VYTAUTO DIDŽIOJO UNIVERSITETAS
INFORMATIKOS FAKULTETAS
TAIKOMOSIOS INFORMATIKOS KATEDRA
SISTEMŲ ANALIZĖS KATEDRA

Kipras Bielinskas

APSKAITOS SISTEMA INDIVIDUALIA VEIKLA UŽSIIMANTIEMS VARTOTOJAMS

Bakalauro baigiamasis darbas

Multimedijos ir interneto technologijų studijų programa, valstybinis kodas 6121BX017
Informatikos inžinerijos studijų kryptis

Vadovas (-ė) Rita Valterytė
Moksl. laipsnis, vardas, pavardė parašas, data

Apginta prof. dr. Tomas KRILAVIČIUS
Fakulteto dekanas parašas, data

Kaunas, 2021

TURINYS

SANTRUMPŲ IR TERMINŲ ŽODYNAS

API	Programos programavimo sąsaja (angl. *application programing interface*), skirta trečiųjų šalių partneriams ar programuotojams, leidžianti programiškai gauti informaciją iš sistemos, kuri turi šią sąsają.
SaaS	Programų pardavimo būdas, kuomet prieiga prie programos yra suteikiama prenumeratos principu, o ne parduodama licencija programą naudoti neribotą laiką.
Debesis	Tai per internetą pasiekiami serveriai ar programinė įranga, duomenų bazės veikiančios tuose serveriuose.
Individuali veikla	„<...> tai dokumentas, kuris patvirtina, kad asmuo yra įregistravęs vykdomą veiklą savo gyvenamosios vietos VMI ir gali verstis pažymoje nurodyta veikla“.
PVM	Pridėtinės vertės mokestis, mokamas juridinių asmenų valstybei, kurių metinės pajamos viršija 45 000 eurų.
Klaidos kodas 500	Tai HTTP protokolo klaidos kodas, nurodantis, kad serveris dėl nenumatytos priežasties nesugebėjo įvykdyti užklausos.
Laisvoji profesija	„<...> profesija, kuria reikiamą kvalifikaciją turintys gyventojai verčiasi asmeniškai <...>“
Leidžiamieji atskaitymai	Tai tokios sąnaudos, kurios yra įprastai patiriamos vykdant veiklą.
Programavimo karkasas	Programinio kodo rinkiniai, suteikiantys pagrindinę programos struktūrą ir funkcionalumą.
AJAX	Technologija, leidžianti svetainėms būti atnaujintoms asinchroniniu būdu, neperkraunant puslapio.
CSS klasė	Tai specialus žymeklis, kuriuo galima pažymėti HTML elementus ir pritaikyti jiems bendrą CSS stilių.
MVC	Programos kodo architektūros tipas, kuris padalina visą programa į tris dalis: modelį, vaizdą ir kontrolerį (angl. *model-view-controller*).
OOP	Objektinis programavimas, tai programavimo būdas, paremtas „objektų“ konceptu.
Galutinė klasė	Tokia klasė, kuri nebegali būti tėvinė kitai klasei.
Konstruktorius	Automatiškai iškviečiama funkcija objekto sukūrimo pradžioje.
WYSIWYG	Vartotojos sąsajos programavimo metodika (angl. *what you see is what you get,* liet. ką matai, tą ir gausi), kuomet vartotojo sąsaja simuliuoja, kaip atrodys galutinis produktas, kuris bus atvaizduojamas kitame mediume.

Modalas	Vartotojo sąsajoje naudojamas iššokantis langas, kuriame gali būti sudėta vartotojui aktuali informacija ar prašymas užpildyti formą.

SANTRAUKA

Autorius	Kipras Bielinskas
Pavadinimas	Apskaitos sistema individualia veikla užsiimantiems vartotojams
Vadovas	Rita Valterytė
Darbas pristatytas	Vytauto Didžiojo Universitetas, Informatikos fakultetas, Kaunas 2021-06-07
Puslapių skaičius	73
Priedų skaičius	3

Darbo tikslas - sukurti buhalterijos sistemą, pritaikytą individualia veikla besiverčiantiems juridiniams asmenims, kurioje šie galėtų generuoti ir valdyti sąskaitas-faktūras ir išlaidas bei skaičiuoti mokėtinus mokesčius.

Bakalauro darbo metu pristatomos ir palyginamos kelios individualia veikla besiverčiantiems asmenims pritaikytos buhalterijos sistemos. Išnagrinėjami jų trūkumai ir pateikiami funkcinių bei dizaino problemų sprendimo būdai. Pristatomi naujai kuriamos apskaitos sistemos funkciniai ir nefunkciniai reikalavimai, sistemoje įdiegti įvairūs mokesčių tarifai. Pateikti argumentuotai pasirinkti sistemos kūrimo darbams skirti įrankiai ir technologijos. Projektavimo etape pateikiamos veiklos diagramos, duomenų bazės modelis. Realizavimo etape aiškinamas programavimo procesas, numatytų funkcinių ir nefunkcinių reikalavimų įgyvendinimas ir naudojamų įrankių savitumas. Atliktas veikiančios sistemos darbo testavimas, kuris aprašytas testavimo dalyje, pateikti rezultatai ir išvados.

ABSTRACT

Author	Kipras Bielinskas
Title	Accounting system for self-employed users
Supervisor	Rita Valterytė
Presented at	Vytautas Magnus University, Faculty of Informatics, Kaunas 2021-06-07
Number of pages	73
Number of appendices	3

The aim of the project is to create an accounting system designed specifically for self-employed users in which they could generate and manage invoices, expenses and calculate taxes.

During the Bachelor Project, several accounting systems adapted for self-employed people were presented and compared. The limitations were identified, and the solutions to functional and design problems were presented. Furthermore, certain functional and non-functional requirements of the newly developed accounting system were discussed, various tax rates were introduced. Also, the justifiably selected tools and technologies for system development work were presented. In the design stage, activity diagrams, a database model were demonstrated. In the implementation stage, the programming process, the realization of the intended functional and non-functional requirements, and the specificity of the tools used were explained. Finally, the operation of the developed system was tested and thoroughly described in the testing part, the results and conclusions were presented.

1. ĮVADAS

Augant individualia veikla besiverčiančių asmenų skaičiui, kartu didėja ir patogesnio buhalterijos valdymo poreikis. Šiam poreikiui patenkinti egzistuoja kelios mokamos internetinės buhalterijos sistemos, kurios siūlo galimybes išrašyti sąskaitas, sekti išlaidas, stebėti mokesčius ir kitas juridiniams asmenims aktualias funkcijas. Tačiau tokios buhalterijos sistemos skirtos itin plačiam juridinių asmenų spektrui ir dažnai pasižymi perteklinėmis funkcijomis, prasta vartotojo sąsaja ar netiksliais, neaktualiais mokesčių apskaičiavimo būdais.

Siekiant išvengti paminėtų problemų, šio bakalauro darbo metu sukurta nauja buhalterijos sistema, kurios pagrindinis klientas yra individualia veikla besiverčiantis asmuo. Tokios sistemos sukūrimas išspręstų būtent šio segmento vartotojų su mokėtinais mokesčiais ir individualios veiklos valdymu susijusias problemas, kurių visapusiško sprendimo rinkoje dar nėra sukurta. Kuriant visiškai naują apskaitos sistemą, galima susitelkti į svarbiausius individualios veiklos apskaitos kriterijus ir įtraukti visus aktualiausius mokesčių skaičiavimo būdus. Tokia sistema vienareikšmiškai palengvintų individualia veikla užsiimančių asmenų buhalterijos valdymą, nes dar nesulaukus mokesčių deklaravimo periodo toks vartotojas aiškiai matytų mokėtinus mokesčius, klientams galėtų patogiai ir lengvai išrašyti sąskaitas faktūras. Taip pat, toks vartotojas galėtų stebėti savo veiklos balansą, vesti patirtas išlaidas ir atitinkamai planuoti savo finansus. Sistema leistų laisvai samdomiems darbuotojams būti labiau užtikrintiems savo veiklos finansine padėtimi, priimti ateities sprendimus. Dėl šių priežasčių buvo nuspręsta sukurti būtent šio vartotojo poreikius atitinkančią sistemą.

1.1 Darbo tikslas

Sukurti buhalterijos sistemą, pritaikytą individualia veikla besiverčiantiems juridiniams asmenims, kurioje šie galėtų generuoti ir valdyti sąskaitas-faktūras ir išlaidas, skaičiuoti mokėtinus mokesčius, siųsti elektroninius laiškus su prisegtomis sąskaitomis.

1.2 Darbo uždaviniai

1. Ištirti, kokio funkcionalumo trūksta kitose panašaus pobūdžio sistemose;
2. Išsiaiškinti, kokie mokesčiai yra aktualūs individualia veikla besiverčiančiam asmeniui;
3. Apžvelgti sistemos kūrimui tinkamus programavimo įrankius;
4. Suprojektuoti sistemos duomenų bazę ir funkcinę logiką;
5. Suprogramuoti apskaitos sistemą, kurioje vartotojas galės generuoti ir valdyti sąskaitas-faktūras;

6. Integruoti galimybę pridėti ir sekti vartotojo patirtas išlaidas;
7. Sukurti intuityvią sąskaitos-faktūros išrašymo vartotojo sąsają;
8. Suprogramuoti mokėtinų valstybei mokesčių apskaičiavimą pagal pasirinktą veiklos tipą ir papildomus parametrus;
9. Sukurti galimybę siųsti el. laiškus gavėjams su prisegtais PDF priedais;
10. Integruoti automatinius priminimus, siunčiamus el. paštu nurodytiems adresams dėl vėluojančių apmokėjimų;
11. Integruoti paaiškinimus vartotojui apie mokesčių apskaičiavimą ir galimas parinktis;
12. Integruoti detalesnį sąskaitos-faktūros būsenų valdymą;
13. Atlikti sistemos testavimą.

2. ANALITINĖ DALIS

Asmeniui, pradėjusiam vykdyti individualią veiklą, reikia nuspręsti, kaip šis norės tvarkyti buhalteriją: ar sąskaitas-faktūras rašyti ranka ir naudoti įvairias skaičiuokles, tokias kaip *Microsoft Excel* norint sekti sąskaitas, patirtas išlaidas ir mokėtinų mokesčių sumą, ar pasirinkti vieną iš nedaugelio internetinių sistemų, kurios siūlo automatizuoti daugelį daug laiko užimančių procesų. Įprastai viskas priklauso nuo asmens patirties individualios veiklos buhalterijos vedime. Jei šis puikiai supranta, kaip reikia teisingai išrašyti sąskaitą-faktūrą, kaip vesti išlaidas ir apskaičiuoti mokesčius, o į metus jam tereikia išrašyti keletą sąskaitų, logiškas sprendimas - naudoti savo kurtas skaičiuokles. Tačiau jei asmuo tik pradėjo vykdyti individualią veiklą ar paprasčiausiai neturi laiko pats kruopščiai vesti buhalterijos – geras pasirinkimas yra perkelti buhalteriją į internetinę apskaitos sistemą, kuri gali pasiūlyti įvairias funkcijas, kurių neįmanoma gauti vedant buhalteriją skaičiuoklėje: automatinį sąskaitos generavimą PDF formatu, tikslų mokesčių apskaičiavimą, intuityvią vartotojo sąsają.

2.1 Esamų apskaitos sistemų analizė

„Apskaitos sistema“ arba „Buhalterijos sistema“ – sistema pritaikyta patogiam ir efektyviam buhalterijos sekimui. Šios sistemos dažniausiai pasiekiamos per naršyklę, prisijungiant naudojantis asmenine ar įmonės paskyra. Sistemoje vartotojas gali valdyti įvarius su juridine veikla susijusius funkcionalumus.

Lietuvoje individualia veikla besiverčiantis asmuo turi kelis buhalterijos sistemų pasirinkimus. Šiame darbe nagrinėjamos trys populiarios apskaitos sistemos, kurios atitinkamai suteikia reikiamą funkcionalumą bet kokiam vartotojui: FreshBooks, Sąskaita123 ir i.APS.

