VILNIAUS UNIVERSITETAS
MATEMATIKOS IR INFORMATIKOS FAKULTETAS
PROGRAMŲ SISTEMŲ KATEDRA

Automatinė ūkio valdymo sistema

Automatic farm management system

Laboratorinis darbas I

Atliko: 2 kurso 3 grupės studentai
Matas Savickis (parašas)
Justas Tvarijonas (parašas)
Greta Pyrantaitė (parašas)
Rytautas Kvašinskas (parašas)

Darbo vadovas: Karolis Petrauskas, Doc., Dr. (parašas)

Vilnius – 2017

TURINYS

Įvadas

Automatinė ūkio valdymo sistema (toliau - Auto ūkis) yra programa, leidžianti ūkininkui valdyti jo ūkį skaitmeniniu būdu. Auto ūkis leidžia registruoti gyvūnus ir stebėti kiekvieno jų bioparametrus (kraujo spaudimą, svorį, sveikatą) bei matyti ūkio technikos judėjimą po žemės plotą. Taip pat sistema vartotojui leidžia sekti dirvos parametrus (drėgmę, pH lygį), oro prognozes ir gyvūnų ligų paplitimą aplinkinėse teritorijose. Auto ūkis padeda ir su verslo valdymu: nesunkiai galima samdyti darbuotojus, atlikti buhalterinę apyskaitą, stebėti rinkos kainas ir apskaičiuoti bei numatyti galimą pelną. Iškilus nelaimei per Auto ūkio sistemą galima greitai iškviesti greitąją pagalbą, policiją, gaisrinę ar saugos tarnybą. Orų prognozės yra paimtos iš www.gismeteo.lt. Pagrindinė sistemos inovacija yra tai, kad, kai sistema yra pilnai įdiegta, darbuotojų skaičius, reikalingas palaikyti ūkį, tampa minimalus. Kadangi ūkio technika būtų valdoma automatiškai, vairuotojų ir derliaus nurinkėjų nereiktų. Gyvūnų sekimas yra įgyvendinamas mikro kontrolerio su Arduino pagalba. Šis kontroleris nedidelis ir lengvai pritaikomas visokio pobūdžio darbams. Jį, kartu su WiFi moduliu, sistema naudoja gauti gyvūno lokaciją per Google Maps,- taip pasiklydę ar pavogti gyvūnai būtų greitai surandami ir grąžinami. Žemės laistymas ir tręšimas taip pat būtų automatizuotas: parametrai gaunami per Arduino detektorius, kurie pagal pasikeitusią dirvos kompoziciją nusprendžia, ko trūksta žemei, ir aktyvuoja laistymo ir tręšimo sistemas. Darbuotojų samdymas yra įgyvendintas per darbo biržos puslapį, kur greitai ir nesunkiai galimą įdėti skelbimą arba surasti darbuotoją. Buhalterija yra tvarkoma naudojantis nemokama buhalterijos programa Wave Accounting, kuri yra implementuota į Auto ūkį. Auto ūkio sistema yra parašyta JAVA kalba - tai leidžia programą paleisti ant bet kurios operacinės sistemos. Ateityje numatoma galimybė programą perkelti į išmaniuosius telefonus. Sistema buvo projektuojama pasitelkiant www.planttext.com ir www.draw.io funkcionalumą.

1. Sukurtos sistemos aprašymas(v1.0)

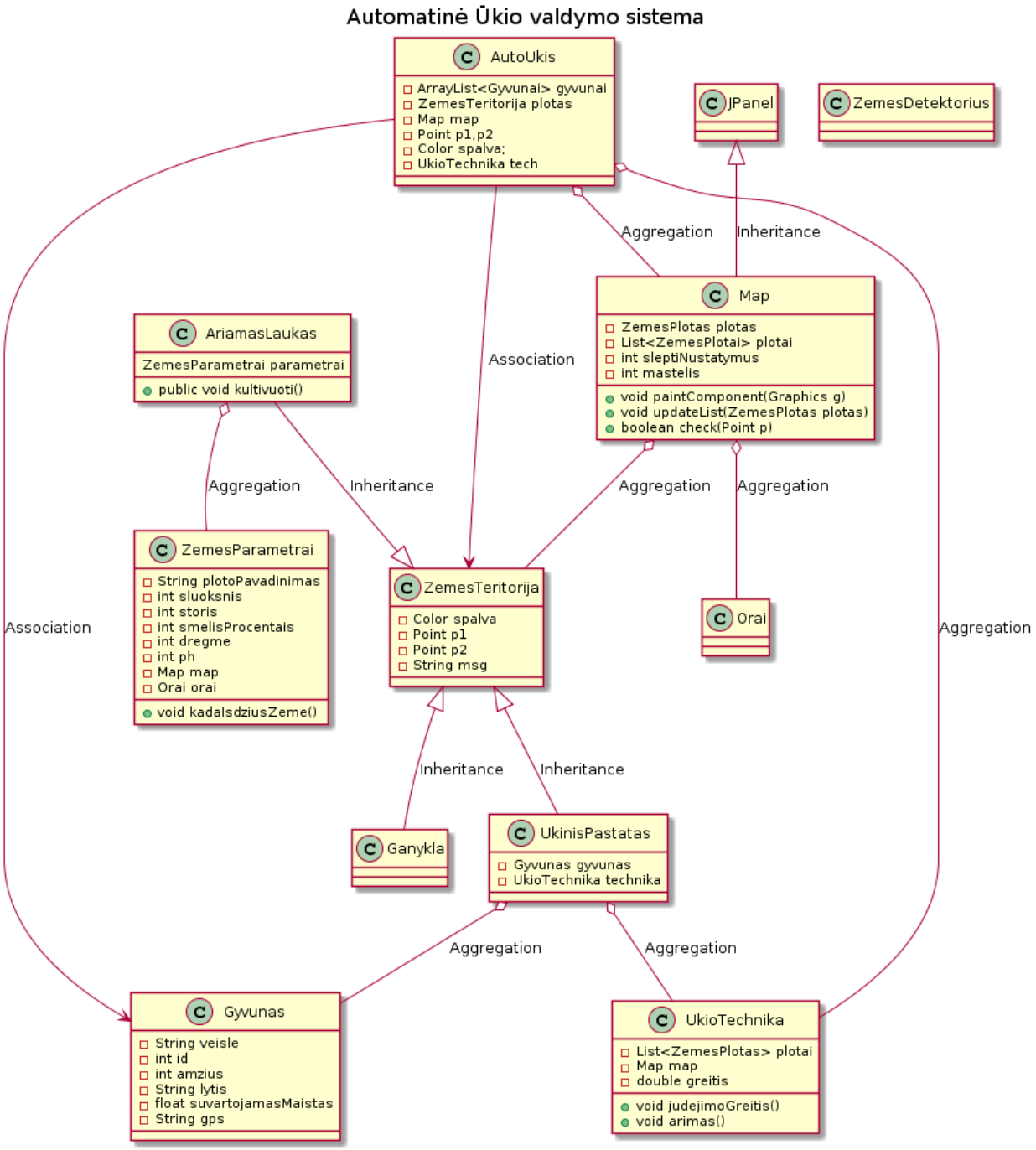
1.1. Loginis pjūvis

1.1.1. Žodynas

- Klasės:
  * AutoUkis - pagrindinė (main) programos klasė. Ši klasė piešia grafinę vartotojo sąsają ir laiko savyje kitų klasių objektus, kurių informacija reikalinga piešimui.
  * Map - teritorijos piešimui skirta klasė.
  * ZemesTeritorija - apskaičiuoja tam tikros teritorijos plotą.
  * Gyvunas - klasė, skirta gyvūno rodmenims ir metodams saugoti.
  * AriamasLaukas - laiko savyje reikšmes, apibūdinančias unikalų lauką, ir metodus, susijusius su lauko darbu.
  * Ganykla - laiko parametrus ir metodus darbui su ganyklomis, kurios yra žemės plote.
  * UkinisPastatas - saugo ūkinę techniką arba gyvūnus.
  * UkioTechnika - laiko ūkio technikos duomenis ir apskaičiuoja technikos judėjimo greitį.
  * ZemesParametrai - saugo įvairius žemės parametrus (drėgmė, pH...).
  * Orai - klasė, skirta gauti vartotojui reikalingas orų prognozes iš www.gismeteo.lt.
  * ZemesDetektorius - klasė, skirta bendrauti su žemės detektoriumi.
- Bendri terminai:
  * Žemės plotas - vieta, kurią valdo ir gali stebėti vartotojas(ūkininkas).
  * Detektorius - Arduino mikro kontroleris.
  * Ūkininkas - žmogus, kurio valdomoje teritorijoje įdiegtas Auto ūkis.
  * Arduino - mikro kontroleris, skirtas ūkio sekimui.
  * Automatiškai valdoma - valdymui nereikalinga žmogaus pagalba.
  * Darbuotojas - žmogus, dirbantis ūkininko versle.
  * Gyvūnas - visi gyvūnai, kurie priklauso ūkininkui, ir yra registruoti Auto ūkis sistemoje.

1.1.2. Klasių diagrama

Pagal suprogramuotą šabloninį programos karkasą nubraižėme *UML diagramą(1 pav)* minėta PlnatText programa.

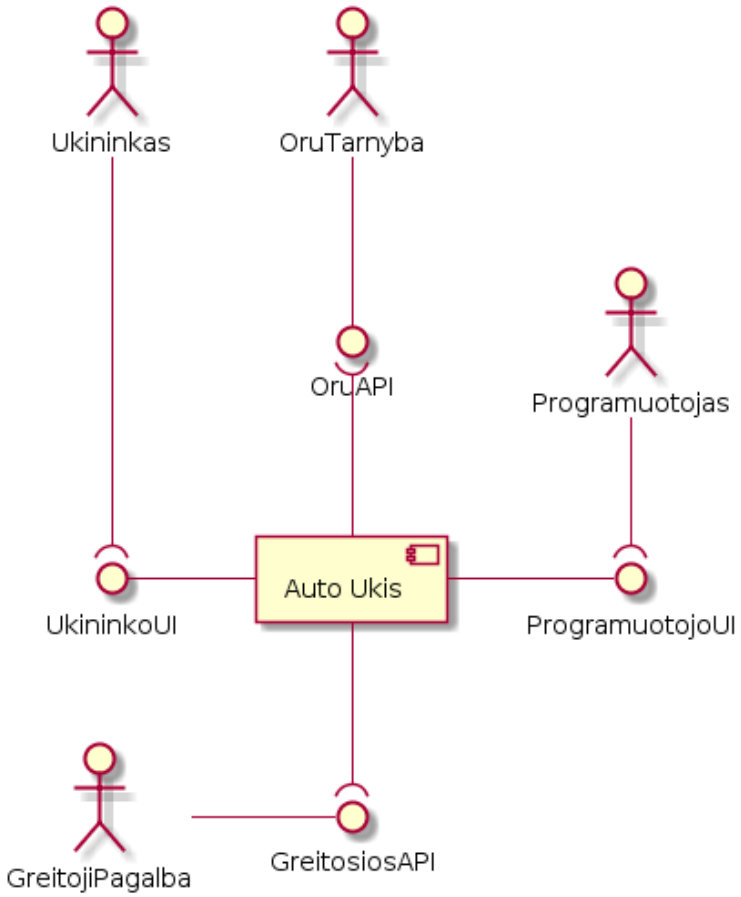

Automatinė Ūkio valdymo sistema
AutoUkis
ArrayList<Gyvunai> gyvunai
ZemesTeritorija plotas
Map map
Point p1,p2
Color spalva;
UkioTechnika tech
JPanel
ZemesDetektorius
Aggregation
Inheritance
Map
ZemesPlotas plotas
List<ZemesPlotai> plotai
int sleptiNustatymus
int mastelis
void paintComponent(Graphics g)
void updateList(ZemesPlotas plotas)
boolean check(Point p)
AriamasLaukas
ZemesParametrai parametrai
public void kultivuoti()
Association
Aggregation
Inheritance
Aggregation
Aggregation
ZemesParametrai
String plotoPavadinimas
int sluoksnis
int storis
int smelisProcentais
int dregme
int ph
Map map
Orai orai
void kadaIsdziusZeme()
ZemesTeritorija
Color spalva
Point p1
Point p2
String msg
Orai
Association
Aggregation
Inheritance
Inheritance
Ganykla
UkinisPastatas
Gyvunas gyvunas
UkioTechnika technika
Aggregation
Aggregation
Gyvunas
String veisle
int id
int amzius
String lytis
float suvartojamasMaistas
String gps
UkioTechnika
List<ZemesPlotas> plotai
Map map
double greitis
void judejimoGreitis()
void arimas()

