



VILNIAUS KOLEGIJA
ELEKTRONIKOS IR INFORMATIKOS FAKULTETAS

RESTORANO UŽSAKYMŲ VALDYMO SISTEMA

BAIGIAMASIS DARBAS
BD 6531BX028 PI21A

DIPLOMANTĖ

2025-01-06

GODA

SAKALAUSKAITĖ

VADOVAS

2025-01-06

MARIUS

GŽEGOŽEVSKIS

2025



VILNIUS KOLEGIJA HIGHER EDUCATION INSTITUTION
FACULTY OF ELECTRONICS AND INFORMATICS

RESTAURANT ORDER MANAGEMENT SYSTEM

FINAL PROJECT
FP 6531BX028 PI21A

UNDERGRADUATE

01/06/2025

GODA
SAKALAUSKAITĖ

SUPERVISOR

01/06/2025


MARIUS
GŽEGOŽEVSKIS

2025

**VILNIAUS KOLEGIJA
ELEKTRONIKOS IR INFORMATIKOS FAKULTETAS**

TVIRTINU

**Elektronikos ir informatikos fakulteto
prodekanė**

 **dr. Laura Gžegoževskė**

2024 m. spalio mėn. 31 d.

BAIGIAMOJO DARBO UŽDUOTIS

Skirta **PI21A** grupės diplomantei **Godai Sakalauskaitei** 2024 m. spalio mėn. 31 d.

Baigiamojo darbo tema: Restorano užsakymų valdymo sistema

Baigiamojo darbo tema anglų kalba: Restaurant Order Management System


Baigiamojo darbo tikslas: sukurti restorano užsakymų valdymo sistemą.

Baigiamojo darbo uždaviniai:

1. Suprojektuoti programėlės architektūrą.
2. Sukurti klientų programėlės vartotojo sąsają. Įgyvendinti meniu, užsakymo ir apmokėjimo funkcionalumus.
3. Sukurti darbuotojų programėlės sąsają. Įgyvendinti prisijungimo, meniu tvarkymo ir užsakymų patvirtinimo funkcionalumus.
4. Sukurti duomenų mainų sistemą tarp klientų ir darbuotojų programėlių.

Baigiamojo darbo realizavimo priemonės: IntelliJ IDEA Ultimate, Firebase, GitHub, Expo Go.

Baigiamasis darbas bus ginamas Programinės įrangos katedros posėdyje 2025 m. sausio mėn. 6 d.

Diplomantė.......... Goda Sakalauskaitė
(parašas) (vardas, pavardė)

Baigiamojo darbo vadovas..... Marius Gžegoževskis
(parašas) (vardas, pavardė)

Patvirtinta:

Programinės įrangos katedros vedėjas..... Justinas Zailskas
(parašas) (vardas, pavardė)

Baigiamojo darbo konsultantai:

..........dr. Igor Katin
(parašas) (vardas, pavardė)

..........dr. Joana Katina
(parašas) (vardas, pavardė)

Anglų kalbos konsultantė:

..........Milda Kiškytė
(parašas) (vardas, pavardė)

SANTRAUKA

Vilniaus kolegija

Elektronikos ir informatikos fakultetas

Programinės įrangos katedra

Studijų programa: Programų sistemos, valstybinis kodas – 6531BX028

Baigiamojo darbo tema: **Restorano užsakymų valdymo sistema**

Diplomantė **Goda Sakalauskaitė**

Vadovas **Marius Gžegoževskis**

Darbo apimtis – 80 p. teksto be priedų, 36 paveikslai, 8 lentelės, 14 informacijos šaltinių, 1 priedas.

Baigiamojo darbo tikslas – sukurti modernią restoranų užsakymų valdymo sistemą, kuri optimizuotų užsakymų apdorojimą, pagerintų klientų aptarnavimą ir supaprastintų restoranų darbuotojų darbo procesus. Šiam tikslui pasiekti buvo sukurta programa, apimanti klientų ir darbuotojų skirtas vartotojo sąsajas, meniu valdymą, užsakymų patvirtinimą bei realaus laiko duomenų mainus.

Darbo metu buvo suprojektuota lanksti ir pritaikoma programėlės architektūra, aprašyti funkciniai ir nefunkciniai reikalavimai, sukurti duomenų bazės modeliai bei įdiegtos prisijungimo, užsakymų valdymo ir meniu redagavimo funkcijos. Programos vystymui naudoti įrankiai, tokie kaip IntelliJ IDEA Ultimate, Firebase, GitHub ir Expo Go, užtikrino efektyvų ir šiuolaikišką programinės įrangos kūrimo procesą.

Rezultatai parodė, kad uždaviniai buvo sėkmingai įgyvendinti. Sistema efektyviai valdo užsakymus ir realaus laiko informaciją, tačiau tolesniam vystymui rekomenduojama optimizuoti sprendimus, siekiant pritaikyti didesniam naudotojų srautui.

Reikšminiai žodžiai: užsakymų valdymas, restoranas, hibridinė mobilioji programėlė, meniu, duomenų mainai.

SUMMARY

Vilniaus Kolegija Higher Education Institution

Faculty of Electronics and Informatics

Department of Software Development

Study Programme: Software Engineering, state code– 6531BX028

Title of the Final Project: **Restaurant Order Management System**

Undergraduate **Goda Sakalauskaitė**

Supervisor **Marius Gžegoževskis**

Length of the work – 80 p. text without annexes, 36 pictures, 8 tables, 14 references, 1 annexes.

The goal of the thesis is to create a modern restaurant order management system that would optimize order processing, improve customer service, and simplify the work processes of restaurant employees. To achieve this goal, an application was created that includes user interfaces for customers and employees, menu management, order confirmation, and real-time data exchange.

During the work, a flexible and adaptable app architecture was designed, functional and non-functional requirements were described, database models were created, and login, order management, and menu editing functions were implemented. Tools used for program development, such as IntelliJ IDEA Ultimate, Firebase, GitHub, and Expo Go, ensured an efficient and modern software development process.

The results showed that the tasks were successfully implemented. The system effectively manages orders and real-time information, but for further development, it is recommended to optimize solutions in order to adapt to a larger user flow.

Keywords: order management, restaurant, hybrid mobile app, menu, data exchange.

TERMINŲ IR SANTRAUPŲ PAAIŠKINIMŲ SĄRAŠAS

- *Užsakymų valdymas* – Tai procesų rinkinys, skirtas tvarkyti ir apdoroti užsakymus nuo jų pateikimo iki įvykdymo.
- *Hibridinė mobilioji programėlė* – Tai mobilioji programėlė, kuri sukurta naudojant universalias technologijas (pvz., JavaScript, React Native), leidžiančias tą pačią programėlę naudoti ir „Android“, ir „iOS“ įrenginiuose.
- *Duomenų mainai* – Tai procesas, kai duomenys perduodami tarp skirtingų sistemų, įrenginių ar programų.
- *Emuliatorius*- Tai programinė įranga arba įrenginys, kuris imituoja kitos sistemos veikimą.
- *Klaidų žurnalas(Log angliškai)*- Tai sistema arba failas, kuriame registruojama išsami informacija apie klaidas, atsirandančias programinės įrangos veikimo metu. Ši informacija yra automatiškai fiksuojama ir naudojama problemų diagnostikai bei sistemų veikimo stebėsenai.

TURINYS

LENTELIŲ SĄRAŠAS	9
PAVEIKSLŲ SĄRAŠAS.....	10
ĮVADAS	12
1. UŽDUOTIES FORMULAVIMAS	14
1.2. Funkciniai reikalavimai	14
1.3. Nefunkciniai reikalavimai.....	16
2. UŽDUOTIES ANALIZĖ	17
2.1. Panaudos atvejų diagrama ir aprašas	17
2.2. Esybių–ryšių diagrama ir aprašas	19
2.3. Veiklos diagramos ir aprašas	19
3. PROGRAMINĖS REALIZACIJOS APRAŠYMAS	32
3.1. Duomenų bazės aprašymas.....	32
3.2. Programinis kodas.....	34
4. DIEGIMO IR NAUDOJIMO INSTRUKCIJA	62
4.1. Programinės realizacijos priklausomybė nuo kitų programinių produktų	62
4.2. Kompiuterinės technikos parametrai	62
4.3. Programinės realizacijos paleidimas	63
4.4. Programėlės naudojimosi instrukcija.....	64
4.5. Programinės realizacijos šalinimo žingsniai	77
5. IŠVADOS IR SIŪLYMAI	78
INFORMACIJOS ŠALTINIŲ SĄRAŠAS	80

LENTELIŲ SĄRAŠAS

1 lentelė. Meniu peržiūra	20
2 lentelė. Užsakymo pateikimas ir apmokėjimas	21
3 lentelė. Krepšelio peržiūra.....	23
4 lentelė. Prisijungimas prie sistemos	24
5 lentelė. Registracija prie sistemos	26
6 lentelė. Užsakymų peržiūrėjimas.....	28
7 lentelė. Užsakymų būsenos tvarkymas	29
8 lentelė. Meniu tvarkymas	31

PAVEIKSLŲ SĄRAŠAS

1 pav. Kliento panaudos atvejų diagrama.....	18
2 pav. Darbuotojo panaudos atvejų diagrama	18
3 pav. Esybių ryšiu diagrama	19
4 pav. Meniu peržiūra.....	20
5 pav. Užsakymo pateikimas ir apmokėjimas.....	21
6 pav. Krepšelio peržiūra.....	23
7 pav. Prisijungimas prie sistemos	24
8 pav. Registracija prie sistemos	26
9 pav. Užsakymo peržiūrėjimas.....	27
10 pav. Užsakymų būsenos tvarkymas.....	29
11 pav. Meniu tvarkymas	30
12 pav. Paleidimas.....	64
14 pav. Meniu ekrana	65
13 pav. Qr kodo ekranas	65
15 pav. Rodyti daugiau.....	66
16 pav. Užkandžių kategorija	66
17 pav. Pridėti į krepšelį.....	67
18 pav. Krepšelio vaizdas.....	67
19 pav. Apmokėjimo vaizdas	67
20 pav. Blogas formatas	68
21 pav. Užsakymas pateiktas.....	68
23 pav. Prisijungimo ekranas	69
22 pav. Pradinis ekranas	69
24 pav. Registracijos ekranas	70
25 pav. Suvesti registracijos duomenys.....	70
26 pav. Užsakymų ekranas	71
27 pav. Užsakymo patvirtinimas	71
29 pav. Patiekalo informacija	72
28 pav. Meniu ekranas.....	72
31 pav. Redaguoti ekranas kai nėra	73
30 pav. Redaguoti ekranas.....	73
32 pav. Ištrinti ekranas.....	74
33 pav. Pridėti ekranas.....	74
35 pav. Naujas patiekalas	75
34 pav. Užpildytas pridėti ekranas	75

36 pav. Naujas patiekalas pas klientą	76
---	----

IVADAS

Restorano užsakymų valdymo sistemos tema buvo pasirinkta dėl nuolatinio poreikio gerinti restoranų veiklos efektyvumą bei užtikrinti kokybišką klientų aptarnavimą. Šiuolaikinėje visuomenėje restoranai ir kitos maitinimo įstaigos susiduria su daugybe iššūkių, tokių kaip lėtai apdorojami užsakymai, netikslumai užsakymų perdavime, didėjantis klientų reikalavimas dėl aptarnavimo greičio ir kokybės. Šie iššūkiai dažnai kyla dėl pasenusių sistemų arba jų nebuvimo.

Mano pasirinkta tema yra aktuali ir svarbi, nes šiandien technologijos tampa neatsiejama restoranų veiklos dalimi. Tačiau egzistuojančios rinkoje sistemos, tokios kaip „UberEats“ ar „Wolt“, orientuojasi tik į išorinius procesus – užsakymų pateikimą klientų programėlėje ir jų pristatymą, tačiau jos neoptimizuoja vidinių restoranų procesų. Šios sistemos dažnai neatsižvelgia į individualius restoranų poreikius, pavyzdžiui, meniu administravimą ar tiesioginį užsakymų patvirtinimą.

Restoranų užsakymų valdymo sistema, kurią planuoju sukurti, siūlo ne tik klientams patogų užsakymo procesą, bet ir efektyvius sprendimus darbuotojams, tokius kaip meniu valdymas, užsakymų patvirtinimas ir realaus laiko informacijos mainai. Ši sistema yra inovatyvi, nes ji integruoja kelis svarbius funkcionalumus į vieną sprendimą ir leidžia restoranams efektyviau organizuoti darbą.

Pasirinkta tema yra aktuali ne tik vietiniu, bet ir pasauliniu mastu. Restoranai siekia išlikti konkurencingi, todėl investuoja į technologijas, kurios užtikrina greitą ir kokybišką aptarnavimą. Tokios sistemos padeda mažinti darbo sąnaudas, išvengti klaidų ir padidinti klientų pasitenkinimą, todėl jų svarba kasmet auga.

Darbo tikslas - sukurti restorano užsakymų valdymo sistemą.

Uždaviniai:

1. Suprojektuoti programėlės architektūrą, užtikrinančią sistemos lankstumą, patikimumą ir pritaikomumą įvairioms restoranų veikloms.
2. Sukurti klientams skirtą vartotojo sąsają, kuri leistų peržiūrėti meniu, pateikti užsakymą ir jį apmokėti.
3. Sukurti darbuotojams skirtą vartotojo sąsają, kurioje būtų įdiegtos prisijungimo, meniu valdymo, užsakymų patvirtinimo ir užsakymų stebėjimo funkcijos.
4. Sukurti duomenų mainų sprendimus, leidžiančius užtikrinti sklandų ryšį tarp klientų ir darbuotojų programėlių.

Realizavimo priemonės. Šiam darbui realizuoti buvo naudojami šiuolaikiniai programinės įrangos kūrimo įrankiai, įrenginiai bei technologijos užtikrinančios greitą ir efektyvų sistemos kūrimą:

- **IntelliJ IDEA Ultimate** pasirinkta kaip pagrindinė programavimo aplinka dėl jos siūlomų funkcionalumų, tokių kaip kodo rašymo patogumas, klaidų diagnostika ir integracija su kitomis programavimo bibliotekomis.
- **Firebase** bus naudojama kaip duomenų valdymo platforma, kuri užtikrina realaus laiko duomenų mainus ir patikimą autentifikacijos mechanizmą.
- **GitHub** – tai versijų kontrolės sistema, kuri padės išlaikyti darbo nuoseklumą ir leis sekti atliktus pakeitimus.
- **Expo Go** pasirinktas dėl galimybės greitai testuoti ir paleisti mobilias aplikacijas tiek „Android“, tiek „iOS“ įrenginiuose
- **Nešiojamasis kompiuteris** buvo naudotas programavimui, duomenų bazių kūrimui ir naudojimui bei programėlių paleidimui.
- **Mobilusis telefonas(IOS)** naudotas pirmos programėlės testavimui, stebėjimui duomenų mainų realiu laiku bei funkcijų veikimo tikrinimas.
- **Emulatorius(Android Studio Device Manager)** naudotas antros programėlės testuotavimui, stebėjimui duomenų mainų realiu laiku bei funkcijų veikimo tikrinimas.
- **Draw.io** naudotas piešti diagramas.

1. UŽDUOTIES FORMULAVIMAS

Šioje dalyje apibrėžiami restorano užsakymų valdymo sistemos funkciniai ir nefunkciniai reikalavimai. Ši sistema skirta supaprastinti restoranų veiklą, apimant pagrindinius procesus, tokius kaip meniu administravimas, užsakymų pateikimas, apdorojimas. Sistema sudaryta iš dviejų pagrindinių komponentų: klientų programėlės ir darbuotojų programėlės.

Klientų programėlė suteikia galimybę patogiai peržiūrėti meniu, pateikti užsakymus ir atlikti mokėjimus. Ji pritaikyta naudoti tiek mokėjimui grynais vietoje, tiek užsisakant maistą sumokant PayPal. Darbuotojų programėlė leidžia administruoti meniu, tai yra pridėti naujus patiekalus arba pašalinti, redaguoti esamus patiekalus. Valdyti užsakymus, priimti bei atmesti Atnaujinti informacija realiu laiku.

Sistema orientuota į dvipusį duomenų mainų mechanizmą, kuris užtikrina greitą ir efektyvą ryšį tarp klientų ir restorano darbuotojų. Ji sukurta naudojant šiuolaikinius technologinius sprendimus, užtikrinant intuityvų naudojimą ir aukštą sistemos našumą.

1.2. Funkciniai reikalavimai

Funkciniai reikalavimai nustato pagrindinius sistemos funkcionalumus ir vartotojo sąsajos ypatybes, kurios užtikrins vartotojo patirtį ir darbuotojų darbo efektyvumą. Reikalavimai skirstomi į dvi pagrindines kategorijas: klientų programėlės funkcijas ir darbuotojų programėlės funkcijas.

Klientų programėlės funkcijos:

- **Galimybė peržiūrėti meniu**
 - **Pradiniai duomenys:** duomenų bazėje saugomas meniu su patiekalų pavadinimais, aprašymais, kainomis ir nuotraukomis.
 - **Atliekami veiksmai:** vartotojas prisijungia prie programėlės, pasirenka meniu kategoriją ir peržiūri patiekalus.
 - **Rezultatas:** vartotojui pateikiamas pasirinktos kategorijos meniu sąrašas.
- **Galimybė pateikti užsakymus**
 - **Pradiniai duomenys:** kliento pasirinkti patiekalai, jų kiekis.
 - **Atliekami veiksmai:** vartotojas pasirenka patiekalus, patvirtina užsakymą ir nurodo apmokėjimo būdą.
 - **Rezultatas:** užsakymas perduodamas darbuotojų programėlei apdoroti.
- **Galimybė apmokėti užsakymus**

- **Pradiniai duomenys:** užsakymo suma ir kliento pasirinktas apmokėjimo būdas (korte, grynaisiais).
- **Atliekami veiksmai:** sistema priima mokėjimą, patvirtina jo sėkmingą įvykdymą ir išsiunčia klientui patvirtinimą.
- **Rezultatas:** užsakymas pažymimas kaip apmokėtas.

Darbuotojų programėlės funkcijos:

○ **Galimybė prisijungti**

- **Pradiniai duomenys:** darbuotojo prisijungimo vardas ir slaptažodis.
- **Atliekami veiksmai:** sistema patikrina įvestą informaciją su duomenų bazėje esančiais duomenimis.
- **Rezultatas:** sėkmingai prisijungus, vartotojas pasiekia sistemos valdymo funkcijas.

○ **Galimybė prisiregistruoti**

- **Pradiniai duomenys:** darbuotojo prisijungimo vardas, pavardė, pozicija, telefono numeris, el. Pašto adresas ir slaptažodis.
- **Atliekami veiksmai:** sistema išsaugoja duomenis į duombazę ir leidžia prisijungti.
- **Rezultatas:** sėkmingai prisiregistravus, vartotojas pasiekia sistemos valdymo funkcijas.

○ **Galimybė valdyti meniu**

- **Pradiniai duomenys:** administratoriaus įvesti patiekalų duomenys (pavadinimas, aprašymas, kaina).
- **Atliekami veiksmai:** darbuotojas prideda naują patiekalą, redaguoja esamus įrašus arba pašalina pasenusius patiekalus.
- **Rezultatas:** atnaujintas meniu matomas klientų programėlėje.

○ **Galimybė peržiūrėti ir patvirtinti užsakymus**

- **Pradiniai duomenys:** klientų pateikti užsakymai, kuriuos sistema saugo duomenų bazėje.
- **Atliekami veiksmai:** darbuotojas peržiūri naujus užsakymus, patvirtina jų vykdymą arba pažymi kaip įvykdytus.

- **Rezultatas:** užsakymas pažymimas kaip patvirtintas arba įvykdytas.

1.3. Nefunkciniai reikalavimai

Nefunkciniai reikalavimai apibrėžia techninius standartus, kurie užtikrins sistemos kokybę, našumą ir saugumą.

1. **Našumo reikalavimai.** Sistema turi apdoroti iki 100 užsakymų per valandą be veikimo trikdžių.
2. **Patikimumas ir atsparumas:**
 - Sistema turi būti atspari tinklo trikdžiams ir užtikrinti duomenų sinchronizavimą, kai ryšys atnaujinamas.
 - Sistema turi perduoti duomenis tarp klientų ir darbuotojų programėlių greitai ir stabiliai, be trikdžių ar užlūžimų.
3. **Naudojamumas.** Programėlė turi būti intuityvi, o pagrindiniai veiksmai (užsakymo pateikimas ar patvirtinimas) turi užtrukti ne ilgiau nei 5 paspaudimus.
4. **Suderinamumas.** Programėlė turi veikti „Android“ ir „iOS“ platformose.
5. **Saugumas.** Klientų ir darbuotojų prisijungimo duomenys turi būti saugomi naudojant šifravimo metodus.