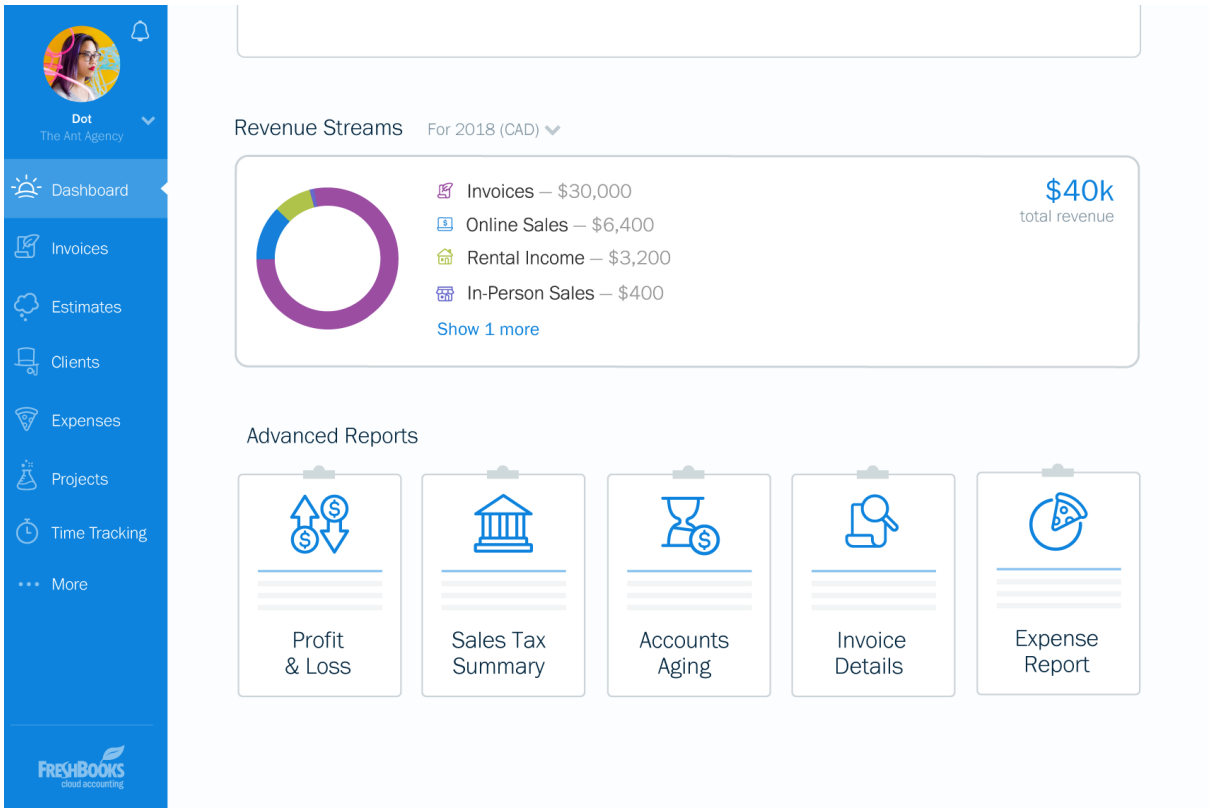
Šios sistemos lygintos pagal šiuos kriterijus:

- kiek vartotojui kainuoja paslauga,
- kokį bendrą funkcionalumą paslauga siūlo,
- ar šis funkcionalumas aktualus asmeniui, vykdančiam individualią veiklą Lietuvoje,
- vartotojo sąsajos intuityvumas,
- bendras paslaugos tiekėjo įvaizdis,
- ar yra išskirtinių sąlygų taikomų vartotojams, dėl kurių šie negalėtų naudotis paslauga,
- ar paslauga gali būti integruota su kitomis sistemomis,
- ar paslauga prieinama lietuvių kalba.

Kartu su pasirinktomis sistemomis buvo lyginama ir šio projekto metu sukurta apskaitos sistema.

2.1.1 FreshBooks

„PCmag“ svetainės duomenimis, FreshBooks (žiūrėti 1 pav.) yra populiariausia ir geriausiai pritaikyta apskaitos sistema individualia veikla užsiimantiems asmenims [1]. 2003 metais įsteigta įmonė pristatė savo pirmąją programinę įrangą kaip paslaugą (angl. *software as a service*, toliau SaaS) skirtą generuoti sąskaitas-faktūras, o pagrindiniai naudotojai buvo interneto technologijomis užsiimantys laisvai samdomi asmenys [2]. Šiuo metu FreshBooks siūlo itin platų paslaugų asortimentą: sąskaitų sukūrimą, valdymą ir generavimą, darbo laiko sekimą, automatinius neapmokėtų sąskaitų priminimus ir galimybę automatiškai rinkti apmokėjimus per jų atvirą ir viešai prieinamą API [3]. Tačiau, norint pasinaudoti šiomis paslaugomis, tenka mokėti mėnesinį mokestį, kuris prasideda nuo 15 dolerių (12.67 €) iki 50 dolerių (42.24 €). Pasirinkus brangiausią paketą, vartotojui taip pat suteikiamas saugus duomenų saugojimas *debesyje*, klientų profilių kūrimas ir išsaugojimas bei automatinis banko sąskaitos tikrinimas [4].

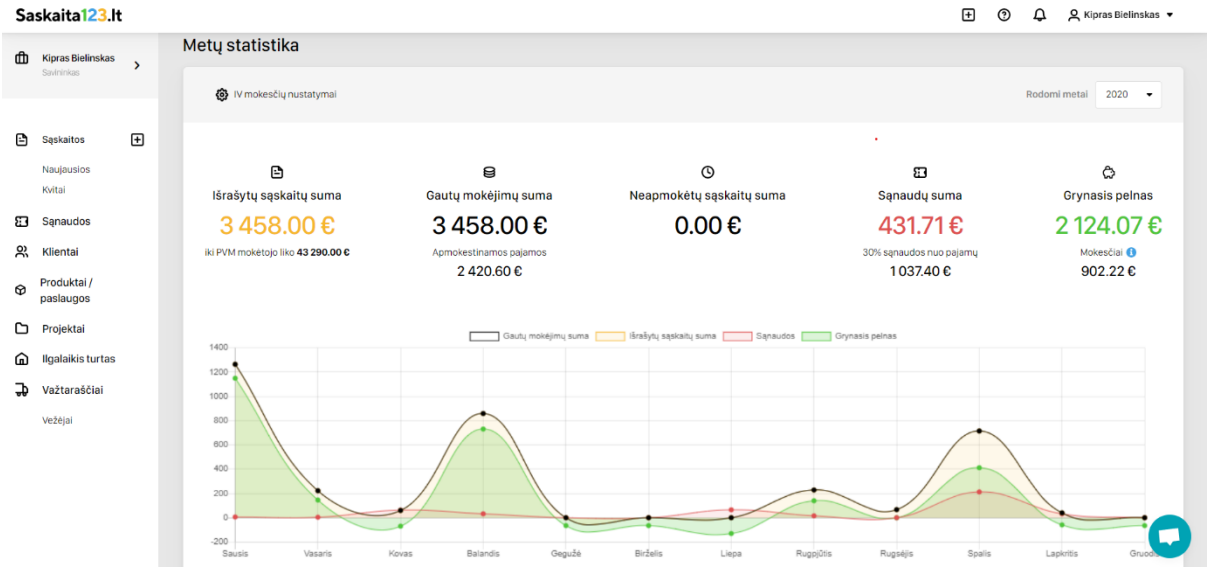


1 pav. FreshBooks valdymo skydelis

Ši apskaitos sistema nesiūlo automatinio mokesčių skaičiavimo dėl itin plataus vartotojų geografinio pasiskirstymo, kurio didžiąją dalį vartotojų sudaro Šiaurės Amerikoje gyvenantys asmenys. Asmuo, besinaudojantis šia sistema ir užsiimantis individualia veikla Lietuvoje, mokesčius nuo gautų pajamų turi apsiskaičiuoti pats, nes FreshBooks sistemoje nėra numatyti Lietuvoje juridiniams asmenims aktualūs mokesčių tarifai. Ši sistema puikiai tinka asmenims, užsiimantiems individualia veikla, norintiems turėti didelį funkcionalumą apskaitos sistemoje ir mokantiems apskaičiuoti mokėtinus mokesčius nuo gautų pajamų sumos patiems.

2.1.2 Sąskaita123

SimilarWeb duomenimis, www.saskaita123.lt [5] – viena populiariausių Lietuvoje veikiančių apskaitos sistemų (žiūrėti 2 pav.), kuri veiklą vykdo *SaaS* principu. 2016 metais veiklą pradėjusi įmonė šiuo metu, jų duomenimis, turi 8 tūkstančius registruotų vartotojų [6]. Asmenys, norintys pradėti naudotis Sąskaita123 teikiamomis paslaugomis, gali tai padaryti užsisakant paslaugų paketus, kurių kainos svyruoja nuo 6,99 € iki 11,99 € per mėnesį. Individualia veikla užsiimantys asmenys yra skatinami rinktis pigesnį planą, kuris leidžia paskyroje valdyti vieną veiklą, išrašyti neribotą sąskaitų kiekį, eksportuoti sąskaitas įvairiais formatais ir kurti pasikartojančias sąskaitas.



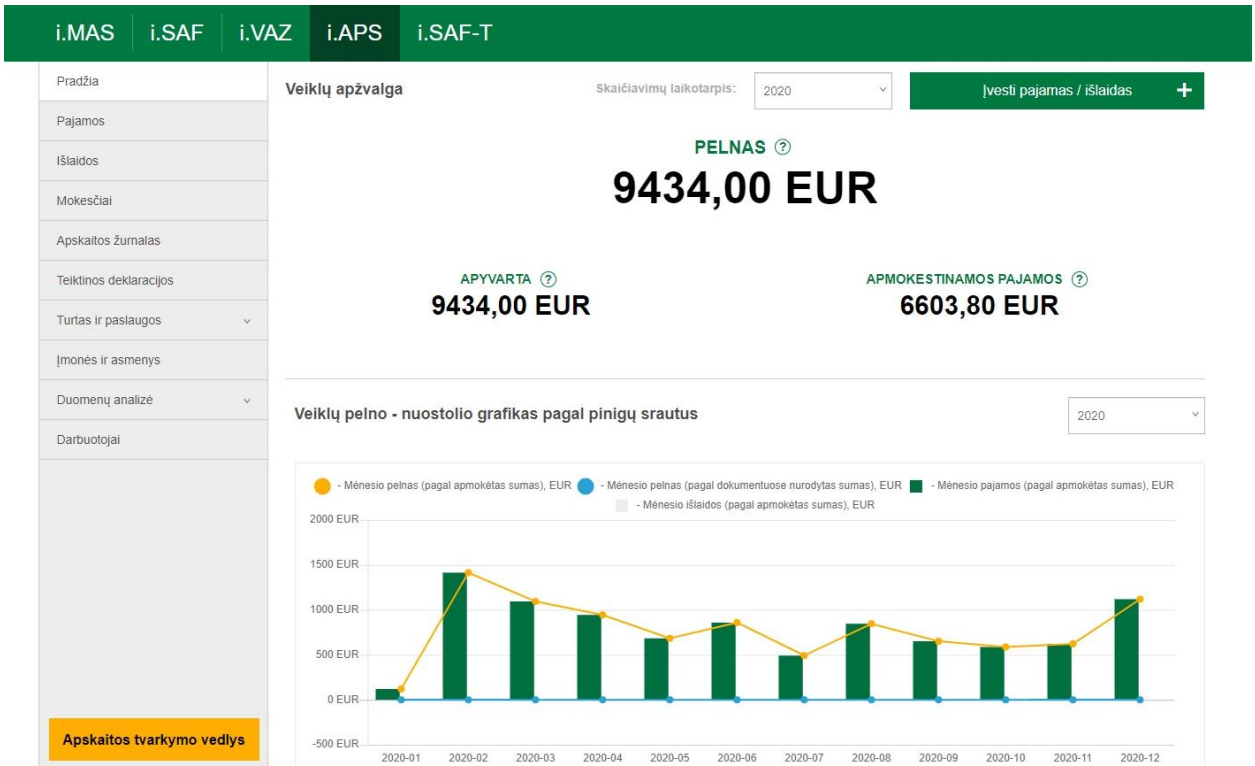
2 pav. Sąskaita123 valdymo skydelis

Įmonėms siūloma rinktis brangesnį paslaugų planą su galimybe naudotis šiomis papildomomis funkcijomis: el. laiškų siuntimu iš įmonės pašto bei darbuotojų valdymu. Vienas didžiausių šios sistemos privalumų – galimybė sekti mokėtinus mokesčius nuo gautų pajamų sumos. Vis tik šioje sistemoje yra įvesti tik populiariausi mokesčių tarifai – jeigu asmens vykdoma

veikla turi specialių reikalavimų, tokiu atveju sistema negeba apskaičiuoti mokesčių tiksliai. Ši sistema puikiai tinka asmenims, užsiimantiems tik individualia veikla ir neturintiems jokių mokestinių lengvatų.

