ir gautų duomenų validavimo integracinis testas.

```
14  pm.test("Test coordinates not null", function () {
15      var jsonData = pm.response.json()[0];
16      pm.expect(jsonData.latitude).to.not.undefined;
17      pm.expect(jsonData.longitude).to.not.undefined;
18  });
```

4.9 pav. Grąžintų duomenų validavimo integracinis testas

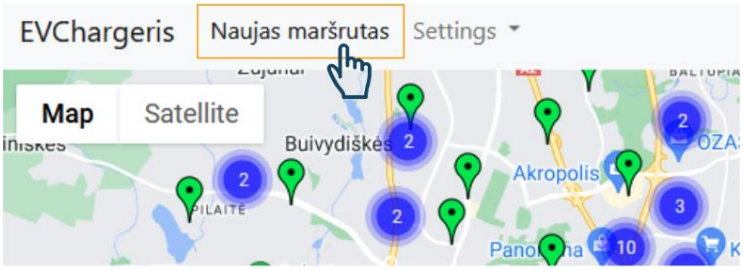
Pateiktas duomenų validavimo integracinis testas (žr. *4.9* pav.) patikrina, ar privalomos reikšmės, šiuo atveju, koordinatės, turi nustatytas reikšmes, t.y. nėra tuščios. Tokiu būdu patikrinama, ar gauti duomenys yra teisingi.

Integraciniais testais padengti ir visų įkrovimo stotelių gavimo (žr. PRIEDAS 8.1 pav.) bei sukurto maršruto gavimo prieigos taškai (žr. PRIEDAS 9.1 pav.), kurie taip pat testuojami pagal grąžintus atsakymo kodus, identifikacinius numerius bei duomenų validumą. Gavus maršruto gavimo prieigos taško atsakymą, papildomai testuojama, ar atskiri duomenų šaltiniai grąžino duomenis.

4.5. Vartotojo vadovas

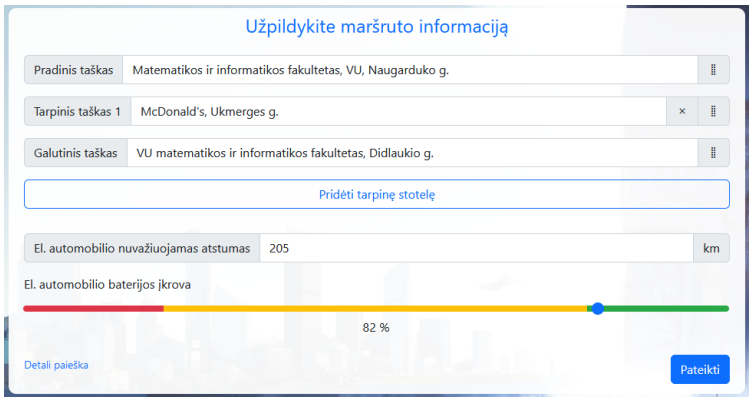
4.5.1. Maršruto sukūrimas

Naudotojas gali sukurti maršrutą įsijungus internetinį puslapį ir paspaudus viršutinėje juostoje esanti mygtuką „Naujas maršrutas“.



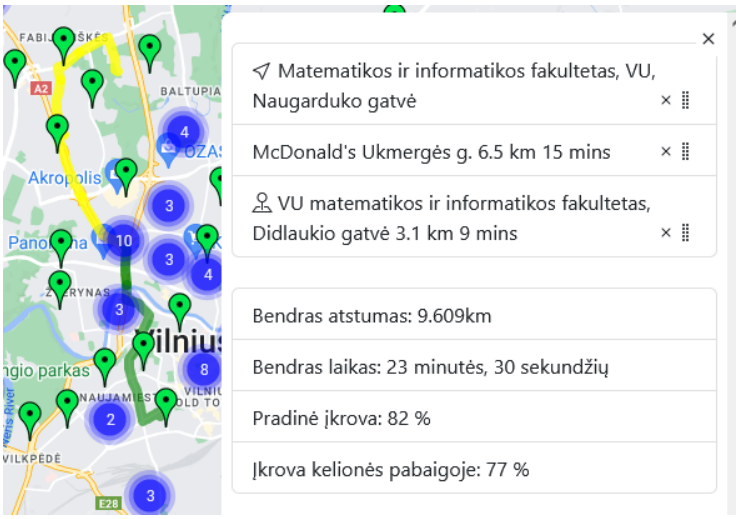
4.10 pav. „Naujas maršrutas“ mygtukas

Paspaudus mygtuką „Naujas maršrutas“ atidaroma maršruto sudarymo forma, kurią reikia užpildyti. Užpildytos formos pavyzdys žemiau.



4.11 pav. Maršruto sukūrimo forma

Užpildžius formoje esančius duomenis, spaudžiamas mygtukas „Pateikti“ ir naudotojas yra perkeliamas į žemėlapį su sudarytu maršrutu.



