VILNIAUS UNIVERSITETAS
MATEMATIKOS IR INFORMATIKOS FAKULTETAS
INFORMATIKOS INSTITUTAS
KOMPIUTERINIO IR DUOMENŲ MODELIAVIMO KATEDRA

Bakalauro baigiamasis darbas

Optimalios dietos parinkimo sistema

Atliko:
Neringa Lukoševičiūtė parašas

Vadovas:
dr. Joana Katina

Vilnius
2019

Turinys

Santrauka

Šiame darbe yra aprašoma optimalios dietos problemą sprendžianti sistema, jos išsamus analizės bei realizavimo procesas. Darbo tikslas - sukurti optimalios dietos parinkimo sistemą, taikančią simplekso metodą ir sprendžiančią dietos problemą, kaip tiesinio programavimo uždavinį. Tikslo įgyvendinimui yra atsižvelgiama į jau egzistuojančias sistemas, kiekvienos iš jų pliusus bei minusus. Pagal atsižvelgiamus kriterijus sistemai buvo suformuluoti pagrindiniai uždaviniai, sudarytas ir aprašytas teorinis sistemos modelis. Pasiektas darbo rezultatas yra sukurta žiniatinklio programų sistema, gebanti generuoti optimalias dietas individualiam naudotojui, pagal naudotojo įvestus asmeninius parametrus. Sistemoje įgyvendintas modelis užtikrina, kad naudotojai turėtų galimybę generuojant dietas pasirinkti jų poreikius tenkinančius produktus, o sugeneruotas dietas - redaguoti. Prisijungę naudotojai taip turi galimybę atlikti su istorijos saugojimu susijusius veiksmus - dietų saugojimą, peržiūrą, trynimą, progreso stebėjimą, naujų sprendimų gavimą atsižvelgiant į ankstesnius veiksmus. Neprisijungusiems naudotojams siūloma gauti rekomenduojamų produktų sąrašą atsižvelgiant į kitų naudotojų įvertinimus, sistemoje pritaikant atraminių vektorių mašinų algoritmą.

Summary

Building an Optimal Diet System

This paper presents a system which solves an optimal diet problem, its detailed analysis and implementation process. The aim of this work is to build a system that would solve an optimal diet problem, by applying Simplex method and interpreting it as a linear programming task. To accomplish the aim, similar existing systems are taken into consideration, by looking into their advantages and disadvantages. Based on main criteria for the system, the goals of the project were formulated, and theoretical system model was created and articulated. The result of the project – a system in website form, which is capable of generating optimal diets for each user individually, based on their personal information. A model implemented in the system ensures that each user, before generating diets, could choose the products they like, and make modifications to the created diets afterwards. Logged in users are able to perform functional tasks related to history, such as saving, viewing and deleting their diets, monitoring personal progress and getting new solutions based on the previously taken steps. System also ensures, that non-logged in users could get the product recommendations, based on the ratings, provided by other users, by applying Support Vector Machines algorithm.

Įvadas

"Jei 1937m. nebūtų įvykęs tam tikras įvykis, tai po 10 metų būtų aišku, kad tiesinis programavimas ir simplekso metodai niekada nebūtų išrasti (bent jau ne tada), o daugelio žmonių gyvenimai bei kai kurių įmonių ateities planai būtų pasisukę ganėtinai kitaip." - taip matematikas George B. Dantzig, 1947m. suformulavęs dietos parinkimo problemą, kaip tiesinio programavimo uždavinį, apibūdino šios formuluotės aplinkybes ir rezultatus [4].

Pirmasis dietos uždavinį suformulavo matematikas Jerry Cornfield, siekdamas surasti mažiausios kainos dietą, kuri atitiktų mitybos poreikius JAV ginkluotųjų pajėgų kariui, Antrojo pasaulinio karo metu (1941-1945m.) [14]. Tą pačią problemą nagrinėjo ir kiti mokslininkai, pavyzdžiui, ekonomistas George Stigler, suformulavęs panašų uždavinį pagal savo asmeninius duomenis. Pilna uždavinio formuluotė: "Kiek vidutiniškai aktyviam žmogui (ekonomistui), sveriančiam 154 svarus, kasdien reikėtų suvalgyti kiekvieno iš 77 valgių, kad devynių maistinių medžiagų (įskaitant kalorijas) suvartojimas būtų ne mažesnis, nei rekomenduojamos mitybos normos, 1943m. pateiktos Nacionalinės Tyrimų Tarybos, kai dietos kaina yra minimali." [7]. Iš pradžių optimalios dietos uždavinys buvo bandomas išspręsti konstruojant įvesties-išvesties modelį, vėliau buvo panaudotas simplekso metodas (standartinis metodas, skirtas maksimizuoti tiesinę funkciją su keliais kintamaisiais), galiausiai George B. Dantzig pritaikė tiesinį programavimą [14]. Tai buvo vienas pirmųjų uždavinių, išspręstų taikant simplekso metodą bei tiesinį programavimą, kas padėjo rasti optimalesnį (mažesnės kainos) uždavinio sprendinį [1].

Remiantis Maliha Agha ir Riaz Agha gautais statistinio tyrimo rezultatais [2], nesubalansuotos mitybos problema gali būti laikoma viena opiausių ir apimančių vis didesnę dalį Europos gyventojų populiacijos. Autoriai pastebi, kad didžiojoje dalyje Europos valstybių antrinis sergamumas (angl. *prevalence*) nutukimu per paskutinius 10 metų išaugo nuo 10% iki 40%. To pasekoje prastėja sergančių žmonių gyvenimo kokybė, jie dažniau serga įvairiomis ligomis, patiria žalą sveikatai. Remiantis šiais duomenimis, galima teigti, kad optimalios dietos problema, nagrinėta visų anksčiau išvardintų autorių, yra aktuali ir šiais laikais. Manoma, kad optimalios dietos parinkimo sistema galėtų spręsti nesubalansuotos mitybos problemą.

Šio darbo tikslas - sukurti optimalios dietos parinkimo sistemą, taikančią simplekso metodą ir sprendžiančią dietos problemą, kaip tiesinio programavimo uždavinį. Sistemoje optimali dieta laikoma tokia, kuri tenkina minimalias individualiam žmogui apskaičiuotas rekomenduojamų maistinių medžiagų normas ir, taip pat kaip ir anksčiau nagrinėtų autorių atveju, minimizuoja rezultatą pagal bendrą dietos kainą.

Tikslo pasiekimui suformuluoti pagrindiniai uždaviniai - sukurtų sprendimų analizė, leidžianti suprasti ir palyginti egzistuojančių ir rinkoje naudojamų sistemų privalumus bei trūkumus, sistemos teorinio modelio kūrimas, apibrėžiantis sistemai keliamus funkcinius ir nefunkcinius reikalavimus, apibūdinantis planuojamą sistemos modelį UML diagramomis, pavaizduojantis sistemos navigacijos bei duomenų bazės struktūras, išskiriantis optimalios dietos sąvokos apibrėžimo pritaikymą ir priskyrimą apibrėžtoms problemoms literatūroje, sukurto modelio programinė realizacija, pritaikanti egzistuojančio optimalios dietos parinkimo modelį pasirinktam tiesinio programavimo metodui.

Pasiekti baigiamojo bakalauro darbo rezultatai - sukurta žiniatinklio programų sistema, kurios pagrindinis funkcionalumas - optimalios dietos problemos sprendimas, taikantis simplekso metodą ir pateikiantis gautus rezultatus naudotojams. Sistemoje įgyvendintas būdas užtikrina, kad optimali dieta bus parenkama atsižvelgiant į naudotojo įvestus asmeninius duomenis, automatiškai apskaičiuojant rekomenduojamų maistinių medžiagų normas. Taip pat, sistemoje sukurtas mode-

lis siekia atsižvelgti į naudotojų poreikius prieš generuojant dietą - šį funkcionalumą įgalina naujų produktų kūrimas, jų įtraukimas į dietą arba ištrynimas iš jos. Sugeneruotos dietos taip pat gali būti redaguojamos, o jeigu naudotojas prisijungęs - išsaugomos o vėliau peržiūrimos bei ištrinamos. Prisijungę naudotojai taip pat turi galimybę sekti savo svorio kitimo progresą, gauti naujus produktų siūlymus atsižvelgiant į ankstesnius veiksmus. Kadangi sistemai apie neprisijungusių naudotojų ankstesnius veiksmus nėra žinoma, jiems siūloma išbandyti atraminių vektorių mašinų algoritmo rekomenduojamas parinktis, atsižvelgiant į kitų naudotojų įvertinimus.

Šio rašto darbo 1 skyriuje yra pateikiama sukurtų sprendimų analizė - aprašomi ir palyginami panašūs egzistuojantys sprendimai tiek Lietuvos, tiek ir užsienio rinkoje, išskiriant kiekvieno iš analizuojamų sprendimų privalumus bei trūkumus. 2 skyrius yra skirtas sistemos teorinio modelio kūrimui ir aprašymui. Pagrinde orientuojamasi į sistemos funkcionalumą, iškeliant aprašomai sistemai funkcinius bei nefunkcinius reikalavimus, pavaizduojant pagrindinius sistemos aspektus UML diagramomis - naudojimo atvejų, klasių, veiklų. Teoriniame sistemos modelyje taip pat pateikiamos sistemos navigacijos bei duomenų bazės struktūros ir jų aprašymai. 3 skyriuje aprašomos programinės realizacijos detalės - įvardijami naudojami įrankiai, apibūdinami pagrindinį funkcionalumą realizuojantys sprendimai, pateikiant ir aprašant naudojamus algoritmus, pristatomi sistemos saugumo įgyvendinti sprendimai. Rašto darbo pabaigoje pateikiamos išvados bei rekomendacijos tolimesniam darbo vystymui.

1. Sukurtų sprendimų analizė

Atlikus panašių sukurtų sprendimų paiešką internete rastos kelios egzistuojančios sistemos, naudojamos Lietuvos arba užsienio rinkoje. Toliau analizuojamos sistemos buvo parinktos pagal susijusį su aprašoma sistema funkcionalumą bei bendrą dietos (mitybos) sritį. Svarbiausias kriterijus atrinktoms sistemoms - dietos sudarymo funkcijos buvimas. Kitos esančios/trūkstamos funkcijos aprašomose sistemose išskirtos ir aprašytos kiekvienu individualiu atveju išsamiau. Toliau analizuojamos Kilo.lt ir ManoMP.lt, priklausančios Lietuvos rinkai bei "Finite mathematics utility: diet problem solver" ir "Eat This Much", pritaikytos užsienio rinkai, sistemos. Panašių sprendimų analizė Lietuvos ir užsienio rinkose yra išskaidytos ir pateiktos žemiau esančiuose poskyriuose.

1.1. Panašūs sprendimai Lietuvos rinkoje

Pirmoji analizuojama Lietuvos rinkai priklausanti žiniatinklio programų sistema - Kilo.lt. Ši sistema yra orientuota į mokamų mitybos (dietų) bei treniruočių programų sudarymo teikimą. Visi optimalumo reikalaujantys sprendimai, priimami sistemoje ir pasiūlomi naudotojui yra sudaromi profesionalių dietologų ir trenerių. Norint gauti parengtą dietą naudotojas yra paprašomas užpildyti pradinį klausimyną, kuriame nurodomi poreikiai, tikslas, gyvenimo būdas ir pan. Vėliau remiantis šiais duomenimis profesionalus dietologas gali sudaryti dietą. Pagrindiniai sistemos privalumai - platus ingredientų ir receptų pasirinkimas, parengtų planų modifikavimas ir prisitaikymas individualiems poreikiams, progreso stebėjimas, įgyvendintas produktų paieškos, maistinių verčių atvaizdavimo funkcionalumas. Taip pat sistemos funkcionalumą papildo kitos, su sveiku gyvenimo būdu susijusios funkcijos - kūno masės indekso, lieknėjimo skaičiuoklės, sporto pratimų patarimai, straipsniai, forumas. Pagrindiniai šios sistemos trūkumai - profesionalų sudaromos ir gaunamos asmeninės dietos ir treniruočių programos ne iš karto, be to, didžioji dalis funkcionalumo (programų sudarymas, konsultacijos ir pan.) yra mokama. Kadangi sudaromos dietos yra profesionalaus dietologo, bet ne kompiuterio, daroma išvada, kad sudarant dietą orientuojamasi į individualius kliento poreikius bei siekiama pateikti jam patrauklų mitybos planą, bet nebūtinai optimizuotą pagal tam tikrą optimalios dietos kriterijų.

Antroji analizuojama Lietuvos rinkai pritaikyta sistema - ManoMP.lt. Sistema yra skirta automatizuotui mitybos planų sudarymui, planavimui, analizavimui. Kaip ir Kilo.lt sistemos atveju, didžioji dalis funkcionalumo yra smarkiai apribota nenusipirkusiems narystės naudotojams. Tačiau tokie naudotojai vis tiek turi galimybę gauti sugeneruotą mitybos planą, pagal savo asmeninius duomenis pamatyti, kiek maistinių medžiagų (kalorijų, baltymų, angliavandenių ir riebalų) turėtų suvartoti per dieną, atlikti produktų paiešką, peržiūrint jų maistines vertes. Nusipirkę narystę naudotojai gali pamatyti žymiai daugiau sistemos privalumų ir jais naudotis - pridėti bei ištrinti pasiūlytus produktus dietoje, kurti naujus produktus, sekti savo asmeninį svorio kitimo progresą, vertinti produktus, kad ateityje mitybos planas būtų sudaromas atsižvelgiant į šiuos rezultatus. Pagal vertintus analizės reikalavimus sistema netenkina vienintelio kriterijaus - optimalios dietos. Sistemoje realizuotas sprendimas atsitiktiniu būdu parenka produktus, tenkinančius rekomenduojamų maistinių medžiagų reikalavimus, bet ne siekia optimizuoti pačią dietą. Visi kiti analizės reikalavimai yra vertinami teigiamai, sistema teikia tikrai platų funkcionalumą, nors didžioji dalis jo yra prieinama tik narystę įsigijusiems nariams.

1.2. Panašūs sprendimai užsienio rinkoje

"Finite mathematics utility: diet problem solver" - optimalios dietos problemą sprendžiantis įrankis. Sistemoje įgyvendintas sprendimas yra realizuotas taikant simplekso metodą, pagal S. Waner ir S. Costenoble autorių [15] knygą ir yra labiau skirtas mokymosi tikslais. Pagrindinis sistemos privalumas - automatizuotas dietos, sprendžiančios optimalios dietos uždavinį, radimas. Sprendimas yra išsamiai dokumentuotas, todėl naudotojui yra aišku, kokie skaičiavimo veiksmai yra atliekami ir į kokią naudotojo įvestį yra atsižvelgiama. Įrankyje taip pat yra realizuota produktų pasirinkimo funkcija, tačiau produktų atvaizdavimas yra labai skurdus - paieška įgalinama pateiktame sąraše, nėra filtravimo pagal įvestį funkcionalumo. Šios sistemos trūkumai - nėra dietos modifikavimo, naudotojų registracijos (todėl ir naudotojo istorijos, progreso sekimo) funkcijų. Taip pat, sistemoje daroma prielaida, kad naudotojas jau žino, kiek jam reikia suvartoti maistinių medžiagų per dieną, todėl nėra realizuotos maistinių medžiagų apskaičiavimo dienai funkcijos.