2. UŽDUOTIES ANALIZĖ

Šiame skyriuje pateikiama restoranų užsakymų valdymo sistemos analizė. Analizė remiasi UML diagramomis ir esybių–ryšių modeliavimu, siekiant tiksliai ir aiškiai aprašyti sistemos struktūrą, procesus bei duomenų sąveiką. Restoranų užsakymų valdymo sistema suteikia galimybę valdyti meniu, priimti ir apdoroti užsakymus, stebėti jų būseną ir apmokėjimą bei užtikrinti patogų sąveikos modelį vartotojams. Analizėje akcentuojamos pagrindinės funkcijos, įskaitant kliento užsakymo teikimą, darbuotojų atliekamą užsakymų apdorojimą bei administratoriaus atliekamą sistemos priežiūrą.

Pagrindiniai analizės tikslai:

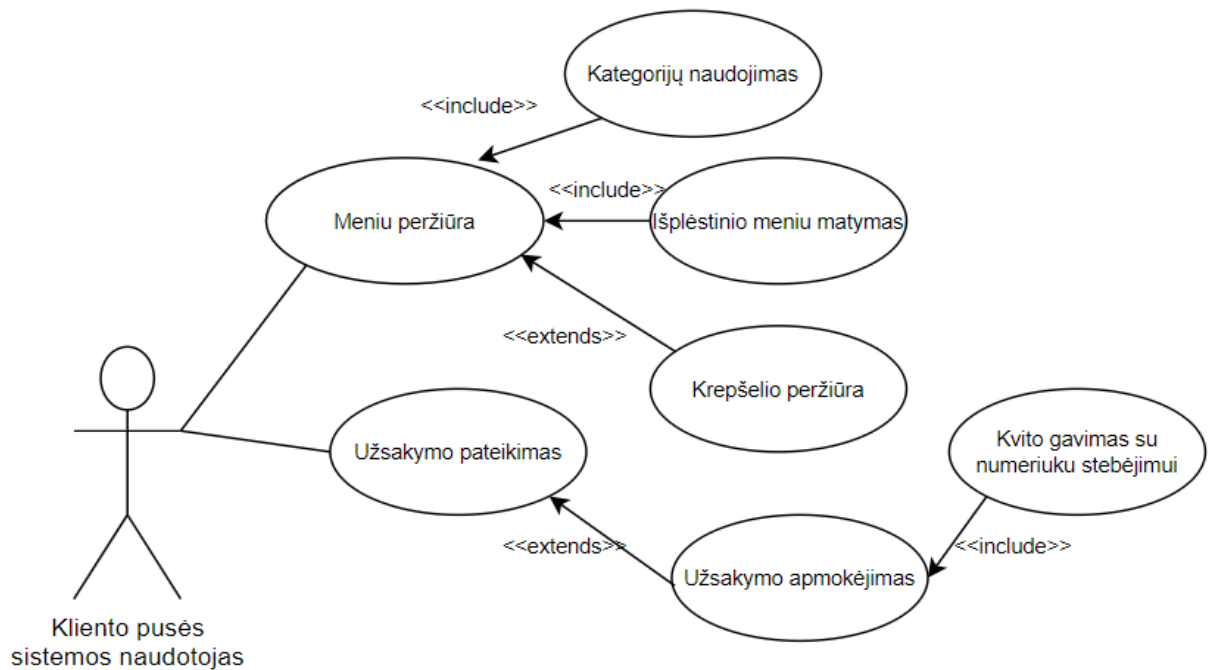
- Vizualizuoti sistemos komponentus, jų funkcijas ir ryšius.
- Apibrėžti vartotojo sąveiką su sistema, pabrėžiant, kaip įvairūs vartotojai atlieka užduotis.
- Modeliuoti duomenų struktūrą ir jos sąryšius, užtikrinant duomenų nuoseklumą ir prieinamumą.

2.1. Panaudos atvejų diagrama ir aprašas

Panaudos atvejų diagrama restorano užsakymų valdymo sistemoje vizualizuoja konkrečias vartotojų ir sistemos sąveikas, identifikuodama funkcijas, reikalingas kliento, darbuotojo ir administratoriaus poreikiams tenkinti. Pavyzdžiui, klientas per sistemą peržiūri meniu, pateikia užsakymą, stebi užsakymo būseną bei apmoka užsakymą. Darbuotojas per sistemą peržiūri gautus užsakymus, atnaujiną jų būsenas ir tvarko meniu turinį. Administratorius valdo naudotojų paskyras, konfigūruoja sistemos nustatymus ir atlieka kitus priežiūros darbus.

Panaudos atvejų kliento pusės diagrama:

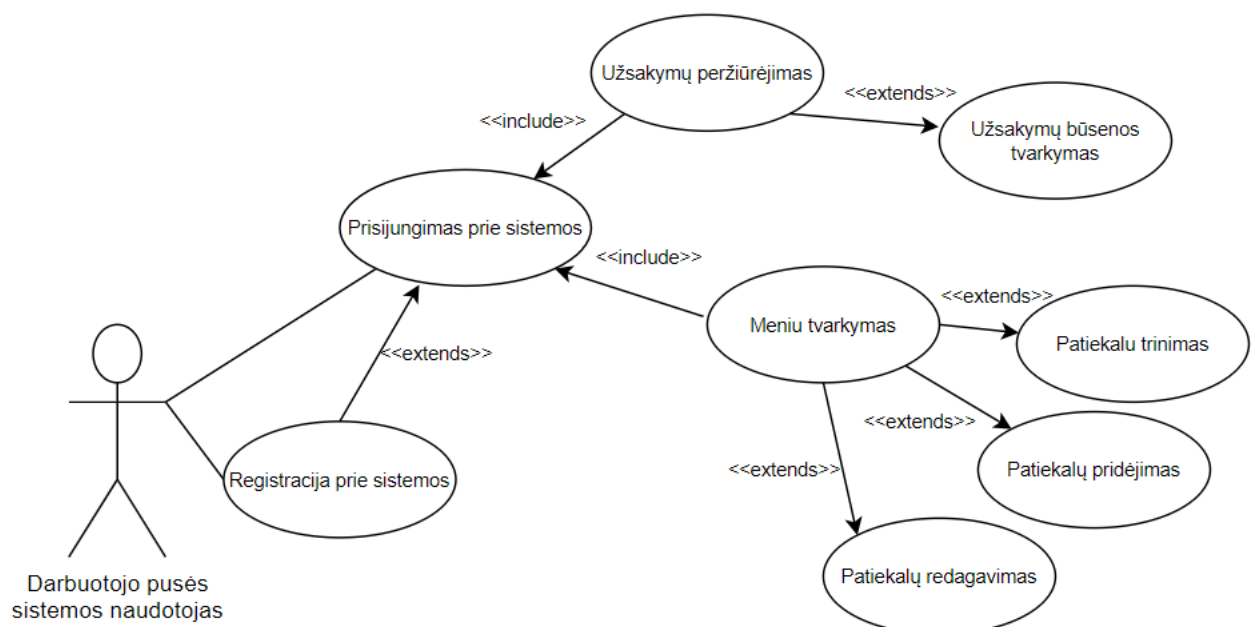
- **Klientas:** peržiūri meniu, gali naudoti kategorijas, matyti išplėsta meniu bei peržiūrėti krepšeli. Pateikia užsakymą, apmoka užsakymą ir gauna kvita su savo užsakymo numeriu (žr. 1 pav.).



1 pav. Kliento panaudos atvejų diagrama

Panaudos atvejų darbuotojo pusės diagrama:

- **Darbuotojas:** prisijungia arba prisiregistruoja prie sistemos, peržiūri užsakymus, patvirtina/atšaukia užsakymus, tvarko meniu trindamas, pridėdamas arba redaguodamas patiekalus.

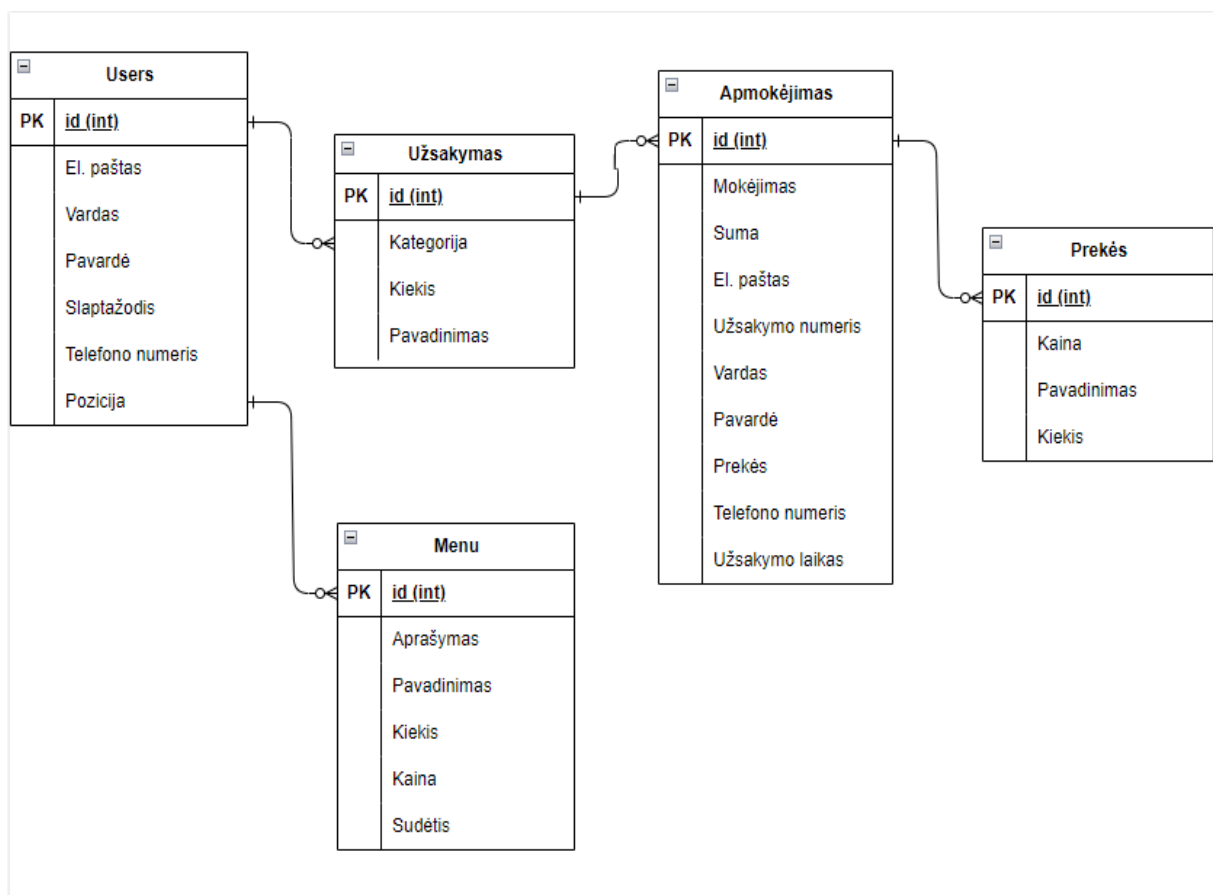


2 pav. Darbuotojo panaudos atvejų diagrama

2.2. Esybių–ryšių diagrama ir aprašas

Restorano užsakymų valdymo sistemos esybių–ryšių modelis tiksliai apibrėžia pagrindines sistemos duomenų esybes, jų atributus ir tarpusavio ryšius. Ryšių modelis atspindi, kaip duomenys organizuojami ir sąveikauja sistemoje.

Diagrama:



3 pav. Esybių ryšių diagrama

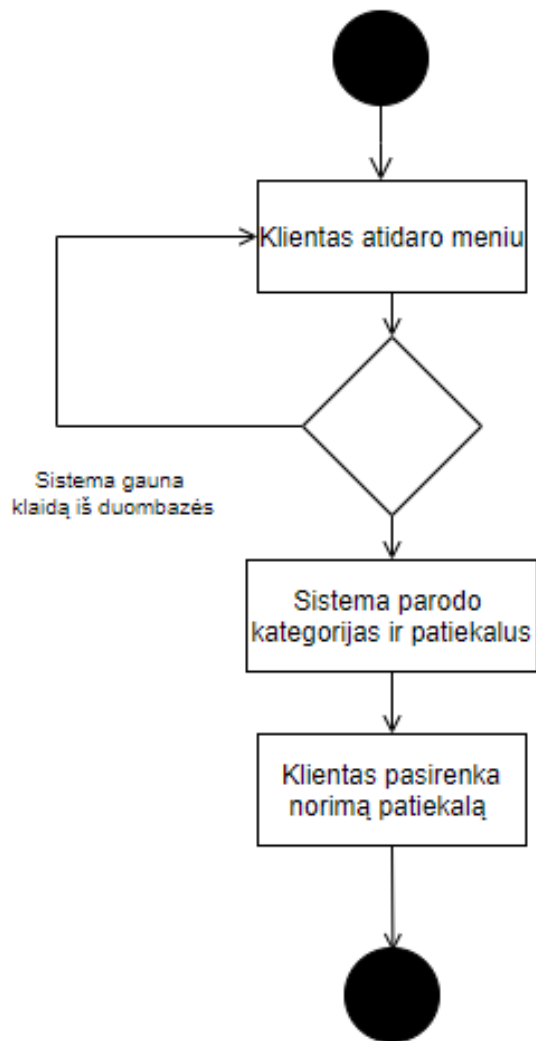
2.3. Veiklos diagramos ir aprašas

Veiklos diagramos aiškiai parodo, kaip vyksta pagrindiniai procesai nuo pradžios iki pabaigos, nurodant veiksmų seką, dalyvių sąveikas ir galimus alternatyvius scenarijus. Kiekviena veiklos diagrama apima alternatyvius scenarijus, pavyzdžiui, klaidas prisijungiant, serverio gedimus ar kitus trikdžius, ir nurodo, kaip sistema reaguoja į tokias situacijas.

1. Veiklos diagrama: Meniu peržiūra

Aprašymas. Veiklos diagrama „Meniu peržiūra“ aprašo procesą, kaip klientas peržiūri restorano meniu, norėdamas pasirinkti patiekalus užsakymui.

Diagrama:



4 pav. Meniu peržiūra

Lentelė:

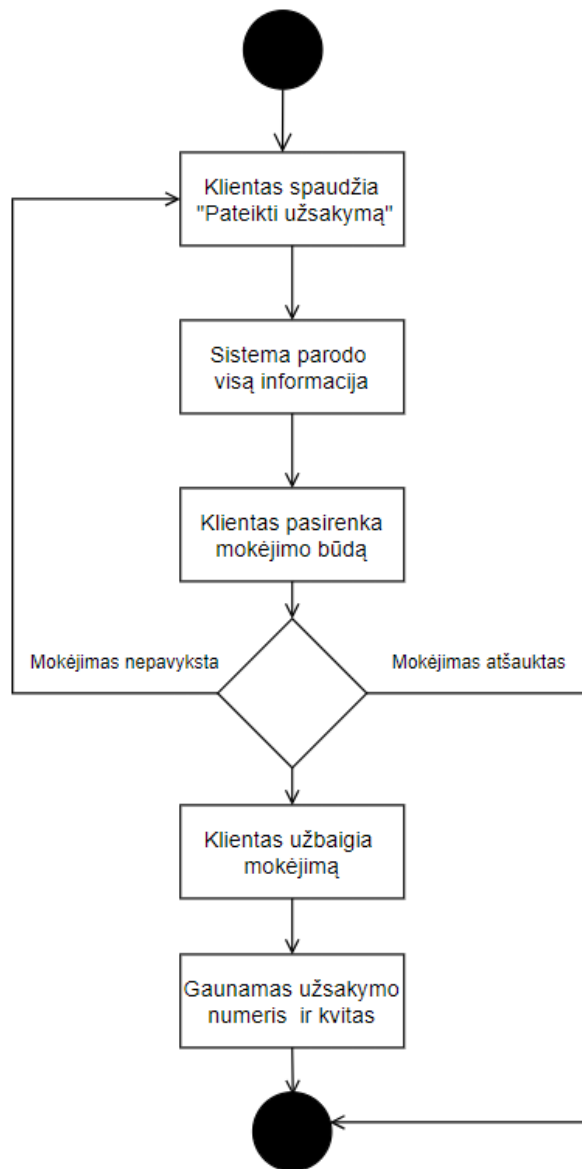
1 lentelė. Meniu peržiūra

Pavadinimas	Meniu peržiūra
ID	VD1
Aprašymas	Procesas, kaip klientas peržiūri meniu.
Aktorius	Klientas
Prieš sąlygos	Klientas turi įsijungti sistemą naudojant QR kodą.
Pagrindinis scenarijus	<ol style="list-style-type: none"> 1. Klientas atidaro meniu peržiūros langą. 2. Sistema parodo meniu kategorijas ir patiekalus. 3. Klientas peržiūri pasirinktus patiekalus.
Alternatyvūs scenarijai	<ol style="list-style-type: none"> 3.1. Klientas nusprendžia neperžiūrėti visų meniu kategorijų ir grįžta į pagrindinį langą. 3.2. Sistema negali parodyti meniu (pvz., dėl serverio klaidos), pateikiamas klaidos pranešimas.
Po sąlygos	Klientas žino patiekalų sudėtį ir kainas.

2. Veiklos diagrama: Užsakymo pateikimas ir apmokėjimas

Aprašymas. Veiklos diagrama „Užsakymo pateikimas ir apmokėjimas“ iliustruoja, kaip klientas pasirenka patiekalus, suformuoja užsakymą ir atlieka apmokėjimą.

Diagrama:



5 pav. Užsakymo pateikimas ir apmokėjimas

Lentelė:

2 lentelė. Užsakymo pateikimas ir apmokėjimas

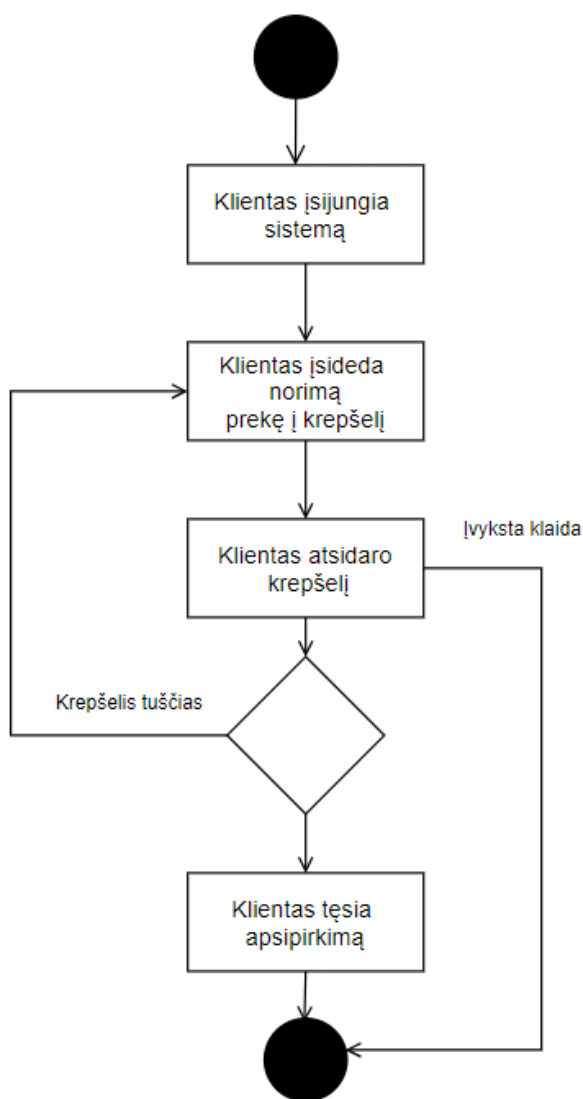
Pavadinimas	Užsakymo pateikimas ir apmokėjimas
ID	VD2
Aprašymas	Procesas, kaip klientas pateikia ir apmoka užsakymą.

Aktorius	Klientas
Prieš sąlygos	Klientas turi pasirinkti patiekalus iš meniu.
Pagrindinis scenarijus	1. Klientas pasirenka patiekalus ir spaudžia „Pateikti užsakymą“. 2. Sistema rodo bendrą kainą ir mokėjimo informaciją. 3. Klientas pasirenka mokėjimo būdą ir užbaigia mokėjimą. 4. Gaunamas užsakymo numeris ir kvitas
Alternatyvūs scenarijai	3.1. Klientas nutraukia procesą ir grįžta į meniu peržiūrą. 3.2. Mokėjimas nepavyksta (pvz., kortelės klaida), klientas gauna pranešimą su instrukcijomis pakartoti mokėjimą.
Po sąlygos	Užsakymas sėkmingai pateiktas, apmokėtas ir gautas užsakymo numeris.

3. Veiklos diagrama: Krepšelio peržiūra

Aprašymas. Veiklos diagrama aprašo, kaip klientas gali peržiūrėti savo užsakymo krepšelį.