4.12 pav. Maršruto ir maršruto informacijos atvaizdavimas

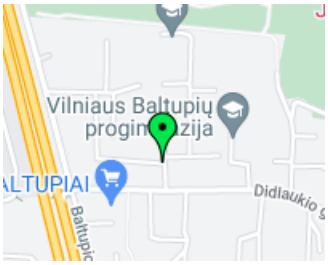
Žemėlapyje sudarytas maršrutas. Žalia maršruto spalva informuoja naudotoją, kad šioje maršruto atkarpoje elektromobilio įkrova yra nuo 80-100 %, geltona nuo 20-80 %, raudona nuo 5-20 % ir juoda spalva nurodo, kad baterijos įkrova mažesnė nei 5%. Taip pat, naudotojas baterijos įkrovą ir nuvažiuotą atstumą bei oro sąlygas tam tikroje maršruto atkarpoje gali peržiūrėti paspausdamas arba užvilkdamas pele ant maršruto atkarpos.

4.13 pav. Maršruto pasirinktos atkarpos informacijos pavyzdys

Užvilkus pelę ant maršruto atkarpos, atvaizduojama elektromobilio įkrova bei kita informacija pasirinktame maršruto atkarpos taške.

4.5.2. Elektromobilio įkrovimo stotelės peržiūra

Naudotojas turi galimybę peržiūrėti informaciją apie elektromobilių įkrovimo stotelę. Žalias segtukas pažymi vietą, kurioje yra įkrovimo stotelė.



4.14 pav. Įkrovimo stotelės segtukas (žalios spalvos)

Su pelyte paspaudus ant segtuko, kairėje ekrano pusėje, gaunama informacija apie pasirinktą elektromobilių įkrovimo stotelę.

4.15 pav. Informacijos apie įkrovimo stotelę pavyzdys

Įkrovimo stotelės peržiūroje yra atvaizduojama tokia informacija kaip: stotelės adresas, stotelės identifikacinis kodas, krovimo įkainiai, adapterių tipai bei galingumas ir taip pat adapterių užimtumas. Adapterių užimtumas rodomas tik LAKD įrengtose stotelėse, nes duomenų apie įkrovimo stotelių užimtumą iš privačių įmonių gauti nepavyko.

5. REZULTATAI IR APIBENDRINIMAS

Analizė parodė, kad oro sąlygos, važiavimo greitis, elektromobilio baterijos būklė ir įkrova, krovimo įpročiai įtakoja elektromobilio nuvažiuojamą atstumą. Taip pat, naudojantis Haversino atstumo formule galima efektyviai planuoti maršrutus ir padidinti elektromobilių kelionės patikimumą. Google Maps buvo pasirinktas kaip tikslus ir patikimas žemėlapių duomenų šaltinis, o Lietuvos automobilių kelių direkcija (LAKD) suteikė išsamius duomenis apie įkrovimo stoteles. Meteorologiniai duomenys iš Meteo duomenų šaltinio leidžia planuoti maršrutus atsižvelgiant į oro sąlygas.

Remiantis analizės rezultatais, optimalus maršrutas įgyvendintas duomenimis gautais iš Google Maps API platformos, Lietuvos automobilių kelių direkcijos įkrovimo stotelių duomenimis, Meteo.lt oro sąlygų duomenimis ir pasitelkus Haversino atstumo formulę, įkrovimo stotelių nuo maršruto skaičiavimui ir maršruto aktualių regionų, oro sąlygoms gauti, paieškai.

Elektromobilių maršruto sudarymo informacinė sistema buvo sukurta taikant mikroservisų struktūrą, užtikrinančią lankstumą ir patikimumą. Sistema veikia net ir tais atvejais, kai vienas iš komponentų veikia neteisingai arba yra išjungtas. Sistema buvo išbandyta įvairiomis testavimo rūšimis ir patvirtintas jos tinkamumas bei atitiktis reikalavimams.

Sistema suteikia naudotojui galimybę sukurti optimalų maršrutą elektromobiliui, gauti rekomendacijas apie tinkamiausias elektromobilių įkrovimo stoteles maršrute, peržiūrėti informaciją apie stoteles, matyti oro sąlygas, planuoti alternatyvius maršrutus ir kitas funkcijas.

Šis baigiamasis darbas prisideda prie elektromobilių populiarinimo ir ekologiškesnio transporto vystymo, suteikdamas efektyvias ir patikimas priemones elektromobilių kelionės planavimui. Informacinė sistema su savo funkcijomis ir architektūra yra gerai įgyvendinta ir atitinka vartotojų poreikius.