Antroji analizuojama užsienio rinkai pritaikyta sistema - "Eat This Much" internetinis puslapis. Sistemoje visi naudotojai prieš generuojant dietą gali sužinoti jiems rekomenduojamų maistinių medžiagų dienai normas, nusistatyti norimos dietos tipą (vegetariška, veganiška ir pan.). Taip pat, bet kuris naudotojas gali pamatyti individualiai jam sugeneruotą dietą, ją modifikuoti pakeičiant pasiūlytų produktų kiekius. Tuo tarpu prisijungusio naudotojo funkcionalumas yra praplečiamas progreso sekimo, dietų išsaugojimo ir vėlesnio jų peržiūrėjimo arba trynimo, produktų paieškos funkcijomis. Be visų išvardintų privalumų sistemoje taip pat egzistuoja produktų vertinimo funkcija, kuri užtikrina, kad naudotojui nepatinkantys produktai nebūtų siūlomi ateityje. Vienintelis pastebėtas analizuojamos sistemos skirtumas aprašomos problemos kontekste yra toks, kad "Eat This Much" internetinis puslapis siūlo dietą atsižvelgiant į naudotojo įvestas arba sugeneruotas maistinių medžiagų sąlygas, bet ne siekia optimizuoti pačią dietą, pagal kažkurį kriterijų.

1.3. Sukurtų sprendimų analizės apibendrinimas

Atlikus aukščiau aprašytų sukurtų sprendimų analizę pastebėta, kad tiek Lietuvos, tiek užsienio rinkoje jau yra sukurta nemažai sistemų, tenkinančių pagrindinius egzistuojančių sprendimų analizei iškeltus reikalavimus. Analizės metu nebuvo rasta Lietuvos rinkai priklausančių sistemų, sprendžiančių būtent optimalios dietos problemą. ManoMP.lt sistema, lyginant su Kilo.lt, pasižymėjo žymiai platesniu funkcionalumu, nors ir didžioji dalis jo yra prieinama tik įsigijus narystę. Tuo tarpu užsienio rinkoje abiejų analizuotų sistemų vertintas tik nemokamas funkcionalumas, kuris buvo pakankamas aprašomos sistemos kontekste. "Finite mathematics utility: diet problem solver" įrankis, lyginant su visais kitais analizuotais sprendimais, turi didžiausią pranašumą, kad dietą parenka sprendžiant optimalios dietos uždavinį. Visas kitas funkcionalumas yra žymiai platesnis "Eat This Much" sistemoje. Apibendrinti analizės rezultatai pateikiami 1 lentelėje.

Aprašoma sistema funkcionalumo prasme siekia orientuotis į ManoMP.lt ir "Eat This Much" sprendimus. Aprašomos sistemos modelis siekia užtikrinti, kad naudotojas turėtų galimybę dietas modifikuoti, sekti savo progresą, peržiūrėti anksčiau sugeneruotas ir išsaugotas dietas. Taip pat, siekiant gauti dietos generavimui privalomą informaciją apie naudotoją, maistinių medžiagų apskaičiavimas dienai yra vykdomas automatiškai, kaip ir šiose sistemose, o ne įvedamas naudotojo. Pagrindinis aprašomos sistemos tikslas - spręsti optimalios dietos uždavinį, t. y. siekti optimizuoti dietos parenkamą rezultatą. Vienintelis toks sprendimas iš analizuotų yra taikomas "Finite mathematics utility: diet problem solver" įrankyje. Būtent šiuo aspektu aprašoma sistema orientuosis į šį įrankį.

1 lentelė. Sukurtų sprendimų analizės apibendrinimas

Funkcija / Sistema	Automatizuotas dietos radimas	Optimali dieta	Dietos modifikavimas	Progreso sekimas	Maistinių medžiagų apskaičiavimas dienai
Kilo.lt	Ne	Ne	Taip	Taip	Ne
ManoMP.lt	Taip	Ne	Taip	Taip	Taip
Finite mathematics utility: diet problem solver	Taip	Taip	Ne	Ne	Ne
Eat This Much	Taip	Ne	Taip	Taip	Taip

2. Sistemos modelis

Šiame skyriuje pateikiamas sukurtas teorinis sistemos modelis - įvardintas sistemos funkcionalumas, apibrėžiant iškeltus funkcinius ir nefunkcinius reikalavimus, pavaizduojamos pagrindinį funkcionalumą apibūdinančios UML diagramos, pateikiamos ir apibūdinamos sistemos navigacijos ir duomenų bazės struktūros.

2.1. Sistemos funkcionalumas

Pagrindinis aprašomos sistemos tikslas - spręsti optimalios dietos parinkimo problemą. Kadangi problemos sprendimui taikomas simplekso metodas, kurio veikimui būtina apibrėžti sprendžiamų lygčių apribojimus (sąlygas), sistema, visų pirma turi gebėti šiuos apribojimus apskaičiuoti kiekvienam žmogui individualiai. Naudotojui nurodžius reikiamus duomenis apie save - amžių, ūgį, svorį, lytį, fizinį aktyvumą ir norimą tikslą, sistema apskaičiuoja rekomenduojamas maistinių medžiagų normas, taikant M.D. Mifflin ir S.T. St Jeor pasiūlytą formulę [12] ir parodo šias reikšmes naudotojui. Vėliau atsižvelgiant į šias reikšmes sistema geba generuoti pačią dietą. Prieš dietos generavimą naudotojas gali pridėti norimus arba pašalinti nenorimus, tačiau sistemos pasiūlytus produktus. Sugeneruota dieta taip pat gali būti redaguojama, keičiant pasiūlytų produktų kiekius. Prisijungęs naudotojas taip pat turi galimybę išsaugoti ir vėliau peržiūrėti išsaugotų dietų istoriją, sekti asmeninius svorio kitimo rezultatus, atnaujinti asmeninius duomenis. Prisijungusio naudotojo vykdomi veiksmai yra saugomi - parenkamos dietos ir produktai yra reitinguojami, todėl toks naudotojas turi galimybę gauti naujas dietas atsižvelgiant į savo ankstesnius veiksmus. Informacija apie neprisijungusius naudotojus nėra saugoma, todėl tiksliai nustatyti, ką konkretus neprisijungęs naudotojas mėgsta, galimybės nėra. Siekiant, kad tokie naudotojai visgi turėtų galimybę gauti bent dalį jiems patinkančių produktų, sistemoje taikomas atraminių vektorių mašinų algoritmas. Taigi, kiekvienas sistemos neprisijungęs naudotojas turi galimybę pamatyti rekomenduojamus produktus, atsižvelgiant į kitų naudotojų vertinimus.

2.1.1. Funkciniai reikalavimai

Atsižvelgiant į aukščiau aprašytą sistemos funkcionalumą, sistemai keliami tokie funkciniai reikalavimai:

- Sistemoje turi būti įgalintas naudotojo duomenų įvedimas ir išsaugojimas.
- Sistemoje turi veikti rekomenduojamų maistinių medžiagų normų apskaičiavimas pagal individualius naudotojo duomenis. Naudotojas turi turėti galimybę pamatyti šių rekomendacijų reikšmes.
- Bet kuris sistemos naudotojas turi turėti galimybę inicijuoti dietos generavimo veiksmą. Rezultatas turi būti apskaičiuojamas taikant simplekso metodą ir pateikiamas atgal naudotojui.
- Sistema turi gebėti apdoroti produktų pridėjimą į dietą bei ištrynimą iš jos.
- Sistema turi leisti, kad individualiai sugeneruotą dietą naudotojas galėtų redaguoti, o jei naudotojas yra prisijungęs - ir išsaugoti.
- Prisijungęs naudotojas privalo turėti galimybę vykdyti išsaugotų dietų peržiūros ir trynimo funkcijas.
- Prisijungęs naudotojas privalo turėti galimybę išsaugoti savo asmeninius duomenis ir juos atnaujinti, peržiūrėti savo rezultatus.

- Sistema turi užtikrinti, kad naudotojams nebus siūlomi jiems nepatinkantys produktai, kuriuos naudotojas jau įvertino.
- Neprisijungęs sistemos naudotojas privalo gebėti inicijuoti tapatumo nustatymo veiksmą ir prisijungti arba, jei dar neturi paskyros, prisiregistruoti prie sistemos.
- Neprisijungęs sistemos naudotojas turi turėti galimybę gauti rekomenduojamų produktų sąrašą atsižvelgiant į kitų naudotojų įvertinimus.

2.1.2. Nefunkciniai reikalavimai

Siekiant užtikrinti sklandų sistemos veikimą, buvo iškelti tokie nefunkciniai reikalavimai:
- Sistema turi užtikrinti, kad nenustatyto tapatumo (angl. *unauthenticated*) asmuo negalėtų atlikti prieigos teisių (angl. *authorization*) reikalaujamų veiksmų.
- Sistema turi veikti sklandžiai, be žymių strigimo (vėlavimo) požymių.
- Sistema turi būti pritaikyta naudoti su stacionariu ar nešiojamu kompiuteriu.

2.2. UML diagramos

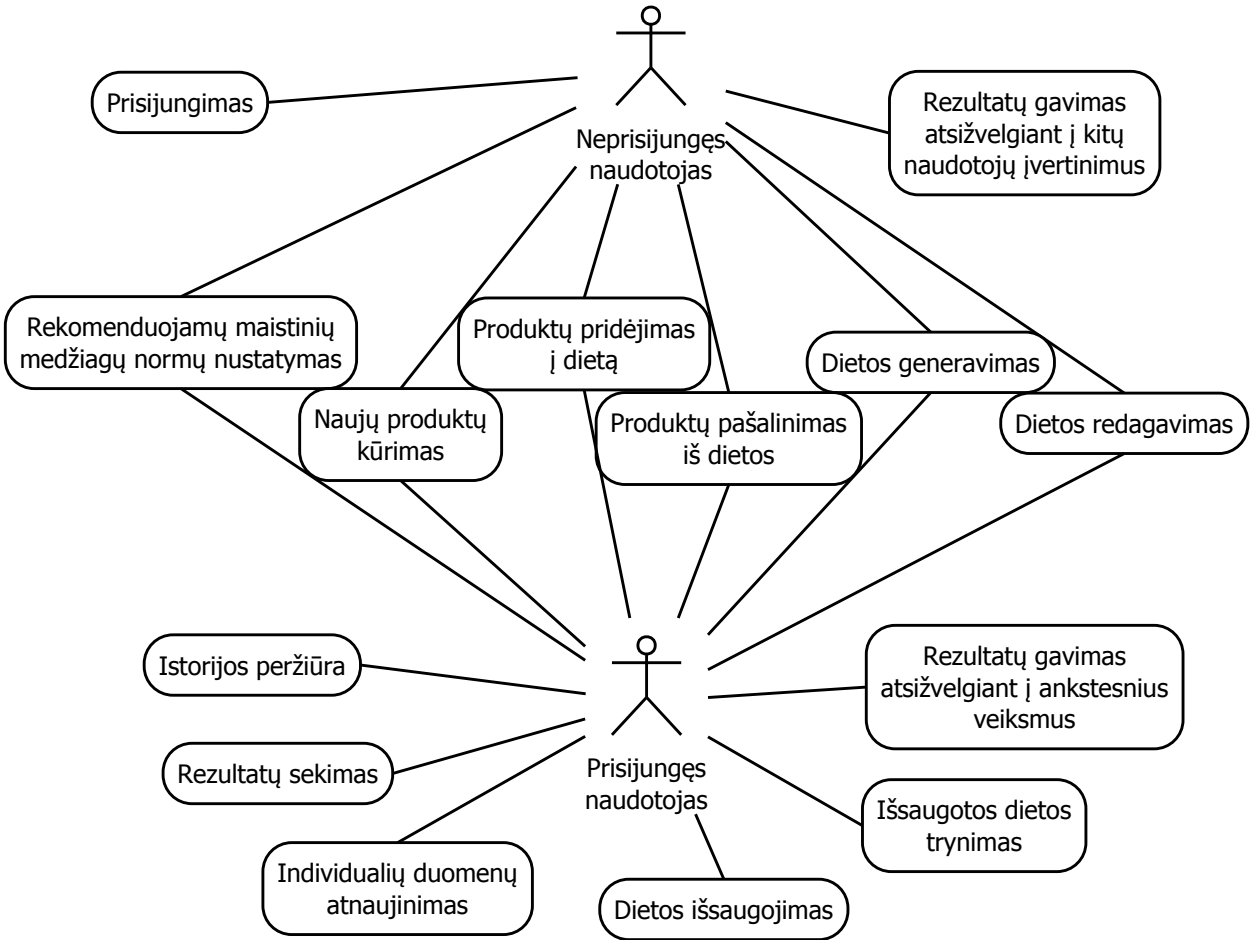
Šiame poskyryje pateikiamos ir aprašomos pagrindinį sistemos funkcionalumą apibūdinančios UML diagramos - naudojimo atvejų, klasių bei veiklų.

2.2.1. Naudojimo atvejų diagrama

Pagrindinis sistemos funkcionalumas sudarytas iš dviejų sistemos aktorių - neprisijungusio ir prisijungusio naudotojų galimų atlikti veiksmų. Neprisijungęs naudotojas sistemoje gali vykdyti tokias funkcijas:
- Prisijungimas;
- Rekomenduojamų maistinių medžiagų normų gavimas;
- Naujų produktų kūrimas;
- Produktų pridėjimas į dietą;
- Produktų pašalinimas iš dietos;
- Dietos generavimas;
- Dietos redagavimas;
- Rezultatų gavimas atsižvelgiant į kitų naudotojų įvertinimus.

Registruotas ir prisijungęs naudotojas apibrėžiamas kaip aktorius, kuris praplečia neprisijungusio naudotojo galimų atlikti veiksmų aibę. Prisijungęs naudotojas negali tik gauti rezultatų atsižvelgiant į kitų naudotojų vertinimus (vietoje to, naudotojui siūlomi produktai parinkti konkrečiai jam, pagal jo asmeninius įvertinimus). Papildomai prisijungusiam naudotojui suteikiamos tokios su istorijos saugojimu susijusios funkcijos:
- Istorijos peržiūra;
- Rezultatų sekimas;
- Individualių duomenų atnaujinimas;
- Dietos išsaugojimas;
- Išsaugotų dietų trynimas;
- Rezultatų gavimas atsižvelgiant į ankstesnius veiksmus.