Diagrama:



6 pav. Krepšelio peržiūra

Lentelė:

3 lentelė. Krepšelio peržiūra

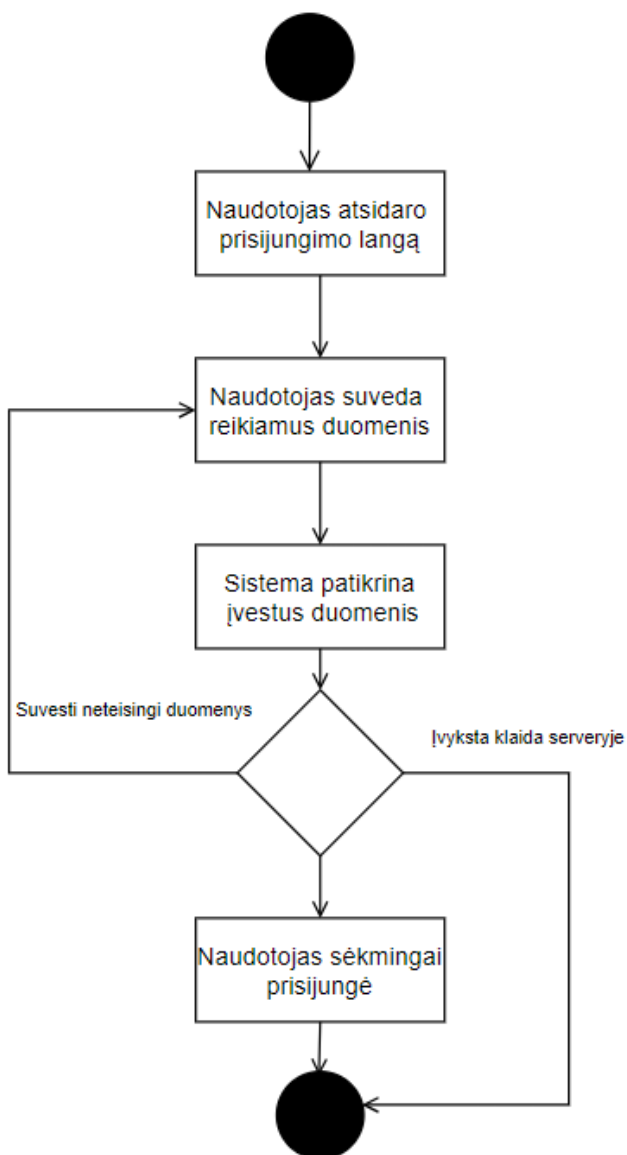
Pavadinimas	Krepšelio peržiūra
ID	VD3
Aprašymas	Procesas, kaip klientas peržiūri savo užsakymo krepšelį.
Aktorius	Klientas
Prieš sąlygos	Klientas turi būti prisidėjęs prekių į krepšelį.
Pagrindinis scenarijus	1. Klientas įsijungia sistemą. 2. Klientas prisideda norimų prekių į krepšelį. 3. Klientas atidaro krepšelį.
Alternatyvūs scenarijai	3.1. Sistemai nepavyksta parodyti krepšelio(pvz., dėl klaidos), pateikiamas klaidos pranešimas.

	3.2. Sistemai nepavyksta parodyti krepšelio, nes klientas nieko neįsidėjo, pateikiamas pranešimas.
Po sąlygos	Klientas mato savo norimas prekes ir kiekius.

4. Veiklos diagrama: Prisijungimas prie sistemos

Aprašymas. Ši diagrama aprašo, kaip darbuotojas prisijungia prie sistemos.

Diagrama:



7 pav. Prisijungimas prie sistemos

Lentelė:

4 lentelė. Prisijungimas prie sistemos

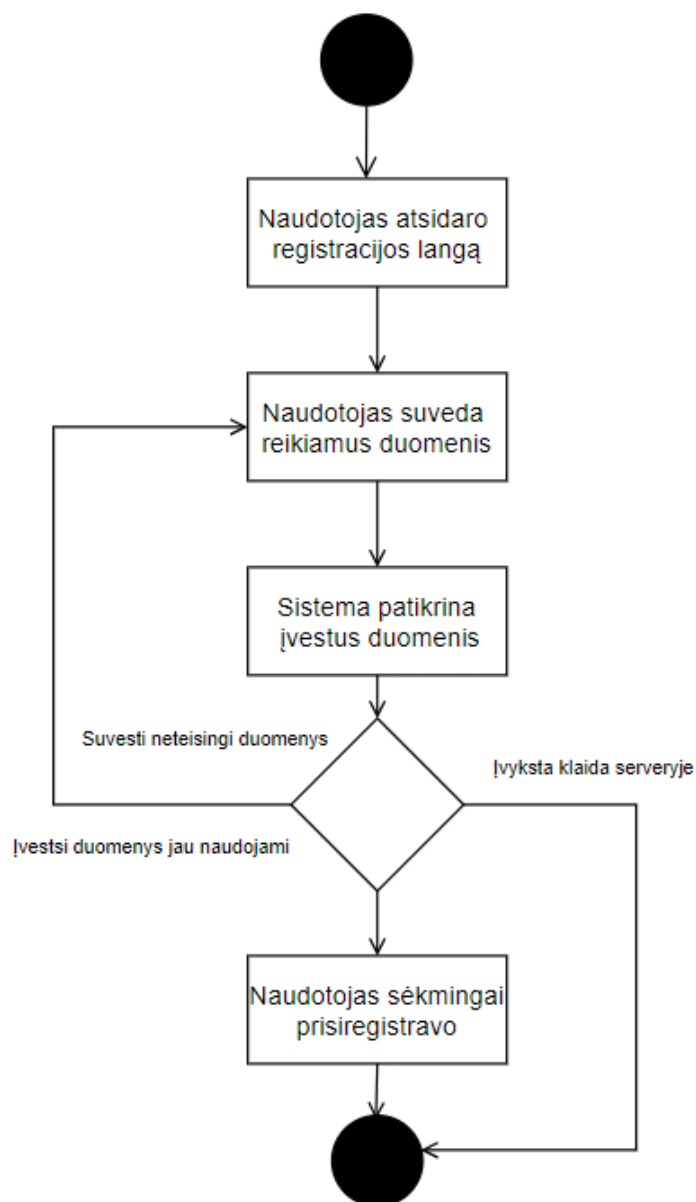
Pavadinimas	Prisijungimas prie sistemos
-------------	-----------------------------

ID	VD4
Aprašymas	Procesas, kaip naudotojas prisijungia prie sistemos.
Aktorius	Darbuotojas
Prieš sąlygos	Naudotojas turi galiojančius prisijungimo duomenis.
Pagrindinis scenarijus	1. Naudotojas įveda el. pašto adresą ir slaptažodį. 2. Sistema patikrina įvestus duomenis. 3. Jei prisijungimas sėkmingas, naudotojas patenka į pagrindinį puslapį.
Alternatyvūs scenarijai	2.1. Sistema aptinka neteisingus duomenis ir pateikia klaidos pranešimą. 2.2. Prisijungimas nepavyksta dėl serverio klaidos, pateikiamas klaidos pranešimas.
Po sąlygos	Naudotojas prisijungęs prie sistemos.

5. Veiklos diagrama: Registracija prie sistemos

Aprašymas. Ši diagrama aprašo, kaip naujas darbuotojas registruojasi prie sistemos.

Diagrama:



8 pav. Registracija prie sistemos

Lentelė:

5 lentelė. Registracija prie sistemos

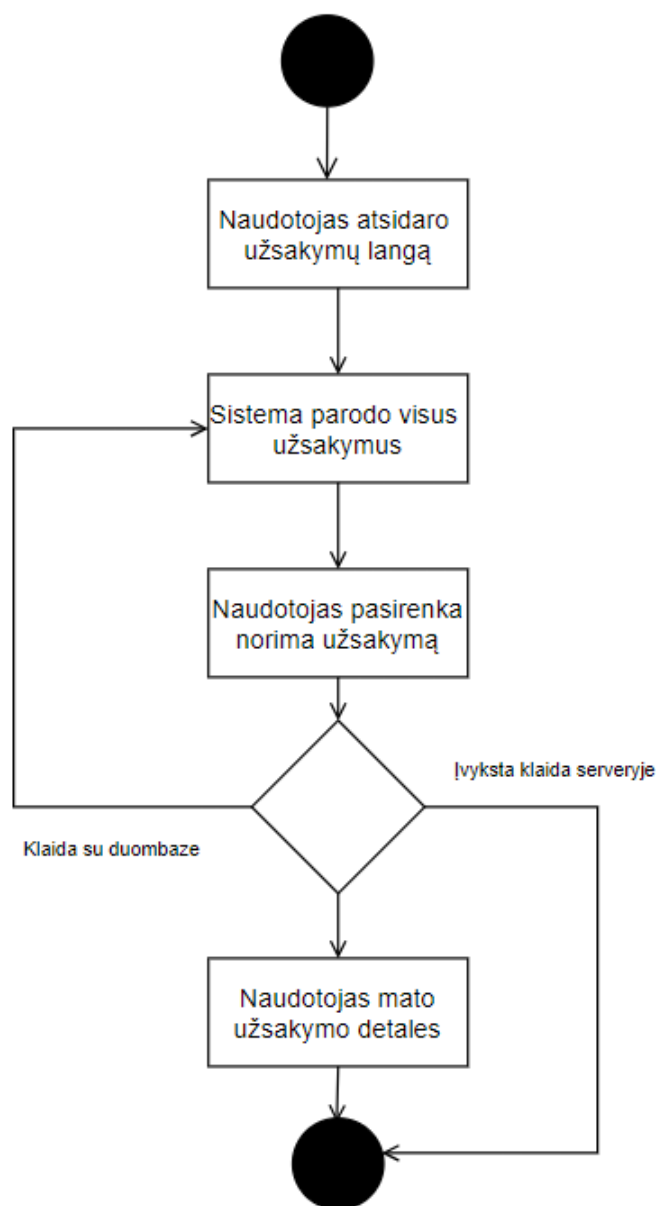
Pavadinimas	Registracija prie sistemos
ID	VD5
Aprašymas	Procesas, kaip kuriamas naujas vartotojas registruojantis prie sistemos.
Aktorius	Darbuotojas
Prieš sąlygos	Naudotojas suveda galiojančius ir tikrus asmeninius duomenis.
Pagrindinis scenarijus	<ol style="list-style-type: none"> 1. Naudotojas įveda vardą, pavardę, poziciją, telefono numerį, el. pašto adresą ir slaptažodį. 2. Sistema patikrina įvestus duomenis užtikrinimui, kad jie tinkami ir dar nenaudojami jau esamo kito darbuotojo. 3. Jei registracija sėkminga, naudotojas patenka į pagrindinį puslapį.

Alternatyvūs scenarijai	2.1. Sistema aptinka neteisingus duomenis ir pateikia klaidos pranešimą. 2.2. Prisijungimas nepavyksta dėl serverio klaidos, pateikiamas klaidos pranešimas. 2.3. Sistema aptinka tokius duomenis pas kita vartotoja, išmetamas klaidos pranešimas.
Po sąlygos	Naudotojas prisiregistruvęs prie sistemos.

6. Veiklos diagrama: Užsakymų peržiūrėjimas

Aprašymas. Veiklos diagrama aprašo, kaip darbuotojas peržiūri gautus užsakymus.

Diagrama:



9 pav. Užsakymo peržiūrėjimas

Lentelė:

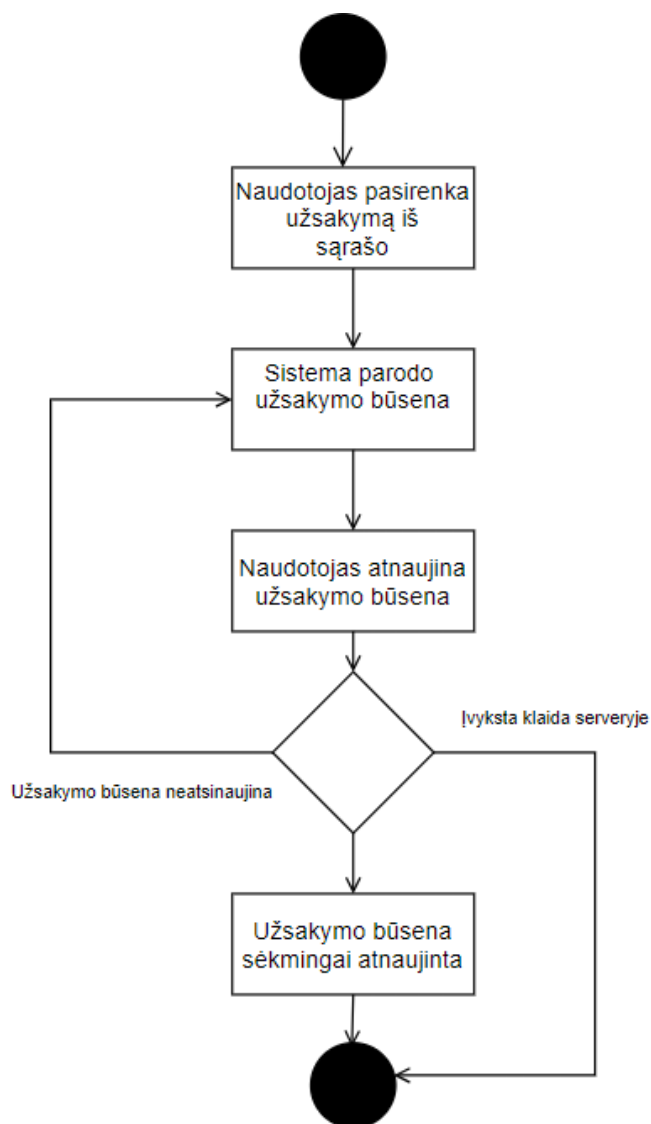
6 lentelė. Užsakymų peržiūrėjimas

Pavadinimas	Užsakymų peržiūrėjimas
ID	VD6
Aprašymas	Procesas, kaip darbuotojas peržiūri užsakymų sąrašą.
Aktorius	Darbuotojas
Prieš sąlygos	Naudotojas turi būti prisijungęs prie sistemos.
Pagrindinis scenarijus	1. Naudotojas atidaro užsakymų peržiūros puslapį. 2. Sistema rodo visų užsakymų sąrašą su pagrindine informacija. 3. Naudotojas pasirenka užsakymą detaliai peržiūrai.
Alternatyvūs scenarijai	3.1. Sistema nerodo užsakymų sąrašo (pvz., dėl serverio klaidos), pateikiamas klaidos pranešimas. 3.2. Naudotojas nusprendžia grįžti į pagrindinį puslapį, neatlikęs peržiūros.
Po sąlygos	Naudotojas mato užsakymų detales.

7. Veiklos diagrama: Užsakymų būsenos tvarkymas

Aprašymas. Ši diagrama aprašo, kaip darbuotojas atnaujina užsakymų būsenas.

Diagrama:



10 pav. Užsakymų būsenos tvarkymas

Lentelė:

7 lentelė. Užsakymų būsenos tvarkymas

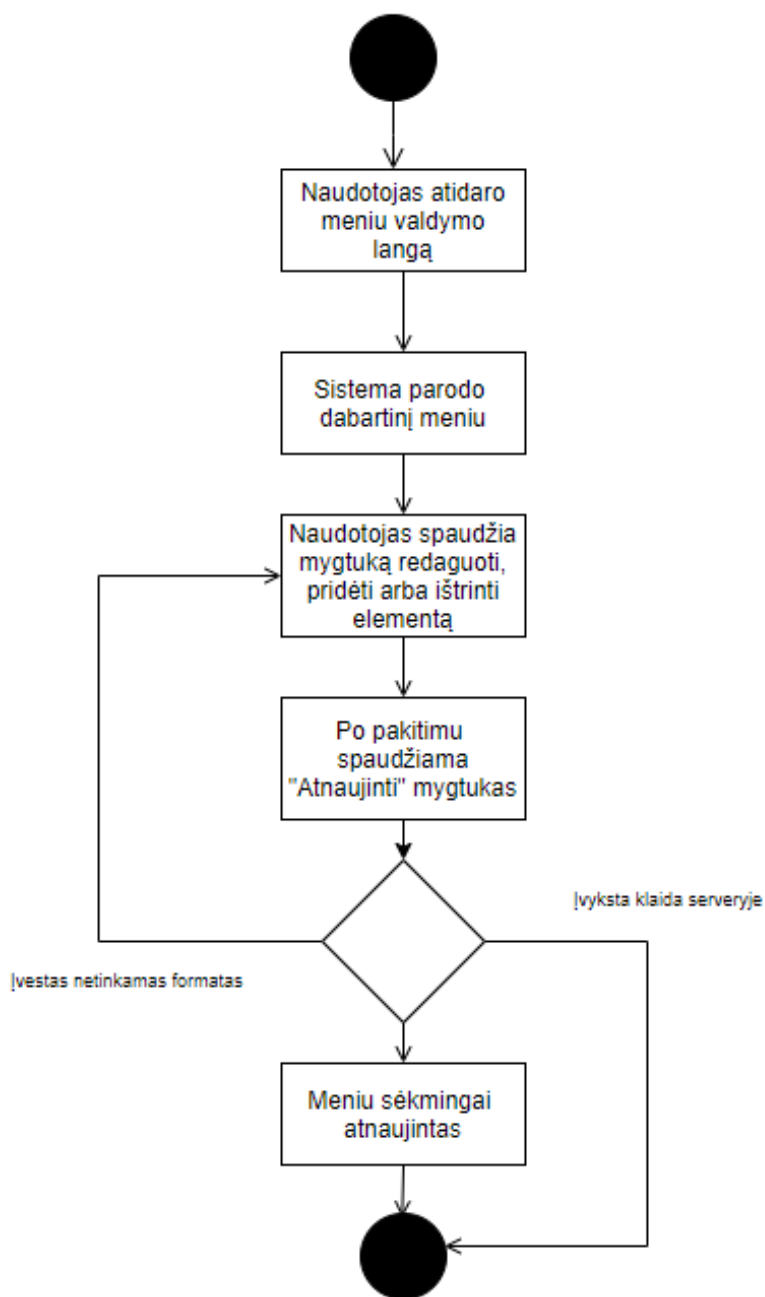
Pavadinimas	Užsakymų būsenos tvarkymas
ID	VD7
Aprašymas	Procesas, kaip darbuotojas atnaujiną užsakymų būsenas.
Aktorius	Darbuotojas
Prieš sąlygos	Darbuotojas turi būti prisijungęs prie sistemos.
Pagrindinis scenarijus	1. Darbuotojas pasirenka užsakymą iš sąrašo. 2. Sistema parodo esamą užsakymo būseną. 3. Darbuotojas atnaujiną užsakymo būseną (pvz., patvirtintas). 4. Sistema išsaugo pakeitimus ir informuoja klientą.
Alternatyvūs scenarijai	2.1. Sistema nerodo užsakymų būsenos, pateikiamas klaidos pranešimas.

	3.1. Neatsinaujina užsakymo būseną. 4.1. Sistema neišsaugo pakitimų, pateikiamas klaidos pranešimas.
Po sąlygos	Užsakymo būseną sėkmingai atnaujinta.

8. Veiklos diagrama: Meniu tvarkymas

Aprašymas. Veiklos diagrama „Meniu tvarkymas“ aprašo, kaip darbuotojas gali pridėti, redaguoti ar pašalinti meniu elementus sistemoje.

Diagrama:



11 pav. Meniu tvarkymas

Lentelė:

8 lentelė. Meniu tvarkymas

Pavadinimas	Meniu tvarkymas
ID	VD8
Aprašymas	Procesas, kaip darbuotojas valdo meniu.
Aktorius	Darbuotojas
Prieš sąlygos	Darbuotojas turi būti prisijungęs prie sistemos.
Pagrindinis scenarijus	1. Darbuotojas atidaro meniu valdymo langą. 2. Sistema rodo dabartinį meniu. 3. Naudotojas pasirenka veiksmą: pridėti naują elementą, redaguoti esamą arba pašalinti. 4. Spaudžiamas mygtukas „Atnaujinti“ 5. Sistema išsaugo pakeitimus ir atnaujina meniu.
Alternatyvūs scenarijai	3.1. Darbuotojas atšaukia veiksmą. 3.2. Sistema aptinka klaidą (pvz., įvestas netinkamas kainos formatai). Parodomas klaidos pranešimas, darbuotojas turi pakartoti veiksmą.
Po sąlygos	Meniu sėkmingai atnaujintas.

3. PROGRAMINĖS REALIZACIJOS APRAŠYMAS

Šioje dalyje pateikiamas detalesnis restorano užsakymų valdymo sistemos programinės realizacijos aprašymas. Sistema buvo sukurta siekiant užtikrinti efektyvų ir sklandų užsakymų valdymą, pradedant nuo užsakymo pateikimo, jo apdorojimo, apmokėjimo ir baigiant užsakymo užbaigimu bei archyvavimu. Šiam tikslui pasiekti buvo sukurta duomenų bazė, kuri efektyviai saugo ir valdo visą informaciją apie klientus, užsakymus, meniu, darbuotojus ir atliktus mokėjimus. Taip pat, programinė dalis apima logiką, užtikrinančią sklandų sistemos veikimą, greitą užsakymų apdorojimą ir sąveiką su vartotojais.