2.1.3 i.APS

Lietuvos valstybinės mokesčių inspekcijos iniciatyva, finansuojama iš Europos regioninės plėtros fondo, sukurta internetinė apskaitos sistema i.APS yra skirta individualia veikla užsiimantiems asmenims (žiūrėti 3 pav.). Ši sistema yra viešai ir nemokamai prieinama visiems vykdantiems individualią veiklą pagal pažymą ar įsigijusiems verslo liudijimus. 2019 metais sukurta sistema yra sąlyginai nauja, todėl turi itin ribotą funkcionalumą: sąskaitų-faktūrų išrašymą, automatinį pajamų ir išlaidų žurnalo pildymą, PVM ribos skaičiuoklę ir automatiškai užpildomą preliminarią metinę pajamų deklaraciją. Taip pat planuojama įdiegti ir automatinius mokesčių skaičiavimus ir patogų jų apmokėjimą [7]. Jeigu asmeniui reikia dažnai tvarkyti finansus, vienintelis būdas prisijungti prie sistemos naudojantis valdžios vartais gali labai apsunkinti visą darbo eigą. Taip pat, nestandartinis ir neintuityvus dizainas, pilnas lietuviškų ir industrijoje retai naudojamų terminų, neparyškina vartotojui aktualiausių funkcijų, labai apsunkina navigaciją ir reikiamų funkcijų radimą. Sistemoje taip pat dažnai atsiranda ir nenumatytų serverio problemų (klaidos kodas 500).



3 pav. VMI i.APS sistemos valdymo skydelis

Nepaisant galybės sistemos trūkumų, ši apskaitos sistema puikiai tinka vartotojams, kurie nėra bandę naudoti kitų apskaitos sistemų, tad jiems netrukdys iš anksčiau susidariusios vartotojo sąsajos žinios. Sistema taip pat tinka asmenims, nenorintiems naudotis SaaS tipo paslaugomis ir pasitikintiems Lietuvos mokesčių inspekcija, jog ši tęs sistemos tobulinimo darbus.

2.1.4 Detalus sistemų palyginimas

1 lentelė Sistemų palyginimo lentelė

	FreshBooks	Saskaita123	i.APS
Kaina	Nuo 12.67 € iki 42.24 € per mėnesį.	Nuo 6,99 € iki 11,99 € per mėnesį.	Nemokama.
Funkcionalumas	Puikus. Daug įvairių funkcijų, kurios aktualios laisvai samdomiems asmenims.	Geras. Pagrindinės funkcijos aktualios laisvai samdomiems asmenims.	Prastas. Tik kelios pagrindinės funkcijos. Visiškai trūksta mokesčių skaičiavimo.
Mokesčių skaičiavimas	Nėra.	Paprastas. Nėra galimybės pasirikti mokesčių mokėjimo lengvatų.	Nėra. Planuojama įdiegti.
Vartotojo sąsajos intuityvumas	Geras. Modernus dizainas padeda rasti reikiamą funkciją.	Puikus. Modernus dizainas padeda itin lengvai rasti reikiamą funkciją iš nedidelio pasirinkimo.	Prastas. Vartotojo sąsaja pilna industrijoje nenaudojamų lietuviškų terminų. Dizainas neturi atitikmenų kitose industrijos sistemose.
Pasitikėjimas duomenų apsauga	Sena ir didelė įmonė su nepriekaištinga reputacija, turi pakankamai resursų, kad apsaugotų savo vartotojus.	Nedidelė ir sąlyginai nauja įmonė, kuri turi nedaug resursų vartotojo apsaugos užtikrinimui.	Valstybinės institucijos inicijuotas ir prižiūrimas projektas, kuris laikosi aukščiausių apsaugos standartų.

Sistemos pasiekiamumas per mobiliuosius įrenginius	Puikus. Sukurta mobili programėlė siūlo panašų funkcionalumą į svetainės versiją, bet kartu ir prideda patogių funkcijų: čekių skanavimą, nuvažiuotų kilometrų sekimą, naujienas.	Prastas. Sistema naudojantis per mobilųjį įrenginį yra patogi tik apskaitos peržiūrai. Kurti įrašus, peržiūrėti statistiką sudėtinga dėl sąsajos elementų neprisitaikymo prie įrenginio ekrano dydžių.	Vidutinis. Prisijungimo procesas vyksta per seną VMI svetainę, kuri visiškai nepritaikyta mobiliesiems įrenginiams. Prisijungus sistema puikiai prisitaiko prie ekrano dydžio.
	FreshBooks	**Sąskaita123**	**i.APS**
Asmenys kurie gali naudotis sistema	Visi asmenys. Sistema tinka tiek individualiai veikla užsiimantiems asmenims, tiek įmonėms.	Visi asmenys. Sistema tinka tiek individualiai veikla užsiimantiems asmenims, tiek įmonėms.	Sistema gali naudotis tik Lietuvos Respublikos piliečiai, užsiimantys individualia veikla pagal pažymą ar verslo liudijimą.
Galimybė integruoti sistemą su kitomis sistemomis.	Galima naudoti atvirą API, su kuriuo galima atlikti buhalterijos veiksmus iš kitų sistemų.	Nėra.	Nėra.
Galimybė prisijungti naudojant socialinius tinklus.	Prisijungti galima naudojant Google, Facebook paskyras ar susikuriant naują FreshBooks paskyrą.	Nėra galimybės.	Nėra galimybės.
Sistema prieinama lietuvių kalba.	Svetainė prieinama tik anglų kalba.	Taip.	Taip.

Iš lentelės matyti, kad geriausias pasirinkimas būtų FreshBooks. Tačiau Sąskaita123 gali pasiūlyti Lietuvoje individualia veikla užsiimantiems asmenims aktualias paslaugas. Taip pat galima teigti, kad ateityje i.APS turės konkurencingą funkcionalumą, kuris užtikrins itin tikslų mokesčių apskaičiavimą.

2.1.5 Apibendrinimas

Kiekviena iš analizuojamų sistemų turi savo privalumų ir trūkumų. FreshBooks tinka anglų kalbą mokančiam ir mokesčių skaičiavimą išmanančiam vartotojui, kuriam aktualus itin patogus apskaitos valdymas su itin plačiu funkcijų paketu. Sąskaita123 tinka vartotojui, kuris individualią veiklą vykdo Lietuvoje ir neturi mokestinių lengvatų. i.APS sistema tinka vartotojui, kuris nori vesti buhalteriją nemokamai ir jam užtenka minimalaus funkcijų paketo.

Projekto metu kuriama apskaitos sistema sieks aprėpti Lietuvoje individualia veikla užsiimantiems asmenims aktualias funkcijas. Buhalterijos sistema gebės generuoti sąskaitas-faktūras su vartotojo nurodytais duomenimis ir konvertuoti jas į PDF formatą. Šias sugeneruotas sąskaitas vartotojas galės nesudėtingai siųsti savo klientams elektroniniu paštu, bei nustatyti automatinius priminimus, jei klientai pamiršta apmokėti sąskaitas per nurodytą terminą. Vartotojas taip pat galės lengvai įsivesti patirtas išlaidas. Sistema, pagal išrašytas sąskaitas-faktūras, įvestas išlaidas ir pasirinktas lengvatas galės tiksliai apskaičiuoti mokėtinus mokesčius. Intuityvus dizainas leis nesudėtingai naviguoti sistemoje ir atlikti norimus veiksmus. Lengvesnei migracijai iš kitų buhalterijos sistemų, bus integruotas sąskaitų-faktūrų importavimas pasinaudojant CSV failą. Šio darbo metu kuriama sistema puikiai tiks vartotojui, kuris nori patogiai vesti buhalteriją ir būti užtikrintam, kad apskaičiuojami mokėtini mokesčiai yra tikslūs.

2.2 Tikslinio segmento poreikių analizė

Individualia veikla užsiimančiam asmeniui bene svarbiausias legalios, kokybiškos veiklos apsektas yra galimybė už suteiktas paslaugas paslaugų gavėjui išrašyti sąskaitą-faktūrą. Sąskaita įrodo paslaugos atlikimo ir pajamų gavimo faktą, todėl šis dokumentas turi atitikti tam tikrus priimtus aspektus, formatą. Atėjus valstybės inicijuojamo mokesčių deklaravimo periodui, toks vartotojas turi už gautas pajamas pasiskaičiuoti reikiamus mokėti mokesčius ir juos sumokėti valstybei. Dėl šios priežasties, prieš kuriant naują sistemą, turi būti apžvelgti šio tikslinio segmento pagrindiniai poreikiai – sąskaitos-faktūros išrašymo principas bei mokėtinų mokesčių apskaičiavimas.

2.2.1 Sąskaitos-faktūros analizė

Individualia veikla besiverčiantis asmuo po paslaugų atlikimo privalo paslaugų gavėjui (fiziniam asmeniui) pateikti sąskaitą-faktūrą arba PVM sąskaitą-faktūrą, jeigu šiam reikalinga

mokėti pridėtinės vertės mokestį. Šiam dokumentui yra keliami tam tikri turinio reikalavimai, kurie aiškiai nurodo dviejų – paslaugos teikėjo ir paslaugos gavėjo – rekvizitus ir įrodo pinigų gavimo faktą.

Tiek sąskaitos-faktūros, tiek PVM sąskaitos-faktūros rekvizitams yra keliami vienodi, tačiau abiem atvejais privalomi reikalavimai. Sąskaitą-faktūrą yra privaloma išrašyti net ir tuo atveju, jeigu paslaugos gavėjas nepageidauja jos gauti. Vadinasi, individualia veikla besiverčiantis vartotojas privalo po kiekvienos suteiktos paslaugos išrašyti šį darbų atlikimą įrodantį dokumentą. Tiek apmokestinamos pridėtinės vertės mokesčiu, tiek neapmokestinamos veiklos sąskaitoje turi būti nurodomi tokie rekvizitai [8]:

- Apskaitos dokumento pavadinimas;
- Apskaitos dokumento data;
- Pardavėjo, t. y. surašiusiojo (PVM) sąskaitą-faktūrą, pavadinimas, kodas;
- Pirkėjo vardas, pavardė, adresas;
- Prekės/paslaugos pavadinimas;
- Mato vienetai, kaina ir suma.

Kuriant apskaitos sistemą, visi šie kriterijai buvo įtraukti, kad paslaugą atlikęs tikslinio segmento vartotojas galėtų teisiškai tvarkingai vesti gaunamas pajamas.