Sistemos aktoriai ir jų galimi atlikti veiksmai pateikiami 1 paveikslėlyje pavaizduotoje sistemos naudojimo atvejų diagramoje.



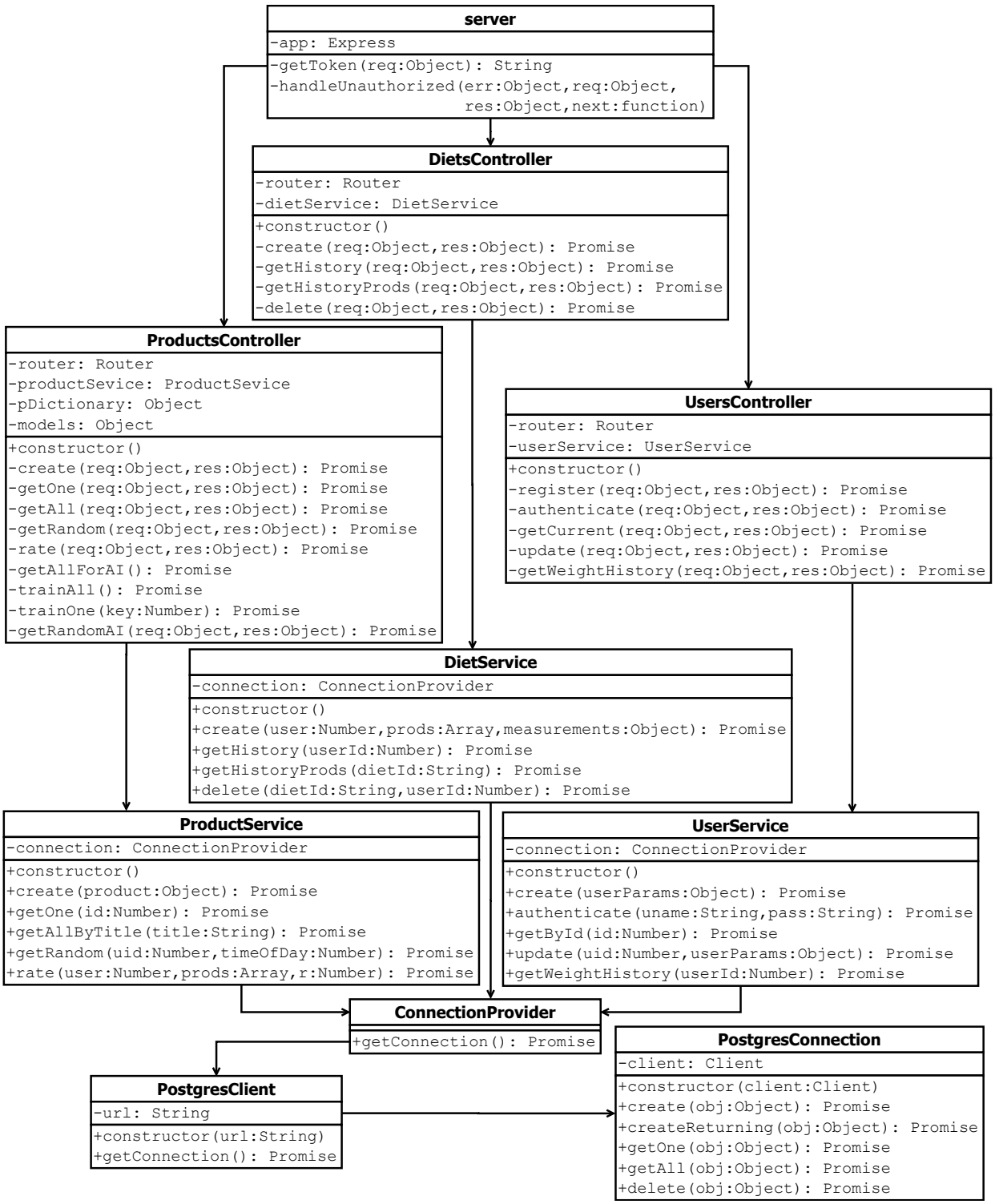
1 pav. Sistemos naudojimo atvejų diagrama

2.2.2. Klasių diagramos

Sistemoje kuriamos vidinės ir išorinės dalių klasės yra atskirtos. Išorinėje dalyje aprašytas funkcionalumas yra atsakingas už gautų arba apskaičiuotų duomenų pavaizdavimą naudotojui, tuo tarpu vidinėje dalyje - už šių duomenų gavimą arba išsaugojimą duomenų bazėje. Sistemos dalys komunikuoja tarpusavyje aprašytų paslaugų (angl. *services*) pagalba. Dėl šios priežasties toliau pateikiamos atskiros klasių diagramos sistemos vidinei bei išorinei dalims.

Vidinė sistemos dalis yra atsakinga už užklausų, gautų iš išorinės dalies, apdorojimą. Gaunamos užklausos yra išskaidytos į 3 pagrindinius tipus, skirtus produktų, dietų ir naudotojų informacijos gavimui bei apdorojimui. Todėl vidinėje sistemos dalyje aprašytos klasės atsakingos būtent už šių tipų užklausų apdorojimą bei persiuntimą duomenų bazei. Visa vidinės sistemos dalies klasių diagrama pavaizduota 2 paveikslėlyje. Pagrindinė veikiančio serverio informacija yra aprašoma *server* klasėje. Joje inicializuojamas *Express* tipo kintamasis *app*, aprašantis, kuriame prievade turi veikti vidinė sistemos dalis, kokias užklausas sistema gali priimti, koks turi būti naudojamas prieigos teisių nustatymo metodas bei kokį atsakymą sistema siunčia gavusi užklausą iš nenustatyto tapatumo arba neturinčio reikiamų prieigos teisių, asmens. Visa ši informacija pateikiama naudojant *Express* internetinio puslapio karkaso funkcijas. Kintamajam *app* apibrėžiant sistemoje veikiančius galinius taškus (angl. *endpoint*) yra kreipiamasi į produktų, dietų ir naudotojų valdiklius (angl. *controller*), kurie veikia kaip atskiros klasės.

Valdikliai aprašomoje sistemoje yra atsakingi už užtikrinimą, kad gautas estafetės paketas (angl. *token*) būtų galiojantis ir priklausantis tam naudotojui, kuris ir siuntė užklausą, gautos už-



2 pav. Vidinės sistemos dalies klasių diagrama

klausos parametrų atrinkimą bei perdavimą paslaugai, grąžinto rezultato apdorojimą, klaidų atveju - statusų parinkimą. Kiekviena iš valdiklių klasių saugo kintamąjį *router*, kuriame aprašomas galinis taškas - gautos užklausos tipas, nuoroda bei funkcija, į kurią kreipiamasi gavus užklausą į atitinkamą galinį tašką. Papildomai kiekviena klasė turi kintamąjį, saugantį nuorodą į reikiamą paslaugą, gebančią apdoroti užklausas. Šie kintamieji yra inicializuojami aprašytuose klasės konstruktoriuose.

Produktų valdiklio klasė *ProductsController* gali apdoroti naujo produkto kūrimo, egzistuo-

jančių produktų gavimo bei produktų vertinimo užklausas. Egzistuojantys produktai gali būti nuskaitomi iš duomenų bazės 3 būdais - norint gauti tik vieną produktą, visus (filtruojamus pagal pavadinimą) arba atsitiktinius (tinkamus konkrečiam dienos metui). Todėl valdiklyje visiems atvejams yra aprašytos atskiros funkcijos. Taip pat, būtent šio valdiklio klasėje esančios funkcijos užtikrina, kad bus laiku inicijuotas atraminių vektorių mašinų algoritmo apmokymas.

Dietų valdiklio klasė *DietsController* atsakinga už naujų dietų išsaugojimo, egzistuojančių dietų bei jų produktų nuskaitymą ir dietų trynimo užklausų apdorojimą.

Naujam naudotojui registruojantis prie sistemos, registruotam naudotojui - prisijungiant, parodant naudotojo informaciją, ją atnaujinant arba nuskaitant ir atvaizduojant naudotojo svorio kitimo rezultatus, būtų kreipiamasi į naudotojų valdiklio klasę *UsersController*.

Sistemoje aprašytos paslaugų klasės užtikrina, kad iš valdiklio gauti parametrai bus apdorojami ir persiunčiami reikiamoms duomenų bazės funkcijoms arba užklausoms. Iš viso sistemoje sukurtos 3 paslaugų klasės - *ProductService*, *DietService* ir *UserService*. Šiose klasėse prijungimas prie duomenų bazės yra inicializuojamas konstruktoriuje ir saugomas *ConnectionProvider* tipo klasės kintamajame.

ConnectionProvider klasė yra atsakinga už prisijungimo prie duomenų bazės inicijavimą. Vienintelė klasės funkcija nuskaito prisijungimo prie duomenų bazės reikšmę iš konfigūracinio failo bei perduoda ją sukurtam *PostgresClient* tipo klasės objekto konstruktoriui.

PostgresClient klasė yra skirta PostgreSQL kliento kūrimui. Šioje klasėje yra inicializuojamas klientas, kuris turi prieigą prie duomenų bazės ir yra perduodamas *PostgresConnection* klasės konstruktoriui.

PostgresConnection klasėje aprašytos visos su duomenų baze susijusios užklausos, skirtos sukurti naujus, gauti bei ištrinti egzistuojančius įrašus.

Visos vidinėje sistemos dalyje aprašytos valdiklių ir paslaugų funkcijos grąžina *Promise* tipo kintamuosius. Tai užtikrina, kad sistema apdoroja užklausas asinchroniškai.

Išorinėje sistemos dalyje egzistuojančios klasės užtikrina, kad duomenys bus tinkamai pavaizduojami naudotojui arba nuskaitomi ir persiunčiami galiniams taškams, aprašytiems vidinėje sistemos dalyje. Taigi, klasės irgi yra susijusios su pagrindiniais 3 apdorojamais - produktų, dietų ir naudotojų - tipais. Pagrindinės išorinės sistemos dalies klasės pavaizduotos 3 paveikslėlyje.

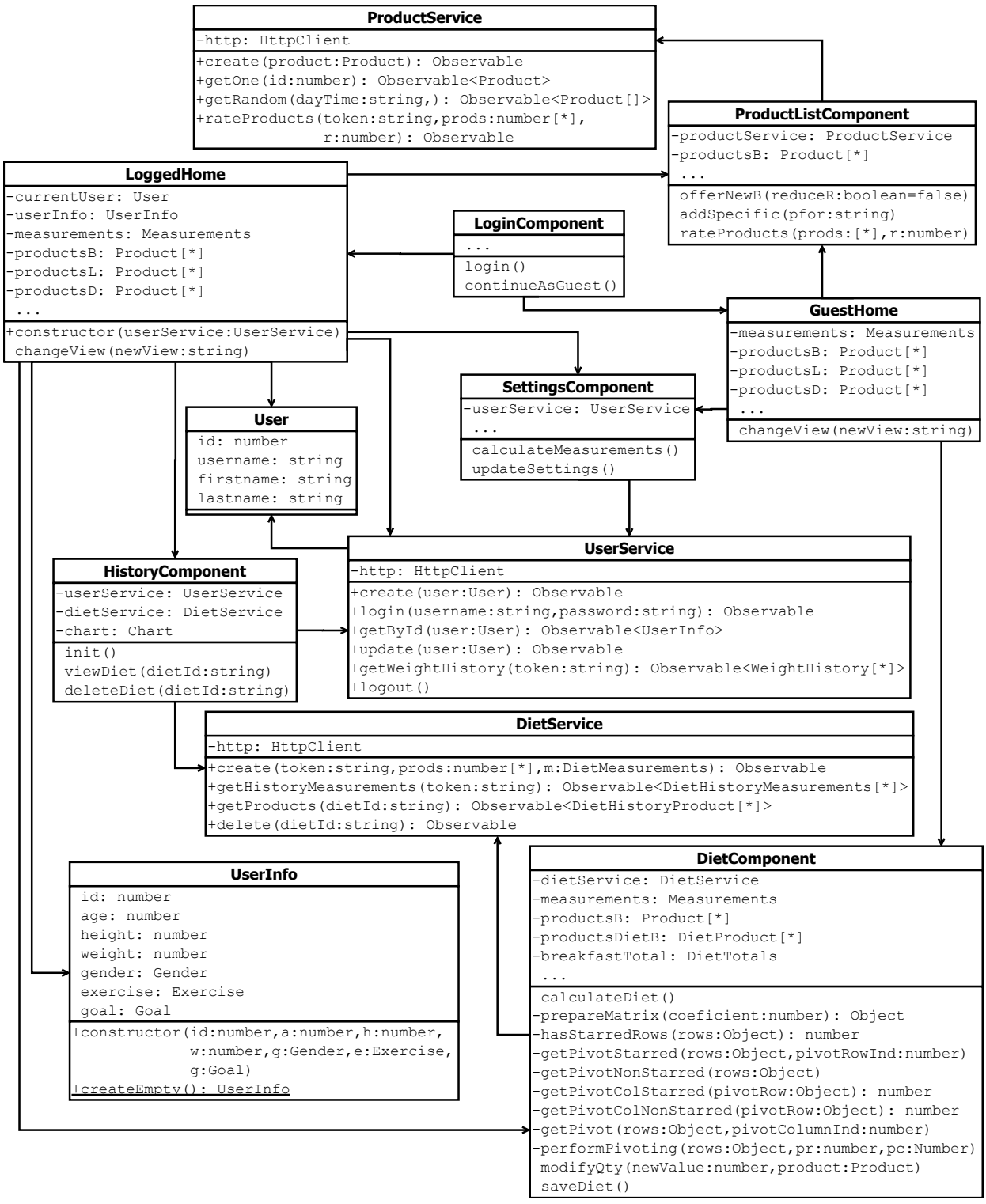
Pirmoji klasė, kuri yra iškviečiama naudotojui atėjus į sistemą yra prisijungimo komponentas (*LoginComponent*). Klasėje aprašytos funkcijos užtikrina, kad sistema gebės apdoroti tapatumo nustatymo veiksmą, jei naudotojas bando prisijungti, arba nukreips jį į neprisijungusio naudotojo aplinką, jei naudotojas prisijungti nenori.

Klasės *LoggedHome* ir *GuestHome* sukuria atitinkamai prisijungusių ir neprisijungusių naudotojų aplinkas. Iš šių aplinkų yra įgalinami kiti sistemos langai, kuriuos inicializuoja klasės *ProductListComponent*, *DietComponent*, *SettingsComponent* ir *HistoryComponent*. Paskutinis langas neprisijungusiems naudotojams nerodomas, todėl *GuestHome* klasė su juo jokio ryšio neturi.