Sekančiuose skyriuose pateikiama detalesnė duomenų bazės struktūros apžvalga ir programinio kodo aprašymas, kuris leidžia užtikrinti sistemos funkcionalumą bei našumą, taip pat apžvelgiamas pagrindinių komponentų tarpusavio ryšys.

3.1. Duomenų bazės aprašymas

Restorano užsakymų valdymo sistemoje naudojama duomenų bazė, kurios tikslas – efektyviai ir saugiai tvarkyti informaciją apie užsakymus, klientus, meniu bei darbuotojus. Duomenų bazė turi būti sukurta taip, kad leistų greitai atlikti įvairius užklausas ir užtikrintų sistemos veikimo našumą bei saugumą.

Pagrindinės duomenų bazės lentelės:

1. Klientai:

- **Aprašymas:** Ši lentelė saugo informaciją apie sistemos vartotojus (klientus).
- **Lentelės struktūra:**
 - `id` (INT, Pirmyn-užsakymo numeris) - Unikalus kliento ID.
 - `vardas` (VARCHAR) - Kliento vardas.
 - `pavardė` (VARCHAR) - Kliento pavardė.
 - `el_pastas` (VARCHAR) - Kliento el. paštas.
 - `telefono_numeris` (VARCHAR) - Kliento telefono numeris.
 - `registracijos_data` (DATETIME) - Kliento registracijos data.

2. Užsakymai:

- **Aprašymas:** Ši lentelė saugo informaciją apie kiekvieną pateiktą užsakymą.
- **Lentelės struktūra:**
 - `id` (INT, AUTO_INCREMENT) - Unikalus užsakymo ID.

- kliento_id (INT) - Nuoroda į Klientą (FK).
- data (DATETIME) - Užsakymo pateikimo data ir laikas.
- mokėjimo_būdas (VARCHAR) - Mokėjimo būdas (kreditinė kortelė, grynaisiais).
- būseną (VARCHAR) - Užsakymo būseną ("Sukurtas", "Apmokėtas", "Atšauktas").
- kaina (DECIMAL) - Užsakymo bendra suma.

3. Meniu:

- **Aprašymas:** Ši lentelė saugo meniu informaciją apie restoraną siūlomus patiekalus.
- **Lentelės struktūra:**
 - id (INT, AUTO_INCREMENT) - Unikalus patiekalo ID.
 - pavadinimas (VARCHAR) - Patiekalo pavadinimas.
 - aprašymas (TEXT) - Patiekalo aprašymas.
 - kaina (DECIMAL) - Patiekalo kaina.
 - kategorija (VARCHAR) - Patiekalo kategorija ("Patiekalai", "Gėrimai").

4. Užsakymo_Patiekalai:

- **Aprašymas:** Ši lentelė saugo informaciją apie patiekalus, susijusius su konkrečiu užsakymu.
- **Lentelės struktūra:**
 - id (INT, AUTO_INCREMENT) - Unikalus įrašas ID.
 - uzsakymo_id (INT) - Užsakymo ID, nuoroda į užsakymą.
 - patiekalo_id (INT) - Patiekalo ID, nuoroda į meniu.
 - kiekis (INT) - Užsakytas patiekalo kiekis.

5. Darbuotojai:

- **Aprašymas:** Ši lentelė saugo darbuotojų informaciją.
- **Lentelės struktūra:**
 - id (INT, AUTO_INCREMENT) - Unikalus darbuotojo ID.
 - vardas (VARCHAR) - Darbuotojo vardas.

- `pavardė` (VARCHAR) - Darbuotojo pavardė.
- `pareigos` (VARCHAR) - Darbuotojo pareigos ("Padavėjas", "Virtuvės šefas").
- `telefonas` (VARCHAR) - Darbuotojo telefono numeris.

6. Mokėjimai:

- **Aprašymas:** Ši lentelė saugo informaciją apie atliktus mokėjimus.
- **Lentelės struktūra:**
 - `id` (INT, AUTO_INCREMENT) - Unikalus mokėjimo ID.
 - `užsakymo_id` (INT) - Užsakymo ID, nuoroda į užsakymą.
 - `suma` (DECIMAL) - Sumokėta suma.
 - `mokėjimo_data` (DATETIME) - Mokėjimo atlikimo data ir laikas.
 - `mokėjimo_būdas` (VARCHAR) - Mokėjimo būdas ("PayPal", "Grynaisiais").

Ryšiai tarp lentelių:

- **Klientai ↔ Užsakymai:** Klientas gali pateikti daug užsakymų, todėl tarp lentelių yra vienas į daugelį ryšys.
- **Užsakymai ↔ Užsakymo_Patiekalai:** Kiekvienas užsakymas gali turėti kelis patiekalus, todėl čia yra ryšys „vienas į daug“.
- **Menu ↔ Užsakymo_Patiekalai:** Kiekvienas patiekalas gali būti užsakytas daugybę kartų, todėl čia taip pat yra „vienas į daug“ ryšys.
- **Užsakymai ↔ Mokėjimai:** Kiekvienas užsakymas turi tik vieną mokėjimą, tačiau gali būti atlikti keli užsakymai su įvairiais mokėjimo būdais.

3.2. Programinis kodas

Restorano užsakymų valdymo sistemos programinis kodas apima keletą pagrindinių komponentų, kurie užtikrina veiksmingą ir saugų užsakymų valdymą. Šios sistemos dalys susijusios su vartotojo sąsaja, serverio logika ir duomenų bazės sąveika.

Pagrindiniai kodavimo aspektai:

1. Vartotojo sąsaja (UI):

- Pagrindinis vartotojo sąsajos tikslas – suteikti klientams ir darbuotojams paprastą ir intuityvią aplinką užsakymams atlikti ir apdoroti.

- Klientui pateikiamas meniu, leidžiantis pasirinkti norimus patiekalus ir atlikti užsakymą.
- Darbuotojams pateikiama užsakymų valdymo sistema, leidžianti stebėti užsakymų būsenas ir užtikrinti, kad užsakymai būtų vykdomi laiku.

2. Serverio logika:

- **Užsakymų apdorojimas:** Užsakymų pateikimo ir apdorojimo funkcijos apima užsakymo informacijos gavimą iš kliento, patikrinimą dėl mokėjimo ir duomenų įrašymą į duomenų bazę.
- **Mokėjimo sistemos integracija:** Programoje yra funkcija, kuri susijungia su išorinėmis mokėjimo sistemomis (pvz., PayPal), kad patikrintų, ar mokėjimas buvo atliktas sėkmingai.
- **Užsakymų stebėjimas:** Sistema nuolat stebi užsakymų būsenas ir praneša darbuotojams, kai užsakymas yra paruoštas pristatymui.

3. Duomenų bazės sąveika:

- Programos backend'as naudoja SQL užklausas, kad gautų, atnaujintų, įrašytų ir ištrintų duomenis iš duomenų bazės.
- Pavyzdžiui, kai klientas pateikia užsakymą, sistema įrašo informaciją apie užsakymą ir pasirinktus patiekalus į duomenų bazę.

Programinio kodo fragmentai klientų programėlės:

1. *App.js*:

Šiame faile yra aprašyta pagrindinė aplikacijos navigacijos sistema, naudojant **React Navigation** biblioteką. Failas taip pat inicializuoja Firebase paslaugą, kad būtų galima naudoti Firestore duomenų bazę.

Pagrindinės funkcijos ir komponentai:

- **Firestore Inicializacija:**
 - **initializeApp(firebaseConfig):**
 - Ši funkcija inicijuoja Firebase paslaugą su užkoduotais konfigūracijos duomenimis (firebaseConfig). Tai leidžia naudoti Firebase funkcijas, tokias kaip autentifikacija ir duomenų bazė.
 - **getFirestore(app):**

- Funkcija, kuri užtikrina prisijungimą prie Firestore duomenų bazės, leidžiant programai skaityti ir rašyti duomenis.
- **Navigacijos Sistema:**
 - **NavigationContainer:**
 - Apima visą navigaciją ir užtikrina, kad vartotojas galės pereiti per įvairius programos langus.
 - **createStackNavigator:**
 - Funkcija, kuri sukuria "stack" tipo navigacijos sistemą, leidžiančią vartotojui pereiti per skirtingus ekranus (screen'us), rodomus vienas po kito.
- **Ekranų Aprašymas:**
 - **HomeScreen:**
 - Pagrindinis ekranas, kur vartotojas pradeda savo interakciją su programėle.
 - **WebViewScreen:**
 - Ekranas, kuriame vartotojas gali peržiūrėti interneto svetainę, naudodamasis WebView komponentu.
 - **CartScreen:**
 - Ekranas, kuriame vartotojas gali peržiūrėti ir valdyti savo prekių krepšelį.
 - **PaymentScreen:**
 - Ekranas, skirtas apmokėjimo procesui. Čia vartotojas gali įvesti savo apmokėjimo informaciją ir atlikti apmokėjimą.
 - **ConfirmationScreen:**
 - Ekranas, kuriame vartotojas gauna patvirtinimą apie atliktą užsakymą ir apmokėjimą.

2. *MenuItem.js*:

Šis komponentas atsakingas už kiekvieno meniu elemento, įskaitant jo informaciją, rodyklę, kiekio keitimą ir pridėjimą į krepšelį, atvaizdavimą.

Funkcijos:

- **handlePress():**
 - Ši funkcija valdo meniu elemento plėtimo arba suskleidimo būseną.

- Kodo iškarpa:

```
const handlePress = () => {
    setIsExpanded(!isExpanded);
};
```

- **handleAddToCart():**

- Atsakinga už pridėjimą į krepšelį. Kai naudotojas nuspaudžia mygtuką "Pridėti į krepšelį", parodomas patvirtinimo langas, kuris klausia, ar tikrai nori pridėti prekę su nurodytu kiekiu.

- Kodo iškarpa:

```
const handleAddToCart = () => {
    Alert.alert(
        'Ar tikrai pridėti į krepšelį',
        `${item.PAVADINIMAS} (Kiekis: ${quantity}) keliauja į krepšelį.`,
        [
            {
                text: 'Pridėti',
                onPress: () => {
                    onAddToCart(item, quantity);
                },
            },
            {
                text: 'Atšaukti',
                style: 'cancel',
            },
        ],
        { cancelable: false }
    );
};
```

- **increaseQuantity() ir decreaseQuantity():**

- Šios funkcijos leidžia didinti arba mažinti prekių kiekį, kurį naudotojas nori įdėti į krepšelį.

- Kodo iškarpa:

```
const increaseQuantity = () => {
    setQuantity(quantity + 1);
};
```

```
};
const decreaseQuantity = () => {
  if (quantity > 1) {
    setQuantity(quantity - 1);
  }
};
```

3. *QRCodeGenerator.js*:

Šis komponentas generuoja QR kodą, kuris leidžia vartotojui nuskenuoti ir pasiekti meniu.

Funkcijos:

- **QRCode:**
 - Naudojama **react-native-qrcode-svg** biblioteka, kad sugeneruotų QR kodą su nurodytu URL, kuris yra perduodamas kaip **props**.

4. *QRCodeScannerComponent.js*:

Šis komponentas leidžia naudotojui nuskaityti QR kodus naudojant kamerą.

Funkcijos:

- **handleQRCodeScan():**
 - Kai QR kodas yra nuskaitytas, ši funkcija gauna nuskaitytą URL ir perduoda jį per **onScanSuccess** callback'ą. Jei nuskaityti duomenys nėra tinkami, bus išvedama klaida.
 - Kodo iškarpa:

```
const QRCodeScannerComponent = ({ onScanSuccess }) => {
  const handleQRCodeScan = (e) => {
    if (e.data) {
      const scannedUrl = e.data;
      console.log("Nuskaitytas URL: ", scannedUrl);
      onScanSuccess(scannedUrl);
    } else {
      console.log("Nepavyko nuskaityti QR kodo");
    }
  };
  return (
    <QRCodeScanner
      onRead={handleQRCodeScan}
      reactivate={true}
```

```

        reactivateTimeout={2000}
        cameraStyle={styles.cameraStyle}
      />
    );
  };
};

```

5. *HomeScreen.js*:

Šis komponentas yra pagrindinis vartotojo sąsajos ekranas, kuris rodo QR kodo generatorių su fiksuotu URL, taip pat suteikia vartotojui galimybę pasirinkti vieną iš dviejų veiksmų. Vartotojas gali nuspręsti atidaryti kamerą, kad nuskaitytų QR kodą, arba atidaryti WebView ekraną, kuris rodo interneto puslapį. Komponentas yra skirtas pagrindinei navigacijai tarp pagrindinių funkcionalumų – QR kodo generavimo ir nuskaitymo, bei naršymo naudojant WebView.

Funkcijos:

- Pateikia QR kodo generatorių, kuriame yra fiksuotas URL (<http://localhost:19006>), skirtas pateikti meniu arba svetainės informaciją.
- Turi navigacijos logiką, kuri leidžia vartotojui pereiti į kitus ekranus (QRCodeScreen ir WebView).
- QRCodeGenerator(url): Pateikia QR kodą su nurodytu URL.
- navigation.navigate('QRCodeScreen'): Naviguoja į QR kodo nuskaitymo ekraną.
- navigation.navigate('WebView', { url }): Naviguoja į WebView ekraną ir pateikia URL, kad jį būtų galima peržiūrėti.
- Kodo iškarpa:

```

const HomeScreen = ({ navigation }) => {
  const webUrl = 'http://localhost:19006';

  return (
    <><ImageBackground
      source={require('../assets/images/logo1.png')}
      style={styles.image}></ImageBackground><View
        style={styles.container}>
      <QRCodeGenerator url={webUrl} />
      <Button
        title="Open Camera to Scan QR Code"
        onPress={() =>
          navigation.navigate('QRCodeScreen')}
      />
    </>
  );
};

```

```

        <Button
            title="Open WebView"
            onPress={() => navigation.navigate('WebView', {
url: webUrl })}
        />
    </View></>
);
};

```

6. *WebViewScreen.js*:

WebViewScreen komponentas rodo prekių meniu, įskaitant įvairias kategorijas: patiekalus, gėrimus ir užkandžius. Komponentas integruoja Firestore duomenų bazę, kad dinamiškai užkrautų meniu elementus ir leistų vartotojui filtruoti juos pagal kategorijas. Be to, jis leidžia vartotojui pridėti prekes į krepšelį ir pereiti prie kitos pirkimo proceso dalies.

Funkcijos:

- Gali filtruoti meniu pagal kategorijas (Patiekalai, Gėrimai, Užkandžiai) ir rodyti tik atitinkamus elementus.
- Leidžia vartotojui pridėti prekes į krepšelį, taip pat patikrinti, ar prekė jau yra krepšelyje.
- Dinamiškai užkrauna duomenis iš Firebase Firestore duomenų bazės ir pateikia juos sąrašė.
- `fetchMenuItems()`: Užkrauna prekes iš Firestore duomenų bazės.
- `filterItems(category)`: Atlieka filtravimą pagal kategorijas ir atnaujiną matomus meniu elementus.
- `handleAddToCart(item)`: Prideda prekę į krepšelį, jeigu ji dar nėra pridėta, arba atnaujiną jos kiekį, jei jau yra.
- `handleContinueToPurchase()`: Naviguoja į krepšelio ekraną, leidžiantį atlikti pirkimo veiksmus.
- Kodo iškarpa:

```

useEffect(() => {
    const fetchMenuItems = async () => {
        try {
            const menuSnapshot = await getDocs(collection(db, 'Menu'));

```



```

        const items = menuSnapshot.docs.map((doc) => ({ id: doc.id,
...doc.data() }));
        setMenuItems(items);
        setFilteredMenuItems(items);
    } catch (error) {
        console.error('Klaida gaunant meniu dalis:', error);
    } finally {
        setLoading(false);
    }
};

fetchMenuItems();
}, [db]);

const filterItems = (category) => {
    setSelectedCategory(category);
    if (category === 'All') {
        setFilteredMenuItems(menuItems);
    } else {
        const translatedCategory = getCategoryTranslation(category);
        const filteredItems = menuItems.filter(item => item.KATEGORIJA
=== translatedCategory);
        setFilteredMenuItems(filteredItems);
    }
};

const getCategoryTranslation = (appCategory) => {
    switch (appCategory) {
        case 'Food':
            return 'PATIEKALAI';
        case 'Drinks':
            return 'GĖRIMAI';
        case 'Snacks':
            return 'UŽKANDŽIAI';
        default:
            return appCategory;
    }
};

const renderItem = ({ item }) => (
    <MenuItem
        item={item}

```

```

        onSelect={() => handleSelectItem(item)}
        onAddToCart={() => handleAddToCart(item)}
    />
);
const handleSelectItem = (item) => {
};
const handleAddToCart = (item) => {
    const existingItem = basketItems.find(basketItem =>
basketItem.id === item.id);

    if (existingItem) {
        Alert.alert(
            'Prekė jau yra',
            `${item.PAVADINIMAS} jau yra krepšelyje. Ar tikrai norite
pridėti?`,
            [
                {
                    text: 'Pridėti ',
                    onPress: () => updateBasketItemQuantity(item,
existingItem.quantity + 1),
                },
                {
                    text: 'Atšaukti',
                    style: 'cancel',
                },
            ],
            { cancelable: false }
        );
    } else {
        setBasketItems([...basketItems, { ...item, quantity: 1 }]);
    }
};
const updateBasketItemQuantity = (item, quantity) => {
    setBasketItems((prevItems) =>
        prevItems.map((basketItem) =>
            basketItem.id === item.id
                ? { ...basketItem, quantity }
                : basketItem
        )
    );
};

```

```

    )
  );
};
const handleContinueToPurchase = () => {
  navigation.navigate('CartScreen', { basketItems });
};
7. CartScreen.js:

```

Tai ekranas, kuriame rodomas pirkinių krepšelis su visomis pasirinktomis prekėmis. Vartotojas gali valdyti prekių kiekį (pridėti arba pašalinti prekes) ir pereiti į mokėjimo ekraną.

Funkcijos:

- Prekių grupavimas pagal ID, jų kiekio ir kainos apskaičiavimas.
- Vartotojas gali pritaikyti kiekį ir pašalinti prekes iš krepšelio.
- Pasirinkus "Užsakyti", prekių informacija perduodama į „PaymentScreen“ komponentą.
- Kodo iškarpa:

```

const updatedGroupedItems = Array.from(groupedItemsMap.values());
setGroupedItems(updatedGroupedItems);

const updatedTotalAmount = updatedGroupedItems.reduce((total,
item) => total + item.visoKaina, 0);
setTotalAmount(updatedTotalAmount);
}, [basketItems]);

const handleIncrement = (itemId) => {
  setGroupedItems((prevItems) =>
    prevItems.map((item) =>
      item.id === itemId
        ? { ...item, kiekis: item.kiekis + 1, visoKaina:
item.visoKaina + item.KAINA }
        : item
    )
  );
};

updateItemInFirestore(itemId, 1);
};

const handleDecrement = (itemId) => {
  setGroupedItems((prevItems) =>
    prevItems.map((item) =>
      item.id === itemId && item.kiekis > 1

```

```

        ? { ...item, kiekis: item.kiekis - 1, visoKaina:
item.visoKaina - item.KAINA }
        : item
    )
  );
  updateItemInFirestore(itemId, -1);
};

const updateItemInFirestore = async (itemId, quantityChange) =>
{
  const db = getFirestore();
  try {
    const cartRef = collection(db, 'Cart');
    const q = query(cartRef, where('id', '==', itemId));
    const querySnapshot = await getDocs(q);
    querySnapshot.docs.forEach(async (doc) => {
      const itemRef = doc.ref;
      await updateDoc(itemRef, {
        kiekis: doc.data().kiekis + quantityChange,
        visoKaina: doc.data().visoKaina + (quantityChange *
doc.data().KAINA),
      });
    });
  } catch (error) {
    console.error('Klaida atnaujinant prekes Firestore:',
error);
  }
};

const handleRemove = (itemId) => {
  setGroupedItems((prevItems) => {
    const updatedItems = prevItems.filter((item) => item.id !==
itemId);
    const updatedTotalAmount = updatedItems.reduce((total, item)
=> total + item.visoKaina, 0);
    setTotalAmount(updatedTotalAmount);
    return updatedItems;
  });
  removeFromFirestore(itemId);
};

const removeFromFirestore = async (itemId) => {

```

```

try {
  const db = getFirestore();
  const cartRef = collection(db, 'Cart');
  const q = query(cartRef, where('id', '==', itemId));
  const querySnapshot = await getDocs(q);
  querySnapshot.docs.forEach(async (doc) => {
    await deleteDoc(doc.ref);
  });
} catch (error) {
  console.error('Klaida trinant prekę iš Firestore:', error);
}
};

const handleOrder = async () => {
  if (groupedItems.length === 0) {
    Alert.alert('Tuščias krepšelis', 'Jūsų pirkinių krepšelis yra tuščias. Pridėkite prekių, kad testumėte.', [
      { text: 'Gerai', onPress: () => console.log('Gerai paspausta') },
    ]);
  } else {
    try {
      const db = getFirestore();
      const batch = [];
      groupedItems.forEach((item) => {
        for (let i = 0; i < item.kiekis; i++) {
          const newItemRef = collection(db, 'Cart');
          const newItemDoc = {
            PAVADINIMAS: item.PAVADINIMAS,
            KAINA: item.KAINA,
            kiekis: 1,
            visoKaina: item.visoKaina,
            //category: item.category,
          };
          batch.push(addDoc(newItemRef, newItemDoc));
        }
      });
      await Promise.all(batch);
    }
  }
};

```

```

        navigation.navigate('PaymentScreen', { groupedItems,
totalAmount });
        setGroupedItems([]);
        setTotalAmount(0);
    } catch (error) {
        console.error('Klaida pridedant prekes į Firestore:',
error);
    }
}
};

```

8. *PaymentScreen.js*:

Tai ekranas, kuriame vartotojas peržiūri savo užsakymą ir užbaigia mokėjimą. Jis leidžia įvesti asmens duomenis ir pasirinkti apmokėjimo metodą.