1 pav

- Dizainas:
  - Pagrindinė klasė yra AutoUkis.form. Joje sukurtas ir aprašytas Graphical User Interface (GUI) ir visas vartotojo bendravimas su programa vyksta per ją, nes per ją pasiekiami visi duomenys iš kitų klasių, pavyzdžiui, duomenys, esantys klasėje Gyvūnas, kurioje įrašoma vartotojo įvesta informacija apie gyvūną (veislė, amžius, t.t.). Taip pat AutoUkis klasėje kuriama dauguma objektų ir jie ten laikomi, sudedami į sąrašus. Visos kitos klasės turi savo atskiras paskirtis, tokias kaip žemėlapio braižymas, oro prognozių sekimas ir įvairių parametrų laikymas. Kai kurios klasės (pvz., ZemesDetektorius) buvo sukurtos vėlesniam panaudojimui, bet šiuo metu nėra niekur panaudotos. Dėl to, ką būtų buvę galima daryti kitaip,- GUI perkėlimas į atskirą klasę padarytų programą skaitomesnę ir tvarkingesnę, būtų lengviau rasti atskirą kodą. Dar viena alternatyva būtų įgyvendinti front-end dalį web aplinkoje, bet šiuo metu nematome tam būtinybės.
- Funkcionalumas:
  - Viso užsibrėžto programos funkcionalumo įgyvendinti nepavyko. Kai kurios klasės buvo sukurtos ateityje planuojamoms funkcijoms, kurios dar nėra implementuotos. Programa kol kas veikia tik ant kompiuterio ir vienintelis jos bendravimas su internetu yra per Orai klasę, kuri skirta vartotojo pasirinkto miesto orų prognozėms gauti iš gismeteo.lt svetainės. Klasėse Gyvunas, Map, ZemesTeritorija ir iš jos išeinančiose klasėse saugomi atitinkami duomenys apie sukurtus objektus bei aprašyti dar neišplėtoti metodai, tokie kaip žemės teritorijos žymėjimas. Planuojama, kad klasė ZemesDetektorius generuos atsitiktinius parametrus, kurie bus perduoti ZemesParametrai klasei. Nepilnai įgyvendintas finkcionalumas ir neišbaigtos klasės sukelia nepatogumų aprašant programą, nes sunku braižyti diagramas, suvokti aiškius ryšius tarp komponentų ir vykdomas funkcijas.

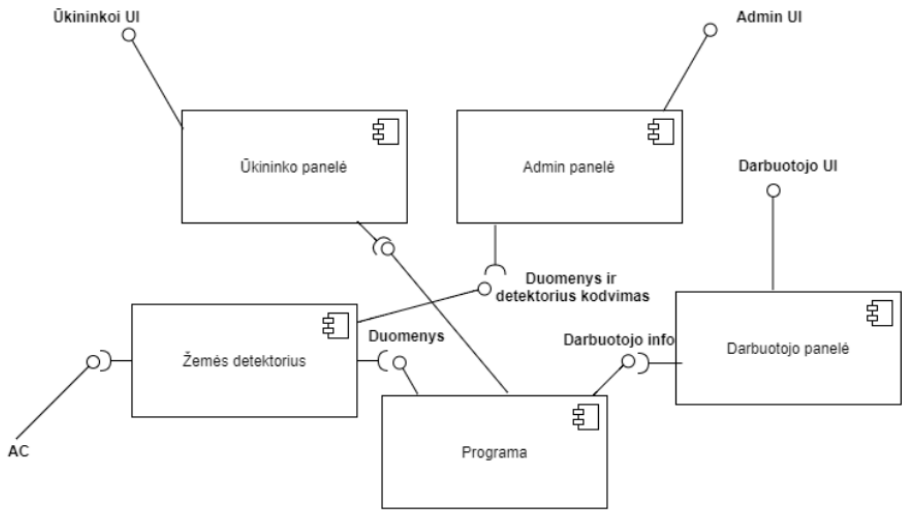
1.2. Kūrimo pjūvis

- Dizainas:
  - Pradėjus rašyti programą nepagalvojome apie kūrimo pjūvį ir kaip teisingiau būtų galima pradėti viską. Žiūrint dabar, visa programa buvo pradėta kurti pagal Bottom -> Up principą. Iš pradžių apsirašėme daugybę mažų klasių ir paskui jas bandėme apjungti į didesnę sistemą. Išskyrėme tokius komponentus kaip ūkininkas, programa, orų tarnyba, žemės detektorius ir administratorius. Kiekvienas komponentas turi skirtingas prieigas prie informacijos ir skirtingas funkcijas, reikalingas ūkio visapusiškam funkcionavimui. Kai kurios klasės liko nepanaudotos, nes šiuo metu jos neatrodo pakankamai svarbios pradiniam projekto variantui. Ūkiniko, Admin ir darbuotojo panelės įtrauktos į dokumentaciją norint pavaizduoti skirtingas prieigas prie sistemos.



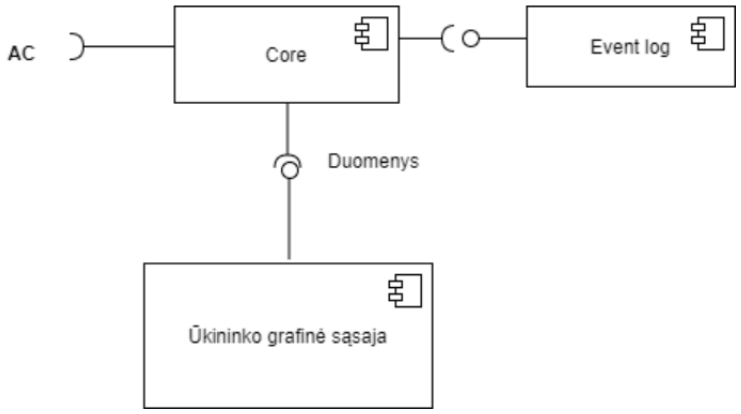
2 pav

- *L0*:
  - *Šioje diagramoje (2 pav)* pavaizdavome sistemos bendravimą su išoriniais agentais, tokiais kaip Greitoji pagalba, Ūkininkas ir t.t. . Ši diagrama aiškiai ir paprastai parodo kuriamus ir įgyvendinamus interfeisus. Galbūt būtų galima Greitosios Pagalbos interfeisą išskaidyti į kelis detalesnius interfeisus, bet apskritai didelių problemų nepastebime.



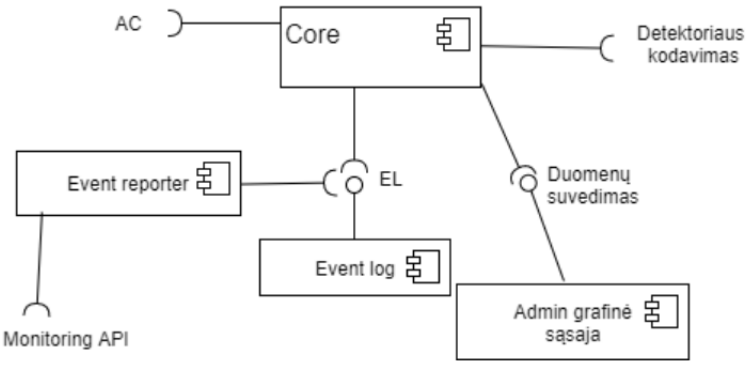
3 pav

- L1: Sudėjus komandos idėjas apie tai, kaip turėtų atrodyti *L1 diagrama(3 pav)*, supratome, kad mūsų sistema neturi normalios struktūros ir gerai nebuvome pagalvoję kaip visi komponentai siesis vieni su kitais, todėl ir diagrama atrodo chaotiška. Trūksta konkretumo, kaip turi Admin sietis su kitas komponentais. Programa atsiranda kaip komponentas, o tai greičiausiai yra nekorektiška.



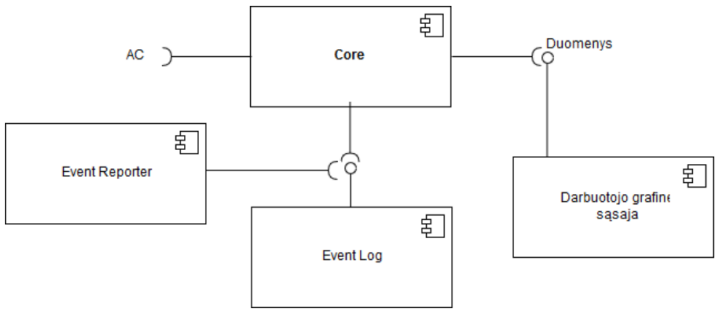
4 pav

- L2(Ūkininkas): *Šioje diagramoje (4 pav)* parodyta, kad programos pagrindas (Core) kuria interface'ą, kuriuo perduoda duomenis grafinei vartotojo sąsajai. Duomenys yra perduodami Event Log komponentui.



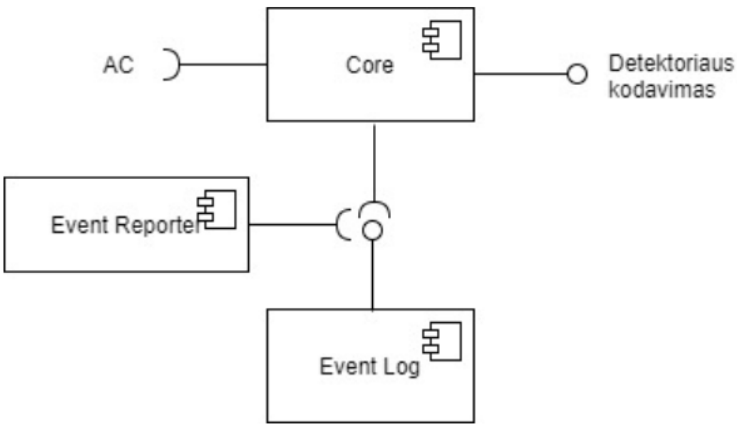
5 pav

- L2(Admin) *Šioje diagramoje (pav 5)* parodyta, kad programos pagrindas naudoja duomenų suvedimo interface, kurį suteikia admin grafinė sąsaja, bei naudoja Detektoriaus kodavimo interface. Visus įvykius įrašo į Event Log.