Kiekviena iš komponentų klasių yra atsakinga už tam tikro lango atvaizdavimą naudotojui, reikiamų reikšmių skaičiavimą, jeigu reikia - duomenų nuskaitymą iš naudotojo ir perdavimą paslaugoms.

Išorinėje sistemos dalyje yra aprašytos 3 paslaugų klasės - *ProductService*, *UserService* ir *DietService*. Kiekviena iš jų yra atsakinga už kreipimąsi į vidinėje sistemos dalyje aprašytus galinius taškus.

Išorinėje sistemos dalyje taip pat yra sukurtos klasės, apibrėžiančios tam tikrų duomenų tipus. Tokiose klasėse dažniausiai apibrėžiamas konstruktorius, inicializuojantis klasės kintamuosius bei funkcijos, gebančios atlikti su jais skaičiavimus. 3 diagramoje tokios klasės yra *User* ir *UserInfo*.



3 pav. Išorinės sistemos dalies klasių diagrama

2.2.3. Veiklų diagramos

Šiame poskyryje pateikiamos pagrindinį funkcionalumą apibūdinančios veiklų diagramos bei jų aprašymai. Toliau išskirtuose paragrafuose detaliau aprašomi pagrindiniai rekomenduojamų maistinių medžiagų normų nustatymo, dietos generavimo bei produktų reitingavimo ir siūlymo sprendimai.

Rekomenduojamų maistinių medžiagų normų nustatymas. Pagrindinis kriterijus, kuris privalo būti nustatytas prieš generuojant dietą - reikiamų suvartoti kalorijų kiekis. 2005m. Amerikos Dietologų Asociacijos atliktu siūlomų formulių palyginimu [6] buvo patvirtinta, kad M.D. Mifflin ir S.T. St Jeor pasiūlyta bazinio kalorijų normos skaičiavimo formulė [12] efektyviausiai apskaičiuoja individualiam žmogui rekomenduojamų suvartoti kalorijų kiekį. Anot autorių, minimalus kalorijų kiekis, kurio reikia vyro kūno funkcionavimui, turėtų būti apskaičiuojamas pagal 2.1 pateiktą formulę, tuo tarpu moteris per dieną turėtų suvartoti kalorijų kiekį, gautą pritaikius 2.2 formulę.

$$10 * svoris(kg) + 6.25 * \textit{ūgis (cm)} - 5 * \textit{amžius (m)} + 5 \tag{2.1}$$

$$10 * svoris(kg) + 6.25 * \textit{ūgis (cm)} - 5 * \textit{amžius (m)} - 161 \tag{2.2}$$

Kalorijų kiekis, suskaičiuotas pagal aukščiau įvardintas formules, vadinamas baziniu, ir yra tinkamas tik organizmo funkcionavimui - kvėpavimui, virškinimui, mąstymui ir pan. Kalorijų kiekis privalo būti didesnis fiziškai aktyviems žmonėms. Sistemoje numatyti 6 fizinio aktyvumo lygiai: 0, 1-2, 3, 4, 5-6, arba 7 fiziškai aktyvios dienos per savaitę. Pasirinkus vieną iš šių lygių, išskyrus pirmąją, suvartojamų kalorijų kiekis atitinkamai dauginamas iš 1.2, 1.3, 1.4, 1.5 arba 1.6 koeficiento. Galiausiai, sistemoje numatytas naudotojo tikslo įvedimas. Tikslas gali būti vienas iš trijų numatytų reikšmių - svorio mažinimas, išlaikymas arba didinimas. Pasirinkus svorio mažinimą rekomenduojamų suvartoti kalorijų kiekis sumažinamas 20%, pasirinkus didinimo tikslą - didinamas 20%. Veiklos diagrama, apskaičiuojanti aukščiau aprašytą veikimo principą, pateikta 4 paveikslėlyje.

Sistemos naudojamam simplekso metodo apribojimams nurodoma ne tik reikiamų suvartoti kalorijų norma, bet ir kitų maistinių medžiagų rekomendacijos - angliavandenių, baltymų ir riebalų kiekis. Aprašomoje sistemoje numatyta, kad kiekvienas žmogus per dieną turėtų suvartoti atitinkamai 50%, 20%, 30% minėtųjų maistinių medžiagų nuo paros rekomenduojamų kalorijų kiekio.

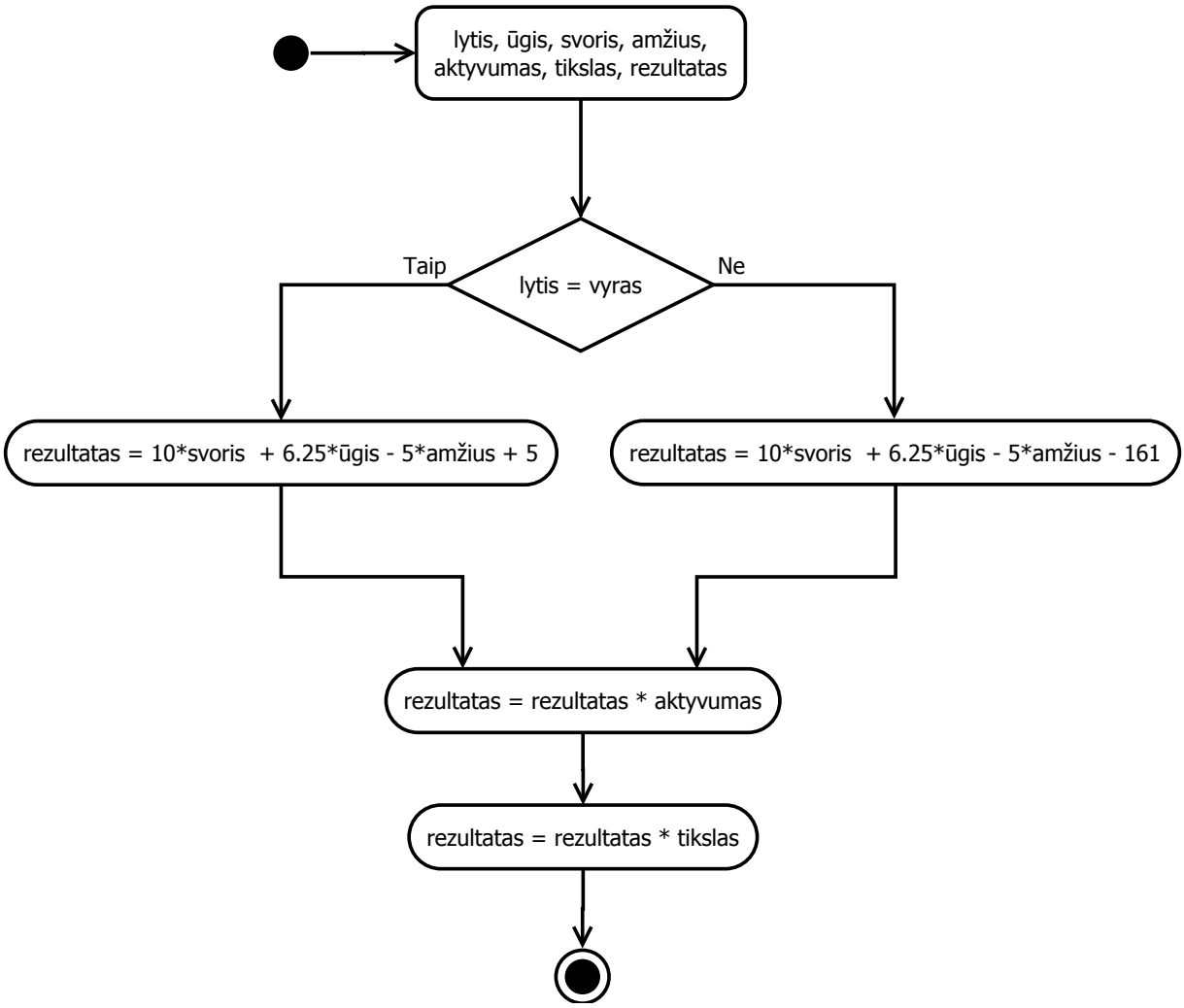
Dietos generavimas. Sistemoje optimalios dietos problema yra sprendžiama kaip tiesinio programavimo uždavinys. Remiantis S. Waner ir S. Costenoble apibrėžimu [15], toks uždavinys formuluojamas, kaip problema, kurios sprendiniu siekiama rasti maksimalią (arba minimalią) tiesinės funkcijos, pateiktos 2.3 lygybėje, reikšmę, apibrėžtą 2.4 nurodytomis sąlygomis.

$$p = a_1x_1 + a_2x_2 + ... + a_nx_n \tag{2.3}$$

$$b_1x_1 + b_2x_2 + ... + b_nx_n \leq c \text{ arba } b_1x_1 + b_2x_2 + ... + b_nx_n \geq c \tag{2.4}$$

Sistemoje parenkant optimalią dietą, funkcija sudaroma remiantis Cristina-Elena HREŢCANU ir Ciprian-Ionel HREŢCANU [8] suformuluotu modeliu. Autoriai apibrėžia, jog problemos tikslas - minimizuoti funkciją *p*, tenkinančią apibrėžtas sąlygas. Taigi, sistemoje siekiama minimizuoti funkciją *p*, apibrėžiančią dietos kainą, kur a_1, a_2, ..., a_n - kiekvieno produkto kaina, b_1, b_2, ..., b_n - produktų maistinių medžiagų kiekiai, *c* - sąlyga, nurodanti konkrečios maistinės medžiagos rekomenduojamą paros normą individualiam žmogui. Apibrėžta funkcija toliau nagrinėjama kaip bendra tiesinio programavimo problema. Tokia problema gali priklausyti standartinėms maksimizavimo (angl. *standard maximization*) problemoms. Standartinė maksimizavimo problema apibrėžiama tokiomis sąlygomis [15]:

1. Tiesinės funkcijos tikslas - maksimizavimas.
2. Tiesinės funkcijos sąlygos kintamieji x_1, x_2, ..., x_n yra ne neigiami.
3. Visos likusios funkcijos sąlygos aprašomos bx_1 + bx_2 + ... + $bx_n \leq c$ forma, kur c - ne neigiamas.

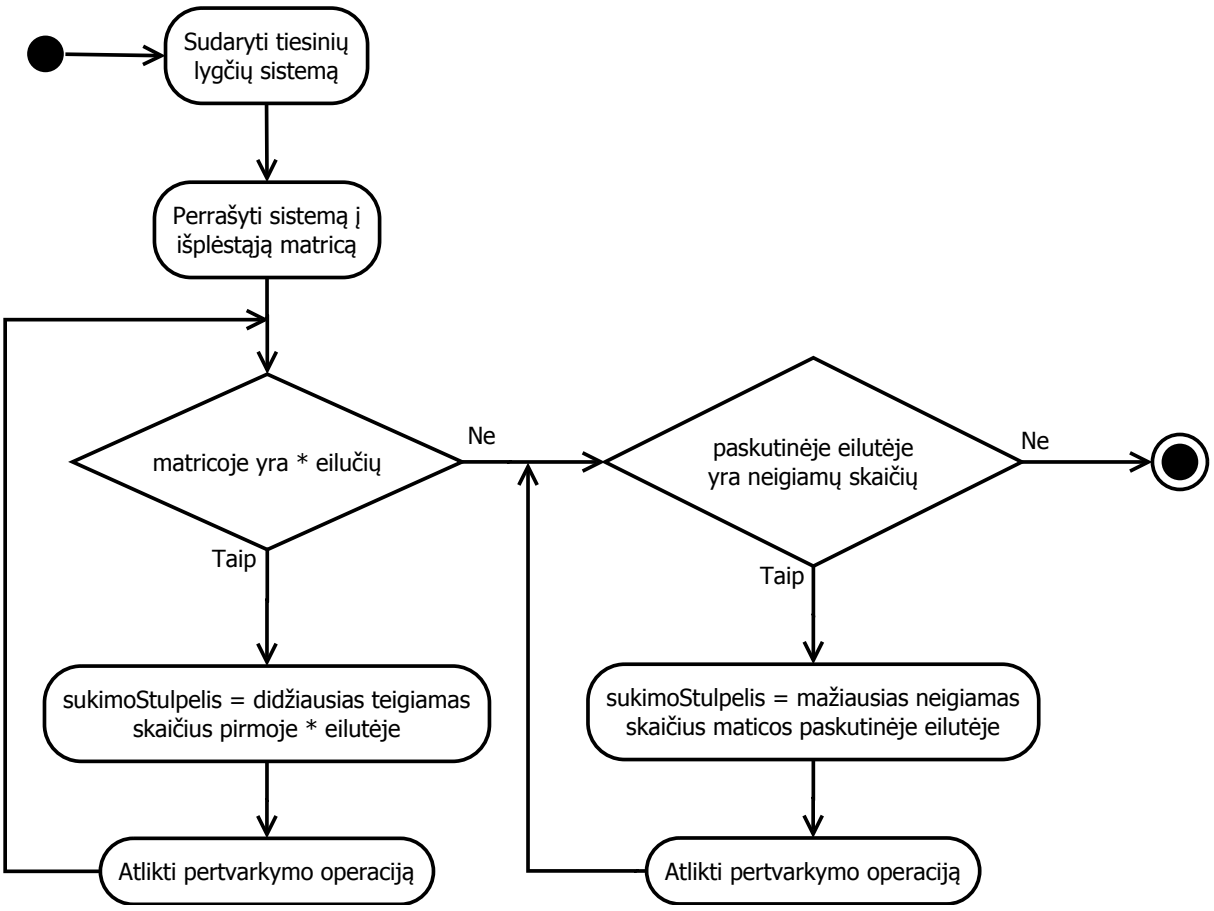


4 pav. Rekomenduojamų suvartoti kalorijų apskaičiavimo veiklos diagrama

Iš sąlygų matyti, kad Cristina-Elena HREŢCANU ir Ciprian-Ionel HREŢCANU suformuluotas modelis netenkina pirmojo ir trečiojo punkto, todėl problema nepriklauso standartinėms maksimizavimo problemoms. Bendros tiesinio programavimo problemos gali būti sprendžiamos simplekso metodu. S. Waner ir S. Costenoble [15] pateiktame sprendime, pavaizduotame 5 paveikslėlyje esančioje veiklos diagramoje, autoriai siūlo spręsti problemą pagrindiniais 6 žingsniais.