Funkcijos:

- Vartotojas gali įvesti vardą, pavardę, telefono numerį ir el. pašto adresą.
- Galima pasirinkti apmokėjimo būdą (pvz., PayPal arba grynaisiais).
- Jeigu visi duomenys įvesti teisingai, užsakymas išsaugomas „Firestore“ duomenų bazėje.
- Po užsakymo pateikimo vartotojas nukreipiamas į patvirtinimo ekraną.
- Kodo iškarpa:

```

const handlePaymentMethodChange = (method) => {
    setSelectedPaymentMethod(method);
};

const handleSubmitOrder = async () => {
    if (!Vardas || !Pavardė || !TelefonoNumeris || !ElPaštas) {
        Alert.alert('Neužpildyta Informacija', 'Prašome užpildyti
visus laukelius.');
```

return;

```

    }

    if (!/^\\d{9}$/.test(TelefonoNumeris)) {
        Alert.alert('Klaida', 'Telefono numeris turi būti 9
skaitmenys.');
```

return;

```

    }

    if (!/\\S+@\\S+\\.\\S+/.test(ElPaštas)) {
        Alert.alert('Klaida', 'Prašome įvesti galiojantį el. paštą.');
```

return;

```

    }
    if (!selectedPaymentMethod) {
        Alert.alert('Pasirinkite mokėjimo būdą', 'Prašome pasirinkti
apmokėjimo metodą.');
```

return;

```

    }
    try {
        const orderNumber = Math.floor(Math.random() * 1000);
        const paymentSuccess = selectedPaymentMethod === 'PayPal' ?
Math.random() < 0.8 : true;
        if (selectedPaymentMethod === 'PayPal' && !paymentSuccess) {
            Alert.alert('Mokėjimo Klaida', 'PayPal apmokėjimas
nepavyko. Bandykite dar kartą.');
```

return;

```

        }
        const db = getFirestore();
        const orderRef = collection(db, 'Apmokėjimas');
        const orderItem = Array.isArray(groupedItems)
            ? groupedItems.map((item) => ({
                PAVADINIMAS: item.PAVADINIMAS,
                KAINA: item.KAINA,
                kiekis: item.kiekis,
                visoKaina: item.visoKaina,
            }))
            : [];
        const orderDoc = {
            vardas: Vardas,
            pavardė: Pavardė,
            telefonoNumeris: TelefonoNumeris,
            elPaštas: ElPaštas,
            Apmokėta: selectedPaymentMethod,
            bendraSuma: totalAmount.toFixed(2) + ' €',
            prekės: orderItem,
            timestamp: serverTimestamp(),
            orderNumber: orderNumber,
        };
        await addDoc(orderRef, orderDoc);
        navigation.navigate('ConfirmationScreen', {

```

```

        orderNumber: orderNumber,
        items: orderItem,
        totalAmount: totalAmount
    });
} catch (error) {
    console.error('Klaida pateikiant užsakymą į Firestore:',
error);
    Alert.alert('Klaida', 'Nepavyko pateikti užsakymo. Bandykite
dar kartą.');
```

9. ConfirmationScreen.js:

Tai paskutinis ekranas, kuriame vartotojas mato užsakymo patvirtinimą, įskaitant užsakymo numerį ir užsakytas prekes.

Funkcionalumas:

- Rodo užsakymo numerį ir prekes.
- Leidžia grįžti į pagrindinį puslapį.
- Kodo iškarpa:

```

const ConfirmationScreen = ({ route, navigation }) => {
    const { orderNumber, items, totalAmount } = route.params;
    return (
        <ScrollView style={styles.container}>
            <Text style={styles.title}>Užsakymas Pateiktas</Text>
            <Text style={styles.orderNumber}>Užsakymo numeris:
{orderNumber}</Text>
            <Text style={styles.sectionTitle}>Įsigytos prekės:</Text>
            {formattedItems.length > 0 ? (
                formattedItems.map((item, index) => (
                    <View key={index} style={styles.itemContainer}>
                        <Text style={styles.itemName}>{item.PAVADINIMAS}
x{item.kiekis}</Text>
                    </View>
                ))
            ) : (
                <Text style={styles.noItems}>Nėra prekių sąrašė.</Text>
            )}
        </ScrollView>
    );
}
```



```

    </ScrollView>
  );
};

```

Programinio kodo fragmentai darbuotojų programėlės:

Kadangi darbuotojų programėlė yra dvigubai didesnė nei klientų aprašynėsiu tik pačias svarbiausias kodo dalis.

1. App.js:

Tai pagrindinis failas, kuriame nustatyta „React Navigation“ struktūra ir „Redux“ teikėjas. Jis leidžia naudoti navigaciją ir globalų valstybės valdymą.

- Provider store={store} - tai „Redux“ teikėjas, kuris suteikia prieigą prie globalios būsenos visiems komponentams.
- NavigationContainer - pagrindinis konteineris, reikalingas naudoti navigacijai React Native programoje.
- BottomTabNavigator - navigacijos komponentas, kuriame apibrėžtos skirtos ekranų navigacijos parinktys (pvz., ekrano perėjimas tarp registracijos ir prisijungimo ekranų).

2. WelcomeScreen.js:

Tai ekranas, kuriame vartotojas pasitinka programą ir gali pasirinkti, ar registruotis, ar prisijungti, jei jau turi paskyrą.

Funkcijos:

- navigation.navigate('Register') - navigacija į registracijos ekraną.
- navigation.navigate('Login') - navigacija į prisijungimo ekraną.
- ImageBackground - naudoja fono nuotrauką.
- SafeAreaView - apsaugo nuo kraštų, užtikrinant, kad turinys nebus užgožtas telefono kraštų.

3. RegisterScreen.js:

Tai registracijos ekranas, kuriame vartotojas įveda savo asmens duomenis, tokius kaip vardas, pavardė, telefono numeris, el. paštas ir slaptažodis, kad sukurti paskyrą.

- **signUp** - tai funkcija, kuri siunčia duomenis į Firebase ir sukuria vartotoją.
- **useState** - naudojama komponento būsenos valdymui, kad galėtumėte saugoti ir atnaujinti įvestus duomenis.

- handleSignUp - šioje funkcijoje atliekamas tikrinimas, ar visi laukai užpildyti, ir jei taip, vartotojas registruojamas.
- navigation.navigate('Pagrindinis') - sėkmingai užregistravus vartotoją, jis nukreipiamas į pagrindinį ekraną.
- Kodo iškarpa:

```
const handleSignUp = async () => {
  try {
    if (!email || !password || !firstName || !lastName ||
    !phoneNumber || !position) {
      console.error("All fields are required");
      return;
    }
    const user = await signUp(email, password, firstName,
    lastName, phoneNumber, position);
    console.log("User created:", user);
    navigation.navigate('Pagrindinis');
  } catch (error) {
    console.error("Registration error:", error.message);
  }
};

return (
  <ScrollView>
    <ImageBackground
    source={require('../assets/images/back.png')}
    style={styles.image}>
      <TouchableOpacity
        onPress={() => navigation.goBack()}
        style={styles.backButton}>
        <Icon name="keyboard-arrow-left" size={35} color="black" />
      </TouchableOpacity>
    <SafeAreaView>
      <Text style={styles.header}>Registracija</Text>
      <Text style={styles.label}>Vardas</Text>
      <TextInput
        style={styles.input}
        value={firstName}
        onChangeText={(text) => setFirstName(text)}
      </TextInput>
    </SafeAreaView>
  </ScrollView>
);
```

```

        placeholder='Įveskite vardą'
    />
<Text style={styles.label}>Pavardė</Text>
<TextInput
    style={styles.input}
    value={lastName}
    onChangeText={(text) => setLastName(text)}
    placeholder='Įveskite pavardę'
/>
<Text style={styles.label}>Pozicija</Text>
<TextInput
    style={styles.input}
    value={position}
    onChangeText={(text) => setPosition(text)}
    placeholder='Įveskite poziciją'
/>
<Text style={styles.label}>Telefono numeris</Text>
<TextInput
    style={styles.input}
    value={phoneNumber}
    onChangeText={(text) => setPhoneNumber(text)}
    placeholder='Įveskite telefono numerį'
/>
<Text style={styles.label}>El. pašto adresas</Text>
<TextInput
    style={styles.input}
    value={email}
    onChangeText={(text) => setEmail(text)}
    placeholder='Įveskite el. paštą'
/>
<Text style={styles.label}>Slaptažodis</Text>
<TextInput
    style={styles.input}
    value={password}
    onChangeText={(text) => setPassword(text)}
    secureTextEntry
    placeholder='Įveskite slaptažodį'

```

```

        />
        <TouchableOpacity
            onPress={handleSignUp}
            style={styles.signupButton}
        >
            <Text style={styles.buttonText}>Registruotis</Text>
        </TouchableOpacity>
        <Text style={styles.orText}>Arba</Text>
        <View style={styles.loginContainer}>
            <Text style={styles.loginText}>Jau turite paskyrą?</Text>
            <TouchableOpacity
                onPress={() =>
                    navigation.navigate('Login')}
            >
                <Text style={styles.loginLink}>Prisijungti</Text>
            </TouchableOpacity>
        </View>
    </SafeAreaView>
</ImageBackground>
</ScrollView>
);
}

```

4. *LoginScreen.js*:

Prisijungimo ekranas, kuriame vartotojas gali įvesti savo el. paštą ir slaptažodį, kad prisijungtų prie paskyros.

- **signInWithEmailAndPassword** - Firebase funkcija, kuri leidžia vartotojui prisijungti su el. pašto ir slaptažodžio kombinacija.
- **getAuth()** - ši funkcija naudojama gauti autentifikacijos objekto pavyzdį, kad būtų galima prisijungti prie Firebase.
- **handleLogin** - funkcija, kuri apdoroja prisijungimo procesą ir naudoja „Firebase“ funkciją „signInWithEmailAndPassword“.
- **navigation.navigate('Home')** - vartotojas nukreipiamas į pagrindinį puslapį sėkmingai prisijungus.
- Kodo iškarpa:

```

const auth = getAuth();

const handleLogin = async () => {

```

```

    try {
        const userCredential = await signInWithEmailAndPassword(auth,
email, password);

        console.log('Prisijungimo sėkmingas!', userCredential);
    } catch (error) {
        console.error('Prisijungimo klaida:', error.message);
    }
};

return (
    <ImageBackground
source={require('../assets/images/back.png')}
style={styles.image}>
    <TouchableOpacity
        onPress={() => navigation.goBack()}
        style={styles.backButton}>
        <Icon name="keyboard-arrow-left" size={35} color="black" />
    </TouchableOpacity>
    <SafeAreaView>
        <Text style={styles.header}>Prisijungimas</Text>
        <Text style={styles.label}>El. Paštas</Text>
        <TextInput
            style={styles.input}
            placeholder="Enter Email Address"
            value={email}
            onChangeText={(text) => setEmail(text)}
        />
        <Text style={styles.label}>Slaptažodis</Text>
        <TextInput

```

```

        style={styles.input}

        secureTextEntry

        placeholder="Enter Password"

        value={password}

        onChangeText={ (text) => setPassword(text) }

    />

    <TouchableOpacity                                onPress={handleLogin}
style={styles.loginButton}>

        <Text style={styles.buttonText}>Prisijungti</Text>

    </TouchableOpacity>

    <Text style={styles.orText}>Arba</Text>

    <View style={styles.signupContainer}>

        <Text style={styles.signupText}>Neturite paskyros?</Text>

        <TouchableOpacity                                onPress={ ()                                =>
navigation.navigate('Register')}>

            <Text style={styles.signupLink}> Registruotis</Text>

        </TouchableOpacity>

    </View>

</SafeAreaView>

</ImageBackground>

);
}

```

5. *MenuDetails.js*:

Tai komponentas, kuris rodo pasirinkto meniu elemento detales ir leidžia vartotojui jį redaguoti arba ištrinti.

- **State:**

- `isExpanded`: Tai būseną, kuri nustato, ar detalės yra išplėtos ar ne. Tai valdo detalių matomumą.

- o `fadeOut: Animated.Value(0)` naudojama animacijai, kad būtų sklandžiai rodomas ir paslėptas turinys.
- **useEffect:**
 - o Pakeičia animacijos vertę, kai keičiasi `isExpanded`. Jei išplečiama, animacija keičiasi į 1 (visiškai matoma), jei ne - į 0 (paslėpta).
- **toggleDetails funkcija:**
 - o Keičia `isExpanded` reikšmę, kad atvertų arba uždarytų detales.

6. *MenuList.js*:

Tai komponentas, kuris rodo meniu elementus. Kiekvienas meniu elementas pateikiamas `MenuDetails` komponente su galimybėmis redaguoti ir ištrinti.

- **onEdit ir onDelete funkcijos:**
 - o Naudojama `navigation` funkcija, kad būtų nukreipta į kitus ekranus (Redaguoti, Ištrinti).
- **Tikrinimas, ar yra galiojantis vartotojas:**
 - o Komponentas tikrina, ar `currentUser` turi `uid` reikšmę, prieš atliekant bet koki veiksmą.
- Kodo iškarpa:

```
const MenuList = ({ menuItem, navigation, currentUser }) => {
  const currentUserId = currentUser?.uid;
  console.log('MenuList item:', menuItem);
  if (!currentUserId) {
    console.error('Invalid current user or missing uid:',
currentUser);
    return null;
  }
  console.log('Current User:', currentUser);
  return (
    <View style={styles.itemContainer}>
      <MenuDetails
```

```

menuItem={menuItem}

onEdit={() => navigation.navigate('Redaguoti', { menuItem
}))}

onDelete={() => navigation.navigate('Ištrinti', { menuItem
}))}

/>

</View>

);

};

```

7. OrderDetails.js:

Šis komponentas rodo užsakymo informaciją, kuri gali būti išplėsta paspaudus, ir leidžia vartotojui patvirtinti arba atšaukti užsakymą.

- **State:**
 - `isExpanded`: Nustato, ar detalės yra išplėtos.
 - `fadeAnim`: Naudojama animacija, kad būtų sukurta sklandi perėjimo animacija.
 - `containerHeight`: Nustato aukštį, priklausomai nuo to, ar detalės išplėtos, ar ne.
- **toggleDetails:**
 - Pakeičia `isExpanded` reikšmę, kad būtų rodomos prekės ir kitos užsakymo detalės.
- **useEffect:**
 - Animacija naudojama norint rodyti arba paslėpti detales. Taip pat atnaujinamas konteinerio aukštis, kad jis tiktų detalėms.
- **prekės:**
 - Tai prekės, esančios užsakyme. Jos rodomos sąraše su pavadinimu, kiekiu, kaina ir kategorija.
- **onConfirm ir onCancel:**
 - Patvirtinimo ir atšaukimo funkcijos, kurios siunčia užsakymą į kitus ekranus (Patvirtinti, Atšaukti).
- **Kodo iškarpa:**

```
const toggleDetails = () => {
```



```

        setIsExpanded(!isExpanded);
    };

    useEffect(() => {
        Animated.timing(fadeAnim, {
            toValue: isExpanded ? 1 : 0,
            duration: 500,
            useNativeDriver: false,
        }).start();
    }, [fadeAnim, isExpanded]);

    useEffect(() => {
        setContainerHeight(isExpanded ? 'auto' : 60);
    }, [isExpanded]);

    return (
        <TouchableOpacity style={styles.detailsContainer}
onPress={toggleDetails}>
            <Text style={styles.heading} numberOfLines={1}
ellipsizeMode="tail">
                {order.vardas}
            </Text>
            <Animated.View style={[styles.detailsExpanded, { opacity:
fadeAnim, height: containerHeight }]}>
                {isExpanded && (
                    <View>
                        <View style={styles.actionsContainerT}>
                            <Text style={styles.subheading}>Prekès:</Text>
                            <ScrollView style={styles.scrollView}>
                                {prekès && prekès.length > 0 && prekès.map((prekè,
index) => (

```

```

        <View key={index} style={styles.prekėContainer}>
            <View style={styles.prekėInfoContainer}>
                <Text style={styles.prekėText}>Pavadinimas:
{prekė.PAVADINIMAS}</Text>
                <Text style={styles.prekėText}>Kiekis:
{prekė.kiekis}</Text>
                <Text style={styles.prekėText}>Kaina:
{prekė.KAINA}</Text>
                <Text style={styles.prekėText}>Viso Kaina:
{prekė.visoKaina}</Text>
                <Text style={styles.prekėText}>Kategorija:
{prekė.category}</Text>
            </View>
        </View>
    ))}
</ScrollView>
</View>
<View style={styles.actionsContainer}>
    <TouchableOpacity style={styles.actionButton}
onPress={() => onConfirm(order)}>
        <Text style={styles.actionText}>Patvirtinti</Text>
    </TouchableOpacity>
    <TouchableOpacity style={styles.actionButtonDelete}
onPress={() => onCancel(order.id)}>
        <Text
style={styles.actionTextDelete}>Atšaukti</Text>
    </TouchableOpacity>
</View>
</View>

```

```

    ) }

    </Animated.View>

    </TouchableOpacity>

  );
};

```

8. *OrderList.js*:

Tai komponentas, kuris rodo užsakymų sąrašą. Kiekvienas užsakymas pateikiamas naudojant `OrderDetails` komponentą su galimybe patvirtinti arba atšaukti užsakymą.

- **currentUserId:**
 - Tikrinama, ar vartotojas yra prisijungęs ir turi teisingą uid.
- **order:**
 - Užsakymas rodomas su informacija apie užsakymo numerį, vardą, pavardę, telefono numerį ir el. pašta. Taip pat rodomos prekės, susijusios su užsakymu.
- **Patvirtinimo ir atšaukimo mygtukai:**
 - Mygtukai, kurie veda į ekranus, kur galima patvirtinti arba atšaukti užsakymą.
- **Kodo iškarpa:**

```

const OrderList = ({ order, navigation, currentUser }) => {
  const currentUserId = currentUser?.uid;

  if (!currentUserId) {
    console.error('Invalid current user or missing uid:',
currentUser);

    return null;
  }

  return (
    <View style={styles.itemContainer}>

      <Text style={styles.orderNumber}>Užsakymo numeris:
{order.orderNumber}</Text>

```

```

        <Text                                style={styles.orderDetails}>Vardas:
{order.vardas}</Text>

        <Text                                style={styles.orderDetails}>Pavardė:
{order.pavardė}</Text>

        <Text                                style={styles.orderDetails}>Telefono        numeris:
{order.telefonoNumeris}</Text>

        <Text                                style={styles.orderDetails}>E-paštas:
{order.elPaštas}</Text>

        {order.prekės && order.prekės.map((prekė, index) => (
            <View key={index} style={styles.itemDetails}>
                <Text style={styles.itemText}>{prekė.PAVADINIMAS}      x
{prekė.kiekis} - {prekė.visoKaina} €</Text>
            </View>
        ))}

        <View style={styles.buttonContainer}>
            <TouchableOpacity
                style={styles.actionButton}
                onPress={() => navigation.navigate('Patvirtinti', { order
}}}
            >
                <Text style={styles.buttonText}>Patvirtinti</Text>
            </TouchableOpacity>
            <TouchableOpacity
                style={styles.actionButton}
                onPress={() => navigation.navigate('Atšaukti', { order })}
            >
                <Text style={styles.buttonText}>Atšaukti</Text>
            </TouchableOpacity>
        </View>

```

```
    </View>  
  );  
};
```

Paskutinės klasės yra ekranai, visose ekranuose panašios funkcijos kaip buvo kliento dalyje.