LITERATŪRA

1. European Commission. CO_2 emission performance standards for cars and vans. 2023. [Online]. Available: https://climate.ec.europa.eu/eu-action/transport-emissions/road-transport-reducing-co2-emissions-vehicles/co2-emission-performance-standards-cars-and-vans_en
2. Lietuvos Respublikos susisiekimo ministerija. Elektromobilių naudojimą skatinančios priemonės. 2023. [Online]. Available: https://sumin.lrv.lt/lt/veiklos-sritys/kita-veikla/pletra-ir-inovacijos/elektromobiliu-naudojima-skatinancios-priemones
3. European Environment Agency. New registrations of electric vehicles in Europe. 2023. [Online]. Available: https://www.eea.europa.eu/ims/new-registrations-of-electric-vehicles
4. Jinhua Ji, Yiming Bie, Ziling Zeng, Linhong Wang, Communications in Transportation Research, Trip energy consumption estimation for electric buses. 2022. [Online] Available. https://www.sciencedirect.com/science/article/pii/S2772424722000191
5. Sirapa Shrestha, Bivek Baral, Malesh Shah, Sailesh Chitrakar, Bim P Shrestha, Measures to resolve range anxiety in electric vehicle users, International Journal of Low-Carbon Technologies, Volume 17, 2022, Pages 1186–1206, https://doi.org/10.1093/ijlct/ctac100
6. Bedogni, Luca, Luciano Bononi, Alfredo D'Elia, Marco Di Felice, Marco Di Nicola, and Tullio S. Cinotti. "Driving without anxiety: a route planner service with range prediction for the electric vehicles", 2014. International Conference on Connected Vehicles and Expo (ICCVE), New York, IEEE
7. Bedogni, Luca, Luciano Bononi, Marco Di Felice, Alfredo D'Elia, and Tullio S. Cinotti. "A route planner service with recharging reservation: electric itinerary with a click." 2016. IEEE Intelligent Transportation Systems Magazine 8 (3): 75–84.
8. Rahul Roy Jannapura Shivakumar, Christian Winter. Conception of an itinerary planning assistant for EV drivers. 2023. [Online] Available: https://www.sciencedirect.com/science/article/pii/S1877050923003344
9. Indah Setyorini, Desi Ramayanti. Finding Nearest Mosque Using Haversine Formula on Android Platform. 2019. [Online] Available: https://www.researchgate.net/publication/336277930_Finding_Nearest_Mosque_Using_Haversine_Formula_on_Android_Platform/fulltext/5d98940b299bf1c363f97296/Finding-Nearest-Mosque-Using-Haversine-Formula-on-Android-Platform.pdf
10. Dwi Arman Prasetya, Phong Thanh Nguyen, Rinat Faizullin, Iswanto Iswanto, Edmond Febrinicko Armay. Resolving the Shortest Path Problem using the Haversine Algorithm. [Online] Available: https://www.lppm.unmer.ac.id/webmin/assets/uploads/lj/LJ20210109161016316836 6.pdf
11. Rezania Agramanisti Azdy, Febriyanti Darnis. Use of Haversine Formula in Finding Distance Between Temporary Shelter and Waste End Processing Sites. [Online] Available: https://iopscience.iop.org/article/10.1088/1742-6596/1500/1/012104/pdf
12. Dwi Arman Prasetya1, Phong Thanh Nguyen, Rinat Faizullin, Iswanto Iswanto, Edmond Febrinicko Armay. Resolving the Shortest Path Problem using the Haversine Algorithm.