Visų pirma, turi būti sudaroma tiesinių lygčių sistema. Kiekvienai apibrėžtai sąlygos nelygybei turi būti pridedamas perteklinis (angl. *surplus*) kintamasis. Jeigu nelygybė netenkina trečiojo standartinės maksimizavimo problemos žingsnio (t. y. iškelta sąlyga nurodo, kad rezultatas *c* turi būti mažesnis arba lygus, bet ne didesnis arba lygus), perteklinis kintamasis s_n turi turėti neigiamą koeficientą. Pavyzdžiui, norint minimizuoti funkciją $p = 2x + 3y + z$, kur $x + y + z \leq 40$, $2x + y - z \geq 10$, $-y + z \geq 10$, $x \geq 0$, $y \geq 0$, $z \geq 0$, būtų sudaroma 2.5 nurodyta lygčių sistema.

$$\begin{cases} x + y + z + s_1 = 40 \\ 2x + y - z - s_2 = 10 \\ -y + z - s_3 = 10 \\ 2x + 3y + z + s_4 = 0 \end{cases} \tag{2.5}$$



5 pav. Simplekso metodo taikymo žingsniai bendroms tiesinio programavimo problemoms

Antrajame žingsnyje gauta lygčių sistema turi būti perrašoma į išplėstąją matricą, dar vadinamą pirmąją *tableau* lentele. Pirmajame matricos stulpelyje nurodomi pagrindiniai kintamieji (angl. *basic variabes)*, sekančiuose - sudarytos lygčių sistemos kintamųjų reikšmės, paskutiniame - rezultatas. 2 lentelėje pateikiama pirmajame žingsnyje gauta lygčių sistema išplėstosios matricos formoje.

2 lentelė. Pirmoji *tableau* lentelė. Algoritmo antrojo žingsnio rezultatas

	x	y	z	s_1	s_2	s_3	s_4	Rezultatas
s_1	1	1	1	1	0	0	0	40
s_2*	2	1	-1	0	-1	0	0	10
s_3*	0	-1	1	0	0	-1	0	10
s_4	2	3	1	0	0	0	1	0

Kadangi problema sprendžiama kaip ne standartinė maksimizavimo, kai kurios eilutės turi būti žymimos žvaigždutėmis (angl. *starred*). Matricoje turi būti pažymimos tos eilutės, kurių rezultato reikšmė padalinta iš pagrindinio kintamojo reikšmių susikirtimo taško yra neigiama. Pateiktoje pavyzdinėje matricoje tokios eilutės yra s_2 ir s_3, abiejų atveju gaunamas tas pats neigiamas rezultatas: -10 (10 / -1).

Tolimesnis algoritmo veikimo principas priklauso nuo to, ar sudarytoje matricoje yra žvaigždutėmis pažymėtų eilučių. Jeigu taip, trečiajame algoritmo žingsnyje sukimo stulpelis yra tas, kuris turi didžiausią teigiamą skaičių pirmoje žvaigždute pažymėtoje eilutėje. Jeigu ne - mažiausią

neigiamą skaičių turintis stulpelis paskutinėje matricos eilutėje. Kadangi aprašomu atveju egzistuoja dvi žvaigždute pažymėtos eilutės, nagrinėjama pirmoji - s_2. Didžiausias teigiamas skaičius eilutėje - 2, todėl sukimo stulpelis - *x*.

Sekančiame žingsnyje turi būti parenkama sukimo reikšmė (angl. *pivot*). Ši reikšmė renkama iš anksčiau rasto sukimo stulpelio, kiekvienos eilutės (išskyrus paskutinę) rezultato reikšmę padalinant iš sukimo stulpelio reikšmės, jei ji yra teigiama. Mažiausias gautas rezultatas yra sukimo reikšmė. Aprašomu atveju gaunamos 2 reikšmės - 40 (40 / 1), 5 (10 / 2). Parenkama mažesnioji, todėl sukimo reikšmė - 2.

Penktasis algoritmo žingsnis - atlikti pertvarkymo operaciją. Šios operacijos tikslas - kiekvieną sukimo stulpelio reikšmę (išskyrus sukimo eilutės reikšmę) pertvarkyti taip, kad reikšmė būtų lygi 0. Eilučių pertvarkymo pavyzdys pateikiamas 2.6 pavaizduotose matricose.

$$\begin{bmatrix} 1 & 1 & 1 & 1 & 0 & 0 & 0 & 40 \\ 2 & 1 & -1 & 0 & -1 & 0 & 0 & 10 \\ 0 & -1 & 1 & 0 & 0 & -1 & 0 & 10 \\ 2 & 3 & 1 & 0 & 0 & 0 & 1 & 0 \end{bmatrix} \begin{matrix} \xrightarrow{2R_1 - R_2} \\ \\ \\ \xrightarrow{R_4 - R_2} \end{matrix} \begin{bmatrix} 0 & 1 & 3 & 2 & 1 & 0 & 0 & 70 \\ 2 & 1 & -1 & 0 & -1 & 0 & 0 & 10 \\ 0 & -1 & 1 & 0 & 0 & -1 & 0 & 10 \\ 0 & 2 & 2 & 0 & 1 & 0 & 1 & -10 \end{bmatrix} \tag{2.6}$$

Gautas penktojo žingsnio rezultatas užpildomas antroje matricoje (antroje *tableau* lentelėje). Nagrinėjamo atvejo rezultatas pateiktas 3 lentelėje esančioje išplėstosios matricos formoje.

3 lentelė. Antroji *tableau* lentelė. Algoritmo penktojo žingsnio rezultatas

	x	y	z	s_1	s_2	s_3	s_4	Rezultatas
s_1	0	1	3	2	1	0	0	70
x	2	1	-1	0	-1	0	0	10
s_3*	0	-1	1	0	0	-1	0	10
s_4	0	2	2	0	1	0	1	-10

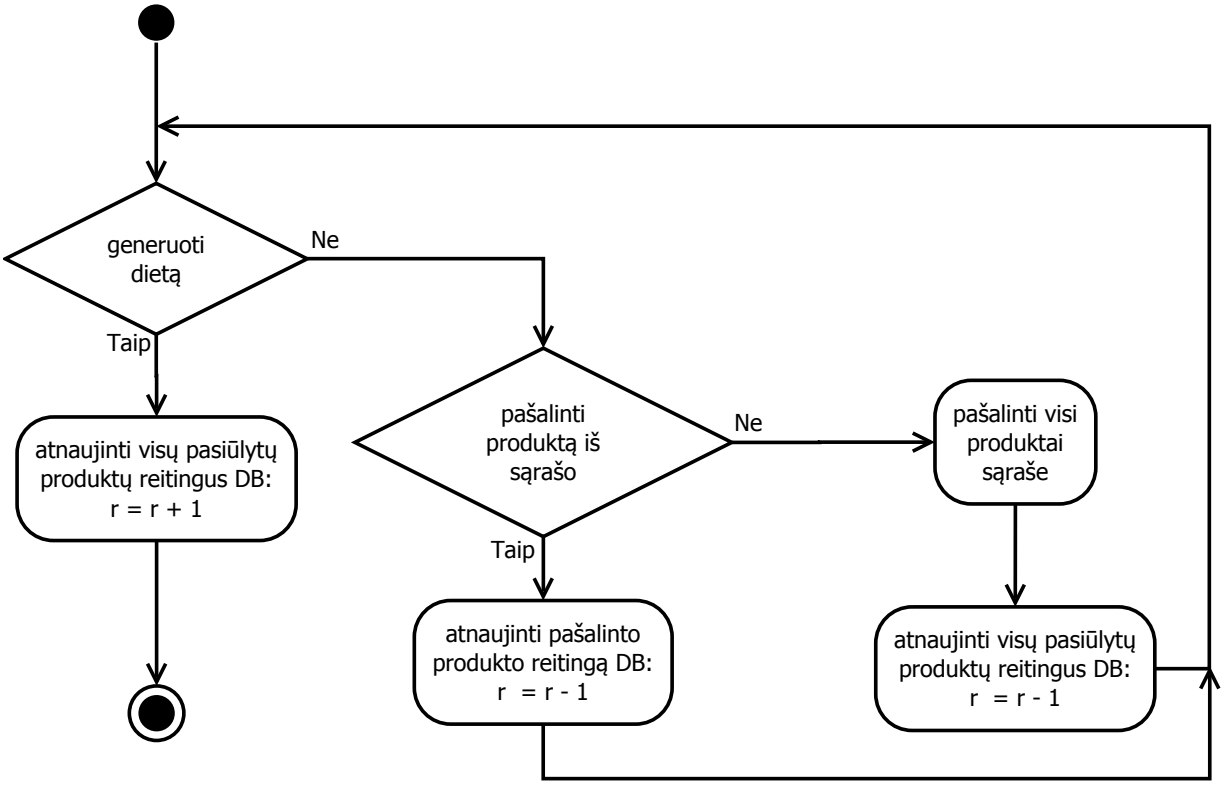
Kadangi gautame rezultate vis dar yra viena eilutė, pažymėta žvaigždute, 3 - 5 žingsniai kartojami lygiai taip pat, kaip ankstesniuoju atveju, sukimo stulpelį parenkant būtent iš žvaigždute pažymėtos eilutės. Todėl tolimesniuoju atveju sukimo reikšmė - 1, esantis s_3 eilutėje. Atlikus pertvarkymo operaciją, gaunama nauja matrica, nebeturinti žvaigždute pažymėtų eilučių, kaip pateikta 4 lentelėje.

4 lentelė. Trečioji *tableau* lentelė

	x	y	z	s_1	s_2	s_3	s_4	Rezultatas
s_1	0	4	0	2	1	3	0	40
x	2	0	0	0	-1	-1	0	20
z	0	-1	1	0	0	-1	0	10
s_4	0	4	0	0	1	2	1	-30

Pagal simplekso metodo apibrėžimą, paskutinis žingsnis: 3 - 5 žingsnių kartojimas tol, kol randamas optimalus sprendinys. Sprendinys laikomas optimaliu tada, kai paskutinėje matricos eilutėje nebelieka neigiamų skaičių. Pateiktame pavyzdyje paskutinėje eilutėje s_4 visos reikšmės $\geq$ 0, todėl gautas sprendinys laikomas optimaliu. Kadangi minimizavimo problema yra sprendžiama konvertuojant funkciją į maksimizavimo uždavinį, kaip pastebėta algoritmo autorių, funkcijas sieja lygybė *p* = -*c*, kur *p* - maksimizavimo, *c* - minimizavimo funkcijos. Todėl aprašomu atveju gaunama optimali reikšmė *p* = *30* (-(-30 / 1)), kai *x* = *10* (20 / 2), *y* = *0*, *z* = *10* (10 / 1).

Produktų reitingavimas ir siūlymas. Siekiant siūlyti naudotojui bent dalį jo mėgstamų produktų, produktai yra reitinguojami. Naudotojui generuojant dietą daroma prielaida, kad visi pasiūlyti arba pasirinkti produktai yra mėgstami. Taigi, generuojant dietą duomenų bazėje pažymima, kad konkretus naudotojas mėgsta tam tikrus produktus. Tuo atveju, kai naudotojas pašalina kažkurį produktą iš dietos arba pergeneruoja visus produktus - kiekvienam iš produktų sumažinamas reitingas duomenų bazėje. Tokiu būdu sumažinama tikimybė, kad produktas bus dar kartą pasiūlytas konkrečiam naudotojui ateityje. Aprašyta produktų reitingavimo veiklos diagrama pateikta 6 paveikslėlyje.



6 pav. Produktų reitingavimas

Jeigu naudotojas yra prisijungęs prie sistemos, siūlant produktus prieš dietos generavimą produktai yra parenkami atsižvelgiant į ankstesnius naudotojo veiksmus - reitingus. Siekiant naudotojui pasiūlyti vis naujų produktų, tačiau nepateikti visiškai atsitiktinės aibės, siūlomų produktų sąrašas sudaromas pasiūlant dalį teigiamai įvertintų ir dalį dar neįvertintų produktų. Sistemoje numatyta iš anksto naudotojui pasiūlyti 7 produktus. Sudarant atsitiktinį produktų sąrašą 4 produktai pasiūlomi iš naudotojo teigiamai įvertintų produktų ir 3 - neturintys jokio įvertinimo. Jeigu naudotojas dar nėra teigiamai įvertinęs pakankamo kiekio produktų, sąraše tiesiog atsiras daugiau atsitiktinai parinktų elementų. Neprisijungusiam naudotojui pasirinkus atsitiktinio produktų sąrašo generavimo funkcionalumą, šis sąrašas būtų sudaromas tokiu pačiu principu, t. y. parenkant visus produktus atsitiktine tvarka.

Neprisijungusiam naudotojui taip pat siūloma gauti produktų sąrašą, atsižvelgiant į kitų naudotojų reitingus. Toks sąrašas sudaromas taikant atraminių vektorių mašinų (angl. *Support Vector Machines*, sutr. SVM) modelį. D. Dirvanausko darbe [5] išskiriama, kad pagrindinė šio metodo idėja - apmokymo duomenų projektavimas į bendrą požymių (angl. *labels*) erdvę ir hiperplokštumos, kuri geriausiai klasifikuotų (atskirtų) skirtingas požymių klases, radimas. To paties autoriaus darbe apibūdinami tiesinės ir netiesinės hiperplokštumos radimo būdai. Netiesinė hiperplokštuma

naudojama tada, kai duomenys tiesiškai negali būti atskirti. Anot L. Jasmonto [11] tam, kad būtų sumažinti skaičiavimai tokių hiperplokštumų radimui, yra naudojamos kernelio funkcijos. To paties autoriaus darbe palyginti tiesinio, polinominio, Gauso bei Sigmoid kerneliai parodė, kad Gauso kernelis tirtiems duomenims buvo tiksliausias ir greičiausias. Be to, aprašomoje sistemoje naudojamo įrankio [13] naudojimosi instrukcijose taip pat nurodoma, kad būtent šis kernelis turėtų būti naudojamas pirmiausia, o kiti pasirenkami tada, jei šio rezultatas netenkina. Dėl šių priežasčių aprašomoje sistemoje naudojamam atraminių vektorių mašinų algoritmo įrankiui nurodoma naudoti būtent Gauso kernelį.

Taigi, sistemoje taikomas SVM algoritmas padeda nustatyti ar tam tikram naudotojui turėtų patikti konkretus produktas. Algoritmas klasifikuoja naudotojus į dvi grupes - tuos, kuriems patiko konkretus produktas ir tuos, kuriems nepatiko (kurie pašalino produktą iš siūlomų produktų sąrašo). Klasifikavimo metodui nurodoma, kad duomenys, pagal kuriuos reikia klasifikuoti naudotojus, yra amžius, svoris bei fizinis aktyvumas. Neprisijungusiam naudotojui nurodžius šiuos duomenis atraminių vektorių mašinų algoritmas priskirs naudotoją vienai iš grupių, t. y. parinks, kuriai grupei naudotojas, pagal savo parametrus, turėtų priklausyti. Jei naudotojas bus priskirtas grupei, kuri tam tikrą produktą mėgsta, šis produktas ir bus pasiūlomas naudotojui.