6 pav

- L2(Darbuotojas): *Šioje diagramoje (pav 6)* parodyta, kad programos pagrindas naudoja grafinę sąsają ir perduoda duomenis į Event Log'ą.



7 pav

- L2(Detektorius): *Ši diagrama (pav 7)* vaizduoja detektoriaus išvedamus duomenis. Įvykiai įrašomi Event Log'e.

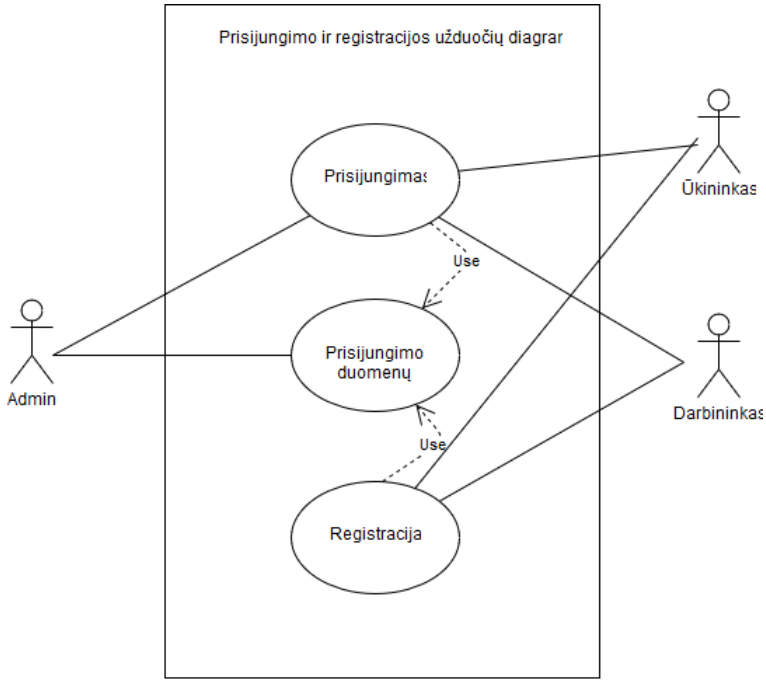
Komponentai / Užduotys	Detektorius	Orų tarnyba	Programa	Ūkininko UI	Admin UI	Greitoji pagalba
Registruoti gyvūnus			+	+		
Orų prognozė		+				
Žemės parametrai	+					
Duomenų suvedimas			+	+	+	
Skaityti ir rašyti event log'ą			+		+	
Samdyti darbuotojus				+		
Kviesti pagalbą			+	+		+
Duomenų skaitymas			+	+	+	
Verslo reikalų tvarkymas			+	+		
Išteklių palaikymas				+		

8 pav

- Elementų ir užduočių ryšių matrica.

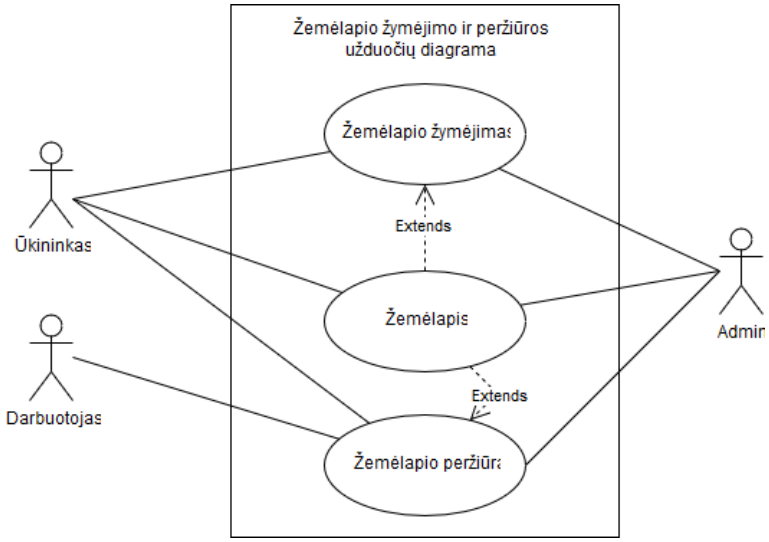
1.3. Use cases

- Prisijungimo ir registracijos Use Case:



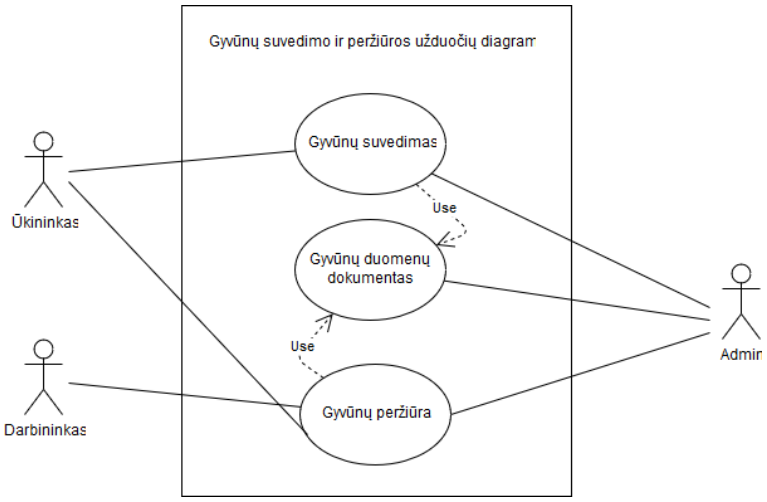
9 pav

- Žemėlapio žymėjimo ir peržiūros Use Case:



10 pav

- Gyvūnų suvedimo ir peržiūros Use Case:



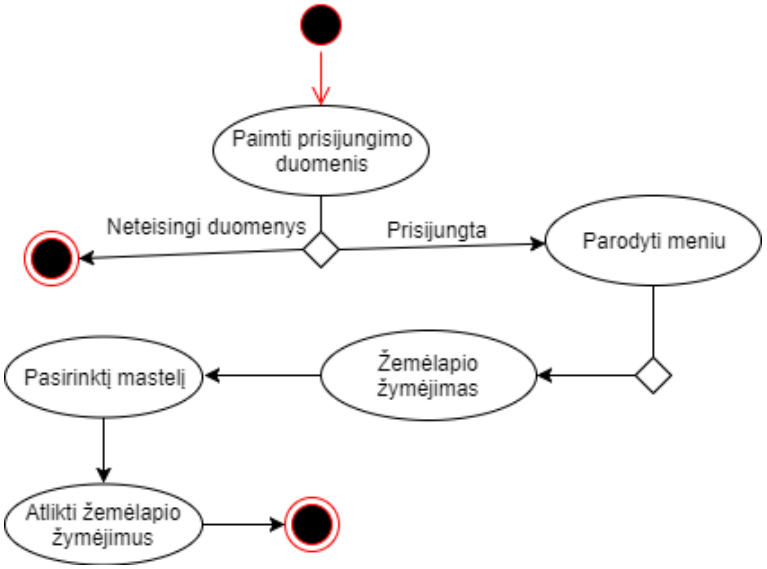
11 pav

1.4. Proceso pjūvis

Šiame skyriuje parodoma programos elgsena jos vykdimo metu.

1.4.1. Veiklos diagramos

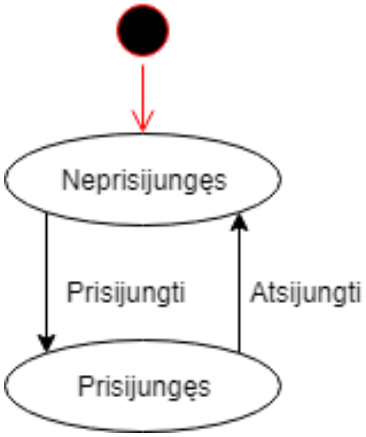
Vaizduojama *žemėlapio žymėjimo veiklos diagrama (pav 8)*. Nurodomi pagrindiniai žingsniai braižant žemėlapį.



12 pav

1.4.2. Būsėnų diagrama

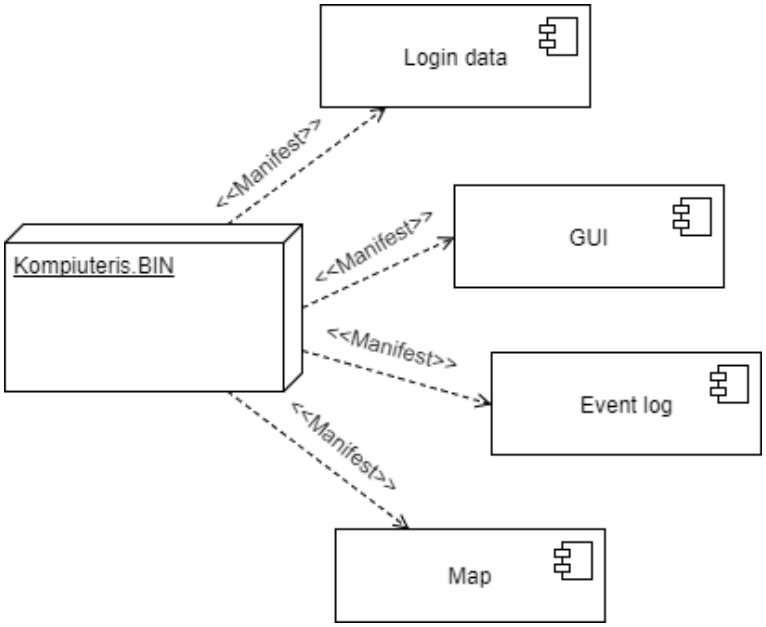
Šioje diagramoje (pav 9) parodomos galimos vartotojo būsenos ir keliai jom pasiekti. Šiuo metu programoje tėra dvi būsenos, taigi diagrama yra labai paprasta.



13 pav

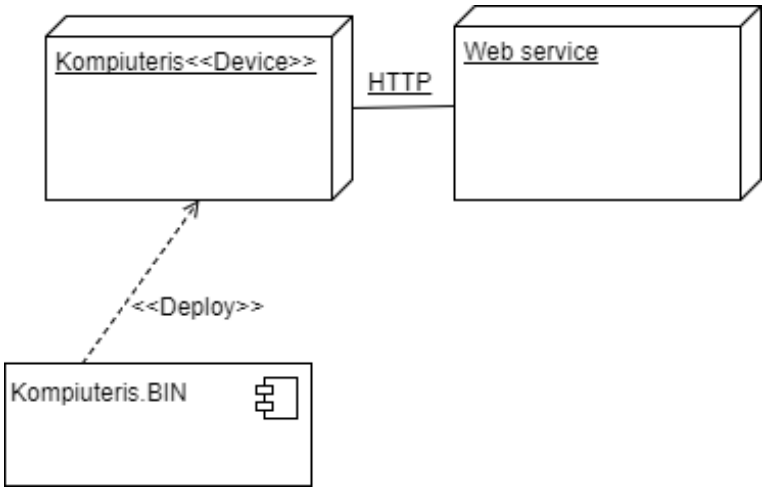
1.5. Fizinis pjūvis

Šiame skyriuje parodoma (10 pav.) šios sistemos techninė įranga, komunikacija, tačiau kadangi šioje šabloninėje versijoje nenaudojame kitų techninių resursų be kompiuterio, fizinis pjūvis parodo tik nedidelį kiekį informacijos.