2023. [Online]. Available: https://www.lppm.unmer.ac.id/webmin/assets/uploads/lj/LJ20210109161016316836 6.pdf
13. Chargeprice API documentation. Vehicles endpoint. 2023. [Online]. Available: https://github.com/chargeprice/chargeprice-api-docs/blob/master/api/v2/vehicles/index.md
14. EV Database. Electric Vehicle Database. 2023. [Online]. Available: https://ev-database.org/
15. United States Environmental Protection Agency. Fuel Economy Data. 2023. [Online]. Available: https://www.fueleconomy.gov/feg/download.shtml
16. European Environment Agency. CO2 emissions from new passenger cars. 2023. [Online]. Available: http://co2cars.apps.eea.europa.eu/
17. OpenStreetMaps. FAQ. Is OpenStreetMap data free?. 2023. [Online] Available: https://wiki.osmfoundation.org/wiki/FAQ
18. Google Maps Platform. Pricing. 2023 [Online] Available: https://mapsplatform.google.com/pricing/
19. Mapbox. Mapbox pricing. 2023 [Online] Available: https://www.mapbox.com/pricing
20. Lietuvos automobilių kelių direkcija. Elektromobilių įkrovos stotelės. 2023 [Online] Available: https://data.gov.lt/dataset/elektromobiliu-ikrovos-stoteles
21. PlugShare. PlugShare API. 2023 [Online] Available: https://company.plugshare.com/api.html
22. Mihai Baboi, Adrian Iftene, Daniela Gîfu, Dynamic Microservices to Create Scalable and Fault Tolerance Architecture, Procedia Computer Science, Volume 159, 2019, Pages 1035-1044, ISSN 1877-0509, https://doi.org/10.1016/j.procs.2019.09.271. (https://www.sciencedirect.com/science/article/pii/S187705091931467X)s
23. Florian Auer, Valentina Lenarduzzi, Michael Felderer, Davide Taibi, From monolithic systems to Microservices: An assessment framework, Information and Software Technology, Volume 137, 2021, 106600, ISSN 0950-5849, https://doi.org/10.1016/j.infsof.2021.106600. (https://www.sciencedirect.com/science/article/pii/S0950584921000793)
24. Martin Fowler, Edward Hieatt and Rob Mee, Repository. [Online] https://martinfowler.com/eaaCatalog/repository.html
25. Martin Fowler, Separated interface, [Online] https://www.martinfowler.com/eaaCatalog/separatedInterface.html
26. Microsoft. Background jobs. 2023. [Online] https://learn.microsoft.com/en-us/azure/architecture/best-practices/background-jobs
27. Google Maps. Measure distance between points. 2023. [Online]. Available: https://support.google.com/maps/answer/1628031?hl=en&co=GENIE.Platform%3DDesktop
28. Mozilla. Firefox Developer tools. 2023. [Online]. Available: https://firefox-source-docs.mozilla.org/devtools-user/
29. Postman. What is Postman? 2023. [Online]. Available: https://www.postman.com/product/what-is-postman/
30. Microsoft. What is the .NET SDK? [Online]. Available: https://learn.microsoft.com/en-us/dotnet/core/sdk
31. Selenium. Selenium Web Driver. 2023. [Online]. Available: https://www.selenium.dev/documentation/webdriver/
32. Postman. Writing tests. 2023. [Online]. Available: https://learning.postman.com/docs/writing-scripts/test-scripts/

PRIEDAS 1

```
from selenium import webdriver
from selenium.webdriver.common.by import By
from selenium.webdriver.support.ui import WebDriverWait
from selenium.webdriver.support import expected_conditions as EC

# Configure web driver
driver = webdriver.Firefox(executable_path="-")

# Navigate to the webpage with the form
driver.get('http://localhost:5001/route')

# Find and click the button
button = driver.find_element(By.CSS_SELECTOR, 'button.btn:nth-child(4)')
button.click()

# Wait for the form fields to load
wait = WebDriverWait(driver, 2)
wait.until(EC.visibility_of_element_located((By.CSS_SELECTOR, '#waypoint1')))

# Fill the form fields
field1 = driver.find_element(By.ID, 'evDistanceField')
field1.send_keys('505')

field2 = driver.find_element(By.ID, 'evBatterySizeField')
field2.send_keys('55')

field2 = driver.find_element(By.ID, 'batteryCharge')
field2.send_keys('80')

field1 = driver.find_element(By.CSS_SELECTOR, '#waypoint0')
field1.send_keys('Vilnius')

field1 = driver.find_element(By.CSS_SELECTOR, '#waypoint1')
field1.send_keys('Šiauliai')

field1 = driver.find_element(By.CSS_SELECTOR, '#waypoint2')
field1.send_keys('Klaipėda')

# Submit the form
submit_button = driver.find_element(By.CSS_SELECTOR, 'button.btn:nth-child(2)')
submit_button.click()

# Close the browser
driver.quit()
```

PRIEDAS 1.1 pav. Selenium testas atliekantis maršruto sukūrimo funkciją

PRIEDAS 2

```
# -*- coding: utf-8 -*-

from selenium import webdriver
from selenium.webdriver.common.by import By
from selenium.webdriver.support.ui import WebDriverWait
from selenium.webdriver.support import expected_conditions as EC
import time

# Configure Selenium to use your preferred web driver
driver = webdriver.Firefox(executable_path="-")

# Navigate to the webpage with the form
driver.get('http://192.168.0.82:5001/route')

# Find and click the button
button = driver.find_element(By.CSS_SELECTOR, 'button.btn:nth-child(4)')
button.click()

# Wait for the form fields to load
wait = WebDriverWait(driver, 2)
wait.until(EC.visibility_of_element_located((By.CSS_SELECTOR, '#waypoint1')))

# Fill the form fields
field1 = driver.find_element(By.ID, 'evDistanceField')
field1.send_keys('505')

field2 = driver.find_element(By.ID, 'evBatterySizeField')
field2.send_keys('55')

field2 = driver.find_element(By.ID, 'batteryCharge')
field2.send_keys('80')

field1 = driver.find_element(By.CSS_SELECTOR, '#waypoint0')
field1.send_keys('Šilutė')

field1 = driver.find_element(By.CSS_SELECTOR, '#waypoint1')
field1.send_keys('Tauragė')

field1 = driver.find_element(By.CSS_SELECTOR, '#waypoint2')
field1.send_keys('Vilnius')

# Submit the form
submit_button = driver.find_element(By.CSS_SELECTOR, 'button.btn:nth-child(2)')
submit_button.click()

# Wait for a page to load
time.sleep(10)

# Find and click the google maps marker
button = driver.find_element(By.CSS_SELECTOR, '.gm-style > div:nth-child(2) >
div:nth-child(2) > div:nth-child(1) > div:nth-child(3) > div[title="EGI-S-160"]')
button.click()
# Do further processing if needed
time.sleep(1000)

# Close the browser
driver.quit()
```