2.3. Sistemos navigacijos struktūra

Pradinis sistemos puslapis, kuris parodomas pirmą kartą į sistemą atėjusiam naudotojui - "LOGIN". Naudotojui norint plačiau paskaityti apie sistemą, jis gali nueiti į "About" puslapį. Naudotojui norint prisiregistruoti prie sistemos yra sukurtas atskiras puslapis "Register".

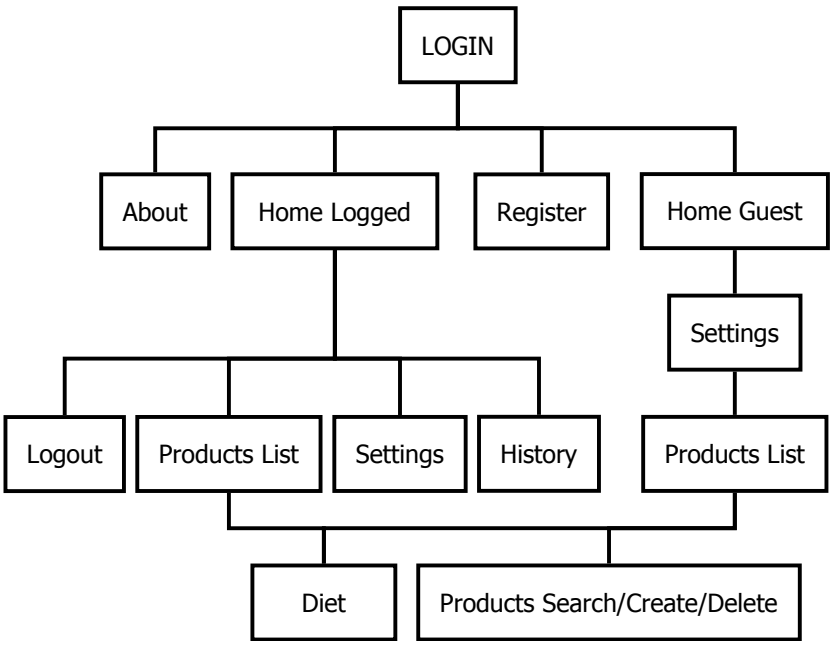
Jeigu naudotojas nenori registruotis, bet nori išbandyti sistemos funkcionalumą kaip svečias, jis yra nukreipiamas į "Home Guest" puslapį, laikantį pagrindinę informaciją, prieinamą neprisijungusiems naudotojams. Neprisijungęs naudotojas yra iš karto paprašomas užpildyti informaciją apie save (kuri vėliau bus naudojama dietos generavime), todėl yra nukreipiamas į "Settings" puslapį. Šiame puslapyje sėkmingai baigus pildyti informaciją atidaromas "Products List" puslapis, kuriame naudotojas gali pamatyti jam sistemos parinktus produktus. Šiame puslapyje naudotojas turi galimybę kurti naujus, pridėti egzistuojančius, pašalinti pasiūlytus produktus dietoje. Galiausiai, kai pasirenkamas tenkinamas produktų sąrašas dieta gali būti generuojama ir parodoma "Diet" puslapyje.

Prisijungęs naudotojas yra nukreipiamas į "Home Logged" puslapį, laikantį informaciją, prieinamą tik prisijungusiems naudotojams. Iš šio puslapio naudotojas gali pasiekti tuos pačius puslapius, kaip ir neprisijungęs naudotojas, tik nebūtinai ta pačia tvarka (jeigu prisijungimo istoriją jau yra išsaugojęs anksčiau). Papildomai prisijungęs naudotojas, nuėjęs į "Settings" puslapį galės atnaujinti savo asmeninę informaciją, o nuėjęs į "Products List" puslapį bei vėliau susigeneravęs dietą, galės ją išsaugoti. Visa ši išsaugota informacija ateityje bus pasiekiama naudotojui "History" puslapyje. Galiausiai, prisijungęs naudotojas, norėdamas atsijungti nuo sistemos gali rinktis "Logout" funkciją, matomą tik prisijungusio naudotojo aplinkoje.

Visa aprašyta sistemos navigacijos struktūra pateikiama 7 paveikslėlyje.

2.4. Duomenų bazės struktūra

Sistemos naudojamoje nutolusioje duomenų bazėje saugoma visa pagrindinė informacija apie sistemos naudotojus, jų išsaugotus veiksmus bei kitus, dietų generavimui reikiamus duomenis. Duomenų bazės lentelės ir ryšiai tarp jų pavaizduoti 8 paveikslėlyje.



7 pav. Sistemos navigacijos struktūra

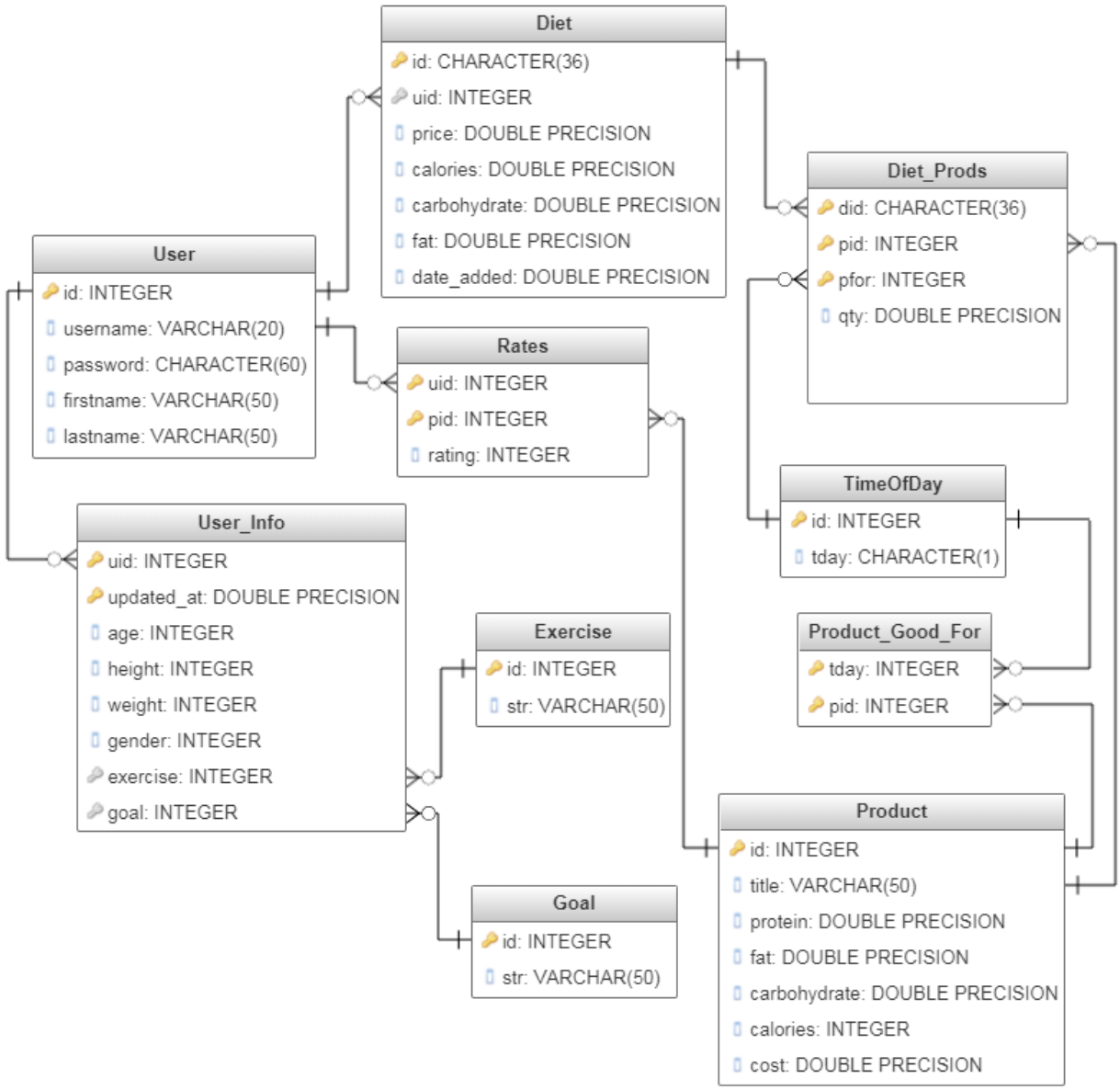
Naujam naudotojui prisiregistravus prie sistemos visų pirma išsaugoma reikiama informacija apie naudotoją. Ji talpinama duomenų bazės lentelėje *User*. Be naudotojo identifikacijos numerio (*id*) taip pat yra išsaugomas naudotojo prisijungimo vardas (*username*), užšifruoto slaptažodžio reikšmė (*password*), vardas (*firstname*) bei pavardė (*lastname*).

Naudotojui pirmą kartą prisijungus prie sistemos paprašoma užpildyti duomenis, nuo kurių priklauso, kiek maistinių medžiagų naudotojui bus rekomenduojama suvartoti. Be to, sistemos veikimo metu, naudotojas turi galimybę šiuos duomenis atnaujinti, o išsaugotą svorio kitimo istoriją - peržiūrėti. Taigi, lenteles sieja vienas-su-daug (1:N) ryšys, todėl naudotojo informacijos saugojimo lentelė (*User_Info*) turi naudotojo identifikacijos numerio reikšmę (*uid*). Lentelės pirminis raktas yra naudotojo identifikacijos numerio ir datos, kada buvo išsaugotas lentelės įrašas milisekundėmis (*updated_at*), kombinacija. Lentelėje taip pat saugomas žmogaus amžius (*age*), ūgis (*height*), svoris (*weight*), lytis (*gender*), fizinis aktyvumas (*exercise*), tikslas (*goal*). Paskutiniai 2 turi visada iš anksto nustatytas reikšmes ir yra saugomi atskirose lentelėse:

- *Exercise*, sauganti vieną iš 6 aprašytų fizinio aktyvumo reikšmių - 0, 1-2, 3, 4, 5-6, arba 7 fiziškai aktyvios dienos per savaitę, saugomos tekstiniu formatu *str* stulpelyje.
- *Goal*, sauganti naudotojo pasirinktą tikslą, kuris gali būti vienas iš trijų numatytų reikšmių - svorio mažinimas, išlaikymas arba didinimas.

Prisijungęs ir užpildęs asmeninę informaciją sistemos naudotojas turi galimybę generuoti ir išsaugoti dietas. Naudotojas gali generuoti daug dietų, tačiau kiekviena iš jų gali priklausyti tik vienam naudotojui. Todėl lenteles sieja ryšys 1:N ir dietos lentelėje (*Diet*) saugomas naudotojo, kuris sugeneravo dietą, identifikacijos numeris (*uid*). Lentelėje taip pat išsaugoma dietos kaina (*price*), dietos maistinių medžiagų kiekiai (*calories*), (*carbohydrate*), (*fat*) bei datos milisekundėmis, kada buvo išsaugota dieta, reikšmė (*date_added*). Visa ši informacija yra atvaizduojama naudotojui norint peržiūrėti išsaugotų dietų istoriją.

Jei naudotojas norėtų išsamiau pamatyti dietos sudėtį, t. y. produktus, iš kurių buvo sudaryta dieta, šie būtų nuskaitomi iš atskiros duomenų bazės lentelės (*Diet_Prods*), skirtos būtent išsaugotos dietos produktų laikymui. Lentelėje saugoma dietos, kuriai priklauso produktas, reikšmė (*did*), produkto identifikacijos numeris, kuris siejamas išoriniu raktu su produktų (*Product*) lentele, die-



8 pav. Duomenų bazės lentelės ir ryšiai tarp jų

nos metas (*pfor*), kuriam buvo pasiūlytas konkretus produktas ir produkto kiekio (*qty*) reikšmė.

Dienos metas gali būti vienas iš 3 iš anksto sistemoje saugomų reikšmių - 'b', 'l', 'd', atitinkamai pusryčiams, pietums ir vakarienei. Šios reikšmės saugomos atskiroje lentelėje - *TimeOfDay*.

Dienos meto lentelė siejama taip pat ir su produktais, norint nurodyti produkto tinkamumą valgyti konkrečiam laikui. Ši informacija saugoma *Product_Good_For* lentelėje, kuri išoriniais raktais siejama su dienos meto (*TimeOfDay*) ir produkto (*Product*) lentelėmis. Tos pačios reikšmės sudaro ir lentelės pirminio rakto kombinaciją.

Produktų lentelėje (*Product*) saugoma visa informacija, susijusi su konkrečiu produktu - pavadinimas (*title*), baltymų (*protein*), riebalų (*fat*), angliavandenių (*carbohydrate*), kalorijų (*calories*), kainos (*price*) kiekiai.

Prisijungusiam sistemos naudotojui generuojant dietą veiksmai yra sekami ir reitinguojami: kiekvienam produktui, kurį naudotojas pasirinko, padidinama egzistuojanti reitingo reikšmė, kurį pašalino - sumažinama. Ši informacija yra saugoma atskiroje duomenų bazės lentelėje *Rates*, kuri turi sąryšius su naudotojo (*User*) ir produkto (*Product*) lentelėmis. Pati reitingo reikšmė saugoma *rating* stulpelyje.

3. Programinė realizacija

Šiame skyriuje pateikiama sistemos funkcionalumo programinė realizacija. 3.1 poskyryje įvardinti programinei realizacijai naudojami įrankiai, visuose kituose poskyriuose paaiškinami pagrindinį sistemos funkcionalumą apibūdinantys sprendimai.

3.1. Programinei realizacijai naudojami įrankiai

Aprašomos sistemos vidinės dalies (angl. *back-end*) realizacijai naudojama Node.js serverio kūrimo platforma, paremta JavaScript programavimo kalba. Ši platforma suteikia prieigą prie įvairaus funkcionalumo (bibliotekų), instaliuojant norimus paketus, naudojantis vienu iš didžiausių programinės įrangos registrų - "npm". Sistemai prijungti "express" (internetinio puslapio karkasas), "express-jwt" (tarpinė programinė įranga (angl. *middleware*), leidžianti nustatyti HTTP užklausų tapatumą pagal JWT (JSON Web Token) estafetės paketus), "pg" (PostgreSQL klientas, skirtas komunikacijai tarp sistemos ir jos naudojamos duomenų bazės), "node-svm" (atraminių vektorių mašinų realizacija) moduliai. Sistemos duomenų saugojimui ir valdymui naudojama PostgreSQL duomenų bazių valdymo sistema, pasižyminti lengva komunikacija su sistemos vidinės dalies funkcionalumu, naudojant minėtąjį "pg" modulį.