4. DIEGIMO IR NAUDOJIMO INSTRUKCIJA

Šioje dalyje pateikiami išsamūs instrukcijų, kaip įdiegti, naudoti ir šalinti programinę sistemą, aprašymai. Tai padės vartotojams ir sistemos administratoriui sėkmingai įdiegti ir išnaudoti visą sistemos funkcionalumą, užtikrinant optimalų veikimą bei lengvą naudojimą. Pateikiama informacija apima reikalavimus, įdiegtinas programas, technines charakteristikas, paleidimo ir naudojimo instrukcijas, bei galimus sistemos trikdžius ir jų sprendimo žingsnius.

4.1. Programinės realizacijos priklausomybė nuo kitų programinių produktų

Norint, kad restorano užsakymų valdymo sistema veiktų sklandžiai, reikalingos tam tikros programinės įrangos priklausomybės:

- **Firestore:** Naudojama kaip duomenų bazė ir autentifikacijos sistema. Ji užtikrina greitą ir saugų duomenų mainą bei vartotojų autentifikavimą realiuoju laiku.
- **GitHub:** Versijų kontrolės sistema, kuri leidžia sekti projekto vystymą ir bendradarbiauti su komanda, užtikrinant darbo nuoseklumą.
- **IntelliJ IDEA Ultimate:** Programavimo aplinka, kurioje sukurtos pagrindinės funkcijos. Tai leidžia naudoti integruotą klaidų diagnostiką, bibliotekų integraciją ir patogų kodo rašymą.
- **Expo Go:** Mobiliosios aplikacijos kūrimo ir testavimo platforma, skirta tiek „Android“, tiek „iOS“ įrenginiams.
- **Emulatorius(Android Studio Device Manager)** naudotas programėlės testuotavimui, stebėjimui duomenų mainų realiu laiku bei funkcijų veikimo tikrinimas.

Visos šios technologijos yra būtinos sėkmingam sistemos diegimui ir eksploatavimui. Be jų, sistema negalės veikt efektyviai ir saugiai.

4.2. Kompiuterinės technikos parametrai

Norint užtikrinti sklandų kodo veikimą kompiuterio sistema turi atitikti šiuos techninius reikalavimus:

- **Operacinė sistema:** Windows, Linux arba macOS, priklausomai nuo pasirinktos kūrimo aplinkos.
- **Procesorius:** Mažiausiai 2 GHz dviejų branduolių procesorius (rekomenduojama 4 branduoliai).

- **RAM atmintis:** Bent 4 GB (rekomenduojama 8 GB) RAM atminties.
- **Kietasis diskas:** Reikalinga bent 1 GB laisvos vietos sistemos įdiegimui ir duomenų bazės saugojimui.
- **Tinklo ryšys:** Stabilus interneto ryšys reikalingas, kad užtikrinti duomenų sinchronizavimą ir nuotoline funkcijas. Norint užtikrinti sklandų programos veikimą kompiuterio sistema turi atitikti šiuos techninius reikalavimus:

Norint užtikrinti sklandų programos veikimą mobilusis telefonas turi atitikti šiuos techninius reikalavimus:

- **Operacinė sistema:** Android 8.0 ar naujesnė, iOS 12.0 ar naujesnė versija.
- **Procesorius:** Minimalus keturių branduolių procesorius su 1,8 GHz arba greitesniu taktiniu dažniu.
- **Atmintis (RAM):** Bent 2 GB operatyviosios atminties.
- **Laisva saugyklos vieta:** Mažiausiai 50 MB programėlės diegimui ir papildomai 100 MB duomenų saugojimui.
- **Ekrano raiška:** Ne mažiau kaip 720x1280 pikselių (HD) raiška.
- **Interneto ryšys:** Stabilus interneto ryšys (3G arba greitesnis) duomenų mainams realiuoju laiku.

Šie techniniai parametrai padės užtikrinti, kad sistema veiks greitai ir patikimai

4.3. Programinės realizacijos paleidimas

Paleidimo procesas yra paprastas, tačiau reikalauja kruopštaus sekimo, kad būtų išvengta klaidų:

1. **Įdiekite reikiamas priklausomybes:** Patikrinkite, ar įdiegta **Java Development Kit (JDK)**, ir visi kiti reikalingi įrankiai (**Expo Go**, **Firestore SDK**).
2. **Kopijuokite programos failus** į serverį ar kompiuterį, kad pradėtumėte diegimo procesą.
3. **Kurkite duomenų bazę:** Naudojant **Firestore**, sukurkite duomenų bazę ir atlikite pirminį duomenų įkėlimą (pvz., meniu, vartotojų duomenys).
4. **Pradėkite serverio aplikaciją:** Sistema skirta serveriui, naudokite komandas (`java -jar app.jar` arba `npx expo start`), kad paleistumėte serverį ir patikrintumėte, ar viskas veikia. Jei programa sėkmingai pasileidžia turėtume gauti QR kodą nuskaitymui bei informacija kaip ką paleisti (žr. 12 pav.). Jei bus naudojamas emuliatorius reikia spausti a raidę, jei naudojamas realus mobilus telefonas reikia skanuoti kamera QR kodą(jei naudojamas Android telefonas prašome atsidaryti Expo Go programėlę ir joje skanuoti QR kodą).

```
PS C:\Users\g0d4s\IdeaProjects\Projektas-baigiamasis\Projektas-main\Kliento> npx expo start
Starting project at C:\Users\g0d4s\IdeaProjects\Projektas-baigiamasis\Projektas-main\Kliento
Starting Metro Bundler
The following packages should be updated for best compatibility with the installed expo version:
  expo@52.0.11 - expected version: ~52.0.20
  react-native@0.76.3 - expected version: 0.76.5
  react-native-webview@13.12.2 - expected version: 13.12.5
Your project may not work correctly until you install the expected versions of the packages.



> Metro waiting on exp://192.168.32.145:8081
> Scan the QR code above with Expo Go (Android) or the Camera app (iOS)

> Using Expo Go
> Press s | switch to development build

> Press a | open Android
> Press w | open web

> Press j | open debugger
> Press r | reload app
> Press m | toggle menu
> shift+m | more tools
> Press o | open project code in your editor

> Press ? | show all commands

Logs for your project will appear below. Press Ctrl+C to exit.
```

12 pav. Paleidimas

5. **Patikrinkite ryšį:** Įsitikinkite, kad kliento ir darbuotojo programėlės gali susisiekti tarpusavyje ir užtikrinti realaus laiko duomenų mainus. Tai galima atlikti paleidus abi programėles.

Po šių žingsnių sistema bus parengta naudoti.

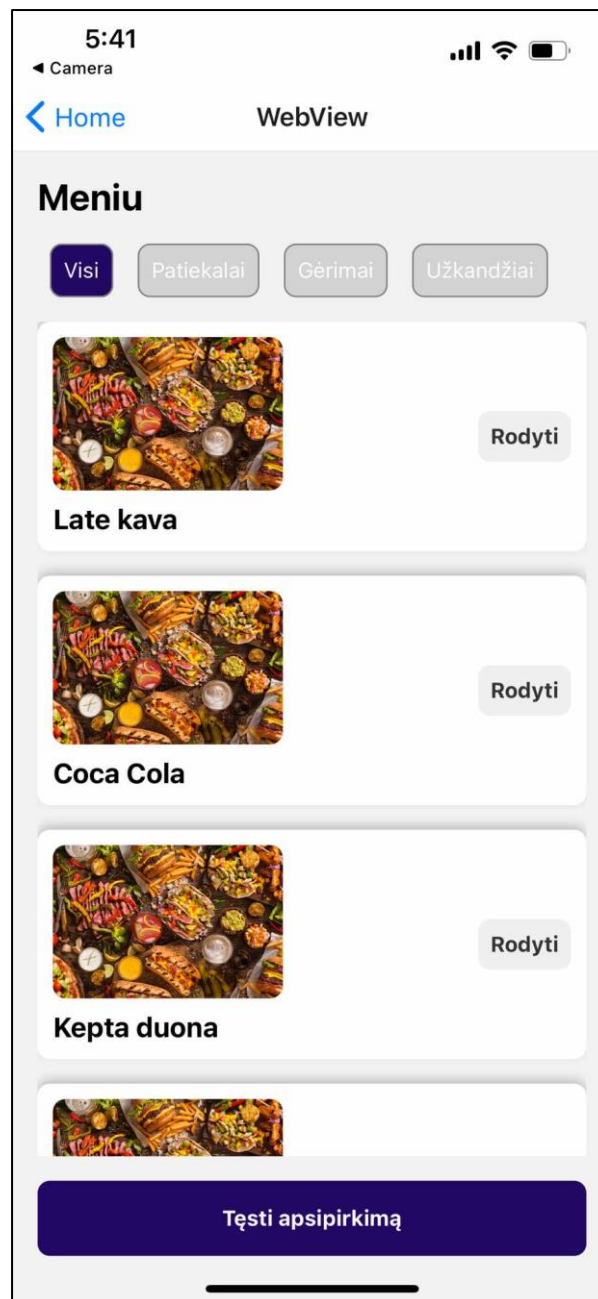
4.4. Programėlės naudojimosi instrukcija

Norint naudotis restorano užsakymų valdymo sistema, tiek klientams, tiek darbuotojams reikia atlikti šiuos veiksmus:

Klientų programėlė:

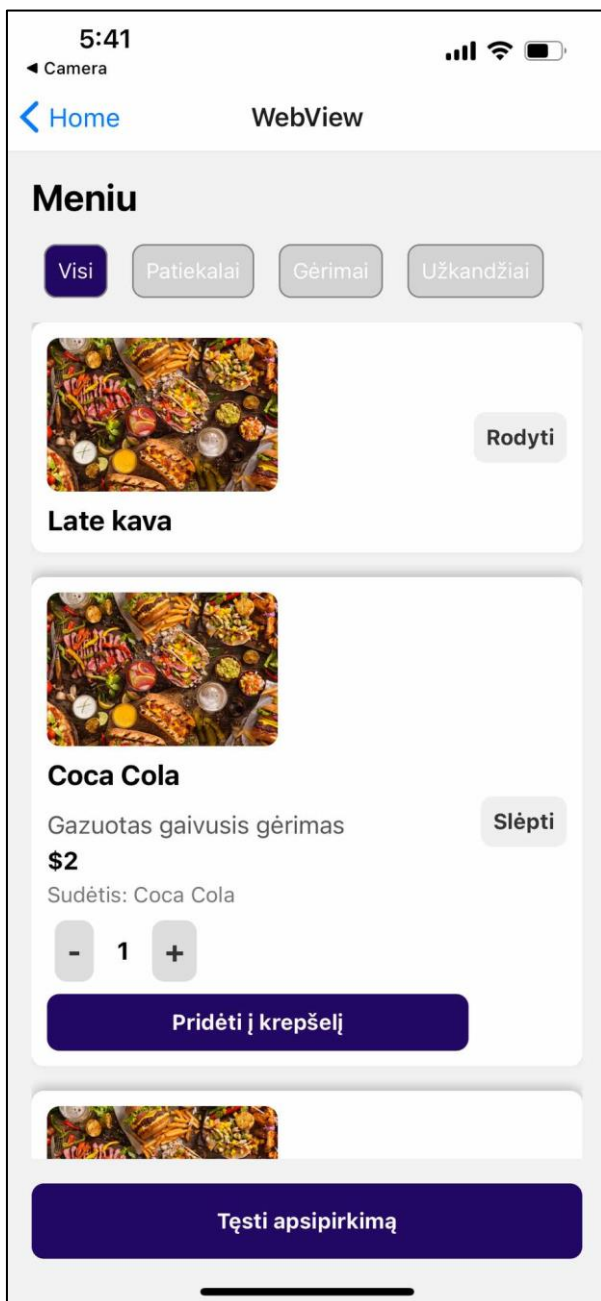


14 pav. QR kodo ekranas

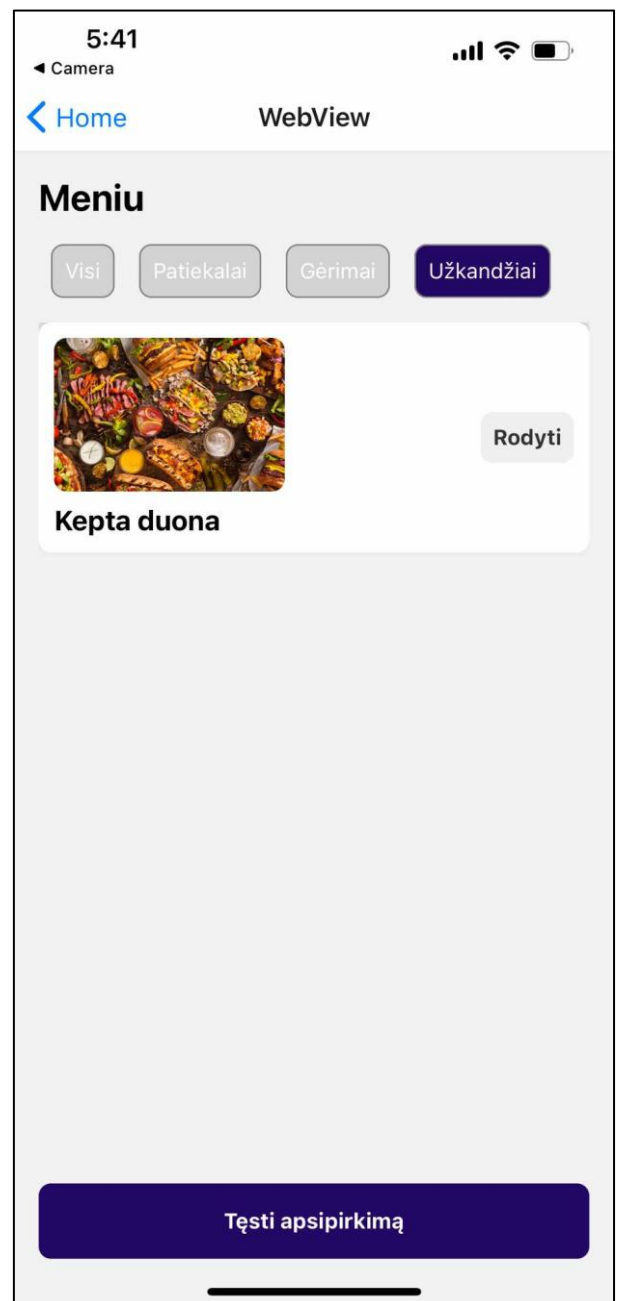


13 pav. Meniu ekrana

1. **QR kodo ekranas.** QR kodo ekranas nuveda į meniu ekrana kuriame galima peržvelgti visą esamą meniu turinį.
2. **Meniu peržiūra.** Klientas pasirenka norimą meniu kategoriją iš esamų ir peržiūri patiekalus arba gali žiūrėti viską bendrai. Paspaudus mygtuką „Rodyti“ galima pamatyti kainą, sudėti ir aprašymą (žr. 15 pav.). Visa informacija gaunama iš duomenų bazės. Pasirinkus tam tikrą kategoriją bus rodomi tik tam tikri patiekalai (žr. 16 pav.).

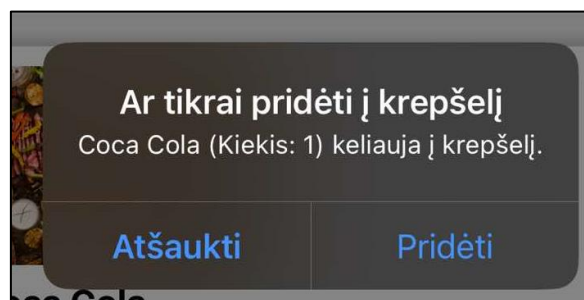


15 pav. Rodyti daugiau



16 pav. Užkandžių kategorija

3. **Užsakymo pateikimas.** Pasirinkus patiekalus, klientas įveda reikiamą kiekį ir spaudžia mygtuką „Pridėti į krepšelį“ tada gaunamas pranešimas su klausimu „Ar tikrai pridėti į krepšelį“ (žr. 17 pav.).

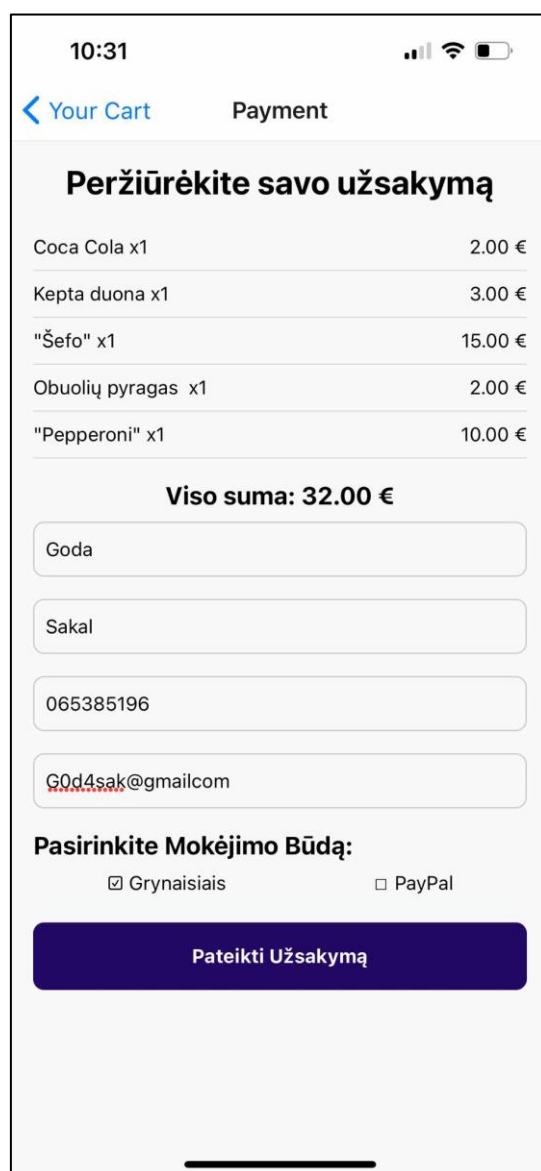


17 pav. Pridėti į krepšelį

4. **Krepšelio vaizdas.** Jame galima pamatyti ką susidėjote į krepšelį (žr. 18 pav.). Galima padidinti kiekį bei pašalinti nebenorima prekę, jei dar ko trūksta galima grįžti vienu ekranu atgal ir prisidėti norimų patiekalų. Nusprendus, kad krepšelyje yra viskas ko norime keliaujame toliau ir spaudžiame „Užsakyti“ kuris mus nuves į apmokėjimo ekraną. (žr. 19 pav.)

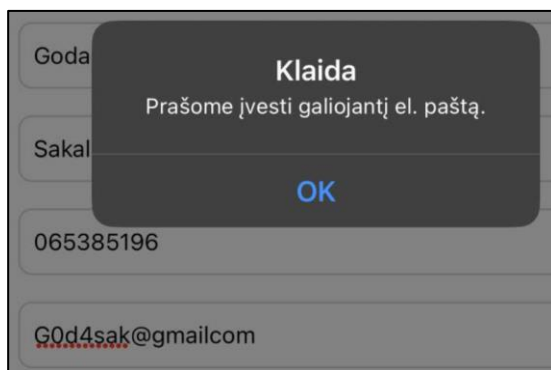


18 pav. Krepšelio vaizdas



19 pav. Apmokėjimo vaizdas

5. **Apmokėjimas.** Šiame ekrane reikia supildyti prašoma informacija kaip vardas, pavardė, telefono numeris bei El. Paštas. (žr. 19 pav.) Pasirinkus apmokėjimo metodą (PayPal, grynaisiais), užsakymas patvirtinamas ir perduodamas darbuotojų programėlei. Jeigu suvedami klaidingi duomenys kaip blogas paštas arba netinkamas telefono numeris gauname pranešimą. Šiuo atveju buvo blogas el. Pašto formatas (žr. 20 pav.).