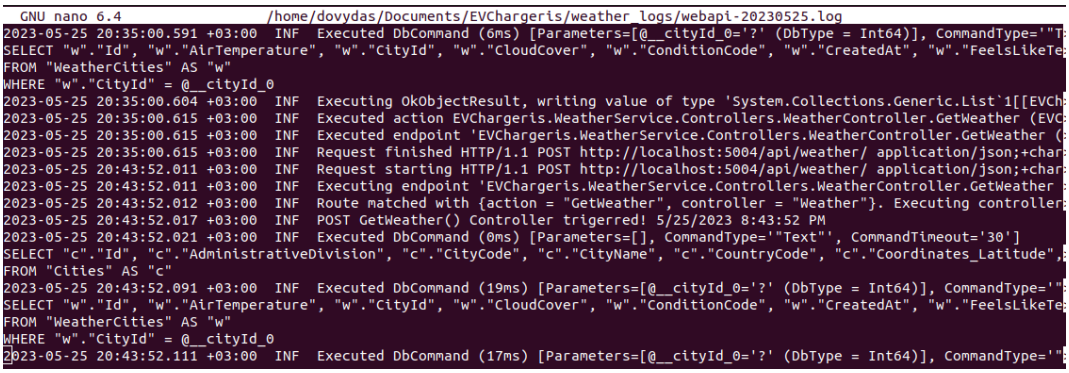
PRIEDAS 2.1 pav. Selenium testas atliekantis maršruto sukūrimo ir įkrovimo stotelės pasirinkimo funkciją

PRIEDAS 3

```
# -*- coding: utf-8 -*-
import time
from selenium import webdriver
from selenium.webdriver.common.by import By
from selenium.webdriver.support.ui import WebDriverWait
from selenium.webdriver.support import expected_conditions as EC

# Configure Selenium to use your preferred web driver
driver =
webdriver.Firefox(executable_path="C:\\Users\\zzdov\\OneDrive\\Documents\\geckodriv
er.exe")

# Navigate to the webpage with the form
driver.get('http://192.168.0.82:5001/route')

# Precondition. Route data submitted.

# Wait for a page to load
time.sleep(10)

# Find and click specific google maps marker
button = driver.find_element(By.CSS_SELECTOR, '.gm-style > div:nth-child(2) >
div:nth-child(2) > div:nth-child(1) > div:nth-child(3) > div[title="EGI-S-160"]')
button.click()

# Wait for station to load
time.sleep(5)

# Select adapter type
select_adapter_button = driver.find_element(By.CSS_SELECTOR, 'div.card:nth-child(1)
> div:nth-child(1) > p:nth-child(1)')
select_adapter_button.click()

time.sleep(2)
# Click add waypoint
add_waypoint_button = driver.find_element(By.CSS_SELECTOR, 'button.btn:nth-
child(7)')
add_waypoint_button.location_once_scrolled_into_view
add_waypoint_button.click()

# Wait for the users validation
time.sleep(1000)

# Close the browser
driver.quit()
```