Sistemos išoriniams programavimo darbams (angl. *front-end*) įgyvendinti yra naudojamas Angular 7 karkasas, veikiantis komponentais-remtu (angl. *component-based*) kūrimo principu. Informacijos apipavidalinimui naudojamos HTML bei CSS technologijos, Bootstrap karkasas, naudotojų svorio istorijos atvaizdavimui grafike naudojama *angular-highcharts* [10] direktyva.

Šios technologijos užtikrina, kad sistemos tiek vidinei, tiek išorinei dalims bus naudojama ta pati programavimo kalba - JavaScript. Renkantis technologijas atsižvelgta ir į tai, kad didelė dalis sistemai būtino funkcionalumo gali būti panaudota turint prieigą prie "npm" registrų, tokiu būdu panaudojant pažangius algoritmus ar sprendimus, jau realizuotus kitų autorių.

3.2. Produktų paieška

Sistemai buvo iškelti keli reikalavimai, susiję su egzistuojančių produktų paieška duomenų bazėje - naujų produktų kūrimas ir produktų pridėjimas į dietą. Kuriant naują produktą ir įvedant jo pavadinimą atliekama paieška duomenų bazėje ir ieškoma, ar panašaus produkto duomenų bazėje dar nėra, taip siekiant pasiūlyti visus variantus, atitinkančius naudotojo įvestį. Kadangi produktus į duomenų bazę pridėti gali visi naudotojai, ateityje duomenų bazė gali sparčiai plėstis. Dėl šios priežasties sistemoje atliekant produktų paiešką siunčiami tik reikiami ir naudotojo įvestį atitinkantys duomenys. Sprendimas buvo realizuotas atliekant reikiamus veiksmus sistemos išorinėje ir vidinėje dalyse, komunikaciją tarp jų įgyvendinant aprašytos paslaugos pagalba.

Išorinėje sistemos dalyje užtikrinama, kad naudotojas nebeveda teksto į produkto pavadinimo lauką. Taip sumažinamas užklausų kiekio siuntimas serveriui - sulaukus, kol žmogus parašys, pavyzdžiui, žodį "Apple" bus galiausiai išsiųsta tik 1 užklausa, o ne 5, naudotojui įvedus kiekvieną žodžio raidę. Optimizacijos pritaikymui panaudotas *debounceTime* Angular 2 operatorius, atidedantis veiksmo vykdymą nurodytam laiko tarpui. Tuomet, kai naudotojas baigia įvedinėti produkto pavadinimą, sistema kreipiasi į aprašytą paslaugą, kuri, savo ruožtu, komunikuoja su sistemos naudojama duomenų baze. Vidinėje sistemos dalyje aprašyta funkcija kreipiasi į duomenų bazėje aprašytą funkciją, atsakingą už produkto paiešką duomenų bazėje, pagal nurodytą produkto pavadinimo dalinę įvestį. Duomenų bazei baigus vykdyti aprašyta funkciją, rezultatas grąžina-

mas vidinei sistemos daliai, ir, paslaugos pagalba, perduodamas išorinei sistemos daliai, kurioje galiausiai atvaizduojamas ir parodomas naudotojui.

Visa paieškos logika vykdoma duomenų bazėje aprašytoje funkcijoje, kuri, visų pirma, atlieka produkto paiešką pagal gautą pavadinimo įvestį. Rezultatui priskiriama visa produktą apibūdinanti informacija (pavadinimo, baltymų, riebalų, angliavandenių, kalorijų kiekių ir kainos reikšmės). Vėliau produktui randamos produkto valgymo tinkamumo dienos metui ('b' *(breakfast)*, 'l' *(lunch)*, 'd' *(dinner)*) reikšmės. Pastarosios reikšmės gaunamos iš atskiros duomenų bazės lentelės. Ateityje pridėjus daugiau valgymo reikšmių (priešpiečiai, pavakariai ir pan.) reikšmės tiesiog būtų prijungiamos kableliais, tai užtikrintų, kad esant tokiems pakeitimams reikėtų išorinėje sistemos dalyje tik naująsias reikšmes apdoroti, nekeičiant vidinės sistemos dalies.

3.3. Produktų parinkimas dietai

Sistemoje produktai dietai parenkami atsižvelgiant į tai, ar naudotojas yra prisijungęs, ar ne. Pirmuoju atveju naudotojams siūlomi produktai taip, kaip aprašyta 3.3.1 poskyryje. Jeigu naudotojas nėra prisijungęs, jam leidžiama rinktis iš dviejų būdų - gauti produktus visiškai atsitiktine tvarka arba gauti atraminių vektorių mašinų algoritmo parinktus produktus, kuriuos naudotojas turėtų mėgti, kaip aprašyta 3.3.2 poskyryje.

3.3.1. Produktų siūlymas prisijungusiems naudotojams

Kaip ir nurodyta sistemos funkcionalumo aprašyme, prisijungusio naudotojo vykdomi veiksmai yra saugomi ir naudojami ateityje generuojant naujas dietas. Taigi, naudotojui pasirinkus dietos generavimo veiksmą, sistemos vidinėje dalyje vykdomas algoritmas, pateiktas 1 pseudokode.

1 algoritmas. Produktų parinkimo algoritmas

Įvestis: $userid \neq \varnothing, dt \neq \varnothing$
Išvestis: $products$
  $pids =$ *naudotojo mėgstamų produktų, tinkamų valgyti dt dienos metu, id*
  if pids.size > 4 **then**
    $products =$ *atsitiktine tvarka parinkti 4 produktai iš pids sąrašo*
  else
    $products =$ *atsitiktine tvarka parinkti visi pids produktai*
  end if
  $lim = 7 - MIN(pids.size, 4)$
  $unrated =$ *naudotojo neįvertinti produktai, tinkami valgyti dt dienos metu*
  $products = products +$ *(atsitiktine tvarka parinkti lim produktai iš unrated sąrašo)*

Aprašyta funkcija gauna 2 įvesties parametrus - naudotojo identifikacijos numerį *(userid)* ir dienos metą (*dt*), kuriam turi būti parenkami produktai - pusryčiams, pietums ar vakarienei. Funkcija grąžina atrinktų produktų sąrašą, atsižvelgiant į ankstesnius naudotojo veiksmus. Visų pirma, kreipiamasi į duomenų bazėje egzistuojančią reitingų lentelę, iš kurios atrenkami visi naudotojo (identifikuoto pagal *userid* parametrą) mėgstami produktai. Taip pat, pagal nurodyto dienos meto *dt* parametrą parenkami tik konkrečiam dienos metui tinkami valgyti produktai. Duomenų bazės užklausa grąžina tik šias abi sąlygas tenkinančių produktų identifikacijos numerius ir išsaugo

juos *pids* sąraše. Jeigu šio sąrašo dydis yra didesnis, nei 4, produktų masyvui atsitiktine tvarka parenkami 4 produktai iš *pids* sąrašo. Jeigu sąraše nėra pakankamo kiekio teigiamai įvertintų ir konkrečiam dienos metui tinkamų produktų, tuomet parenkami visi *pids* identifikacijos numerį turintys produktai. Galiausiai sąrašas pildomas atsitiktinai parenkamais produktais. Pirmiausia nustatoma, kiek tokių produktų reikia parinkti. Kadangi sistemoje naudotojams siūloma rinktis iš 7 produktų, didžiausias atsitiktinai parenkamų produktų skaičius - 7. Iš šios reikšmės atimama jau atrinktų produktų suma - jeigu naudotojo mėgstamų produktų, tinkamų valgyti *dt* dienos metu, kiekis yra mažesnis, nei 4 - atimama atrinktų produktų suma, priešingu atveju reikšmė sumažinama 4. Gautas rezultatas priskiriamas *lim* kintamajam. Atrenkant atsitiktinius produktus parenkami tik tie, kurie neturi konkrečiam naudotojui priskirto vertinimo, ir kurie tinkami valgyti *dt* dienos metu. Galiausiai gauti sąrašai (teigiamai įvertintų ir neįvertintų) sujungiami ir grąžinamas vienas produktų sąrašas, kuris vėliau ir pasiūlomas naudotojui.

3.3.2. Produktų siūlymas neprisijungusiems naudotojams

Neprisijungusiems naudotojams pasirinkus produktų siūlymą atsižvelgiant į atraminių vektorių mašinų algoritmo parinktus variantus, yra siunčiama užklausa, besikreipianti į iš anksto sudarytus klasifikatorius ir parenkanti rekomenduojamus atsakymus. Aprašyti klasifikatoriai yra apmokomi kiekvienos dienos pradžioje, atsižvelgiant į naujai įvertintų produktų reitingus. Šio funkcionalumo realizavimui vidinėje sistemos dalyje yra aprašyta automatiškai pasileidžianti funkcija, inicijuojanti apmokymą. Šios funkcijos veikimo principas pateiktas 2 algoritme.

2 algoritmas. Apmokymo inicijavimo algoritmas

Įvestis: $models = \{\}$
Išvestis: $models \neq \varnothing$
  $rates \leftarrow$ *visi įvertinti produktai, juos įvertinusių naudotojų informacija ir reitingai*
  $pDictionary = \{\}$
  for $i := 0$ to $rates.size - 1$ **do**
    $record = rates[i]$
    $input = [[[record.age, record.weight, record.exercise], record.rating]]$
    if $pDictionary[record.pid] = \varnothing$ **then**
      $pDictionary[record.pid] =$ *papildytas masyvas su naująja input[0] reikšme*
    else
      $pDictionary[record.pid] = input$
    end if
  end for
  for $i := 0$ to $pDictionary.size - 1$ **do**
    $key = pDictionary[i]$
    if $pDictionary[key].length \geq 2$ **then**
      $clf \leftarrow$ *node-svm bibliotekos klasifikatoriaus inicializacija*
      $models[key] = clf.train(pDictionary[key])$
    end if
  end for

Funkcija, visų pirma kreipiasi į duomenų bazę, iš kurios gauna visus tuo metu išsaugotus reitingus ir išsaugo gautą rezultatą kintamajame *rates*. Vėliau per gautą rezultatą iteruojama ir naujame

masyve išsaugomos vieno lentelės įrašo reikšmės - duomenų bazėje išsaugotas naudotojo amžius, svoris, fizinis aktyvumas. Šis masyvas talpinamas į naująjį (*input*), kuriame taip pat išsaugoma ir reitingo reikšmė, t. y. priskiriama, kuriai grupei naudotojas priklauso. Galiausiai žiūrima ar kiti naudotojai jau buvo įvertinę tokį produktą. Jeigu taip, naujasis masyvas tiesiog įrašomas kaip nauja reikšmė, papildanti jau egzistuojantį masyvą. Priešingu atveju produktui kuriamas įrašas, saugantis produkto identifikacijos numerį kaip raktą, o naująjį masyvą kaip to rakto reikšmę. Gautas objektas (*pDictionary*) saugo visus duomenis pagal kuriuos vėliau vyksta mokymas.

Tam, kad konkretus produktas galėtų būti pasiūlytas, jį turi būti įvertinę bent 2 naudotojai. Taigi, tolimesnis algoritmo veikimas užtikrina, kad tik tokie produktai bus atrinkti mokymui. Vėliau kiekvienam iš produktų inicializuojamas naujas klasifikatorius, kurio dėka visų produktą įvertinusių naudotojų duomenys yra suklasifikuojami į atskiras grupes. Klasifikatoriaus inicializavimas ir pats apmokymas vykdomas panaudojant N. Panel sukurtą atraminių vektorių mašinų algoritmo realizaciją *node-svm* [13]. Apmokyti modeliai yra išsaugomi atskirame objekto kintamajame.

Tolimesnis veikimo principas parenka visiškai atsitiktinius produktus duomenų bazėje, tinkamus valgyti tam tikram dienos metui. Vėliau yra kreipiamasi į anksčiau išsaugotus algoritmo apmokytus modelius ir tikrinama, ar konkretus produktas potencialiai galėtų patikti tam tikram naudotojui. Jeigu klasifikatorius naujuosius duomenis (naudotojo asmeninius parametrus) priskiria tai žmonių grupei, kuriems produktas patiko, šis produktas ir yra pridedamas prie vėliau grąžinamų reikšmių. Toks veiksmas kartojamas tol, kol parenkamas pakankamas kiekis produktų. Galiausiai dar kartą kreipiamasi į duomenų bazę ir grąžinama detalesnė atrinktų produktų informacija (jų maistinės vertės), kuri vėliau ir yra parodoma naudotojui.

3.4. Dietos generavimas

Norint užtikrinti naudotojams siūlomų produktų ir generuojamų dietų tinkamumą dienos metui, sistemoje, sudarant dietą, produktai parenkami trejiems apibrėžtiems valgymo metams - pusryčiams, pietums ir vakarienei. Naudotojui produktų pasirinkimo puslapyje pasirinkus tinkamą produktų sąrašą, duomenys perduodami dietos kūrimo komponentui. Taigi, komponentui nusiunčiami 3 produktų sąrašai, skirti skirtingiems valgymo metams. Vėliau pradedamas simplekso metodo taikymas, taip, kaip aprašyta teoriniame sistemos modelyje, 2.2.3 poskyryje.

Pirmieji 2 algoritmo žingsniai yra apjungti - aprašytoje sistemos funkcijoje užtikrinama, kad kreipiantis į funkciją būtų sudaroma ir grąžinama *tableau* lentelė, kuri yra antrojo žingsnio rezultatas. Kadangi sistemoje, generuojant dietą, atsižvelgiama į baltymų, riebalų, angliavandenių ir kalorijų kiekius, pradinė matrica yra apibrėžta statiškai, kur pertekliniai kintamieji, nurodantys rekomenduojamų maistinių medžiagų normas, turi statines reikšmes (1, jei ribojama maksimali arba -1, jei ribojama minimali reikšmė). Taigi, taikant Cristina-Elena HREŢCANU ir Ciprian-Ionel HREŢCANU [8] pasiūlytą modelį, sudaroma 5 lentelėje pateikta pradinė *tableau* lentelė.