14 pav

- D0: *Šioje diagramoje(11 pav)* parodyta, kas saugoma device kompiuteris. Iš diagramos matome, kad šiame įrenginyje saugomi prisijungimo duomenys, vartotojų grafinės sąsajos, teritorijos žemėlapis bei programoje atliktų veiksmų išrašas. Alternatyva buvo saugoti šiuos duomenis išnuomuotame web service, tačiau daug negalvoję nusprendėme duomenis saugoti kompiuteryje.



15 pav

- D1: Tai *bendresnė D0 diagrama*, joje matome, kad kompiuteris, norėdamas gauti pranešimus apie orus, bendrauja su web HTTP ryšiu.

1.6. Pirmos dalies išvada

Pirmoji programos versija buvo suprogramuota ir suprojektuota perdaug negalvojant apie sistemos plėtimą ateityje. Nors klasėse ir išlaikėme enkapsuliavimo principą, bendras objektinio programavimo principas nebuvo išlaikytas, klasės yra per daug viena nuo kitos priklausomos. Rimtesniai sistemai kurti reiktų naudotis Top to Bottom principu lengvesniam naujų funkcionalumų pridėjimui. Stipri ir paprasta sistemos dalis pasimato kūrimo pjūvyje, kuriame yra aiškūs ryšiai tarp kuriamų ir įgyvendinamų interface'ų. Aprašinėdami L1 ir L2 supratome, kad nepagalvojome apie tai, kaip sistemos vartotojai bendrauja su sistema. Šią dalį reiks pergalvoti antroje programos versijoje, kad viskas taptų aiškiau ir paprasčiau, reikia labiau pasidomėti, kaip tokios sistemos veikia realiame pasaulyje. Proceso pjūvį pavyko aprašyti pakankamai paprastai ir aiškiai, tačiau jam dar trūksta detalumo ir kitų scenarijų numatymo, pavyzdžiui, prisijungti prie sistemos. Fiziniame pjūvyje aprašyta techninė sistema yra ganėtinai paprasta ir primityvi. Pagrindinis trūkumas - nepagalvota, kas nutiktų kompiuterio išsijungimo atveju, ar kas nutiktų atsiradus poreikiui plėsti sistemą. Žiūrint bendrais idėjos bruožais, brangiausia sistemos dalis yra automatinės mašinos, kurios dar yra sąlyginai nauja technologija, ir mikro kontroleriai kiekvienam gyvūnui. Jeigu ferma turi jų daug, visas instaliavimas kainuotų gana brangiai ir turbūt neatneštų didelės naudos. Tai labiau inovacija dėl inovacijos, ne dėl funkcionalumo. Tačiau kitos dalys visai sekmingai pritaikomos. Tokią sistemą,kokią padarėme dabar, įgyvendinti būtų įmanoma, tačiau praplėsti ir palaikyti ją būtų nepatogu, ir sistema turbūt nedirbtų taip greitai, kaip norėtūsi. Trūksta funkcionalumo su išmaniuoju telefonu, darbo birža, rinkos tendencijomis ir kita.

2. Perprojektuotos sistemos aprašymas(To-Be, v2.0)

2.1. Loginis pjūvis

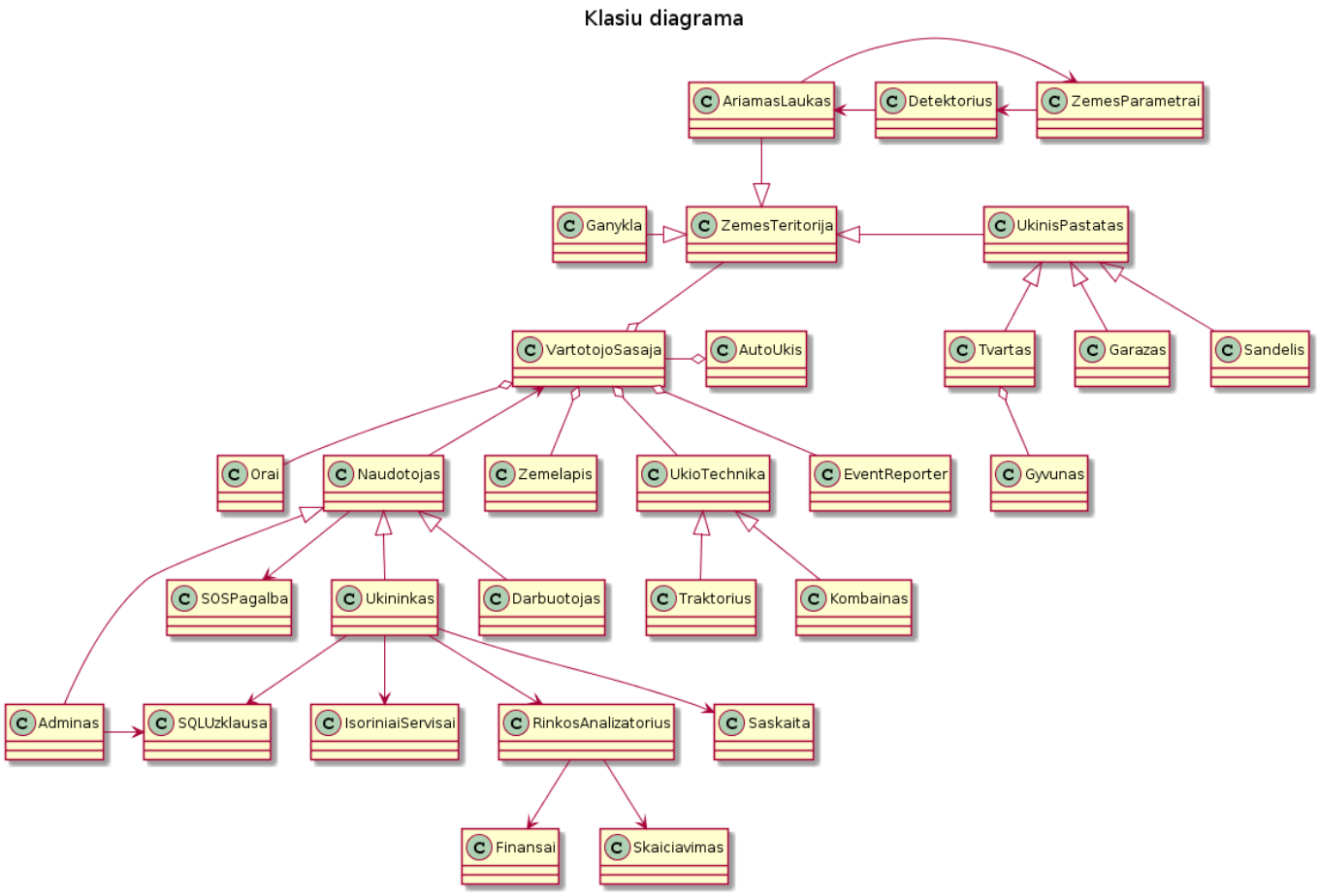
2.1.1. Žodynas

- Klasės:
  * AutoUkis - pagrindinė (main) programos klasė. Ši klasė piešia grafinę vartotojo sąsają ir laiko savyje kitų klasių objektus, kurių informacija reikalinga piešimui.
  * Zemelapis - teritorijos piešimui skirta klasė.
  * ZemesTeritorija - apskaičiuoja tam tikros teritorijos plotą.
  * Gyvunas - klasė, skirta gyvūno rodmenims ir metodams saugoti.
  * AriamasLaukas - laiko savyje reikšmes, apibūdinančias unikalų lauką, ir metodus, susijusius su lauko darbu.
  * Ganykla - laiko parametrus ir metodus darbui su ganyklomis, kurios yra žemės plote.
  * UkinisPastatas - saugo ūkinę techniką arba gyvūnus.
  * UkioTechnika - laiko ūkio technikos duomenis ir apskaičiuoja technikos judėjimo greitį.
  * ZemesParametrai - saugo įvairius žemės parametrus (drėgmė, pH...).
  * Orai - klasė, skirta gauti vartotojui reikalingas orų prognozes iš www.gismeteo.lt.
  * Detektorius - klasė, skirta bendrauti su žemės detektoriumi.
  * VartotojoSasaja - programos grafinė vartotojo sąsaja.
  * Tvartas - pastatas, kuriame laikomi ūkio gyvūnai.
  * Garazas - pastatas, kuriame laikoma ūkio technika.
  * Sandelis - sandėlyje laikomi ūkio ištekliai.
  * Naudotojas - žmogus, kuris naudojasi programa.
  * SOSPagalba - šioje klasėje iškviečiama pagalba pavojaus atveju.
  * Ukininkas - žmogus, kurio valdomoje teritorijoje įdiegtas Auto ūkis.
  * Darbuotojas - žmogus, dirbantis ūkininko versle.
  * Traktorius - atlieka žemės padargų traukimo funkciją.
  * Kombainas - nuima grūdų derlių.
  * Adminas - administratorius, atsakingas už tvarkingą programos veiklą.
  * SQLUzklausa - uzklausa, gauti duomenis iš duombazės.
  * IsoriniaiServisai - kitos paslaugos (pvz., darbo birža).
  * RinkosAnalizatorius - renka duomenis apie rinkos naujienas, aktualias naudotojui.
  * Saskaita - laikomi sąskaitos duomenys ir funkcijos veiksmam su ja atlinkti.

    * Finansai - laikoma informacija apie ūkio finansus ir aprašytos funkcijos veiksmam su jais.
    * Skaiciavimas - atlieka kompleksinius skaičiavimus.
    * EventReporter - kalsė, kuri rašo įvykius į Event Log.
    * EventLog - įvykių programoje įrašai.

- Bendri terminai:
    * Žemės plotas - vieta, kurią valdo ir gali stebėti vartotojas (ūkininkas).
    * Detektorius - Arduino mikro kontroleris.
    * Arduino - mikro kontroleris, skirtas ūkio sekimui.
    * Automatiškai valdoma - valdymui nereikalinga žmogaus pagalba.
    * Gyvūnas - visi gyvūnai, kurie priklauso ūkininkui, ir yra registruoti Auto ūkis sistemoje.

2.1.2. Klasių diagrama



16 pav

- Dizainas:

  - Visas programos dizainas paremtas Top to Bottom principu. Pradėjome galvoti dideliais objektais ir juos išskirstėm į mažesnius. Pagrindinė klasė yra AutoUkis, kuris iškviečia VartotojoSasaja klasę, kuri ir piešia visą prieinamumą vartotojams. Galima nesunkiai pridėti kitokią vartotojo sąsają ar bet kokią kitą UI piešimo galimybę, kaip ,pavyzdžiui, piešti UI išmaniąjame telefone. Programos modalumas leidžia nesunkiai pridėti funkionalumo, nes programos dalys yra atskiros viena nuo kitos. Pavyzdžiui, ištrynus klasę ŪkinisPastatas visos programos veikimas nesutiriktų. Tuo pačiu metu norint pridėti naują ūkinį pastatą galima nesunkiai tai padaryti įvykdžius paveldėjimą iš tėvinės UkinisPastatas klasės taip, kaip tai daro klasės Garažas, Tvartas ir Sandelis.