2.2.2 Mokesčių analizė

Individualia veikla užsiimantiems asmenims mokesčiai susideda iš kelių dalių: gyventojų pajamų, privalomo sveikatos draudimo ir valstybinio socialinio draudimo mokesčių. Kadangi šiuos mokesčius kontroliuoja skirtingos institucijos, juridiniams asmenims gali būti sudėtinga rasti reikiamą mokėtinų mokesčių informaciją ar sekti visų mokesčių pasikeitimus. Šiai problemai dalinai išspręsti Sodra sukūrė „Individualios veiklos skaičiuoklę“, kuri priima tokius parametrus kaip: gautos pajamos, patirtos išlaidos, įvairių lengvatų įvertinimas ir nurodo tiksliai kiek ir kokių mokesčių reikia sumokėti atitinkamoms institucijoms [9]. Tačiau nei viena populiari buhalterijos sistema nėra įsidiegusi panašios skaičiuoklės ir negali pasiūlyti tikslaus mokesčių apskaičiavimo.

2.2.2.1 Gyventojų pajamų mokestis

Gyventojų pajamų mokesčio (toliau GPM) mokėtojai, vykdantys individualią veiklą, dalinami į dvi dalis: užsiimantys *laisvosiomis profesijomis* ir kiti, kurie nėra įtraukti į šių profesijų sąrašą. Jei asmens veikla patenka į laisvosios profesijos apibrėžimą, tuomet GPM sudaro 15 %

nuo apmokestinamo pelno, kitu atveju GPM sudaro 5% [10]. Asmenys, kurie patenka į 5 % ribą yra taip pat skirstomi į tris grupes [11]:

- Uždirbantys iki 20000 eurų apmokestinamo pelno per metus – GPM tarifas yra 5 %;
- Uždirbantys nuo 20000 iki 35000 apmokestinamo pelno per metus – GPM tarifas pagal formule apskaičiuojamas ir gaunamas nuo 5 % iki 15 %;
- Uždirbantys daugiau nei 35000 apmokestinamo pelno per metus – GPM tarifas yra 15 %.

Apmokestinamas pelnas skaičiuojamas iš pajamų atimant *leidžiamuosius* ir kitus atskaitymus [12]. Šis mokestis mokamas Valstybinei mokesčių inspekcijai (VMI).

2.2.2.2 Privalomasis sveikatos draudimas

Valstybė draudžia privalomuoju sveikatos draudimu (toliau PSD) asmenis, vykdančius individualią veiklą. Šis privalomasis mokestis apskaičiuojamas minimalų mėnesinį atlyginimą (toliau MMA) dauginant iš 6,98 % ir dauginant iš 12 (mėnesiai metuose) [13] arba iš apmokestinamųjų pajamų atėmus išlaidas, šias padauginus iš 90 %, o tuomet iš 6.98 % (mokama ta suma, kuri yra didesnė). Tačiau valstybė yra nustačiusi lengvatų asmenims, už kuriuos jau yra mokamas PSD (pvz.: dirbantiems pagal darbo sutartį, valstybės tarnautojams, gaunantiems išmokas), ar draudžiamiems valstybės lėšomis (pvz.: studentams, asmenims, turintiems būtinąjį stažą senatvės pensijai gauti). Privalomo sveikatos draudimo lengvatas turintiems asmenims negalioja MMA * 6,98 % minimalaus PSD mokėjimo suma. Jie moka pagal antrąjį skaičiavimo būdą, nesvarbu, ar šis yra mažesnis nei pirmasis. Asmenys gaunantys pensiją neprivalo mokėti PSD įmokų. PSD mokestis mokamas Sodrai.

2.2.2.3 Valstybinis socialinis draudimas

Valstybinis socialinis draudimas (toliau VSD) yra privalomas mokestis, kuris susideda iš „<...> pensijų draudimo, ligos ir motinystės draudimo, draudimo nuo nedarbo, sveikatos draudimo ir draudimo nuo nelaimingų atsitikimų darbe“ [14]. Šio mokesčio, taip pat kaip ir PSD, įmokos bazė skaičiuojama nuo 90 % apmokestinamųjų pajamų atėmus išlaidas. Mokesčio tarifai:

- jei nekaupiama papildomai pensijai – 12,52 %,
- jei kaupiama papildomai pensijai ir mokama 2,1 % įmokos – 14,62 %,
- jei kaupiama papildomai pensijai su 3% įmokomis – 15,52 %.

Taip pat, VSD turi mokėjimo lubas (didžiausia galima įmoka). Papildomai nekaupiant pensijos, lubos siekia 6683,20 €, kaupiant papildomai su 2 % įmokomis – 7804,19 €, o kaupiant

papildomai su 3 % įmokomis – 8284,61 € [15]. Mokesčio gali nemokėti tie asmenys, kurie gali nemokėti ir PSD (galioja tokie patys reikalavimai) ir asmenys, kurie vykdo individualią veiklą pirmą kartą nuo 2018-01-01 [16].

2.2.2.4 2021 metų pasikeitimai lyginant su 2020 metais

Didžiausią įtaką 2021 metų mokesčių apskaičiavimui turėjo minimalaus mėnesinio atlyginimo (MMA) padidėjimas. 2020 metais MMA buvo lygus 607 eurams, tačiau nuo 2021 metų MMA buvo padidintas iki 642 eurų [17]. Kadangi MMA yra naudojamas Sodros mokesčiams apskaičiuoti (PSD ir VSD), tad atitinkamai pasikeitė ir šių mokesčių tarifai. Taip pat paaugo ir papildomos pensijos mažiausios įmokos procentinė išraiška nuo 2,1 % iki 2,4 %, bet didžiausia, 3 % įmoka, liko nepakitusi. Pakilo ir PSD mokesčio maksimali suma nuo 3725,94 eurų iki 4059,99 eurų.

2.2.3 Išvados

Apibendrinant galima teigti, kad individualia veikla užsiimančiam asmeniui yra itin aktualūs dveji buhalteriniai kriterijai – sąskaitos-faktūros, atitinkančios joms keliamus specifinius reikalavimus, bei mokėtinų mokesčių tikslus apskaičiavimas.

Kiekvienas šios tikslinės grupės vartotojas privalo po paslaugos atlikimo išrašyti paslaugos atlikimą bei darbo atlygį įrodančią sąskaitą-faktūrą, net jeigu paslaugos gavėjas jos ir nepageidauja gauti. Kiekvienoje sąskaitoje-faktūroje yra nurodomi abiejų šalių rekvizitai bei įvardijamos atliktos paslaugos bei jų kainos.

Individualia veikla besiverčiančiam asmeniui yra skaičiuojami trys skirtingi mokesčiai: gyventojų pajamų mokestis, privalomas sveikatos draudimas ir valstybinis socialinis draudimas. Norint įdiegti sistemoje veiksmingą ir naudingą mokesčių skaičiuoklę, reikia įvertinti ne tik kiekvieno mokesčio tarifus, bet ir asmens turimas įvairias lengvatas, kurios gali iš esmės pakeisti mokamų mokesčių galutinę sumą.

Šio darbo metu kuriamoje sistemoje yra integruojamos tiek teisinius reikalavimus atitinkančios sąskaitos-faktūros, tiek ir trys aktualūs mokesčiai, apskaičiuojami nuo išrašytų sąskaitų apmokamos sumos. Apmokama sąskaitų suma gali būti apskaičiuojama dviem būdais: iš galutinės sąskaitų-faktūrų sumos atimamas 30 procentų (kaip teorinės išlaidos) arba iš galutinės sąskaitų-faktūrų sumos atimamos faktinės išlaidos, kurios galės būti įvestos į sistemą. Toks tikslinių vartotojų poreikių išpildymas užtikrins, kad sistema bus itin aktuali šia veikla užsiimantiems žmonėms.

2.3 Sistemos programavimo įrankių analizė

Norint sukurti internetinę sistemą, galima rinktis iš galybės įvairių įrankių, karkasų ir programų, kurie būtų įkomponuoti į galutinį produktą. Tokio produkto kūrimui yra privaloma rinktis programavimo įrankių rinkinį, pritaikytą internetinių sprendimų kūrimui. Visų pirma, reikia pasirinkti programavimo kalbų rinkinį, kurios leistų kurti tiek vartotojo sąsają, tiek ir valdyti duomenų apdorojimą serveryje. Taip pat, siekiant paspartinti sistemos kūrimą, dažniausiai yra naudojami programavimo kalbų karkasai, kurie jau turi išpildytus standartinius, esminius kiekvienai internetinei sistemai reikalingus funkcionalumus. Tačiau, pasirinkimo variantų, kurie būtų tinkami sistemos kūrimui yra gana daug, todėl reikia išsiaiškinti, kokie įrankių rinkiniai būtent tam tikram produktui geriausiai tinka.

Kokybiškai ir interaktyviai internetinei apskaitos sistemai reikia įrankių, kurie leistų ne tik kokybiškai atvaizduoti informaciją vartotojui, bet ir sklandžiai ją apdorotų serveryje. Šiam tikslui bus naudojami įvairūs programavimo karkasai, kurie palengvina programavimo darbą, pridėdami optimizuotą ir puikiai veikiantį funkcionalumą, kuriuo programuotojas gali pasinaudoti, norėdamas sukurti visiškai unikalią sistemą. Tačiau retai kada galima išsiversti tik su vienu karkasu, nes šie dažniausiai būna skirti tik vienam specifiniam programos aspektui: serverio valdymui, vartotojo sąsajos kūrimui ar duomenų bazės valdymui. Šiame skyriuje bus nagrinėjamai ir pasirenkami reikiami programiniai karkasai numatytai sistemai kurti.

2.3.1 PHP

PHP - tai atviro kodo programavimo kalba, skirta apdoroti informaciją serveryje [18]. 2020 metų duomenimis, pateiktais „*W3Techs“* puslapyje, nurodoma, kad net 79,1 % visų svetainių, kurios atlieka duomenų apdorojimą serveryje, naudoja PHP [19]. Specifinio funkcionalumo ar senesnio sukūrimo laikotarpio sistemos dar kartais naudoja Java, Ruby ar kitas programavimo kalbas. Taip pat, šiuo metu sparčiai populiarėja JavaScript (toliau JS) programavimo karkasai. Nors JS programavimo kalba buvo skirta kodo vykdymui vartotojo pusėje, tačiau itin populiarūs karkasai (Vue.js, Angular ar React) sugeba valdyti ir serverio kodo dalį.

Todėl, dėl itin didelio ir plataus naudojimo bei puikaus pritaikymo veikimui serveryje, apskaitos sistema bus kuriama naudojant PHP programavimo kalbą.

2.3.2 Laravel ar Symfony

Išsirinkus PHP programavimo kalbą, galima pradėti programuoti sugalvotą svetainę. Tačiau, naudojant tik PHP kalbą, reikėtų įdėti labai daug bereikalingo darbo tokią sistemą suprogramuoti, nenaudojant jokių pagalbinių funkcijų. Žinoma, tokia programa galėtų būti puikiai pritaikyta iškeltiems reikalavimams, tačiau tam reiktų itin plataus spektro žinių, kad sukurta sistema neturėtų kritinių problemų. Šiai problemai spręsti, labai populiarus sprendimas yra naudoti programavimo karkasus, kurie leidžia pasinaudoti jau optimizuotomis funkcijomis, kurios yra saugios ir patikrintos daugelio kitų programuotojų.

Dažniausiai yra naudojami du populiarūs PHP karkasai: Laravel ir Symfony. Abu karkasai siūlo puikų funkcionalumą, tačiau Laravel yra gerokai populiaresnis - „*Built With*“ duomenimis, turėdamas virš 600 tūkstančių aktyvių svetainių [20], naudojančių šį karkasą, Laravel gerokai lenkia Symfony, kuri turi virš 50 tūkstančių aktyvių svetainių [21]. Toks ryškus vartotojų naudojimo skirtumas gali būti paaiškintas tuo, kad Laravel yra laikomas naujesniu karkasu, kuris integruoja nemažai Symfony funkcionalumo, pridėdamas savo unikalių funkcijų.