20 pav. Blogas formatas

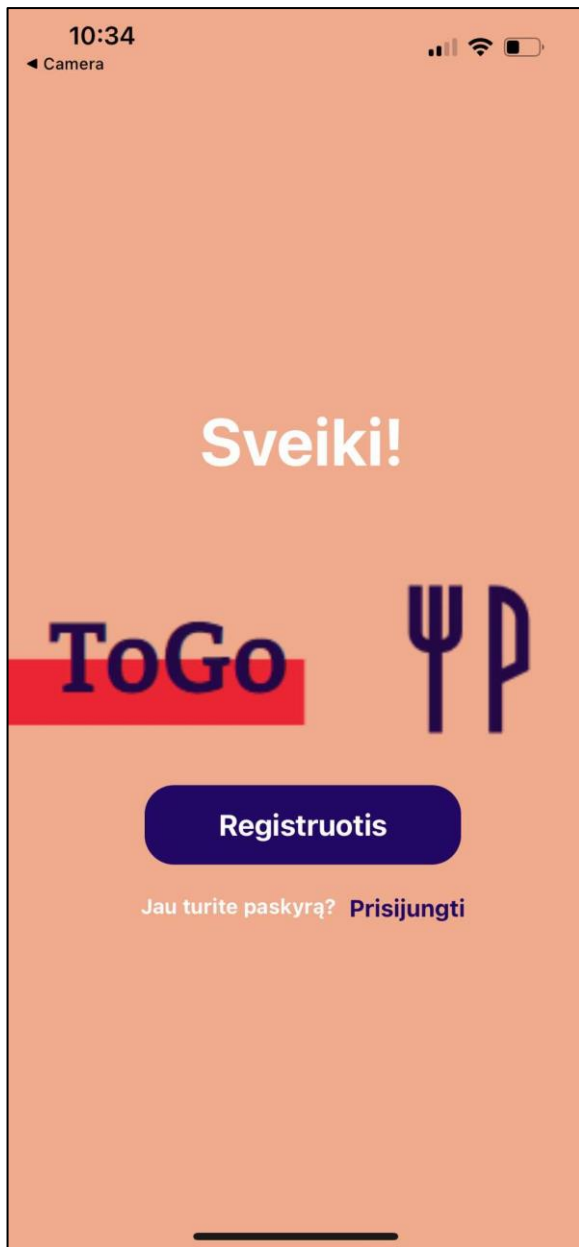
6. **Užsakymo sekimas:** Klientas gauna stebėjimo numerį 171 bei išrašą kas užsakyta, po visko galima grįžti į pagrindinį puslapį.



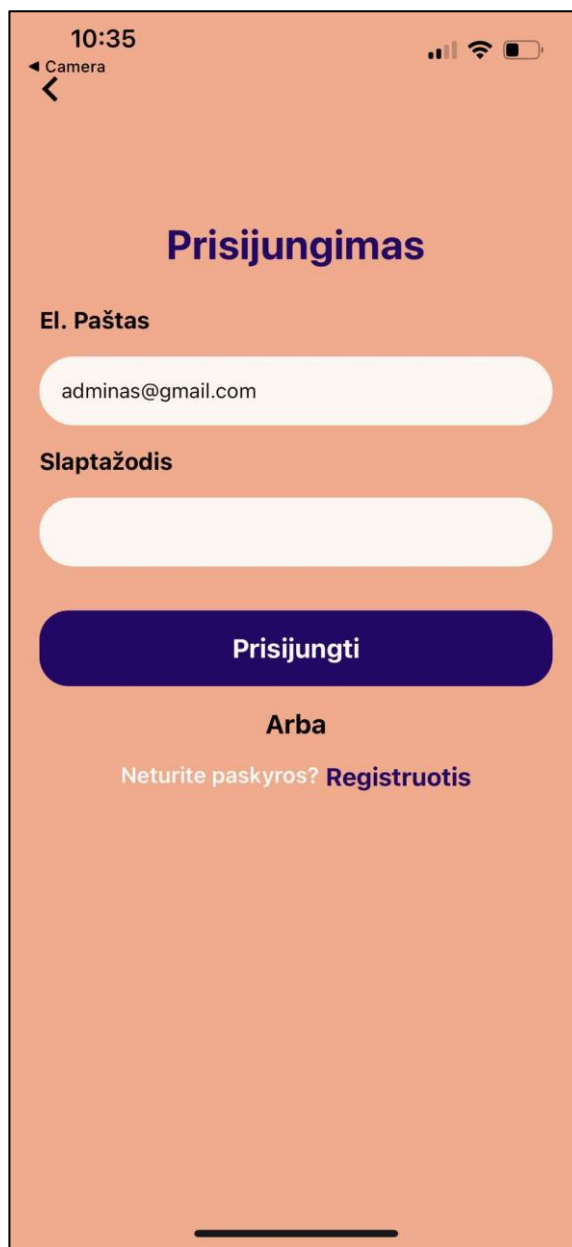
21 pav. Užsakymas pateiktas

Darbuotojų programėlė:

1. **Pradinis ekranas.** Programėlė atidaroma į pradinį ekraną, kuriame darbuotojas turi galimybę pasirinkti tarp registracijos arba prisijungimo. Šis sprendimas užtikrina patogų ir greitą prieigą tiek naujiems, tiek jau esamiems naudotojams.



23 pav. Pradinis ekranas



22 pav. Prisijungimo ekranas

2. **Prisijungimas.** Darbuotojai, turintys registruotą paskyrą, pasirenka prisijungimo parinktį. Jie įveda savo el. pašto adresą ir slaptažodį į tam skirtus laukus. Jei duomenys atitinka duomenų bazėje esančią informaciją, sistema suteikia prieigą prie visų administravimo ir valdymo funkcijų. Prisijungimo sėkmės atveju darbuotojas patenka į pagrindinį meniu, kur gali valdyti užsakymus ir meniu.

3. **Registracija.** Nauji darbuotojai užpildo registracijos formą, kurioje įrašomi būtini duomenys: vardas, pavardė, el. paštas, telefono numeris ir slaptažodis. Ši informacija saugoma duomenų bazėje, leidžiant ateityje darbuotojui prisijungti naudojant šiuos duomenis. Registracijos procesas apima duomenų validaciją, užtikrinant, kad įvesti duomenys atitiktų sistemos reikalavimus (žr. 24 pav. ir 25pav.).

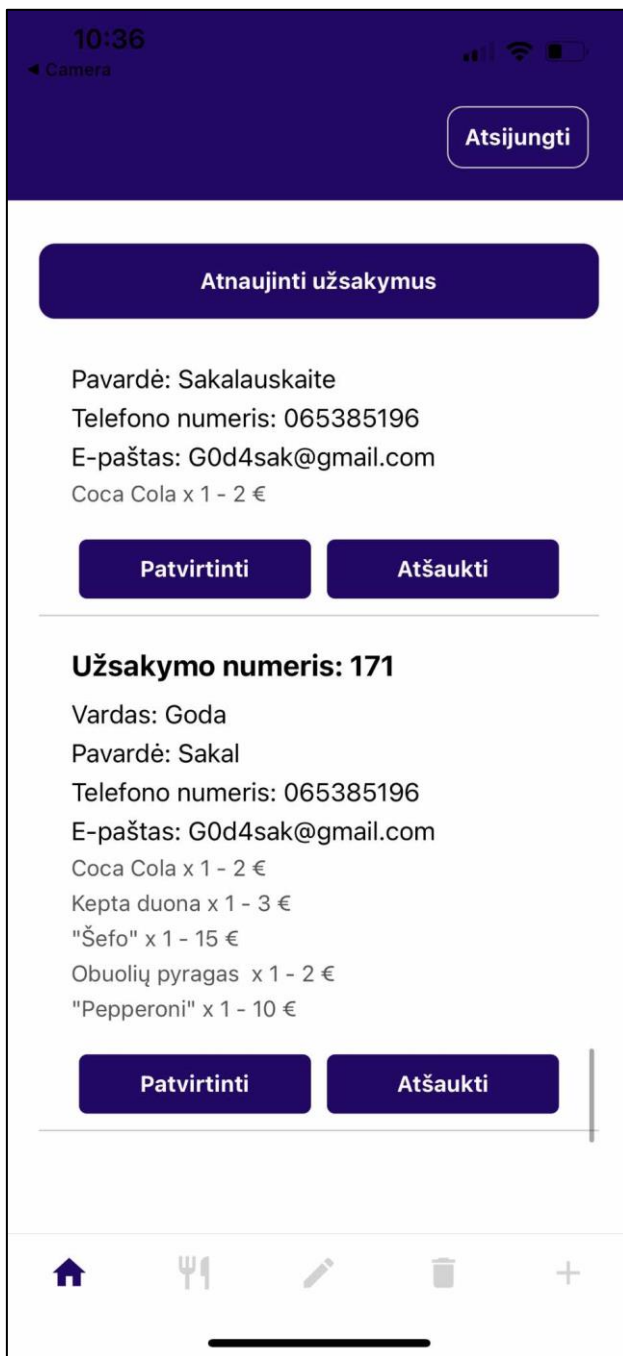
The screenshot shows a mobile app interface for registration. At the top, the status bar displays the time 10:34, signal strength, Wi-Fi, and battery icons. Below the status bar is a navigation bar with a back arrow and the word 'Camera'. The main heading is 'Registracija' in a bold, dark blue font. Below the heading are seven input fields, each with a label and a placeholder text: 'Vardas' (placeholder: įveskite vardą), 'Pavardė' (placeholder: įveskite pavardę), 'Pozicija' (placeholder: įveskite poziciją), 'Telefono numeris' (placeholder: įveskite telefono numerį), 'El. pašto adresas' (placeholder: įveskite el. paštą), and 'Slaptažodis' (placeholder: įveskite slaptažodį). At the bottom of the form is a large, dark blue button labeled 'Registruotis'. Below this button is the text 'Arba' and 'Jau turite paskyrą? **Prisijungti**'.

24 pav. Registracijos ekranas

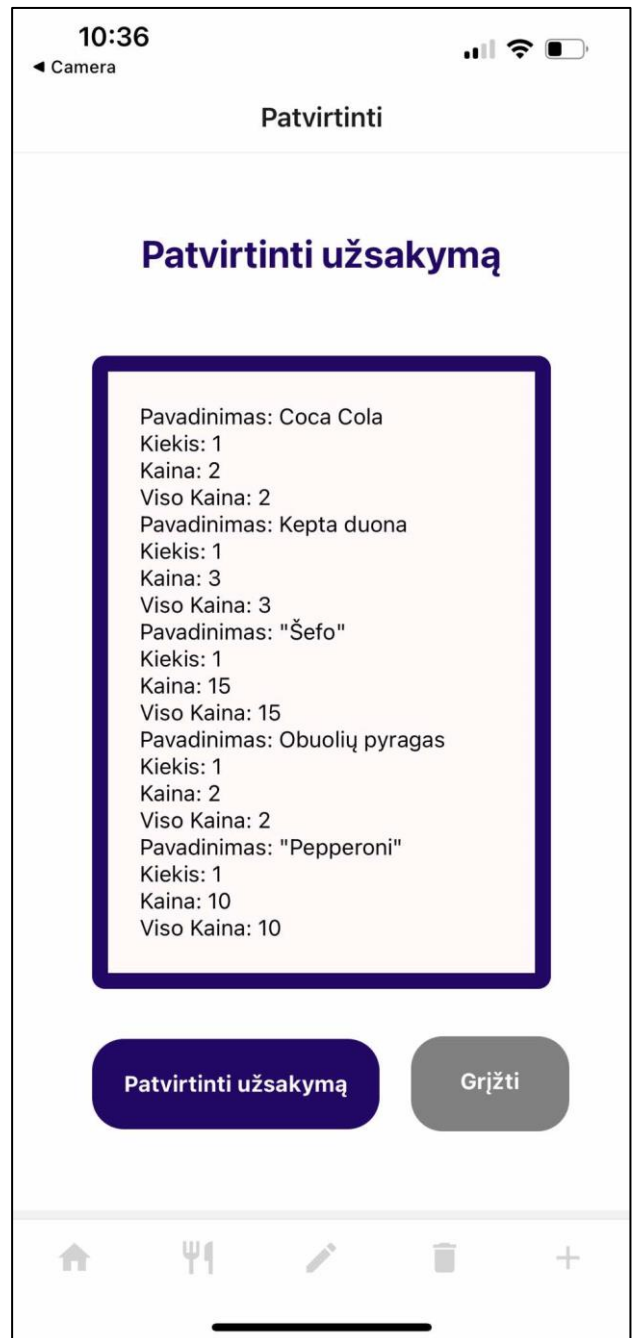
The screenshot shows the same registration form as the previous one, but with example data entered into the fields. The status bar now shows the time 10:36. The input fields contain: 'Vardas' (Goda), 'Pavardė' (Saka), 'Pozicija' (Šefas), 'Telefono numeris' (065385196), 'El. pašto adresas' (G0d4sak@gmail.com), and 'Slaptažodis' (empty). The 'Registruotis' button is still present. Below it is the text 'Arba' and 'Jau turite paskyrą? **Prisijungti**'.

25 pav. Suvesti registracijos duomenys

4. **Užsakymų patvirtinimas arba atšaukimas.** Darbuotojas gali peržiūrėti ir patvirtinti arba atšaukti kliento užsakymus (žr. 26 pav.). Atšauktas užsakymas ištrinamas, o patvirtintas matosi iki kol nėra atliktas arba atšauktas. Suradus katik atliktą užsakymą 171 ir patvirtiname jį (žr. 27 pav.). Šiame ekrane yra „Atnaujinti užsakymus“ mygtukas kurį paspaudus atsiranda nauji užsakymai realiu laiku.



26 pav. Užsakymų ekranas



27 pav. Užsakymo patvirtinimas

5. **Meniu administravimas:** Darbuotojas prideda naujus patiekalus paspaudes „Pridėti įrašą“ arba redaguoja esamus meniu įrašus. Viršuje yra du mygtukai po redagavimo būtina paspausti „Atnaujinti“ mygtuką, tam, kad nauja informacija sugultu duomenų bazėje, po jais yra kategorijos kaip ir kliento programėlėje. Paspaudus ant norimo patiekalo atsiranda daugiau informacijos kurią galima redaguoti arba visiškai ištrinti pasirinktą patiekalą. „Redaguoti“ ekrane galima keisti pavadinimą, aprašymą, sudėtį, kainą bei kategoriją. Jeigu joks patiekalas nepasirinktas „Redaguoti“ ekrane matysime, jog nėra ką redaguoti.

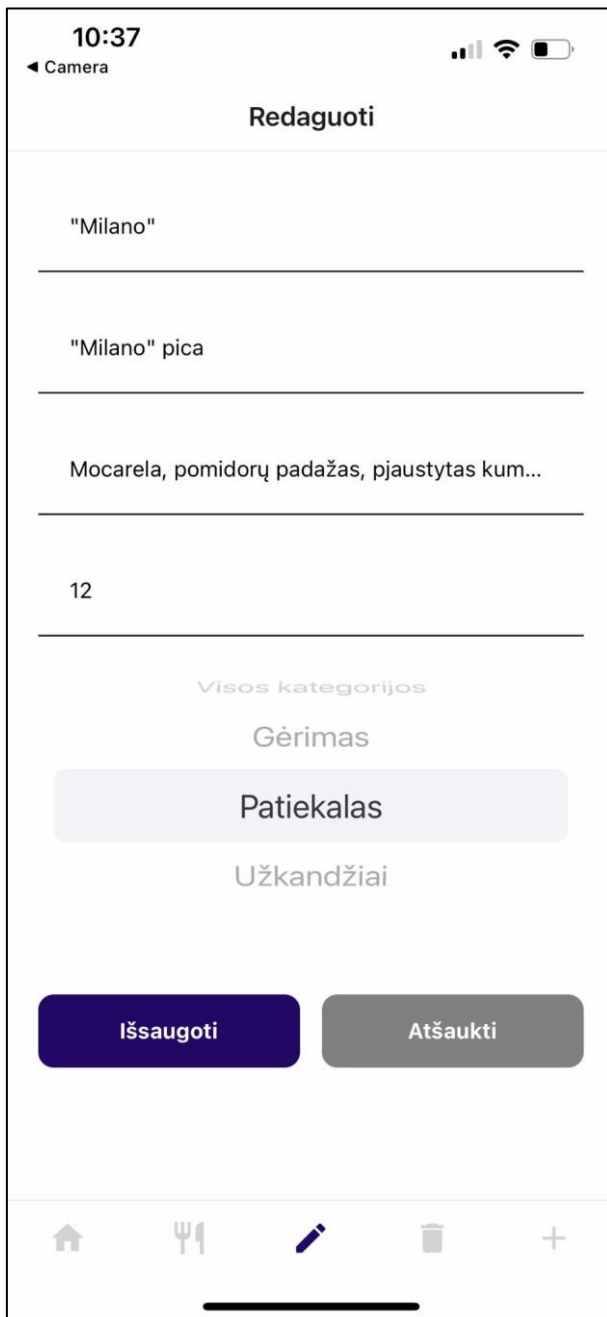
- Patiekalų redagavimas:



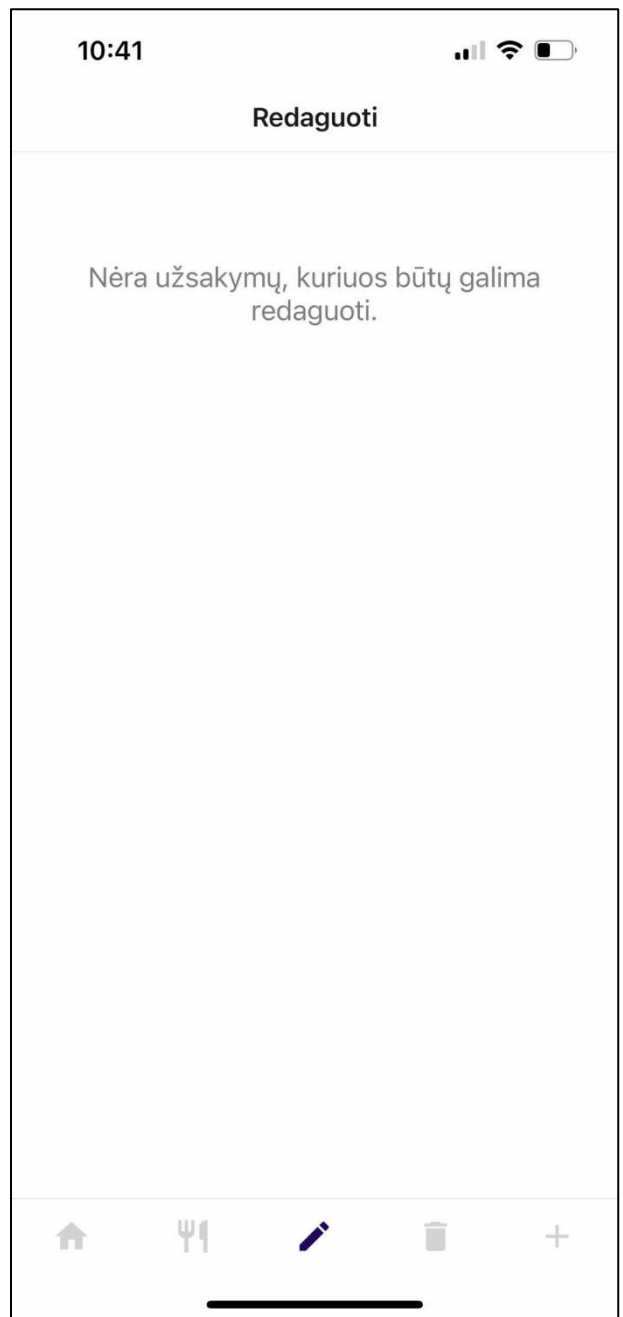
29 pav. Meniu ekranas



28 pav. Patiekalo informacija



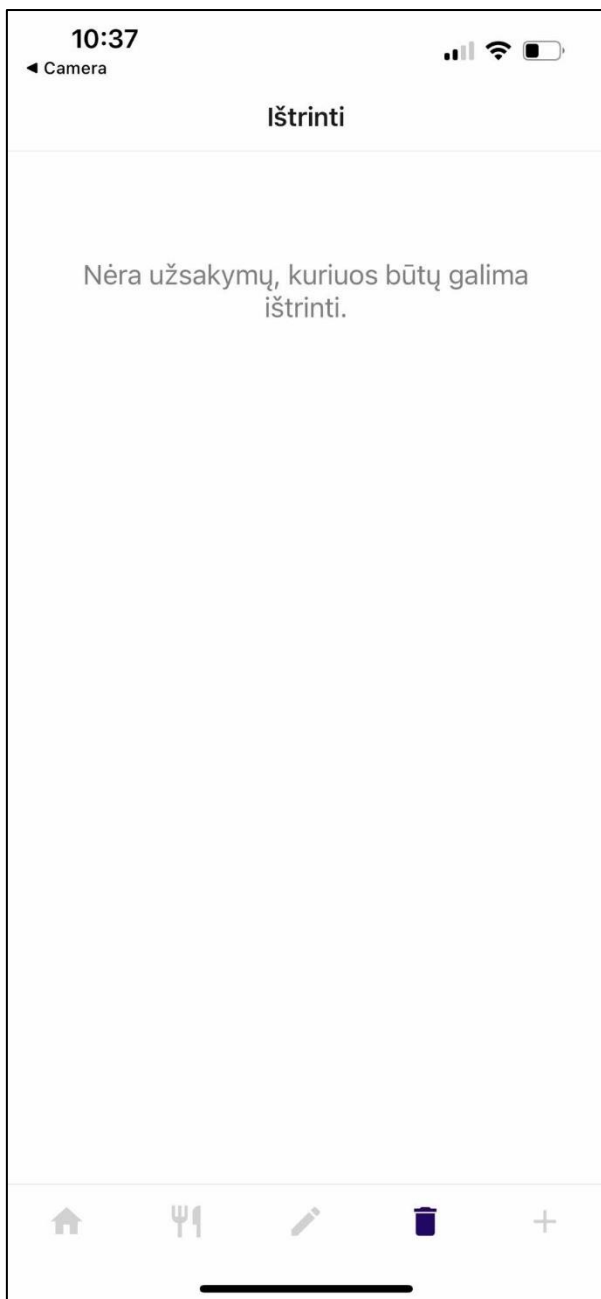
31 pav. Redaguoti ekranas



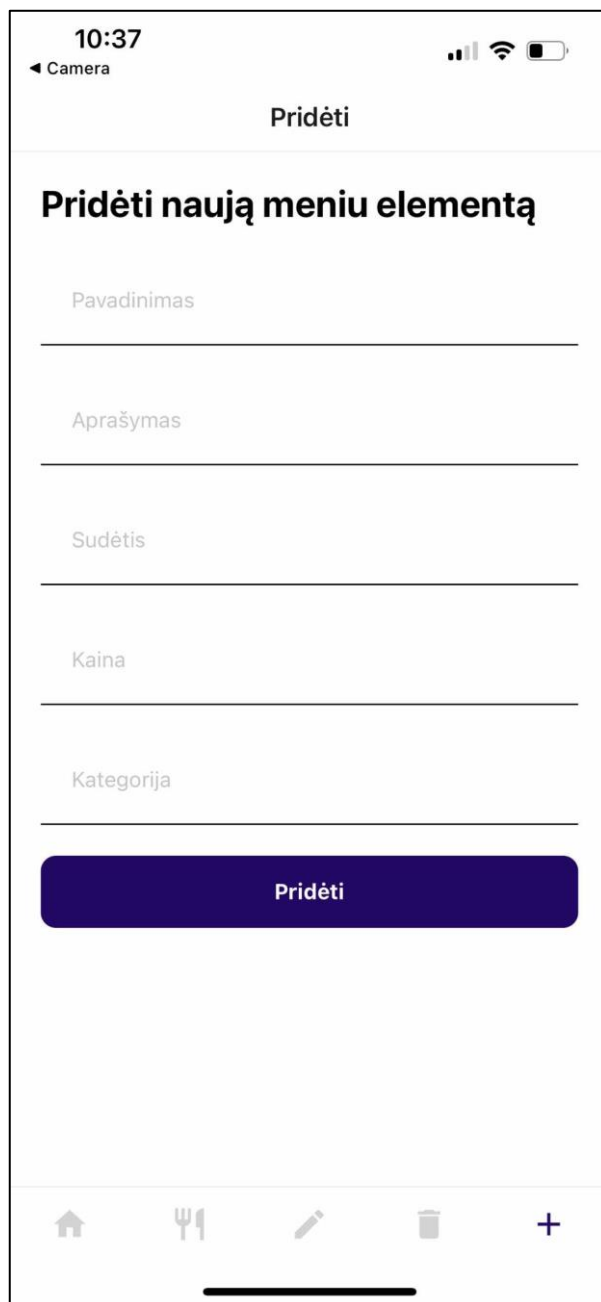
30 pav. Redaguoti ekranas kai nėra

Taip pat egzistuoja „Ištrinti“ ekranas kuriame matomi užsakymai kuriuos bandote ištrinti. Kaip jau minėta yra „Pridėti įrašą“ mygtukas po kurio paspaudimo galime sukurti naują patiekalą. Po sukūrimo reikia tik paspausti „Atnaujinti“ mygtuka ir iškart atsiranda naujas patiekalas.

- Pateikimų trinimas ir pridėjimas:



32 pav. Ištrinti ekranas



33 pav. Pridėti ekranas

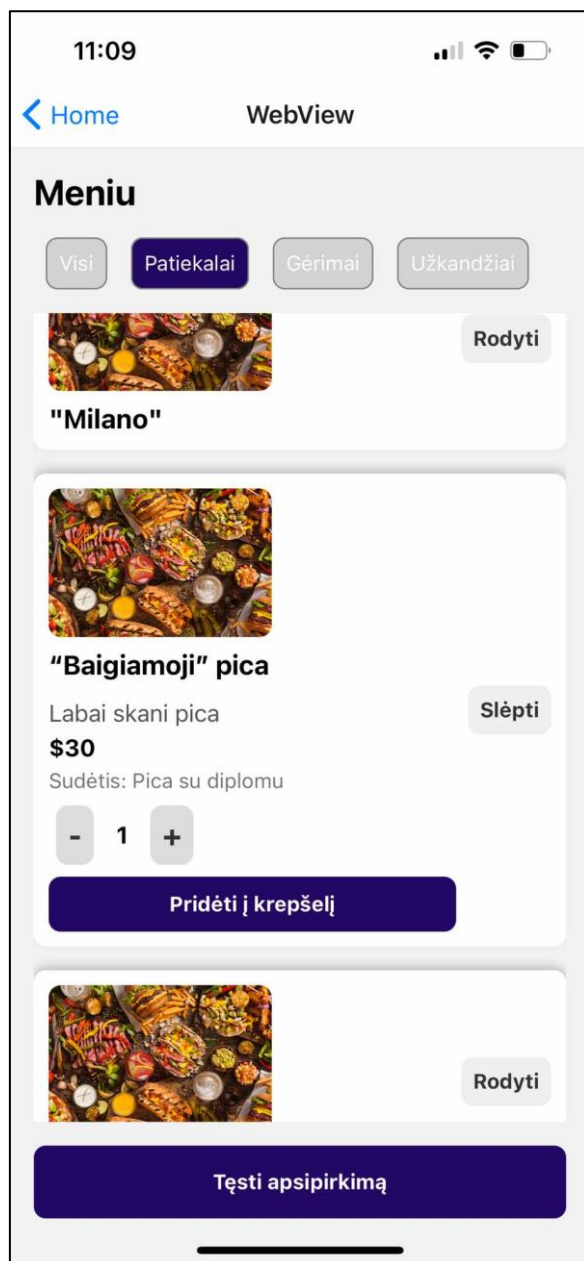


35 pav. Užpildytas pridėti ekranas



34 pav. Naujas patiekalas

Kai darbuotojas prideda ar redaguoja patiekalą, jis akimirksniu tampa matomas kliento programėlėje. Sistema pateikia informaciją apie patiekalą (pavadinimą, sudėtį, kainą ir vizualinius elementus) pagal šiuolaikinius vartotojo patirties principus. Klientai gali iškart užsakyti naują patiekalą, o darbuotojai realiuoju laiku mato šiuos užsakymus sistemoje, užtikrinant efektyvų ir greitą aptarnavimą. Galiausiai, taip atrodo naujas patiekalas, pateiktas klientui realiuoju laiku.



36 pav. Naujas patiekalas pas klientą

Jei sistema neveikia kaip tikėtasi, šie žingsniai padės išspręsti dažniausiai pasitaikančias problemas:

1. **Patikrinkite sistemos reikalavimus:** Uztikrinkite, kad visi techniniai reikalavimai yra įvykdyti (operacinė sistema, techniniai įrankiai).
2. **Perkraukite programą:** Dažnai programa tiesioginiam naudojimui turi būti perkrauta, kad atnaujintų ryšius ar pašalintų trikdžius.
3. **Tas pats interneto ryšys:** Kad programėlė veiktų nešiojamas kompiuteris ir telefonas turi būti prisijugę prie to pačio tinklo(jei naudojamas emuliatorius problemų neturėtų kilti).

4. **Klaidos žurnalai:** Patikrinkite sistemos klaidų žurnalus (log'us) ir identifikuokite galimas klaidas (pvz., klaidingas duomenų įrašas, prisijungimo problemos).
5. **Patikrinkite Firebase duomenų bazę:** Jei duomenų bazės ryšys nutrauktas, patikrinkite Firebase būseną ir įsitikinkite, kad tinklas veikia teisingai.

4.5. Programinės realizacijos šalinimo žingsniai

Norint pašalinti sistemą iš serverių, užtenka ištrinti Kliento ir Darbuotojo kodus bei Firebase ištrinti visą projektą. Šie žingsniai užtikrina, kad sistema bus pašalinta.

5. IŠVADOS IR SIŪLYMAI

1. Programėlės architektūra. Sukurta programėlės architektūra užtikrina sistemos lankstumą, patikimumą ir pritaikomumą įvairioms restoranų veikloms. Architektūra buvo suprojektuota naudojant modulinio dizaino principą, leidžiantį lengvai integruoti naujas funkcijas bei pritaikyti sistemą skirtingiems restoranų modeliams. Be to, sistema yra pagrįsta šiuolaikiniais technologiniais sprendimais, tokiais kaip Firebase ir GitHub, kurie užtikrina greitą duomenų apdorojimą ir aukštą patikimumo lygį. Tolimesniam tobulinimui rekomenduojama įgyvendinti papildomus saugumo sprendimus, tokius kaip HTTPS ryšio protokolą ir papildomas autentifikacijos priemones, siekiant apsaugoti jautrius duomenis.
2. Klientams skirta vartotojo sąsaja. Sukurta klientų vartotojo sąsaja suteikia galimybę patogiai peržiūrėti meniu, pateikti užsakymus ir juos apmokėti. Naudotojo patirtis buvo optimizuota, įgyvendinant intuityvius navigacijos sprendimus. Tobulinant šią sąsają, siūloma pridėti lojalumo programų funkcionalumą, galimybę naudoti nuolaidų kodus bei personalizuoti užsakymus pagal individualius klientų poreikius.
3. Darbuotojams skirta vartotojo sąsaja. Darbuotojams sukurta sąsaja apima prisijungimo, meniu valdymo, užsakymų patvirtinimo ir jų stebėjimo funkcijas. Šios funkcijos leidžia efektyviai organizuoti darbo procesus ir sumažinti klaidų tikimybę. Naudojant šiuolaikinius technologinius sprendimus, darbuotojai gali realiuoju laiku valdyti užsakymus ir sekti jų būseną. Tolimesniam funkcionalumo plėtojimui rekomenduojama pridėti išplėstą analitiką, kuri leistų darbuotojams ir vadovams analizuoti užsakymų statistiką bei identifikuoti veiklos gerinimo galimybes.
4. Duomenų mainų sprendimai. Sistema užtikrina sklandžius ir patikimus duomenų mainus tarp klientų ir darbuotojų sąsajų. Duomenų mainai veikia realiuoju laiku, užtikrinant tikslų ir greitą informacijos atnaujinimą abiejose pusėse. Ši integracija ženkliai sumažina užsakymų klaidų riziką bei pagerina bendrą veiklos efektyvumą. Siekiant dar didesnio sistemos našumo, rekomenduojama įdiegti automatines sesijų valdymo funkcijas, kurios uždarytų neaktyvias sesijas, taip sumažinant serverio apkrovą ir pagerinant duomenų apsaugą.
5. Technologinis pagrindas. Naudotos technologijos, tokios kaip IntelliJ IDEA, Expo Go, ir kitos šiuolaikinės priemonės, užtikrina aukštą sistemos patikimumo, greitą veikos ir lankstumo lygį. Sukurta infrastruktūra leidžia sistemai būti plačiai pritaikomai ir lengvai tobulinamai pagal restorano poreikius. Siūloma išplėsti sistemą, pridėdant mokėjimų per mobiliąs pinigines ar kitokias popiliarias mokėjimo galimybes, kas leistų dar labiau atitikti šiuolaikinių naudotojų poreikius.

Siūlymai tobulinimui:

1. Plėsti sistemos funkcionalumą. Sukurti integracijas su maisto pristatymo platformomis, tokiomis kaip UberEats ar Wolt, užtikrinant sklandų vietinių ir pristatomų užsakymų valdymą.
2. Klientų patirties gerinimas. Įgyvendinti lojalumo programų sekimą, galimybę naudoti nuolaidų kuponus ir pridėti personalizavimo funkcijas.
3. Mokėjimo galimybių išplėtimas. Pridėti mokėjimus mobiliomis pinigėmis ir kitas alternatyvias apmokėjimo formas.
4. Išplėstinė analitika. Sukurti funkcionalumą, leidžiantį restoranų vadovams stebėti pardavimų statistiką, populiariausius patiekalus ir klientų elgsenos duomenis.
5. Papildomos saugumo priemonės. Įdiegti saugumo sprendimus, tokius kaip HTTPS protokolas, siekiant užtikrinti aukštą duomenų apsaugos lygį.
6. Atsiliepimų rinkimas. Leisti klientams teikti atsiliepimus apie patiekalus ir aptarnavimą, taip padedant restoranams gerinti paslaugų kokybę.
7. Darbuotojų našumo įrankiai. Sukurti užduočių paskirstymo ir našumo stebėjimo funkcijas, kurios optimizuotų restoranų personalo darbą.

Tokie patobulinimai užtikrins dar didesnę sistemos pritaikomumą, funkcionalumą ir naudą tiek restoranams, tiek jų klientams.

INFORMACIJOS ŠALTINIŲ SĄRAŠAS

1. Aktas, D., Baltrūnienė, V., Blaževičienė, K., Kubilienė, E., Liepuonienė, R., Miakinkovienė, R., Neverbickaitė, D., Kačinitė-Vrubliauskienė, D., Sindaravičienė, N., & Žėkienė, D. (2023). *Bendrieji studijų rašto darbų reikalavimai: metodinė priemonė. Vilniaus kolegija.* https://www.viko.lt/wp-content/uploads/sites/8/2023/01/Bendrieji-studiju-rasto-darbureikalavimai_NAUJAS_nuo-2023-01-30.pdf
2. Android Developers. (2024). *Building user interfaces with Android.* <https://developer.android.com/guide/topics/ui>
3. Apple Developer. (2024). *Designing for iOS: Best practices for user interfaces.* <https://developer.apple.com/design/human-interface-guidelines/>
4. Blaževičienė, K. (2023). *Studijų rašto darbų šaltinių citavimas ir literatūros sąrašo sudarymas. Vilniaus kolegija.* <https://biblioteka.viko.lt/media/uploads/sites/25/2020/02/STUDIJC5%B2-RA%A0TO-DARB%C5%B2-%C5%A0ALTINI%C5%B2-CITAVIMAS-IR-LITERAT%C5%AAROS-S%C4%84RA%C5%A0O-SUDARYMAS-2023.pdf>
5. Expo. (2024). *Expo for React Native: Build native apps for Android and iOS using JavaScript.* <https://expo.dev/>
6. Firebase. (2024). *Firebase: Real-time database for building apps and websites.* <https://firebase.google.com/>
7. GitHub. (2024). *GitHub: Version control and collaboration platform.* <https://github.com/>
8. PayPal. (2024). *Sandbox testing guide.* <https://developer.paypal.com/tools/sandbox/>

PRIEDAI

1 PRIEDAS. PROGRAMŲ SISTEMŲ STUDIJŲ PROGRAMOS STUDIJŲ REZULTATAI

Studijų pakopos studijų rezultatų aprašymas		Studijų programos rezultatai		Pagrindimas
A.	Žinios ir jų taikymas	A.1	Paaiškinti pagrindinius faktus, sąvokas, teorijas ir matematinius metodus, susijusius su kompiuterių veikimu, kompiuterių technine ir programine įranga, jos savybėmis ir praktinio panaudojimo galimybėmis, kompiuterių komunikacija ir taikomaisiais sprendimais, kurie yra susiję su svarbiais istoriniais, dabartiniais ir galimais informatikos mokslų srities pokyčiais bei tendencijomis ateityje.	Restorano užsakymų valdymo sistemos kūrime buvo taikomos žinios apie duomenų bazės valdymo sistemas, jų optimizavimą, esybių–ryšių modeliavimą bei sąveiką su front-end sistemomis. Taip pat buvo pasitelkti pagrindiniai algoritminiai principai, siekiant užtikrinti efektyvų užsakymų pateikimą ir jų apdorojimą.
		A.2	Paaiškinti algoritmų sudarymo ir analizės principus, programavimo paradigmas, kalbas ir technologijas, žmogaus ir kompiuterio sąveikos principus, tipinius programinės įrangos gyvavimo ciklo etapus ir programinės įrangos kūrimo ir priežiūros metodus.	Projektas įgyvendintas naudojant objektinio programavimo principus (pvz., klasės ir paveldimumas). Žmogaus ir kompiuterio sąveikai užtikrinti buvo parengtos intuityvios naudotojo sąsajos, kurios leidžia lengvai peržiūrėti meniu, pateikti užsakymus.
		A.4	Taikyti programų sistemų krypties studijų žinias, kuriant saugius ir kitus aktualius kriterijus atitinkančius informatikos taikomuosius sprendimus konkrečioms profesinės veiklos problemoms spręsti.	Projekto metu buvo sukurta sistema, kuri leidžia optimizuoti restorano veiklą: efektyviai tvarkyti užsakymus, automatizuoti procesus ir pagerinti klientų patirtį. Buvo naudojamos šiuolaikinės saugumo technologijos, tokios kaip duomenų šifravimas ir vartotojo autentifikacija.
		A.5	Paaiškinti programų sistemų specifikavimą, projektavimą, testavimą ir dokumentavimą, programų sistemų inžinerijos valdymą, procesus, modelius ir metodus.	Sistemos specifikacija parengta remiantis UML modeliais (pvz., veiklos ir esybių–ryšių diagramomis). Kūrimo procesas sekė Agile metodiką, įtraukiant iteracinį požiūrį. Dokumentacija buvo

				rengiama viso proceso metu – nuo reikalavimų analizės iki galutinio produkto pristatymo.
B.	Gebėjimas vykdyti tyrimus	B.1	Apibūdinti duomenų bazių sistemų, internetinių technologijų, išmaniųjų įrenginių programavimo profesinės veiklos problemą bei paruošti konkrečiai profesinės veiklos problemai spręsti reikalingus duomenis ir informaciją iš įvairių šaltinių.	Restorano užsakymų valdymo sistemoje buvo identifikuota problema – neefektyvus užsakymų pateikimo procesas.
		B.2	Išanalizuoti ir įvertinti duomenų bazių sistemų, internetinių technologijų, išmaniųjų įrenginių programavimo konkrečiai profesinės veiklos problemai spręsti reikalingus duomenis, informaciją bei pagrįsti sprendimus argumentuotomis išvadomis.	Restoranų užsakymų valdymo sistemai pasirinktas reliacinės duomenų bazės modelis dėl griežto duomenų struktūruotumo ir galimybės atlikti kompleksines užklausas. Internetinių technologijų, tokių kaip React pasirinkimas buvo grindžiamas jų efektyvumu, lankstumu ir modernumu kuriant vartotojų bei darbuotojų sąsajas.
C.	Specialieji gebėjimai			
		C.3	Projektuoti programų sistemos architektūrą, komponentus, naudotojo sąsają ir testavimo programas pagal programų sistemai keliamus funkcinius ir nefunkcinius reikalavimus.	Restorano valdymo sistema buvo suprojektuota remiantis trijų sluoksnių architektūra, apimančia duomenų bazę, serverio logiką ir naudotojo sąsają. Naudotojo sąsaja buvo sukurta siekiant patogumo ir intuityvumo, o testavimo programos užtikrino sistemos stabilumą.
		C.4	Parengti specifikaciją, projektą ir kitą dokumentaciją, reikalingą programų sistemų produktui ar paslaugai sukurti, įdiegti, plėtoti, naudoti ir administruoti.	Buvo parengtos UML veiklos, sekos ir esybių-ryšių diagramos, dokumentuotos funkcinių reikalavimų specifikacijos.
		C.5	Igyvendinti programų sistemų produktą ar paslaugą konkrečiai profesinės veiklos problemai spręsti pagal programų sistemai keliamus funkcinius ir nefunkcinius reikalavimus.	Sistemoje įdiegta funkcionalumas, leidžiantis klientams patogiai pateikti užsakymus, o personalui – juos efektyviai tvarkyti.

				Buvo atsižvelgta į nefunkcinius reikalavimus, tokius kaip greitis, saugumas ir patikimumas.
D.	Socialiniai gebėjimai	D.1	Profesionaliai komunikuoti valstybine ir bent viena užsienio kalba su specialistų auditorijomis.	Projekto aprašymas buvo atliktas valstybine kalba bei tam tikros reikšmės anglų kalba.
E.	Asmeniniai gebėjimai	E.1	Savarankiškai mokytis ir dirbti, siekiant nuolatinio asmeninio ir profesinio tobulėjimo, imantis iniciatyvos ir prisiimant asmeninę atsakomybę.	Projekto kūrimas buvo vykdomas savarankiškai, teko mokytis naujas technologijas kaip hibridines mobiliąsias programėles.
		E.2	Demonstruoti kūrybingumą, sprendžiant profesinės veiklos uždavinius ir problemas.	Siekiant optimizuoti užsakymų valdymo procesą, buvo naudota inovatyvūs sprendimai, pvz., QR kodo integracija meniu peržiūrai ir užsakymų pateikimui. Šie sprendimai pagerino naudotojo patirtį ir leido sistemai tapti patogesnei bei funkcionalesnei.