- Funkionalumas:

  - Programos funkcionalumas susideda iš trijų pagrindinių dalių, vartotojų prieiga prie sistemos, finansų bei išteklių(žmogiškųjų ir natūraliųjų) valdymas, išteklių sekimas. Prieigos dalyje funkcionalumas pasižymi prieigos išskaidymą. Skirtingi sistemos vartotojai gauna skirtingą prieigą ir galimybes naudotis sistema. Pavyzdžiui, darbuotojas negali tvarkyti įmonės finansų, tačiau visi sistemos vartotojai gali kviesti greitąją pagalbą. Nesunku implementuoti naują vartotojų klasę (pvz. Svečias). Finansų bei išteklių dalyje ūkininkas gali tvarkyti savo išteklius, pavyzdžiui, samdyti arba atleisiti darbuotojus, stebėti rinkos kainą ir nuspręsti, kada jam palankiausia parduoti, sudarinėti sąskaitas ir tvarkyti kitą buhalteriją. Trečiojoje sistemos dalyje yra įgyvendinamas išteklių sekimas. Vartotojai, turintys prieigą gali stebėti žemės parametrus, ūkio technikos sąrašą, ūkinių pastatų sąrašą bei gyvūnų, priklausančių sistemai, sąrašą.

- Elementų ir užduočių ryšių matrica:

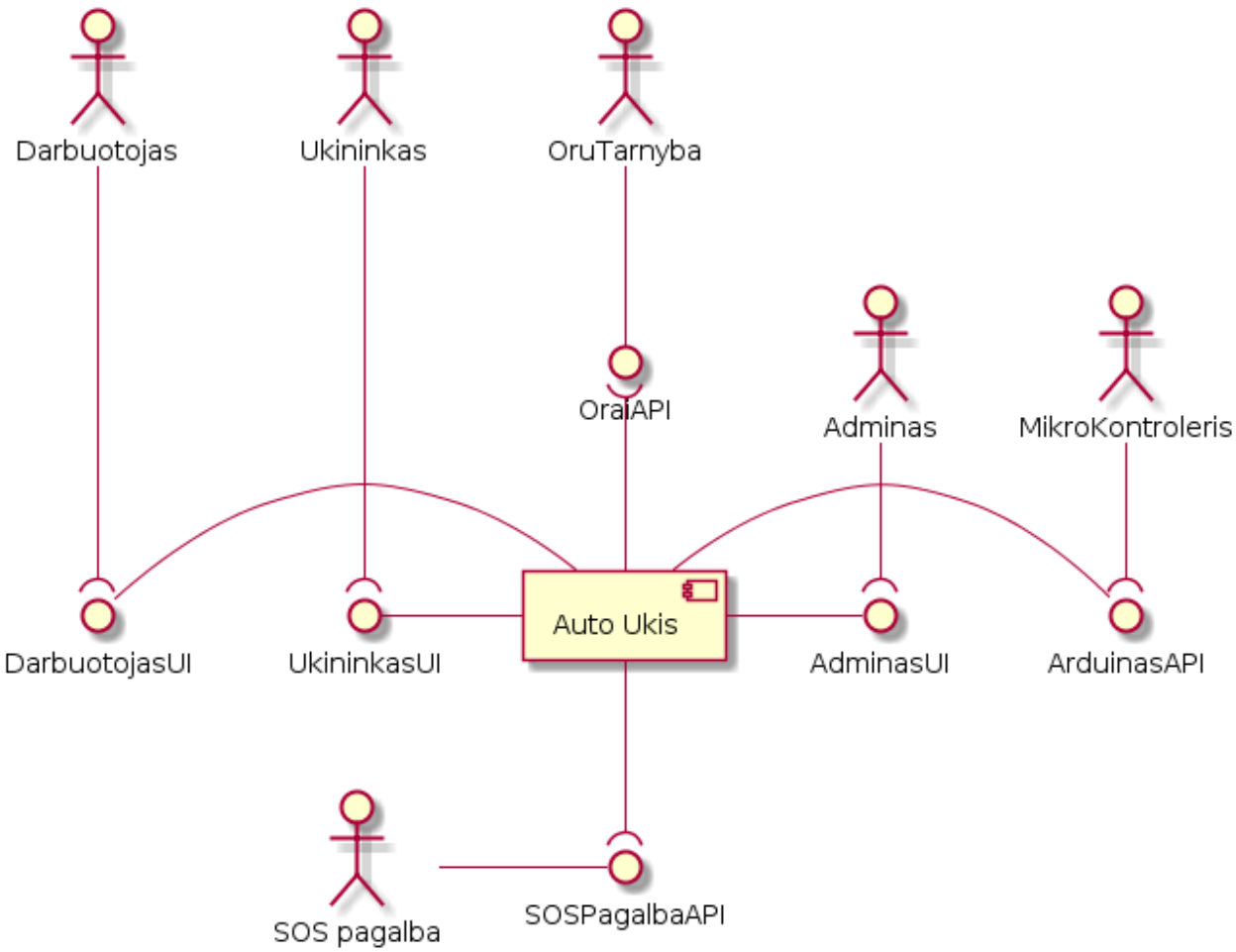
Komponentai / Užduotys	Ūkininkas	Darbuotojas	Adminas	Detektorius	Orų tarnyba
Samdyti darbuotojus	+				
Programuoti detektoriu			+		
SOS pagalbos kvietimas	+	+	+		
Gauti Event Log	+		+		
Resursų sekimas	+				
Technikos lokacijos sekimas	+		+		
Buhalterijos tvarkymas	+				
Pažymėti teritorijas	+				
Orų peržiūra	+	+			
Įvesti gyvūnų duomenis	+				
Peržiūrėti gyvūnų duomenis	+	+			
Peržiūrėti žemėlapį	+	+			
Suvesti žemės parametrus				+	
Sevesti orų prognozes					+

17 pav

2.2. Kūrimo pjūvis

- Dizainas:

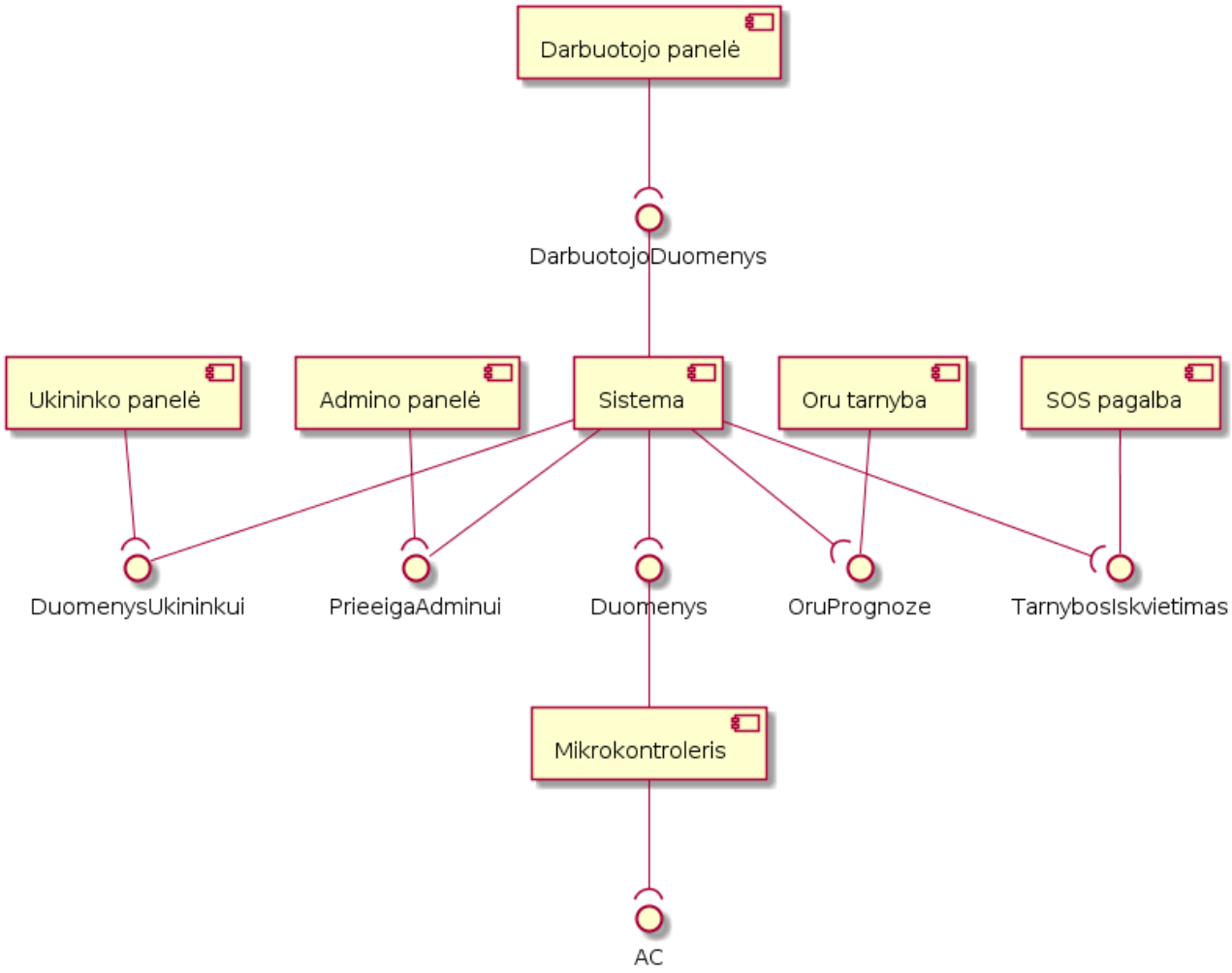
– Programos dizainas stipriai paremtas Top to bottom projektavimo principu ir objektinio programavimo enkapsuliacijos paradigma. Sistema sukuria ir įgyvendina kitų sistemų intefeisus. Pagrindiniai trys interfeisai sukuriami programos yra UkininkasUI, AdminasUI. Šie interfeisai suteikia prieeiga prie sistemos skirtingas privilegijas turintiems vartojams. Auto Ukis taip pat įgyvendina kitų sistemų suteikiamus interfeisus kuriuos naudoja progamos logikoje duomenis gauti. Įgyvendinami interfeisai yra OruTarnybaAPI, SOSPagalbaAPI ir MikrokontrolerisAPI.