Abu pasirinkimai remiasi MVC (*model, view, controller*) struktūra [22] [23], kuomet vartotojas svetainėje mato „vaizdą“ (angl. *view*), kuriame gali įvesti norimą informaciją. Tuomet ši informacija perduodama apdorojimui – kontroleriui (angl. *controller*) - kuris kreipiasi į modelį (angl. *model*), norėdamas gauti tam tikros informacijos iš duomenų bazės. Modelis reikiamą informaciją grąžina kontroleriui, o šis perduoda jau apdorotą informaciją vartotojui peržiūrai atgal į *view*. Tokia sistemų struktūra yra plačiai paplitusi tarp programuotojų, nes leidžia projekto kodą išlaikyti tvarkingą ir lengvai suprantamą.

Laravel taip pat turi integruotą, programuotojų tarpe pagarsėjusį, duomenų bazės objektų vaidmenų modeliavimo (angl. *object-relational mapper*, trumpinys *ORM*) įrankį Eloquent, kuris MVC struktūroje įprasmina modelio sąvoką, kaip duomenų bazės lentelės reikšmę [24]. Šis modulis leidžia duomenų bazėje esančius įrašus pasiekti, redaguoti ar pašalinti ir sukurti naujus, naudojant objektinį programavimą (OOP).

Didelis vartotojų skaičiaus skirtumas ir papildomi moduliai lemia ir tai, kad informacijos įvairiais klausimais apie Laravel yra gerokai daugiau nei apie Symfony. Dėl visų šių aptartų priežasčių, sistema bus kuriama naudojant Laravel karkasą.

2.3.3 Livewire ar Vue.js

Norint sukurti kokybišką, unikalią ir interaktyvią vartotojo sąsają, taip pat logiška pasirinkti karkasą, kuris valdys visą vartotojo sąsajos sąveiką. Šiuo atveju, geras sprendimas būtų

naudoti Vue.js JavaScript kalbos karkasą, kuris yra itin plačiai naudojamas kartu su Laravel karkasu. Tačiau, kaip jau minėta anksčiau, Vue.js iš esmės skirtas naudoti kaip vienas ir vienintelis karkasas, kuris valdytų tiek serverį, tiek ir vartotojo sąsają. Tad šios sistemos kūrimo atveju, kuomet serverio veiklą palaiko Laravel, logiškesnis pasirinkimas būtų rinktis visiškai naują, vos 2020 vasario mėnesį išleistą, Livewire karkasą, kuris buvo specialiai sukurtas naudojimui kartu su Laravel. Šio karkaso didžiausias skirtumas yra tas, kad šis bando perkelti JS karkasų funkcionalumą į PHP. Vartotojo sąsajos elementų pakeitimai yra dinamiškai siunčiami į serverį ir yra grąžinamas naujas elementas AJAX pagalba. Taip pat, šis elementas atnaujinamas dinamiškai, neperkaunant puslapio. Toks būdas kelia problemų, kuomet reikia atnaujinti daug elementų ar tiesiog norint sumažinti serverio užklausų kiekį. Šiai problemai išspręsti buvo sukurtas visiškai atskiras JavaScript karkasas *Alpine JS*, kuris skirtas naudoti kartu su Livewire mažų ir tik vizualinių elementų atvaizdavimui ir valdymui. *Alpine JS* naudojimas yra visiškai neprivalomas, o jis yra pakankamai mažas, kad į programą jį įkelti užtenka tik su internetine nuoroda.

Tad, naudojant Laravel ir Livewire, praktiškai viską galima suprogramuoti - tiek ir serverio tiek ir vartotojo sąsajos dalis, naudojant PHP, HTML ir CSS, minimizuojant JS naudojimą. Taip galima nesiblaškyti tarp visiškai skirtingų programavimo kalbų ir programuoti viena.

2.3.4 Tailwind ar Bootstrap

Išsirinkus vartotojo sąsajos valdymo karkasą, reikia nuspręsti, kaip bus kuriami patys sąsajos elementai. Šiuos elementus galima kurti naudojant tik HTML ir CSS, tačiau šiuo metu yra nusistovėjusios vartotojo sąsajos dizaino normos, tad galima sutaupyti laiko ir naudoti CSS karkasus. Du populiariausi CSS karkasai yra Bootstrap ir Tailwind. Šie yra ganėtinai skirtingi ir palengvina kūrimo procesą kitokiais aspektais. Bootstrap yra laikomas CSS ir JS komponentų karkasu, kuris siūlo itin paprastai naudoti jau sukurtus ir patrauklius dizaino elementus. Tačiau praktiškai visos svetainės, sukurtos naudojant Bootstrap, gali atrodyti šabloniškai ir vienodai, nes Bootstrap komponentus nėra paprasta redaguoti ir pritaikyti individualiai stilistikai. Kita problema, kurią kelia jau pasirinkti programavimo įrankiai, yra ta, kad Bootstrap dinaminiams elementams naudoja JavaScript. Norint kaip įmanoma labiau minimizuoti JS naudojimą, logiškesnis sprendimas yra naudoti kitą CSS karkasą – Tailwind. Šis, skirtingai nei Bootstrap, siūlo visus norimus elementus susikurti patiems, naudojant itin universalią CSS klasių sistemą, kuri leidžia praktiškai pašalinti CSS rašymą, o elementams naudoti tik Tailwind klases. Taip pat, Livewire turi puikią integraciją su Tailwind, tad kuriami elementai galės būti puikiai pritaikyti ir interaktyvūs.

Tailwind siūlo puikią integraciją su jau pasirinktu Livewire karkasu, bei leidžia sukurti sistemos dizainą, kuris nebus panašus į kitas sistemas.

2.3.5 Sinchroninis ar asinchroninis darbų vykdymas

Norint sistemoje patikimai valdyti ir apdoroti laiko ir sistemos resursų atžvilgiu intensyvias užklausas, reikia integruoti asinchroninių darbų vykdymo eigą. Didžiausias darbų vykdymo asinchroniškai privalumas – galimybė šiuos darbus atlikti fone (angl. *background*), o tuo pat metu vartotojas gali tęsti darbą sistemoje nesulaukęs užregistruoto darbo pabaigos.

Iliustruojant keliais pavyzdžiais, privalumai tampa labai aiškūs. Pavyzdžiui, bandant importuoti failą su keliais šimtais sąskaitų tam naudojant sinchroniška darbų metodą, visas funkcionalumas yra įvykdomas paeiliui: įkeliamas failas, failas apdorojamas ir pradedamas naujų įrašų sukūrimas. Sistema, skirdama visus savo resursus, bandytų įvykdyti šį darbą, o vartotojas tuo tarpu turėtų laukti kol importavimas būtų baigtas. Šiuo metodu atliekant laiko atžvilgiu intensyvius darbus kyla ir kita problema – tiek naršyklė, tiek ir kiti sistemos komponentai, negavę atsako iš sistemos per tam tikrą laiką, gali nuspręsti, kad sistema susidūrė su kritine klaida, ir nutraukti visą importavimo procesą.

Tikslesnis ir patikimesnis būdas atlikti šiuos darbus naudoti asinchroninius darbų paskirstymo metodus. Laravel karkasas turi puikiai integruotą modulį *Horizon,* kurio pagalba galima nesudėtingai valdyti asinchronines užklausas. Vartotojui pradėjus sąskaitų importavimą, šis darbas užregistruojamas į darbų eilę, kuri yra saugoma Redis duomenų bazėje. *Horizon,* pagal užklausų kiekį ir sudėtingumą, sukuria darbų paskirstytojus ir darbų vykdytojus. Paskirstytojai darbus ima paeiliui iš darbų eilės ir perduoda darbų vykdytojams apdorojimui. Kadangi darbai yra paskirstomi ir vykdomi fone, nesvarbu kokio sudėtingumo užklausa buvo pateikta, vartotojas ir toliau gali naudotis sistema, o naršyklė ar kitos sistemos dalys nenutrauks importavimo proceso.

2.3.6 Išvados

Sistema bus kuriama naudojant PHP programavimo kalbą. Darbo palengvinimui bus naudojamas Laravel karkasas ir jo moduliai. Vartotojo sąsaja bus kuriama ir valdoma su Livewire ir Tailwind moduliais, o sudėtingas ir laiko atžvilgiu intensyvias užklausas apdoros Laravel Horizon asinchroninių darbų modulis.

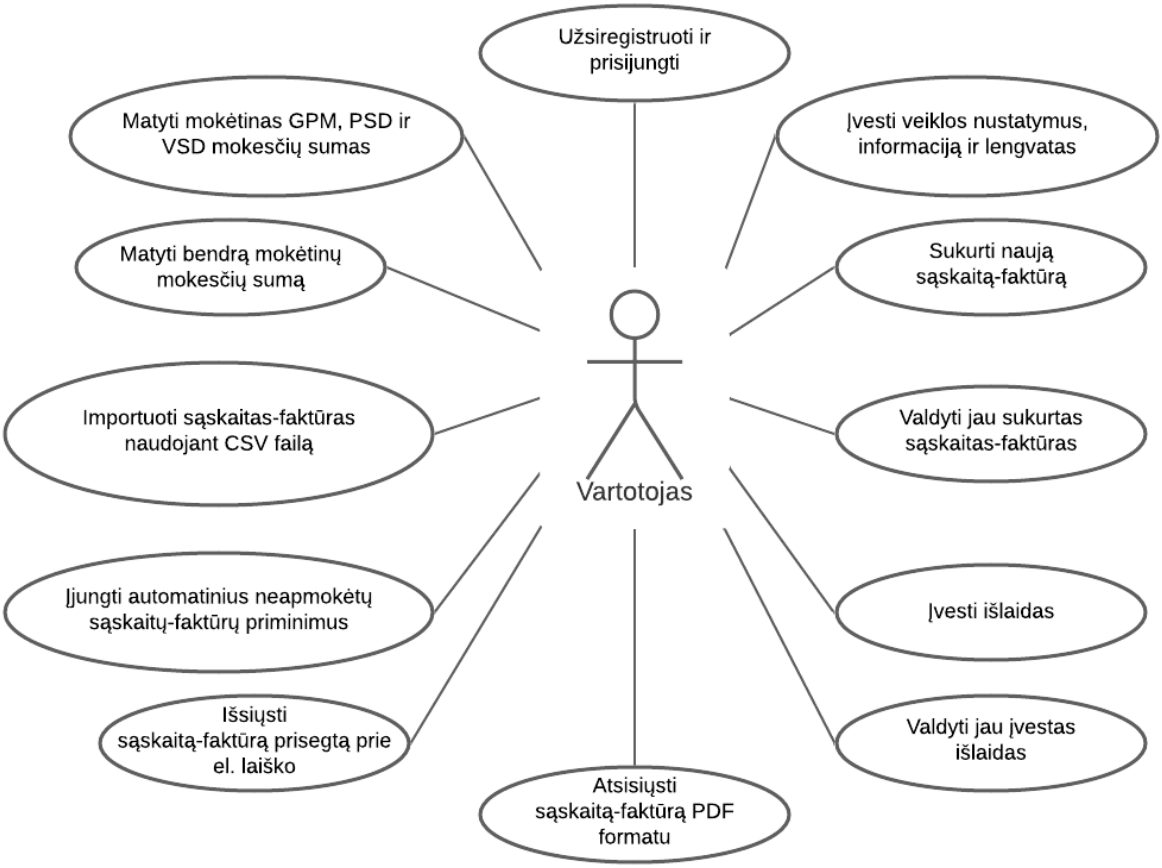
3. APSKAITOS SISTEMOS PROJEKTAVIMAS

Norint sukurti kokybišką ir funkcionalią sistemą, pradžioje reikia apgalvoti ir suprojektuoti reikiamus sistemos aspektus. Šiam tikslui buvo iškelti funkciniai sistemos reikalavimai, kurie aiškiai nurodo, kokio funkcionalumo sistemoje reikia. Taip pat buvo suprojektuota duomenų bazė, kuri leis išsaugoti reikiamą informaciją.