Lentelės kintamieji p_{min}, p_{max}, f_{min}, f_{max}, car_{min}, car_{max}, cal_{min}, cal_{max} nurodo atitinkamai baltymų, riebalų, angliavandenių, kalorijų ribas. Kintamasis *p* apibrėžia pačią funkciją (minimizavimo problemą), o *Ans* - saugomo rezultato reikšmę. Sistemoje, skaičiuojant rezultatą, leidžiama 20% paklaida, tokiu būdu sumažinant atvejus, kai iš siūlomų produktų nebūtų įmanoma sugeneruoti optimalios dietos, tenkinančios apibrėžtus rezultatus. Dėl to kiekviena maistinių medžiagų sąlygos reikšmė dauginama iš 0.8, jei sąlyga nurodo minimalią, arba iš 1.2, jei maksimalią rekomenduojamą reikšmę. Kadangi tokia pati matrica sudaroma ir pusryčių, ir pietų, ir vakarienės dietos generavimo atveju, *Ans* reikšmė dauginama iš koeficiento *c*, nurodančio, kokia visos paros normos dalis konkretaus valgymo atveju turėtų būti suvartojama. Sistemoje nurodyta, kad pus-

5 lentelė. Pradinė *tableau* lentelė

	p_{min}	p_{max}	f_{min}	f_{max}	car_{min}	car_{max}	cal_{min}	cal_{max}	p	Ans
p_{min}*	-1	0	0	0	0	0	0	0	0	p*0.8*c
p_{max}	0	1	0	0	0	0	0	0	0	p*1.2*c
f_{min}*	0	0	-1	0	0	0	0	0	0	f*0.8*c
f_{max}	0	0	0	1	0	0	0	0	0	f*1.2*c
car_{min}*	0	0	0	0	-1	0	0	0	0	car*0.8*c
car_{max}	0	0	0	0	0	1	0	0	0	car*1.2*c
cal_{min}*	0	0	0	0	0	0	-1	0	0	cal*0.8*c
cal_{max}	0	0	0	0	0	0	0	1	0	cal*1.2*c
p	0	0	0	0	0	0	0	0	1	0

ryčių metu turėtų būti gaunama 40%, pietų ir vakarienės - po 30% rekomenduojamų visos paros maistinių medžiagų normų. Taigi, pavyzdžiui, jeigu naudotojui rekomenduojama suvartoti 2000 kalorijų per dieną, sudarant pusryčių planą koeficientas *c* turėtų reikšmę 0.4, o apibrėžtoje išplėstinėje matricoje $cal_{min}*$ eilutės *Ans* reikšmė būtų lygi 640 (2000*0.8*0.4).

Vėliau, remiantis tuo pačiu modeliu, matrica papildoma produktų maistinių medžiagų kiekiais. Taigi, iteruojant per kiekvieną produktą į matricą įterpiamas papildomas stulpelis *x*, nurodantis produkto *x* baltymų, riebalų, angliavandenių, kalorijų ir kainos reikšmes. Funkcijos grąžinama reikšmė - algoritmo antrojo žingsnio rezultatas.

Trečiame algoritmo žingsnyje, visų pirma, tikrinama ar matricoje yra žvaigždute pažymėtų eilučių. Sprendime aprašyta funkcija, skirta rasti sukimo eilutę, iteruoja per kiekvieną matricos eilutę, ir tikrina, ar pirmasis eilutės stulpelis yra pažymėtas. Jei taip, grąžinamas eilutės indeksas, priešingu atveju - reikšmė -1. Vėliau parenkamas maksimalios rastos eilutės reikšmės indeksas, kuris nurodo teoriniame sistemos modelyje aprašyto sukimo stulpelio reikšmę.

Sekančiame žingsnyje apskaičiuojama sukimo reikšmė *pivot*. Už žingsnį atsakinga funkcija iteruoja per visas matricos eilutes, apskaičiuojant koeficientą *c*, nurodantį eilutės paskutinio ir sukimo stulpelių reikšmių dalmenį. Mažiausias gautas rezultatas yra sukimo reikšmė.

Vykdant penktąjį algoritmo žingsnį rezultato matricai turi būti atliekama pertvarkymo operacija. Norint sukimo stulpelyje esančias reikšmes pertvarkyti taip, kad jos būtų lygios 0, algoritmo autoriai [15] siūlo naudoti Gauso-Žordano metodą. Metodas vykdomas tol, kol sukimo stulpelyje visos (išskyrus sukimo) reikšmės, tampa lygios 0. Norint tą pasiekti kiekvienai matricos eilutei reikia atlikti perstatos veiksmą, apibrėžtą kaip $xR_i \pm yR_{pr}$, kur *x* - sukimo reikšmė, R_i - perstatoma eilutė, *y* - $R_i[pc]$, *pc* - sukimo stulpelis, *pr* - sukimo eilutė. Taigi, norint apskaičiuoti perstatytos eilutės reikšmes, sistemoje vykdomi 3 algoritme aprašyti veiksmai.

Visų pirma, algoritmas kaip įvesties parametrus gauna informaciją apie sukimo reikšmę - indeksus *pc* ir *pr*, nurodančius, kurioje pozicijoje masyve *a* yra sukimo elementas. Algoritmas grąžina masyvą *a'*, kuriam pritaikytas pertvarkymo operacijos algoritmas. Algoritmas iteruoja per visas masyvo eilutes. Jeigu eilutė, kuriam atliekama pertvarkymo operacija, nėra sukimo eilutė ir nagrinėjamai eilutei reikia atlikti pertvarkymo veiksmą (nagrinėjamos eilutės sukimo stulpelio reikšmė $\neq 0$), tuomet apskaičiuojamas koeficiento *y* ženklas - nustatoma, ar yR_{pr} reikšmę reikės pridėti ar atimti bei suskaičiuojamas pats *y* koeficientas, parenkant nagrinėjamos eilutės ir sukimo stulpelio susikirtimo elemento reikšmės modulį. Galiausiai, kiekvienam pertvarkomos eilutės elementui pakeičiama reikšmė, kuri apskaičiuojama pagal pritaikytą minėtą $xR_i \pm yR_{pr}$ išraišką.

Visi išvardinti žingsniai sugeneruoja vieną dietą. Kadangi sistemoje dienos dieta yra išskaidyta

3 algoritmas. Pertvarkymo operacijos algoritmas

Įvestis: $pc, pr, a \neq \varnothing$
Išvestis: a'

for $i := 1$ to N **do**
  if $i \neq pr$ **and** $a[i][pc] \neq 0$ **then**
    if $\frac{a[pr][pc]}{a[i][pc]} > 0$ **then**
      $sign = -1$
    else
      $sign = 1$
    end if
    $y = |a[i][pc]|$
    for $j := 1$ to N **do**
      $a[i][j] = a[pr][pc] * a[i][j] + sign * y * a[pr][j]$
    end for
  end if
end for

į 3 (pusryčiams, pietums ir vakarienei), algoritmas kartojamas 3 kartus. Dietos tarpusavyje bendrų ir priklausomų kintamųjų neturi, todėl generuojamos vienu metu, asinchroniškai. Jeigu kažkuriai dietai neegzistuoja optimalus sprendinys, nei viena iš dietų nėra parodoma naudotojui. Tokiu atveju naudotojas yra informuojamas su atitinkama klaidos žinute.

3.5. Sistemos saugumas

Kadangi sistemoje egzistuoja kelių tipų naudotojai, sistemoje užtikrinama, kad kiekviena naudotojų grupė galės vykdyti tik tam tikrus konkrečiai grupei prieigos teisėmis suteikiamus veiksmus. Sistemoje apibrėžti dviejų tipų naudotojai - prisijungę ir neprisijungę. Kadangi prisijungusių naudotojų veiksmus gali atlikti tik nustatyto tapatumo asmenys, priklausantys prisijungusių naudotojų grupei, visi, turimų prieigos teisių patvirtinimo reikalaujami veiksmai yra apsaugoti estafetės paketais. Toks paketas turi atsitiktinę simbolių reikšmę, priklausančią nuo naudotojo identifikacijos numerio. Kiekvieną kartą naudotojui prisijungus prie sistemos priskiriama nauja rakto reikšmė. Ši reikšmė sudaroma naudojant *jsonwebtoken* [9] biblioteką. Naudojama bibliotekos funkcija taip pat padidina saugumą, užšifruojant sugeneruotą estafetės paketą RSA algoritmu bei nurodant, kad paketas galioja tik nurodytą laiko tarpą. Estafetės paketas visuomet perduodamas užklausos antraštėje, todėl vidinės sistemos dalyje gavus tam tikrą užklausą, prieš atliekant teisių patvirtinimo reikalaujamą veiksmą, tikrinama, ar užklausoje toks estafetės paketas yra nurodytas ir ar jis yra galiojantis. Užklausa vykdoma tik tuo atveju, jei estafetės paketas priklauso būtent tam naudotojui, kuris siuntė užklausą. Priešingu atveju serveris grąžina klaidos žinutę su 401 klaidos kodu.

Siekiant užtikrinti sistemoje saugomų naudotojų slaptų duomenų apsaugą, sistemoje yra paslepiama tokių duomenų reikšmė juos užšifruojant. Naudotojų slaptų duomenų grupei yra priskiriami slaptažodžiai. Naujam naudotojui registruojantis prie sistemos slaptažodžio reikšmė, prieš įrašant ją į duomenų bazę, yra užšifruojama "Bcrypt" šifravimo metodu. Slaptažodžių šifravimui sistemoje prijungta *bcrypt* [3] biblioteka. Naudotojui jungiantis prie sistemos nurodytam slaptažodžiui taip pat apskaičiuojama šifro reikšmė, kuri vėliau yra lyginama su šifru, saugomu duomenų bazėje. Tik tuo atveju, kai šios reikšmės yra lygios, naudotojas yra prijungiamas prie sistemos.

Išvados ir rekomendacijos

Darbo metu buvo atlikta sukurtų ir rinkoje veikiančių sprendimų analizė, išskiriant kiekvieno iš jų pliusus bei minusus. Pagal atsižvelgiamus kriterijus sistemai buvo suformuluoti pagrindiniai uždaviniai ir poreikiai. Remiantis išsikeltais sistemos reikalavimais, buvo sukurta žiniatinklio programų sistema, taikanti simplekso metodą ir gebanti spręsti optimalios dietos problemą, kaip tiesinio programavimo uždavinį. Šios sistemos tikslas - suteikti sistemos naudotojams galimybę gauti individualiai jiems sugeneruotas optimalias dietas, pagal jų asmeninius parametrus. Sistemos realizavimo metu buvo siekiama užtikrinti, kad naudotojai turėtų galimybę generuojant dietas pasirinkti jų poreikius tenkinančius produktus, o sugeneruotas dietas - redaguoti. Prisijungusiems naudotojams taip pat realizuotos galimybės atlikti su istorijos saugojimu susijusius veiksmus - dietų išsaugojimą, trynimą, progreso stebėjimą, naujų sprendimų gavimą atsižvelgiant į ankstesnius veiksmus. Neprisijungusiems naudotojams įgyvendinta galimybė gauti rekomenduojamus produktus, atsižvelgiant į kitų naudotojų įvertinimus. Atsižvelgiant į viso sistemos realizavimo procesą bei gautus rezultatus, pateikiamos tokios išvados:

- Atlikus sukurtų sprendimų analizę pastebėta, kad dažniausiai šiais laikais kuriamos automatizuotos dietų parinkimo sistemos orientuojasi į naudotoją, bet ne į dietos optimalumą.
- Remiantis nagrinėta literatūra pastebėta, kad optimali dieta yra tokia, kuri ne tik tenkina išsikeltas maistinių medžiagų normų ribas, bet ir tokia, kurios sprendiniu siekiama optimizuoti rezultatą p, klasikinės dietos uždavinyje dažniausiai apibrėžiamą kaip dietos kainą.
- Sudarant teorinį sistemos modelį nustatyta, kad sistemoje realizuotas tiesinių lygčių sistemos modelis priklauso bendroms tiesinio programavimo problemoms.
- S. Waner ir S. Costenoble [15] pateikiami simplekso metodo žingsniai pritaikyti Cristina-Elena HREŢCANU ir Ciprian-Ionel HREŢCANU [8] siūlomam optimalios dietos parinkimo modeliui patvirtina, kad tiesinis programavimas ir simplekso metodas sėkmingai gali būti taikomas parenkant optimalias dietas.

Siekiant praplėsti sistemos naudojamumą, susidomėjimą tarp potencialių naudotojų, tęsiant darbus sistemoje būtų galima realizuoti įvairesnių problemos sprendimo būdų. Vienas iš rekomenduojamų - surinkus pakankamą kiekį naudotojams patinkančių/nepatinkančių dietų, problema galėtų būti sprendžiama dirbtinio intelekto pagalba. Sistemoje taip pat galėtų būti praplėstas pats funkcionalumas, pavyzdžiui leidžiantis naudotojams susidaryti dietas ilgesniam laiko tarpui, jas atsispausdinti. Be to, būtų naudinga, jei sistema leistų naudotojams apsibrėžti įvairias ligas kuriomis serga ar kurių produktų netoleruoja - dabartinis sistemos funkcionalumas tik užtikrina, kad nemėgstami produktai nebus siūlomi naudotojui, bet jų niekaip negrupuoja, t. y. kiekvieną iš jų naudotojas būtų prašomas vertinti atskirai.

Literatūros šaltiniai

[1] *Stigler's diet problemStigler's diet problem*, pages 784--784. Springer US, Boston, MA, 2001.

[2] Maliha Agha and Riaz Agha. The rising prevalence of obesity: part a: impact on public health. *International journal of surgery. Oncology*, 2(7):e17, 2017.

[3] Nicholas Campbell. Lib to help you hash passwords, 2010. https://www.npmjs.com/package/bcrypt.

[4] George B. Dantzig. The diet problem. *Interfaces*, 20(4):43--47, 1990.

[5] Darius Dirvanauskas. Dirbtinio intelekto metodų taikymas kraštų aptikimui., 2016.

[6] David Frankenfield, Lori Roth-Yousey, and Charlene Compher. Comparison of predictive equations for resting metabolic rate in healthy nonobese and obese adults: A systematic review. *Journal of the American Dietetic Association*, 105:775--89, 06 2005.

[7] Susan Garner Garille and Saul I. Gass. Stigler's diet problem revisited. *Operations Research*, 49(1):1--13, 2001.

[8] Cristina-Elena HREŢCANU and Ciprian-Ionel HREŢCANU. A linear programming model for a diet problem. *Food and Environment Safety Journal*, 9(1), 2017.

[9] Auth0 Inc. An implementation of json web tokens, 2015. https://www.npmjs.com/package/jsonwebtoken.

[10] Felix Itzenplitz. Highcharts directive for angular, 2017. https://www.npmjs.com/package/angular-highcharts.

[11] Lukas Jasmontas. Mašinų mokymosi metodų taikymas didelių duomenų kiekių apdorojimui., 2017.

[12] Mark D. Mifflin, Sachiko T. St Jeor, Lisa A. Hill, Barbara J. Scott, Sandra A. Daugherty, and Young O. Koh. A new predictive equation for resting energy expenditure in healthy individuals. *American Journal of Clinical Nutrition*, 51(2):241--247, 2 1990.

[13] Nicolas Panel. Support vector machine (svm) library for nodejs, 2014. https://www.npmjs.com/package/node-svm.

[14] Corné van Dooren. A review of the use of linear programming to optimize diets, nutritiously, economically and environmentally. *Frontiers in Nutrition*, 5:48, 2018.

[15] S. Waner and S. Costenoble. *Finite Mathematics*. Cengage Learning, 2017.