18 pav

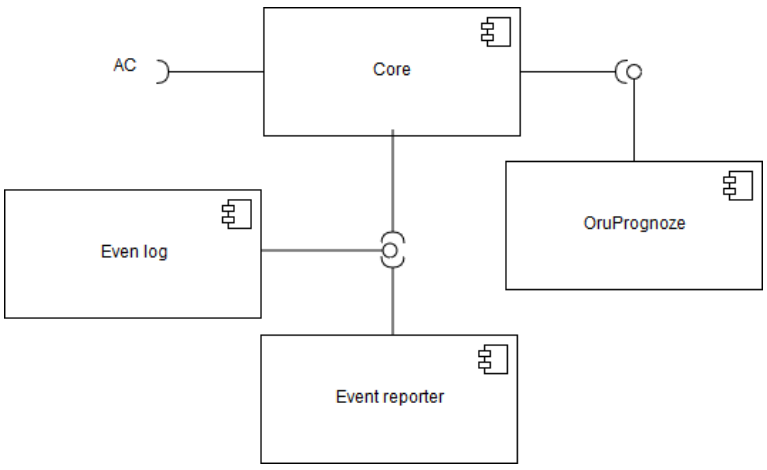
- L0:

– Šioje diagramoje pavaizdavome sistemos bendradarbiavimą su išoriniais agentais, tokiais kaip Mikrokontroleris, Ūkininkas ir t.t. . Ši diagrama parodo sistemos įgyvendinamus ir kuriamus interfeisus. Iš diagramos matome, kad programos pagrindas kuria interface ne tik vartotojams, bet ir tokiems išoriniams agentams, kaip SOS pagalba, bei Orų tarnyba.

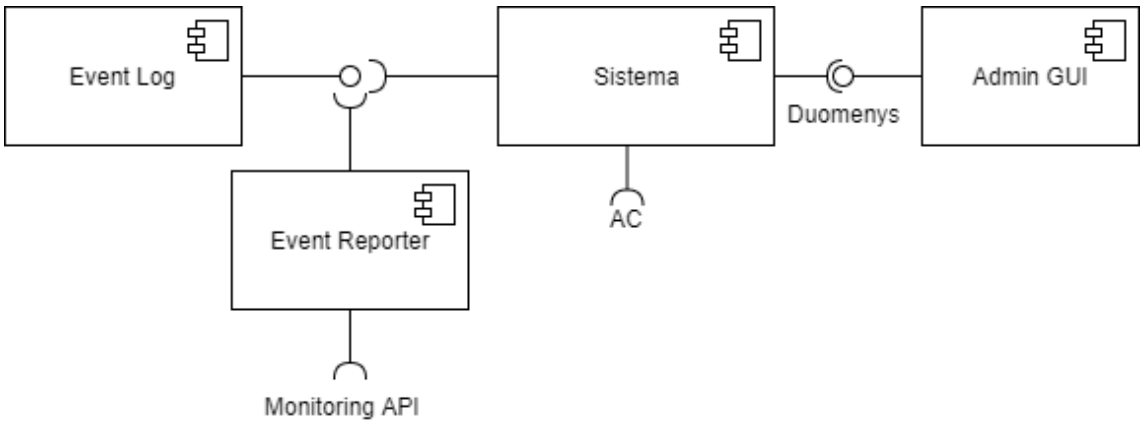


19 pav

- L1: Sistemos struktūra ganėtinai aiški. Yra trys panelės kurios pasiima sisteminius duomeris ir trys panelės kurios suteikia duomenis sistemai. Kadangi struktūra paprasta nesunku prie jos pridėti naujų komponentų ar gauti duomenis iš naujų komponentų.
- L2(Orų tarnyba):

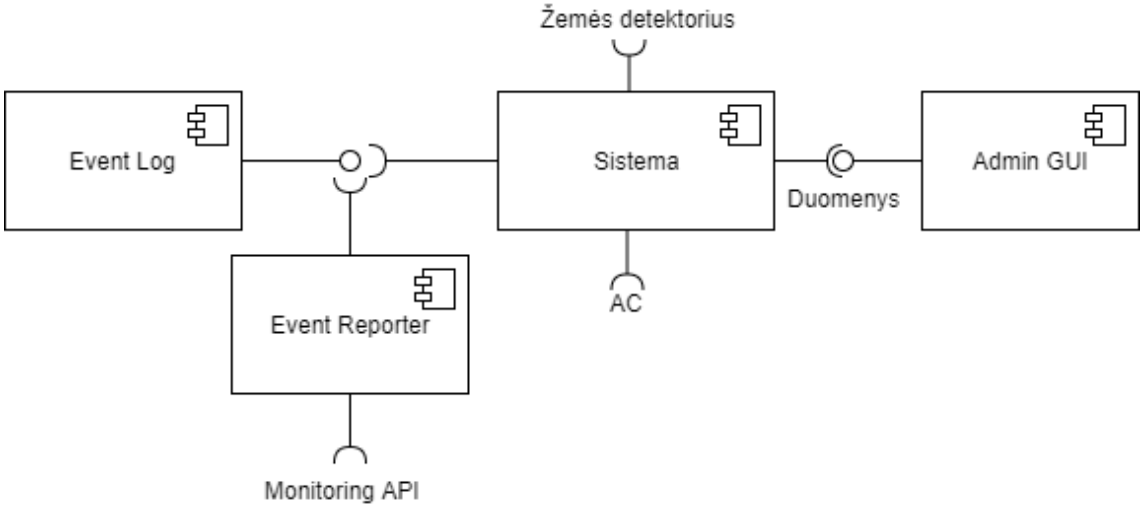


20 pav



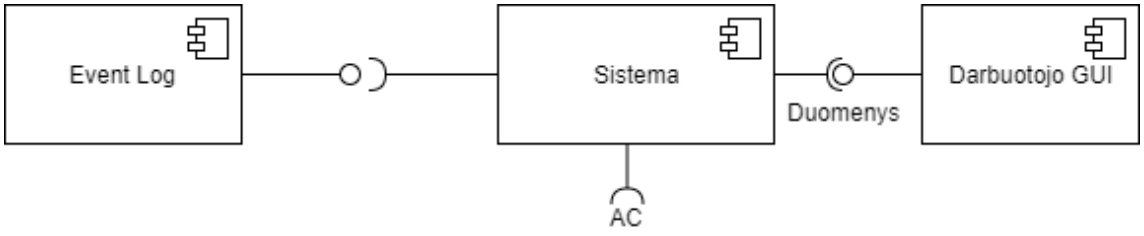
21 pav

- L2(Ūkininkas): Ūkininkas turi prieigą prie sistemos, GUI, Event Reporter ir Event Log.



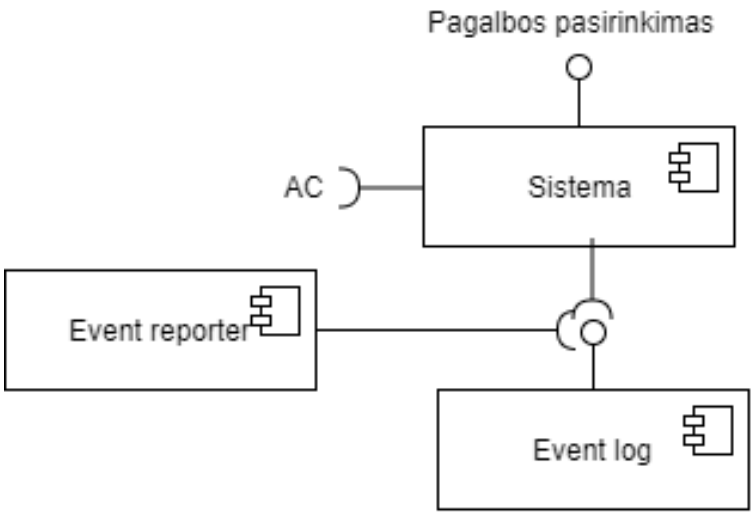
22 pav

- L2(Admin): Diagramoje pavaizduota Admin sąsaja su sistema, GUI ir Event Log. Admin taip pat turi žemės detektoriaus prieigą.



23 pav

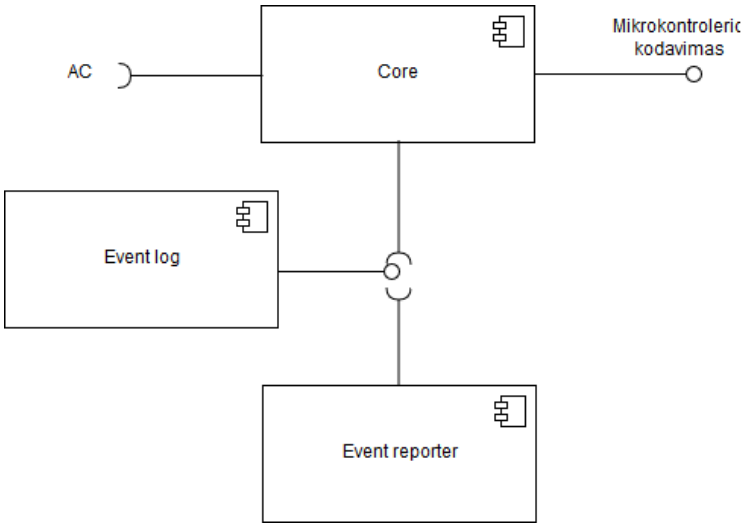
- L2(Darbuotojas): Diagramoje pavaizduota darbuotojo prieiga prie sistemos, GUI ir Event Log.
- L2(SOS pagalba):



24 pav

Šioje diagramoje matome, kad core kuria interface reikiamos pagalbos pasirinkimui, kurį įgyvendins vartotojas iškvietęs SOS pagalbą.

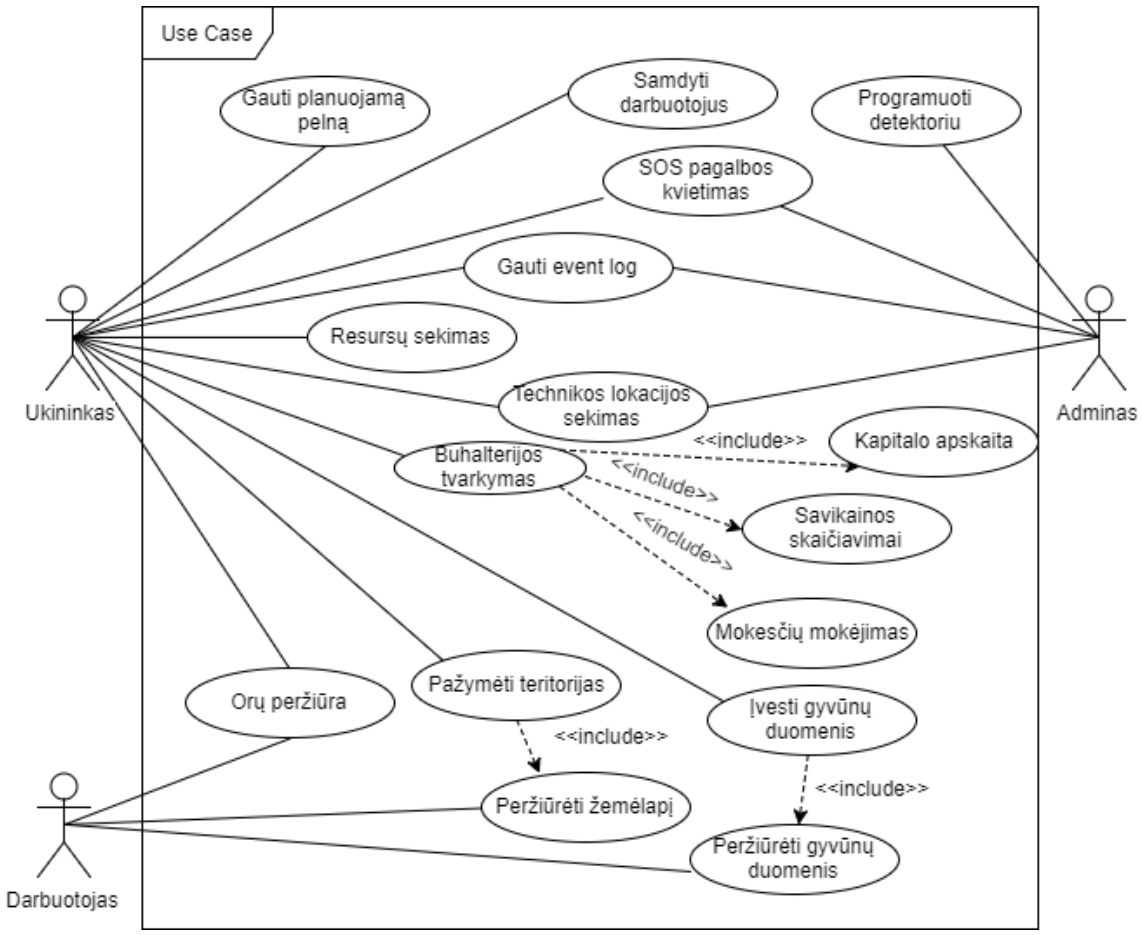
- L2(Mikrokontroleris):



25 pav

2.3. Use case

- Diagramoje (22 pav) pavaizduoti visi galimi veiksmai, kuriuos gali atlikti kiekvienas programos vartotojas.