3.1 Sistemos funkciniai reikalavimai

Planuojamai kurti apskaitos sistemai buvo iškelti šie funkciniai reikalavimai (žiūrėti 4 pav.):

1. Vartotojų autorizacija ir registracija;
2. Vartotojo vykdomos veiklos informacijos pridėjimas ir atnaujinimas;
3. Vartotojo galimybė įvesti mokesčių lengvatas, pagal kurias bus atitinkamai skaičiuojami mokesčiai;
4. Vartotojo galimybė keisti sąskaitos-faktūros numerį bei kodą;
5. Vartotojo galimybė sukurti sąskaitą-faktūrą su neribotu kiekiu produktų;
6. Vartotojo galimybė peržiūrėti, redaguoti ar ištrinti jau sukurtas sąskaitas-faktūras;
7. Vartotojo galimybė įvesti išlaidas;
8. Vartotojo galimybė peržiūrėti, redaguoti ar ištrinti jau sukurtas išlaidas;
9. Vartotojo galimybė atsisiųsti sąskaitą-faktūrą PDF formatu;
10. Vartotojo galimybė tiesiai iš sistemos išsiųsti el. laišką su prisegta sąskaita-faktūra PDF formatu;
11. Vartotojo galimybė nustatyti sąskaitos-faktūros apmokėjimo priminimą, kuris būtų nusiunčiamas klientui artėjant apmokėjimo termino pabaigai;
12. Vartotojo galimybė importuoti sąskaitas-faktūras naudojant CSV failą;
13. Vartotojo galimybė matyti mokėtiną bendrų mokesčių sumą;
14. Vartotojo galimybė matyti gyventojų pajamų, valstybinio socialinio draudimo ir privalomo sveikatos draudimo mokesčių sumas.



4 pav. Funkcinių reikalavimų panaudos diagrama

3.2 Sistemos nefunkciniai reikalavimai

Planuojamai kurti apskaitos sistemai buvo iškelti šie nefunkciniai reikalavimai:

1. Neprisijungęs vartotojas gali pasiekti tik prisijungimo ir registracijos puslapius;
2. Sistemoje apskaičiuojamos mokesčių sumos turi sutapti su Sodros pateiktos individualios veiklos skaičiuoklės gautomis sumomis;
3. Sąskaitos-faktūros išrašymo forma turi atitikti WYSIWYG reikalavimus.

3.3 Duomenų bazės modelis

Iškart numatytas ir gerai apgalvotas duomenų bazės modelis gali gerokai palengvinti visos sistemos kūrimo darbus. Tačiau, norint sukurti tokį modelį, reikia įvertini visų sistemos funkcijų sąveikas su duomenų baze: kokią informaciją reikės saugoti, kaip šią informaciją susieti tarpusavyje bei kaip po to ją perduoti atvaizdavimui.

Kuriama sistema iš viso turės 13 duomenų bazės lentelių, iš kurių 7 bus skirtos saugoti vartotojų informaciją:

1. *invoices* (sąskaitų);
2. *expenses* (išlaidų);
3. *items* (produktų);
4. *users* (vartotojų);
5. *meta* (meta informacija);
6. *import_data* (importavimui skirti failai ir jų informacija);
7. *invoice_emails* (suplanuoti el. laiškai siuntimui).

Kitos lentelės bus skirtos Laravel karkaso informacijai saugoti: duomenų bazės migracijos, sesijos, slaptažodžių logika ir nepavykę darbai. Šios lentelės leidžia Laravel funkcionuoti ir išsaugoti reikalingą informaciją.

5 pav. Duomenų bazės lentelių diagrama

Users lentelėje (žiūrėti 5 pav.) saugomi duomenys apie registruotus vartotojus. Šioje lentelėje iš esmės kol kas bus saugojamas tik vartotojo vardas, el. paštas, slaptažodis, ir bus klausiama, ar vartotojas nori, kad būtų prisimintas prisijungimas bei kada jis užsiregistravo. Visi

kiti laukeliai bus automatiškai sugeneruoti Laravel. Šiuos laukelius bus galima išnaudoti vėlesnėse sistemos integracijose, kuomet reikės daugiau funkcionalumo.

invoices lentelėje saugoma visa pagrindinė išrašytų sąskaitų informacija:

- *user_id* – nurodo sąsają su *users* lentele ir nusako, kam priklauso išrašyta sąskaita;
- *sf_code* – sąskaitos faktūros automatiškai sugeneruotas unikalus kodas;
- *sf_number* – sąskaitos faktūros kodo skaitinė reikšmė, pagal kurią sistema generuos kitos sąskaitos numerį;
- *company_name* – įvestas įmonės pavadinimas, kuriai išrašyta sąskaita;
- *company_code* – įvestas įmonės kodas;
- *company_vat* – įvestas įmonės PVM kodas;
- *company_address* – įvestas įmonės adresas;
- *pay_by* – sąskaitos-faktūros apmokėjimo termino pabaigos data;
- *is_payed* – pažymi, ar sąskaita jau išsiųsta klientui;
- *created_at* – įrašo išrašymo data ir laikas;
- *updated_at* – įrašo atnaujinimo data ir laikas.

expenses lentelėje saugomos išrašytos išlaidos:

- *user_id* – nurodo sąsają su *users* lentele ir nusako, kam priklauso išrašyta išlaida;
- *number* – išlaidos unikalus numeris įrašytas vartotojo;
- *date* – išlaidos patyrimo data;
- *currency* – išlaidos valiuta;
- *seller_name* – įmonės pavadinimas, kuri išrašė išlaidą;
- *seller_code* – įmonės, kuri išrašė išlaidą, kodas;
- *seller_address* – įmonės, kuri išrašė išlaidą, adresas;
- *seller_vat* – įmonės, kuri išrašė išlaidą, PVM kodas.

import_data lentelėje saugomi duomenys apie failus, kurie yra naudojami importuojant naujus įrašus ir leidžia skirtingiems importavimo žingsniams lengviau komunikuoti tarpusavyje bei sekti progresą:

- *user_id* – nurodo sąsają su *users* lentele ir nusako, kam priklauso įkeltas failas;
- *csv_filename* – įkelto failo pavadinimas;
- *csv_header* – nurodo, ar įkeltas failas turi antraštę;

- *csv_data* – įkelto failo informacija;
- *type* – failo tipas;
- *is_imported* – nurodo, ar šis failas jau yra importuotas.

items lentelėje saugoma sąskaitoje įrašytų produktų ar paslaugų informacija:

- *name* – paslaugos pavadinimas;
- *unit* – paslaugos mato vienetas;
- *quantity* – paslaugos kiekis;
- *price* – paslaugos kaina;
- *itemable_type* – sistemos modelio pavadinimas, kuriam priskirtas šis įrašas (gali būti sąskaitos arba išlaidos modelis);
- *itemable_id* – sistemos modelio identifikacinis numeris, kuriam priskirtas šis įrašas (tai sąskaitos arba išlaidos *id* numeris).

meta lentelėje saugoma visa sistemoje generuojama meta informacija, kurios nėra tikslo skirstyti į atskiras lenteles dėl jos išskirtinumo ir retumo:

- *name* – meta informacijos pavadinimas;
- *value* – meta informacijos reikšmė;
- *metable_id* – modelio indeksas, kuriam priklauso ši meta informacija;
- *metable_type* – modelio tipas, kuriam priklauso ši meta informacija.

Šioje lentelėje saugojami vartotojų veiklos nustatymai, kuomet *name* yra nurodomas nustatymo pavadinimas, o *value* JSON formatu išsaugota nustatymų informacija. Kadangi įvairūs nustatymai turi visiškai skirtingus kiekius skirtingos informacijos, o sistemai ganėtinai retai reikia naudoti šias reikšmes, tad logiškiausia šia informaciją saugoti šiuo principu. Taip pat, ši lentelė visiškai neapriboja sistemos naujų funkcijų įdiegimui, pavyzdžiui, naujos nustatymų grupės įdiegimui, kadangi naujai informacijai užregistruoti tereikia sukurti naują įrašą.

invoice_emails lentelėje saugomi suplanuoti el. laiškai, skirti priminti vartotojo klientams apie dar neapmokėtas sąskaitas faktūras. Šią lentelę sudaro šie stulpeliai:

- *invoice_id* – sąskaitos faktūros *id* numeris, kuriai ir yra priskirtas šis priminimas;
- *due_date* – priminimo išsiuntimo data.

Kita matoma lentelė *sessions* yra sugeneruojama automatiškai Laravel, norint palaikyti vartotojų sesijų funkcionalumą. Ši lentelė turi priklausomybę *users* lentelei.

3.4 Vartotojo sąsajos projektavimas

Siekiant sukurti patogią, intuityvią, lengvą naudotis sistemą, reikalinga iš anksto numatyti, kaip atrodys būsimos sistemos vartotojo sąsaja. Buvo nuspręsta kurti minimalistinio dizaino, tik būtiniausius elementus turinčią vartotojo sąsają.

Bendra dizaino stilistika buvo panaudota iš Laravel JetStream modulio, kuris suteikė bendram sistemos vaizdui neapkrautą, dėmesio neblaškančią vartotojo sąsają. Pagrindiniame puslapyje buvo nuspręsta atvaizduoti tik svarbiausią vartotojo vykdomos veiklos informaciją:

- Išrašytų sąskaitų bendrą sumą;
- Suvestų veiklos sąnaudų bendrą sumą;
- Bendrą ir atskirai išskirtą mokėtinų mokesčių sumas.

Sąskaitų išrašymo formai buvo numatyta taikyti WYSIWYG (angl. *What You See Is What You Get*, lietuviškai – ką matai, tą ir gauni) programų kūrimo metodika. Taikant šią metodiką, formos dizainu yra stengiamasi kaip įmanoma tiksliau atkartoti jau sugeneruotos PDF sąskaitos dizainą. Tai leidžia vartotojui geriau suprasti ir įsivaizduoti, kaip atrodys jau išrašyta sąskaita-faktūra. Todėl jau sąskaitos išrašymo formoje galima aiškiai matyti, kokie bus būsimos sąskaitos serija ir numeris, išrašymo data, bei kokia vartotojo veiklos informacija bus matoma išrašytoje sąskaitoje.

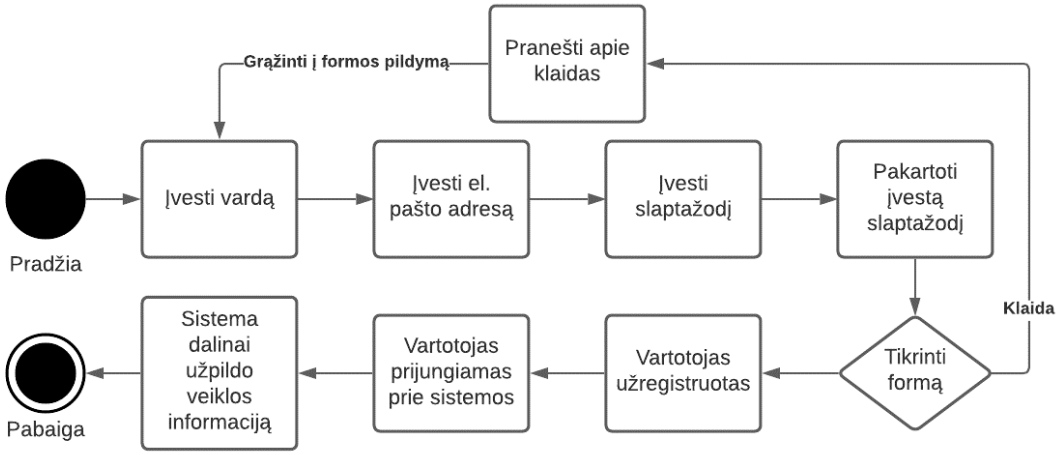
Įvestas sąskaitas ir išlaidas buvo nuspręsta atvaizduoti sąrašu, kuriame iš pirmo žvilgsnio būtų matoma visa pagrindinė su sąskaita ar išlaida susijusi informacija, o paspaudus ant konkrečios sąskaitos ar išlaidos sąrašo elemento būtų galima pamatyti visą su elementu susijusią informaciją. Toks atvaizdavimo būdas leidžia vartotojui lengvai atrasti reikiamą elementą ir sužinoti jo informaciją net ir esant ilgam sąskaitų ar išlaidų sąrašui.