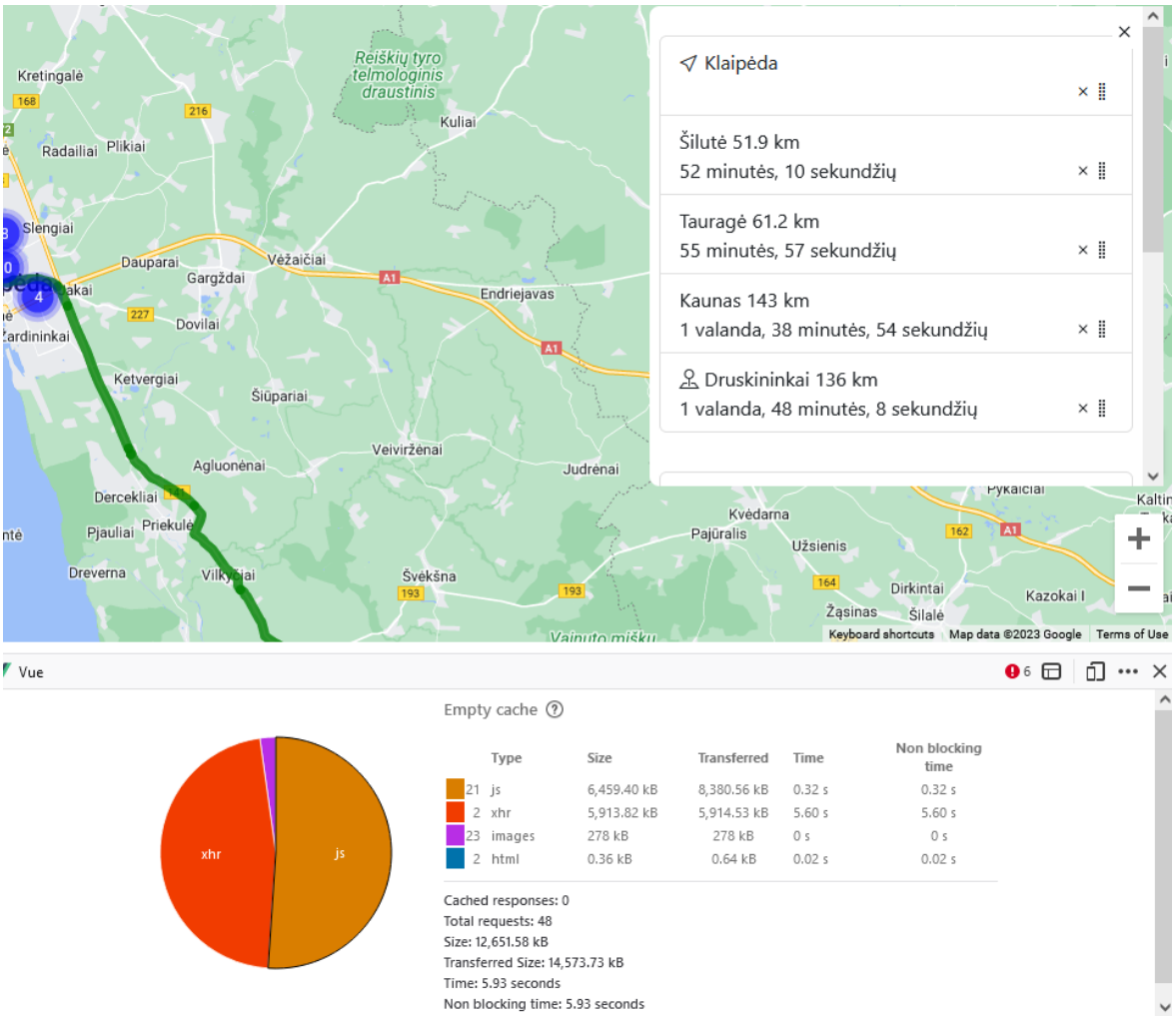
PRIEDAS 3.1 pav. Selenium testas pridedantis elektromobilių įkrovimo stotelę į maršrutą

PRIEDAS 4



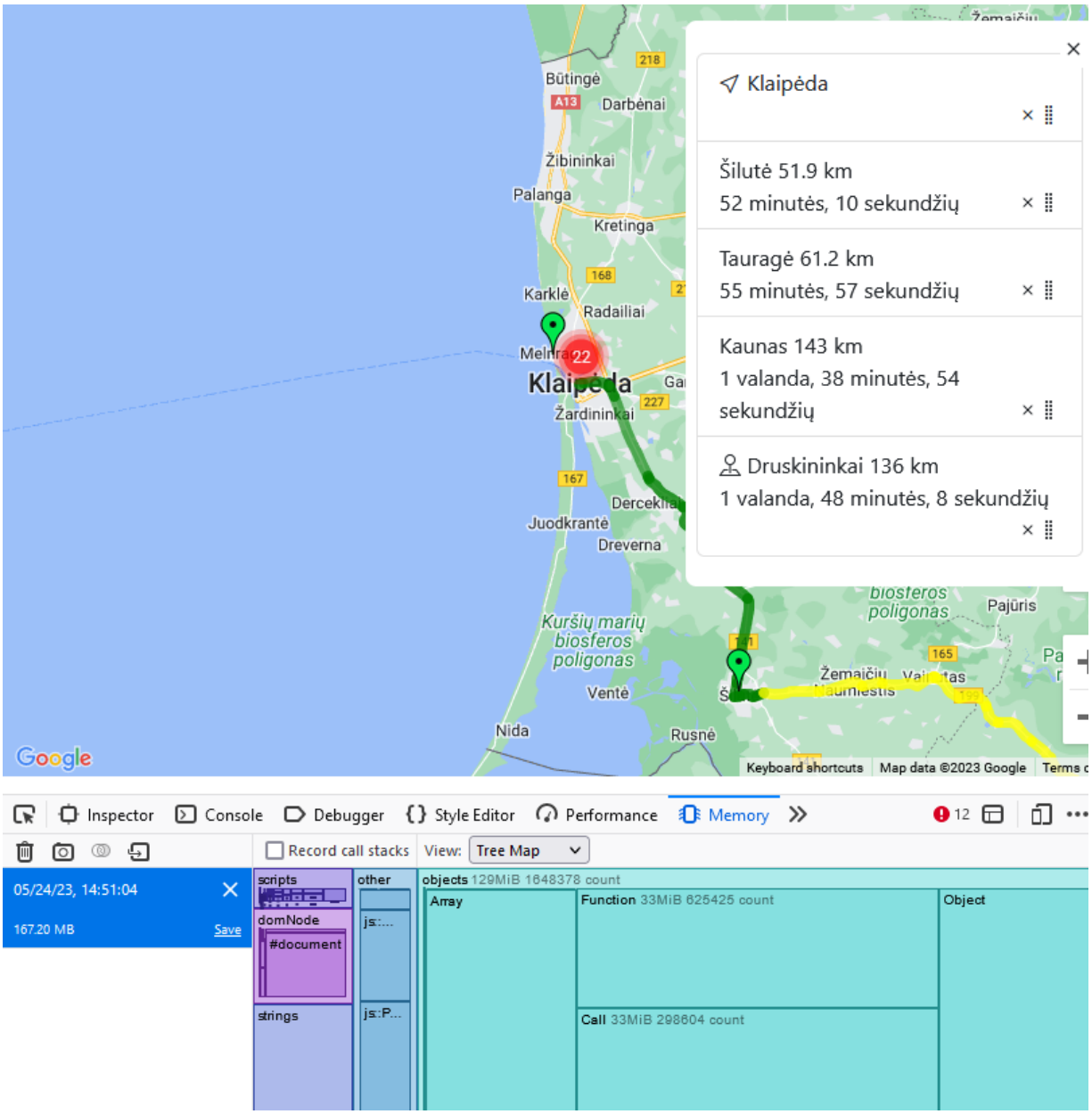
PRIEDAS 4.1 pav. Oro sąlygų komponento log failas