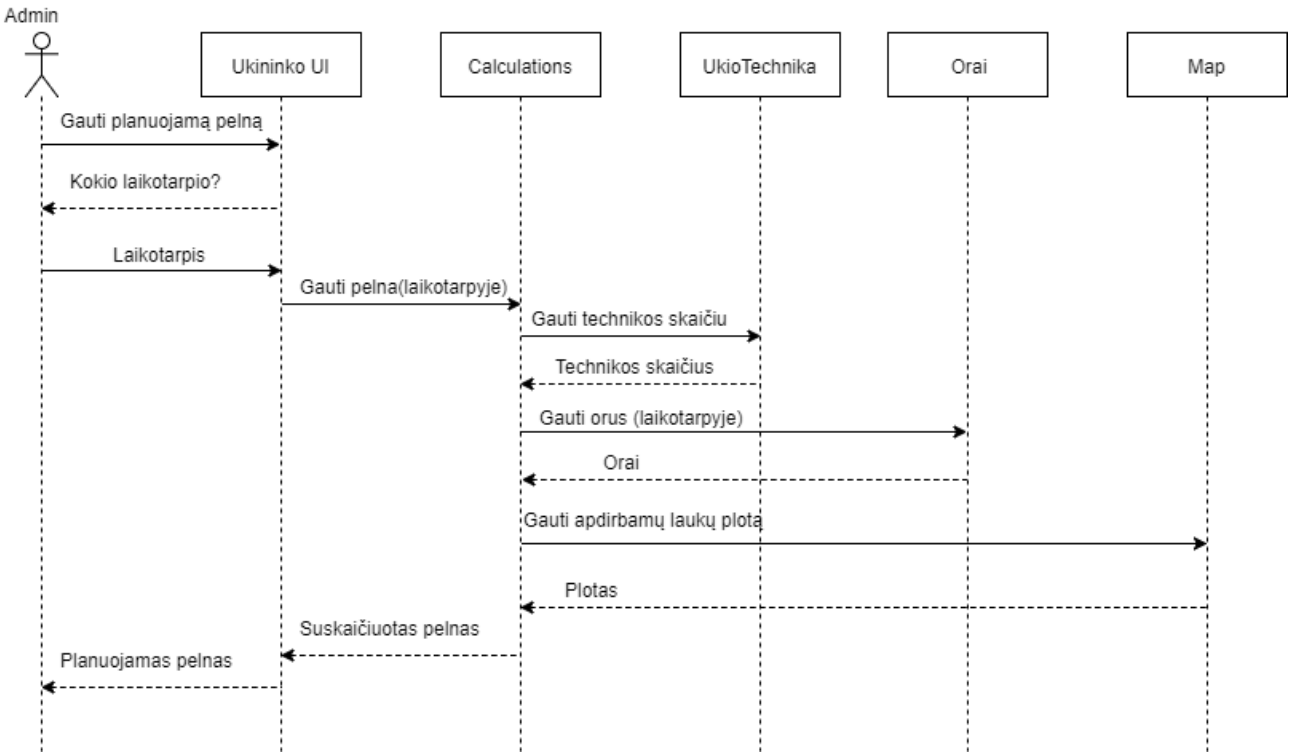
26 pav

2.4. Proceso pjūvis

Šiame skyriuje pagrinde koncentruojamasi į programos elgseną jos vykdymo metu.

2.4.1. Sekų diagramos

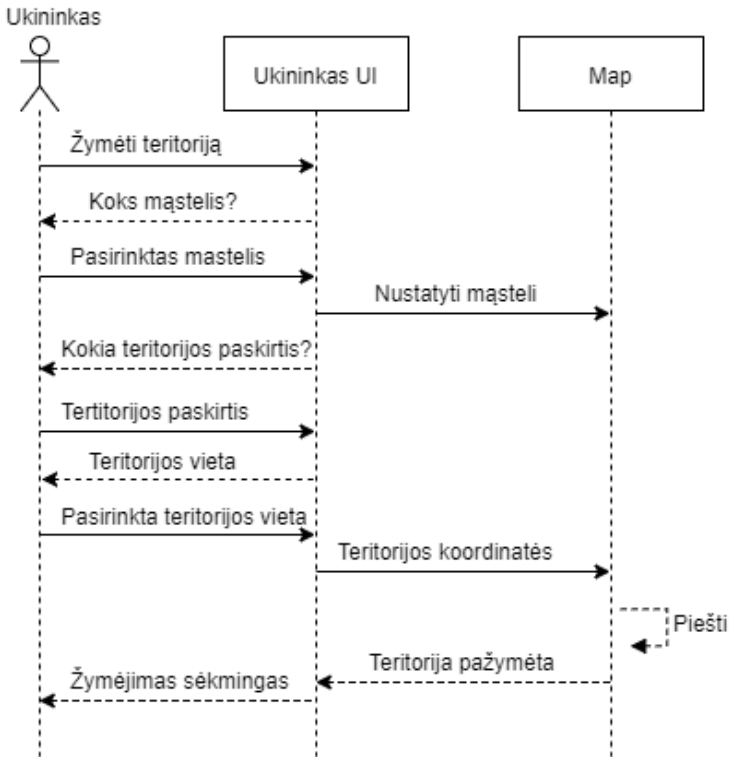
- Diagramoje (12 pav) pavaizduotos seka, kurią programa atlieka Admin norint gauti planuojamą jo pasirinkto laikotarpio pelną.



27 pav

Šioje diagramoje matome, kad ūkininkui pateikus užklausą planuojamui pelnui gauti, Ūkininkas UI kreipiasi į Calculations klasę su prašymu jį apskaičiuoti. Ši savo ruožtu norėdama gauti reikiamus duomenis kreipiasi į klases UkioTechnika, Map ir Orai.

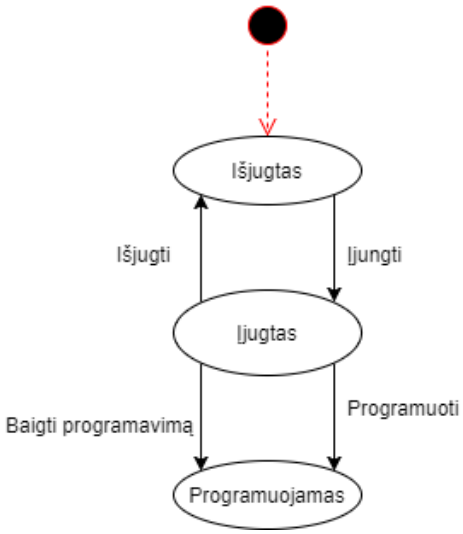
- Teritorijos žimėjimo seka:



28 pav

2.4.2. Būsenų diagramos

- Detektoriaus būsenų diagrama:

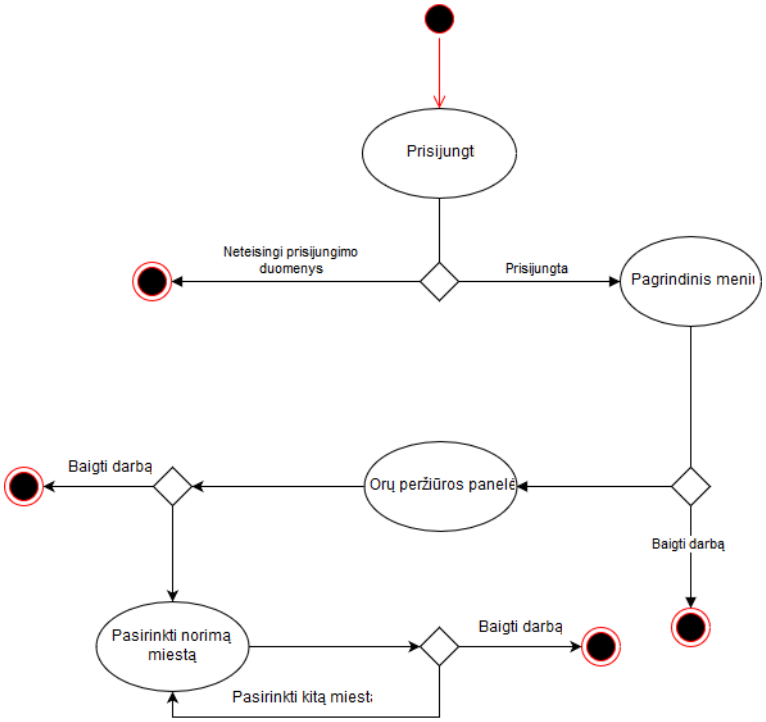


29 pav

Iš šios diagramos matome, kad detektorius negali būt išjungtas programavimo metu, kad nekiltu klaidos vėlesnių paleidimų metu.