3.5 Sistemos funkcinės logikos projektavimas

Sistemos kūrimui labai gelbsti veiksmų schemos, kuriose galima aiškiai aprašyti, kaip sistema turės apdoroti numatytas užduotis.

3.5.1 Registracija

Bet kuris vartotojas gali užsiregistruoti sistemoje. Jis tai gali padaryti suvesdamas savo vardą, el. pašto adresą ir slaptažodį. Jei duomenys suvesti teisingai, tuomet vartotojas yra užregistruojamas ir automatiškai prijungiamas prie svetainės (žiūrėti 6 pav.).

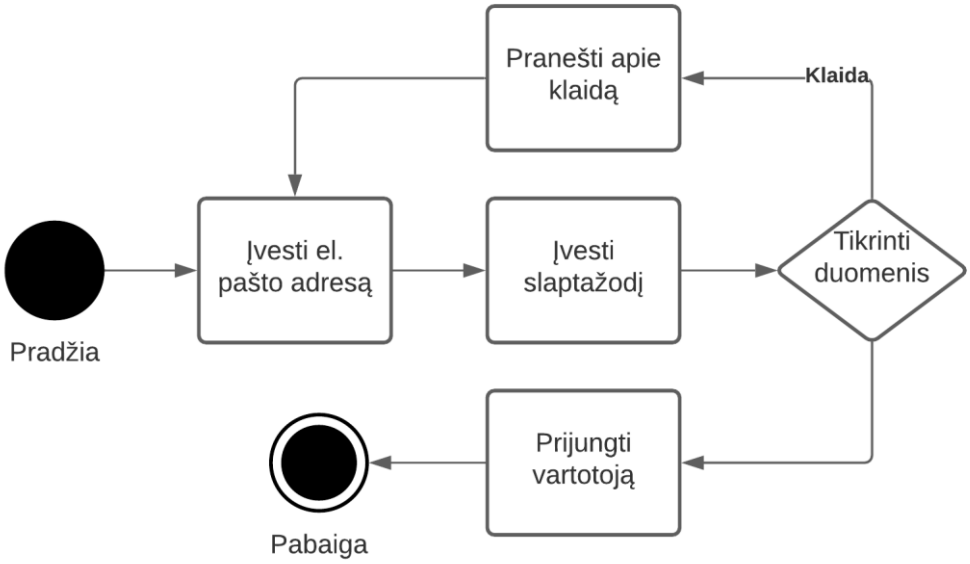


6 pav. Vartotojo registracijos veiklos diagrama

Registracijos formoje visi laukeliai yra privalomi užpildyti. Po sėkmingos registracijos, sistema automatiškai dalinai užpildo veiklos nustatymus su užregistruotu vardu ir el. paštu, nurodo įprastą sąskaitos priedėlį – *SF* kartu su sąskaitos numeriu - 1, nustato, kad vartotojas neturi lengvatų. Prisijungęs vartotojas šią informaciją gali redaguoti.

3.5.2 Prisijungimas

Prie sistemos prisijungti gali tik jau užsiregistravę vartotojai (žiūrėti 7 pav.). Vartotojai gali matyti tik savo informaciją ir negali pasiekti kitų išrašytų sąskaitų ar išsaugotų nustatymų.

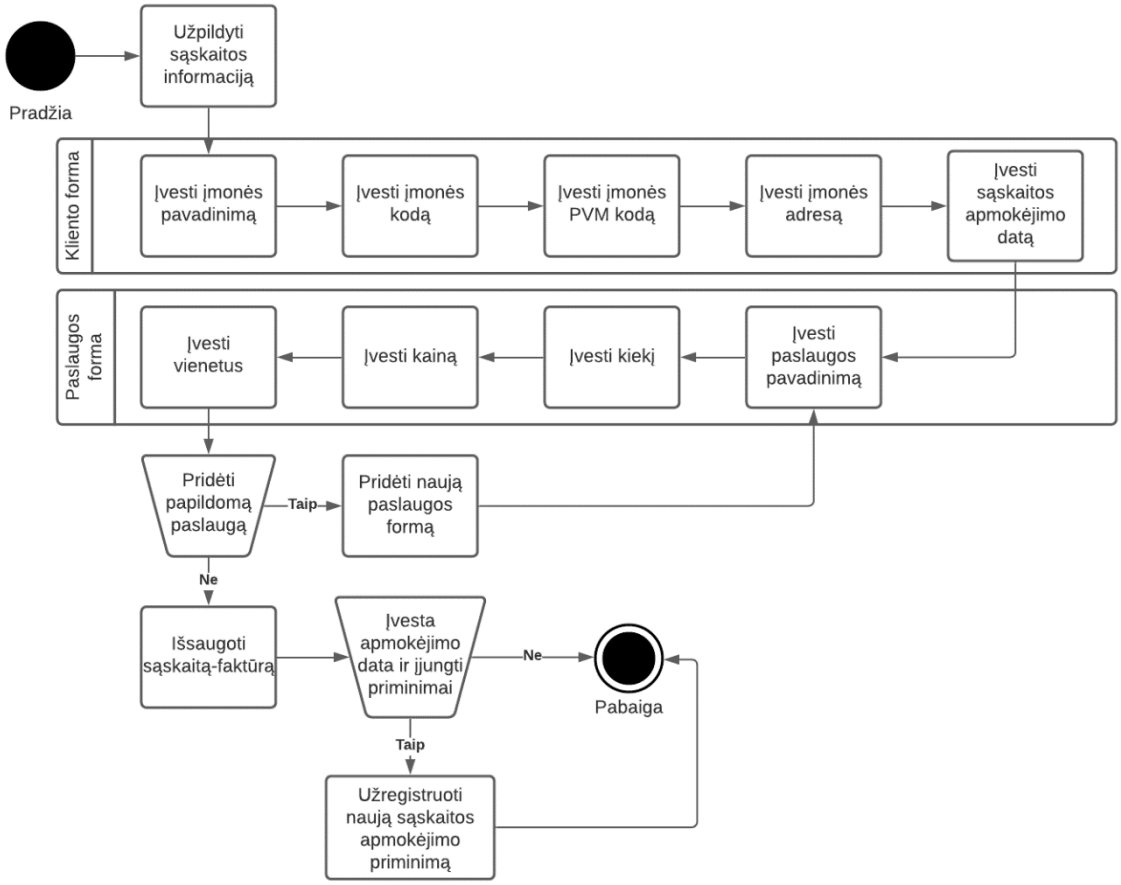


7 pav. Vartotojo prisijungimo veiklos diagrama

Vartotojas, norėdamas prisijungti, privalo įvesti teisingus prisijungimo duomenis, kuriuos naudojo registracijos etape. Jei vartotojas duomenis įveda klaidingai, tuomet sistema nurodo, kurie laukeliai įvesti blogai ir prašo vartotojo dar kartą įvesti prisijungimo duomenis.

3.5.3 Sąskaitos-faktūros sukūrimas

Prisijungęs vartotojas gali sukurti naują sąskaitą faktūrą (žiūrėti 8 pav.).

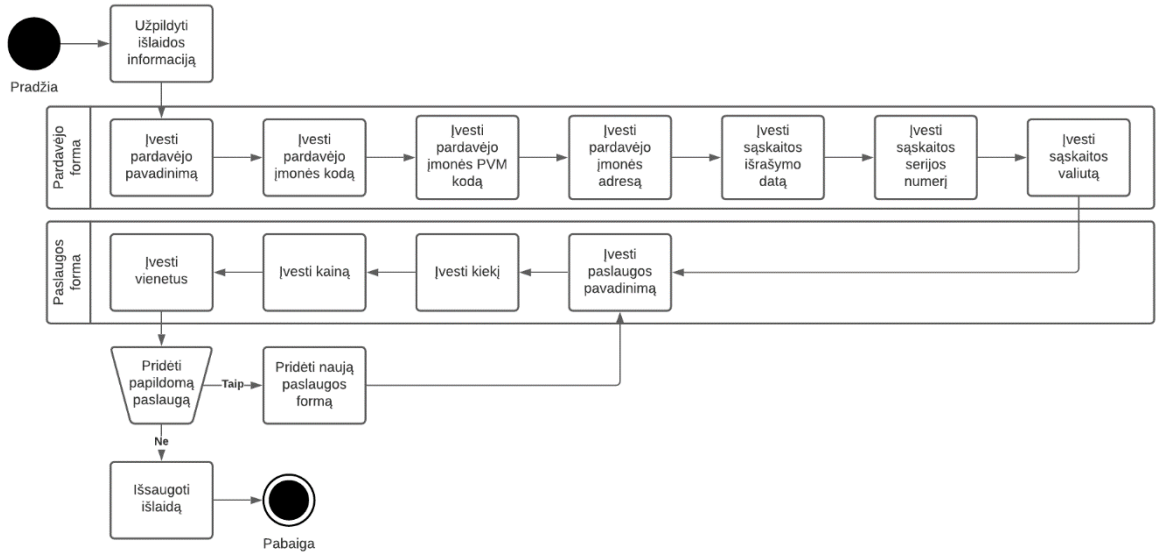


8 pav. Sąskaitos-faktūros sukūrimo veiklos diagrama

Vartotojas privalo įvesti kliento įmonės pavadinimą ir kodą įmonės informacijos formoje ir užpildyti visus paslaugos formos laukelius. Kitu atveju, sistema nesugeneruos naujos sąskaitos-faktūros. Vartotojas taip pat gali pasirinkti, kiek paslaugos įvedimo formų sistemai sugeneruoti. Formų skaičius neribojamas, tad vartotojas gali įvesti kiek nori paslaugų, kurių kainos bus sumuojamos galutinėje sąskaitoje. Vartotojui veiklos nustatymuose nurodžius, kad pageidauja siųsti sąskaitos-faktūros apmokėjimo priminimus klientams ir išrašomoje sąskaitoje įvedus apmokėjimo terminą, sistema užregistruoja naują priminimą, kuris automatiškai bus išsiųstas apmokėjimo datą, jei sąskaita anksčiau laiko nebus pažymėta kaip apmokėta.

3.5.4 Duomenų apie išlaidas pildymas

Vartotojas gali pridėti savo veikloje patirtas išlaidas sąskaitų-faktūrų forma (žiūrėti 9 pav.).