PRIEDAS 5



PRIEDAS 5.1 pav. Maršruto sukūrimo našumo testavimas

PRIEDAS 6



PRIEDAS 6.1 pav. Žemėlapio vartotojo sąsajos atminties testavimas

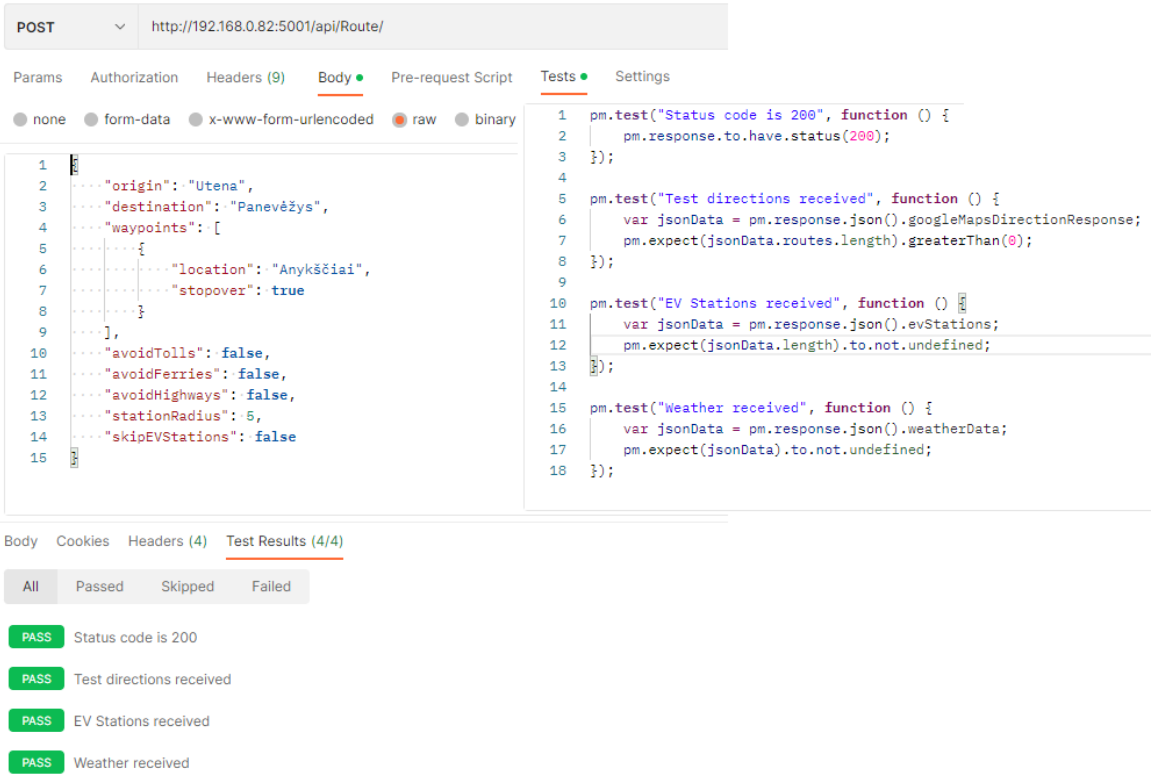
PRIEDAS 7

PRIEDAS 7.1 pav. Elektromobilių įkrovimo stotelės duomenų gavimo prieigos taško testavimas