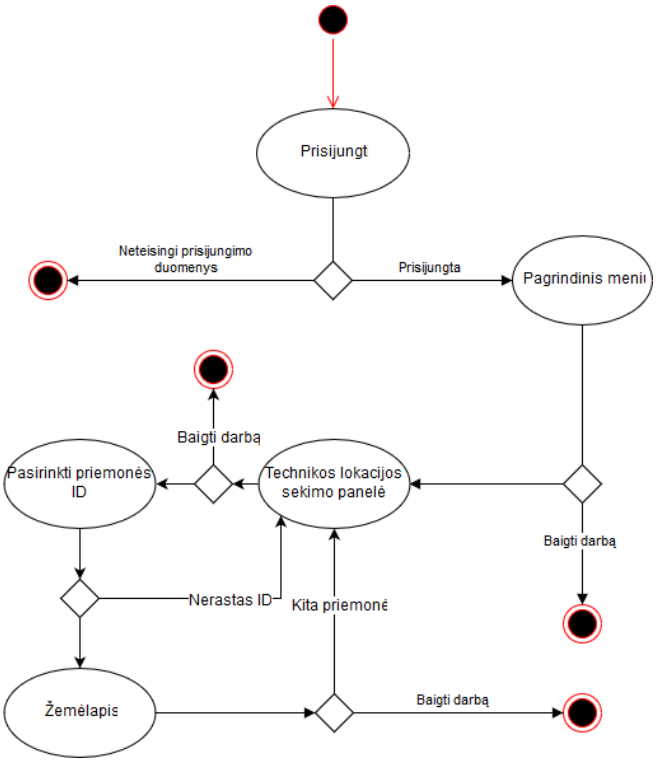
2.4.3. Veiklos diagramos

- Orų sekimo veiklos diagrama:



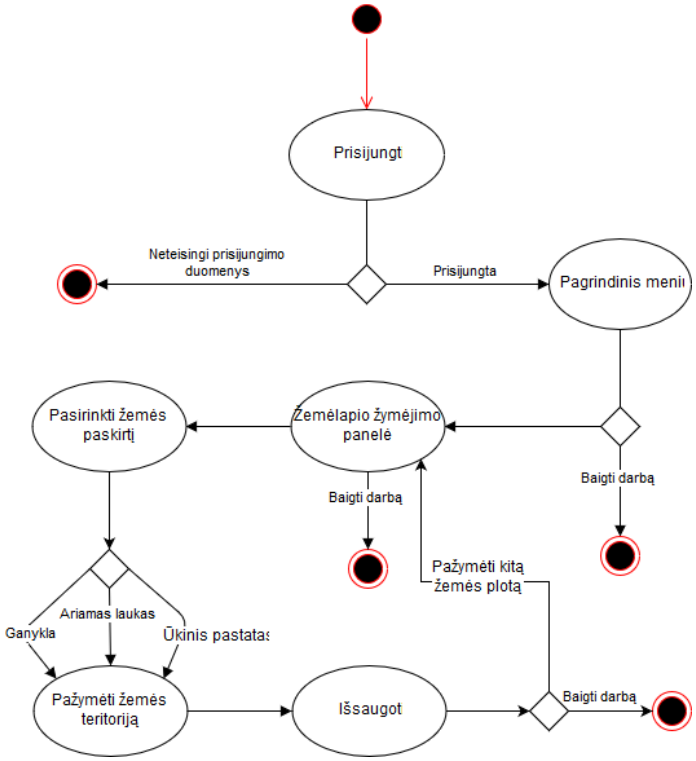
30 pav

- Ūkio technikos sekimo veiklos diagrama:



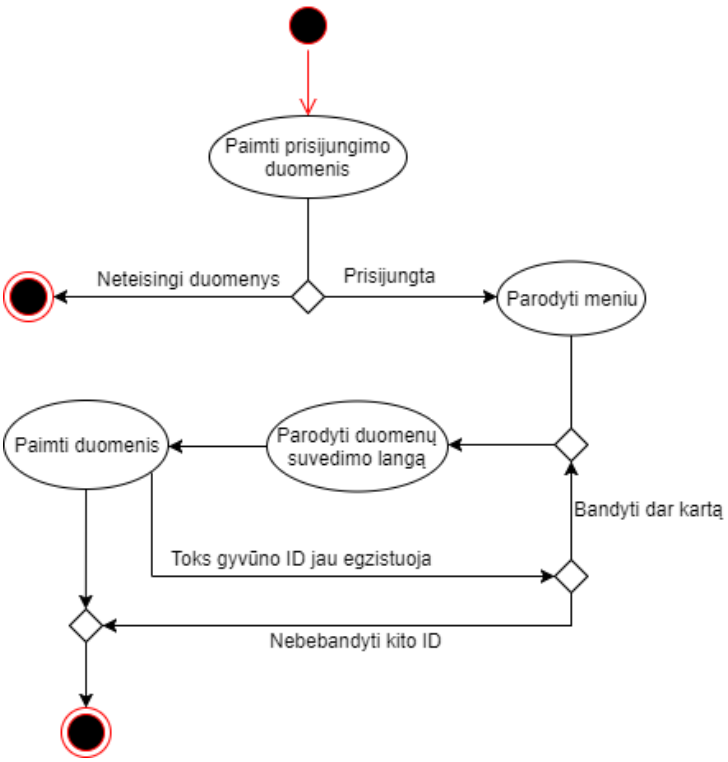
31 pav

- Teritorijos žymėjimo veiklos diagrama:



32 pav

- Gyvūnų suvedimo veiklos diagrama:

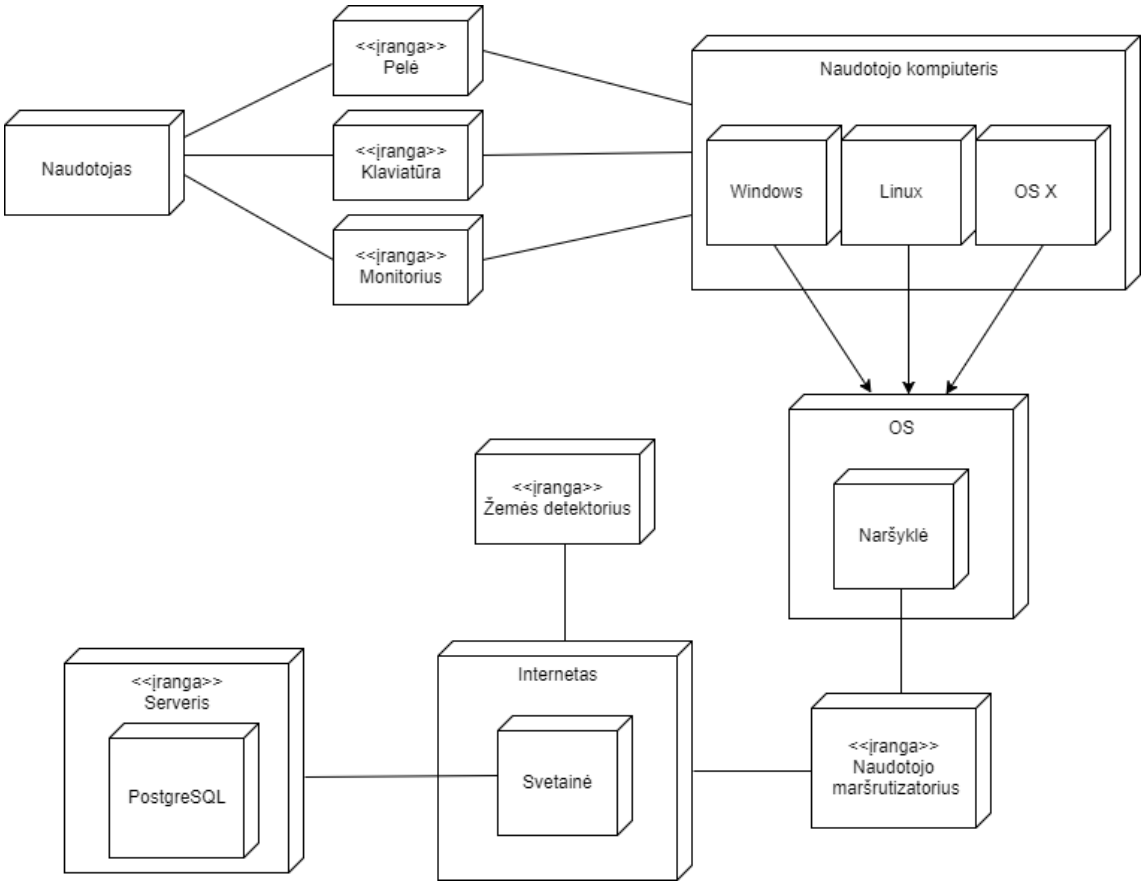


33 pav

- Šiose diagramose matoma kaip įgyvendinamas atitinkamas scenarijus programos vykdymo metu.

2.5. Fizinis pjūvis

Šiame skyriuje parodoma programos naudojama aparatinė įranga, komunikacija tarp tinklo mazgų bei programos komponentų išdėstymas juose.



34 pav

- D0: *Šioje diagramoje (17 pav.)* parodyta, kaip išsaugomi programos duomenys. Pirmoje projekto versijoje visi duomenys buvo saugomi kompiuteryje, vienintelis ryšys su internetu buvo per gismeteo.lt svetainę. Dabartinėje versijoje implementuotas duomenų saugojimas PostgreSQL duombazėje. Šioje diagramoje vaziduojamas kelias nuo naudotojo iki pasirinktos duombazės, naudojami techniniai įrenginiai ir jų sąryšiai su programinė įranga. Taip pat pridėtas žemės detektorius, kuris siunčia programai pasirinkto žemės ploto parametrų duomenis. Informacija, gauta iš žemės detektoriaus, taip pat talpinama PostgreSQL duombazėje.

2.6. Antros dalies išvada

Sistema suprojektuota tokiais principais, kad ją būtų lengva papildyti. Klasių dizainas atitinka Top->Bottom principą, kuris leidžia nesunkiai leidžia pridėti naujo funkcionalumo visai sistemai. Šis principas persikelia ir į kūrimo pjūvi. Sistemos įgyvendinami ir kūremi interfeisai yra labai modularūs ir nesunku pridėti ar išimti interfeisus iš sistermos. Vidinė programos struktūra sekų diagramoje aiškiai apibrėžia vartotojų sąveika su sistema. Sekos išskirstytos diskrečiais ir aiškiais veiksmais kas sumažina klaidų skaičių rašant kodą ir padeda suprasti kaip viskas veikia. Iš fizinės pusės sistema yra serveryje ir dėl pasirinktos JAVA programavimo kalbos sistemą galimą pasileisti iš visų operacinių sistemų. Visa sistema yra serveryje. Duomenims saugoti psirinkome PostgreSQL nes tai populiariausia nemokama duomenų bazių valdymo sistema. Sistemai yra vietos tobulėti. Ateityje egzistuoja galimybe ją pritaikyti išmaniesiams telefonams bei kitiems nešiojamiems įrenginiams.

Rezultatai ir išvados

Projektuodami tą pačią sistemą du kartus skirtingais būdais pamatėme aiškius stilių skirumus. Kuriant sistemą nesilaikant jokių aiškių principų kodas ir pati sistemos struktūra tampa neaiški, po kiek laiko prisimeni apie neįgyvendintus funkcionalumus arba idėjas kurias įdėti į sistemą būtų labai sunku. Projektuojant sistemą antrą kartą buvo prisilaikyta Top->Bottom ir OOP principų kurie leido lengviau struktūrizuoti visą darba. Klasių ir komponentų diagramos pasidarė aiškiai suprantamos ir nauji funkcionalumai būtų nesunkiai įgyvendinami ateityje. Buvo aiškiau apibrėžtas fizinis sistemos principas leidžiantis lanksciau naudotis sistema. Deja ne viską pavyko pridėti. Norėtūsi įdėti išmaniojo telefono palaikymą mūsų sistemoje, bet dėl laiko stokos to nebepadarėme. Tačiau tikime, kad dėl struktūros pranašumų tai padaryti nebūtų sunku. Išmokome fundamentalius PSI aspektus, UML diagramų braižymą, išmokome naudotis keletu UML braižymo programų ir .pdf failų kurimo programa Latex, kuris palengvino darbą komndoje. Dėl gero darbų pasidalijimo projekto kūrimas vyko sklandžiai.