PRIEDAS 8

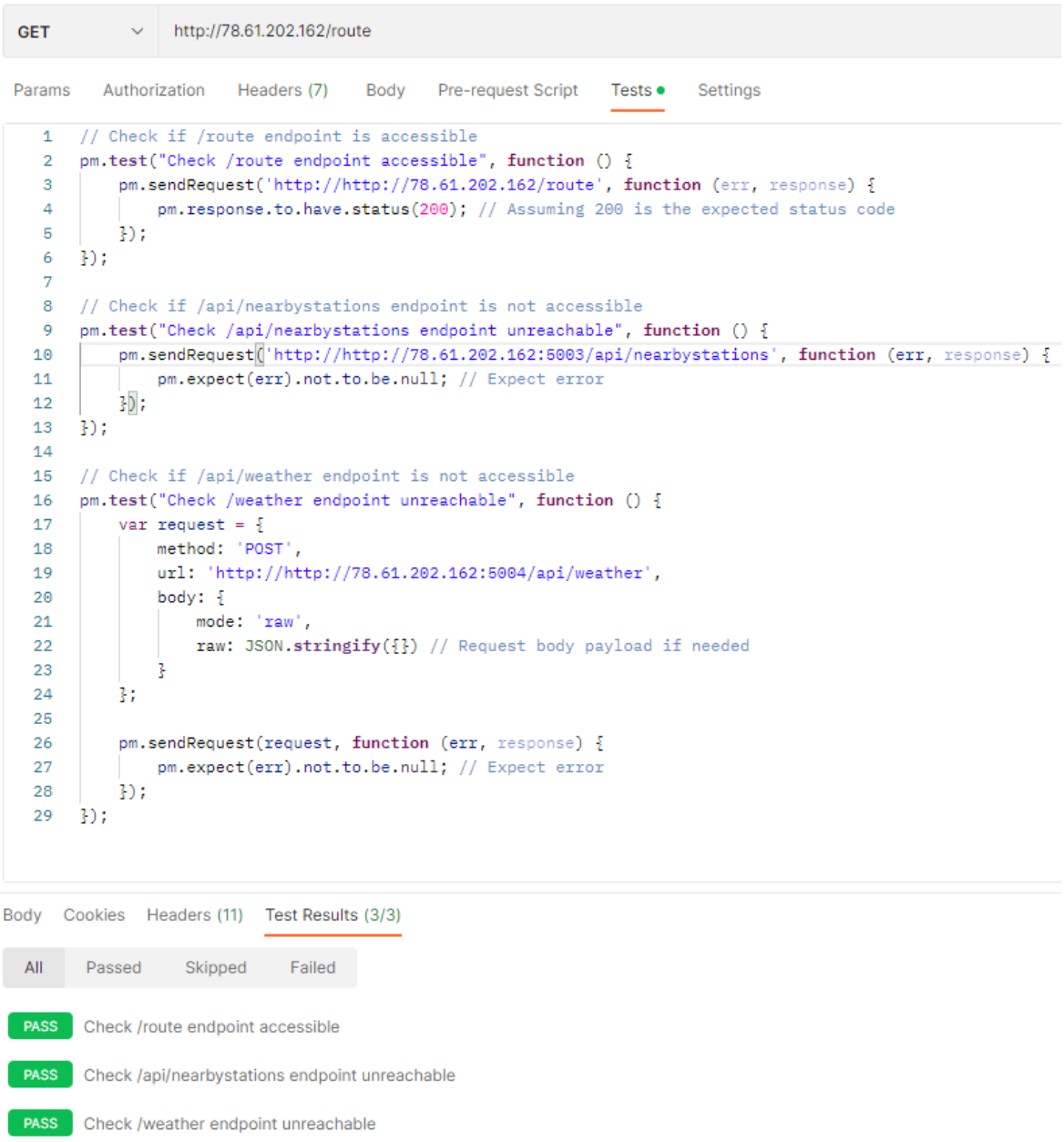
PRIEDAS 8.1 pav. Visų elektromobilių įkrovimo stotelių duomenų gavimo prieigos taško testavimas

PRIEDAS 9



PRIEDAS 9.1 pav. Maršruto sukūrimo prieigos taško testavimas

PRIEDAS 10



PRIEDAS 10.1 pav. Prieinamumo prie GatewayService komponento testavimas iš išorinio tinklo