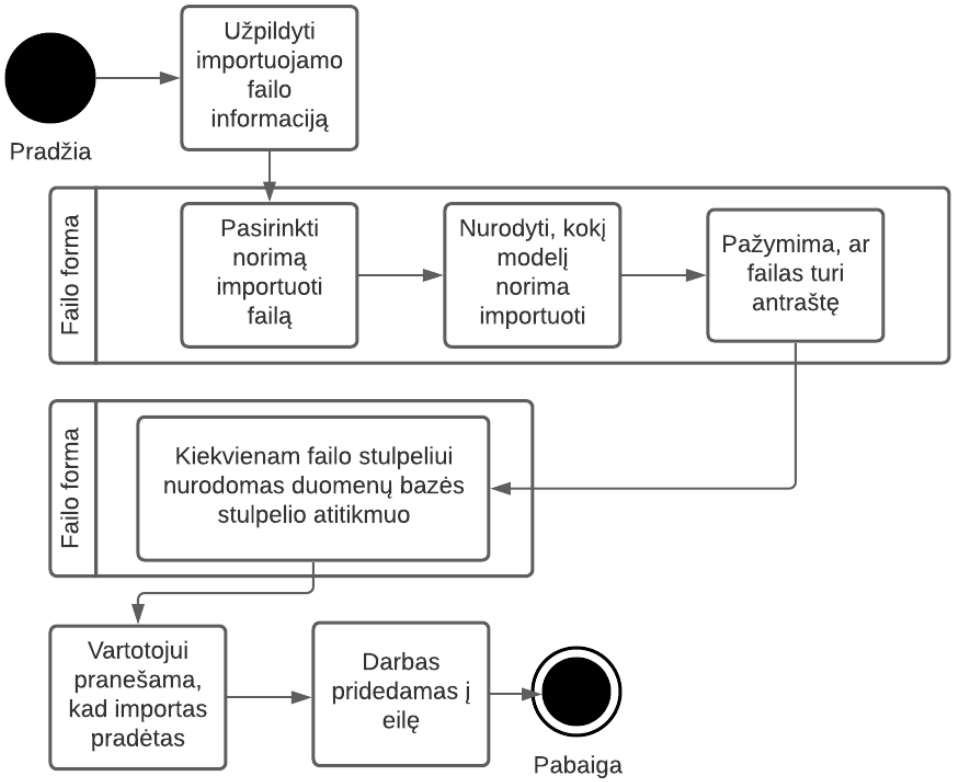


9 pav. Išlaidos įvedimo veiklos diagrama

Vartotojas privalo įvesti pardavėjo pavadinimą, kodą, sąskaitos išrašymo datą, sąskaitos serijos numerį, valiutą, bei pridėti išlaidos produktus, kurių galutinė suma bus susumuota ir taps išlaidos galutine suma. Ši suma bus naudojama mokesčių apskaičiavimui. Forma veikia tokiu pat principu kaip ir sąskaitos-faktūros įvedimo forma, tad integruotas toks pat funkcionalumas.

3.5.5 Sąskaitų importavimas

Vartotojas, pasinaudodamas CSV failu, gali nesudėtingai importuoti visas norimas sąskaitas-faktūras (žiūrėti 10 pav.).

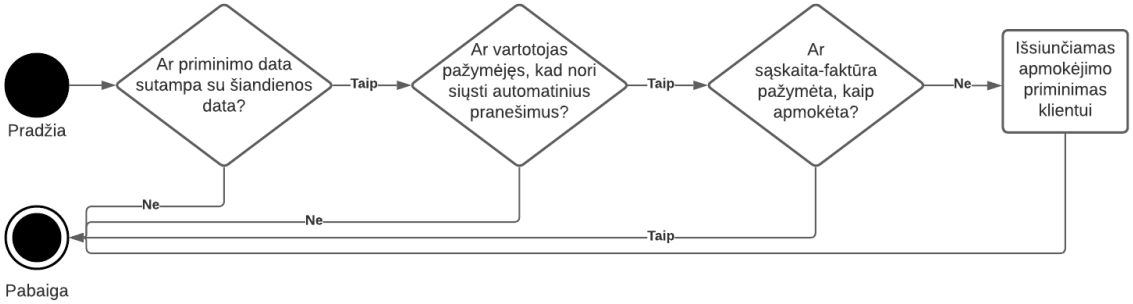


10 pav. Modelių importavimo veiklos diagrama

Vartotojas gali pasirinkti norimą importuoti CSV failą tiesiogiai iš savo kompiuterio. Tuomet jis turi nurodyti, kokią informaciją jis nori importuoti ir ar jo įkeltas failas turi antraštinę eilutę. Patvirtinus įkėlimą, sistema vartotojui atidaro iššokantį langą, kuriame jis gali matyti dvi pirmas savo failo eilutes, kurių stulpeliams turi nurodyti atitinkančius duomenų bazės stulpelius. Nepasirinkti stulpeliai yra praleidžiami ir neįrašomi į duomenų bazę. Pasirinkus norimus stulpelius ir nurodžius jiems reikiamas reikšmes, užregistruojamas naujas asinchroninis darbas, o vartotojui yra pranešama, kad importavimas vyks fone.

3.5.6 Sąskaitų apmokėjimo priminimai

Sistema gali automatiškai priminti sąskaitų gavėjams apie dar neapmokėtas sąskaitas-faktūras (žiūrėti 11 pav.).



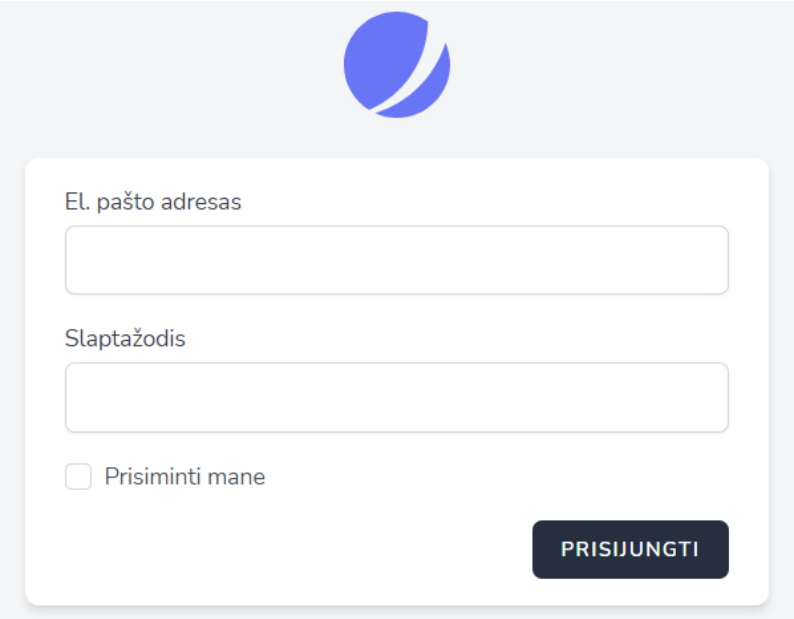
11 pav. Neapmokėtų sąskaitų priminimų siuntimo veiklos diagrama

Sistema kiekvieną dieną surenka visas neapmokėtas sąskaitas, kurių apmokėjimo terminas baigiasi tą dieną. Tuomet tikrinama, ar vartotojas, kuriam priklauso ši sąskaita, yra įjungęs savo veiklos nustatymuose automatinius neapmokėtų sąskaitų faktūrų priminimus. Tada tikrinama, ar sąskaita pažymėta kaip apmokėta. Sąskaitai sėkmingai praėjus visus sistemos patvirtinimus, ši, kartu su priminimu, yra išsiunčiama sąskaitoje nurodytam kliento el. pašto adresui.

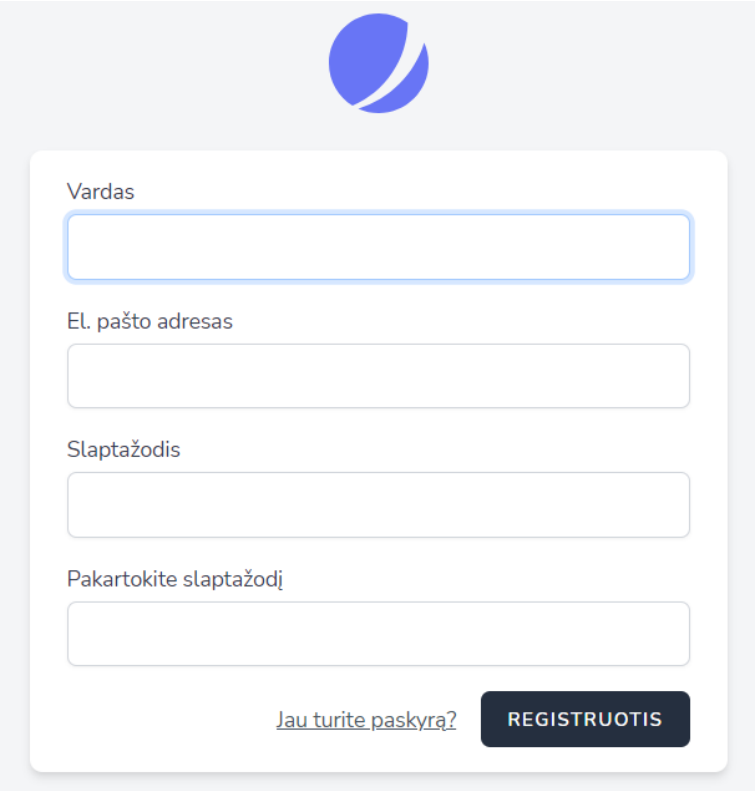
4. REALIZAVIMAS

Atlikus 2 skyriaus analitinę dalį, kurioje buvo išsiaiškinti individualia veikla užsiimantiems vartotojams aktualūs mokesčiai, išsirinkus, kokie programavimo įrankiai bus naudojami, bei atlikus 3 skyriuje įvardintus duomenų bazės ir funkcinius projektavimo darbus, buvo sukurta sistema, atitinkanti jai suformuluotus funkcinius ir nefunkcinius reikalavimus, aprašytus 3 skyriuje.

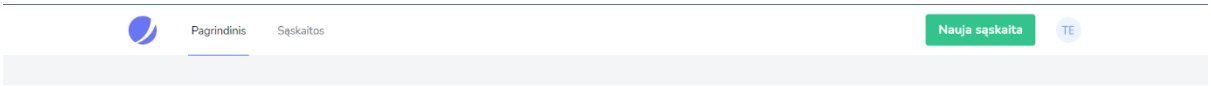
Sistemos kūrimo metu buvo naudota naujausia Laravel 8 versija, kuri pridėjo keletą naujų funkcijų, kurių nebuvo senesnėje Laravel 7 versijoje [25]. Pagrindiniai funkcijų skirtumai, kurie buvo naudojami kuriant sistemą - tai JetStream autentifikacijos ir minimalaus dizaino modulis, kuris pakeitė senąjį Laravel *Auth* modulį, bei dinaminiai Blade komponentai



12 pav. Vartotojo prisijungimo langas



13 pav. Vartotojo registracijos langas

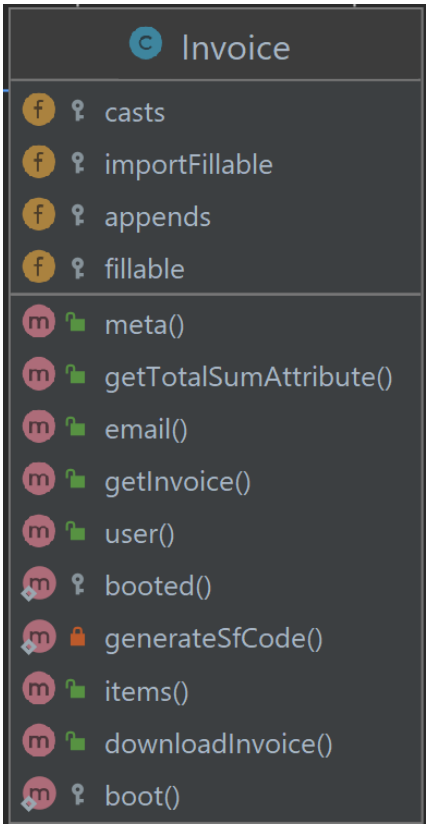


14 pav. Pagrindinis svetainės meniu

Vartotojų valdymui ir autorizacijai bei paprastam sistemos dizainui buvo panaudotas Laravel JetStream, kuris leido nesudėtingai integruoti vartotojų prisijungimą ir registraciją (žiūrėti 12 pav.), autorizacijos valdymą, o tuo pačiu suteikė galimybę vartotojams matyti minimalaus dizaino registracijos ir prisijungimo prie sistemos langus (žiūrėti 13 pav.), pagrindinio meniu dizainą (žiūrėti 14 pav.). Taip pat, Laravel JetStream užtikrino, kad vartotojai galėtų matyti tik jų sukurtą, privačią informaciją, kuri būtų tuo pačiu apsaugota nuo pašalinių asmenų prieigos.

4.1 Modeliai

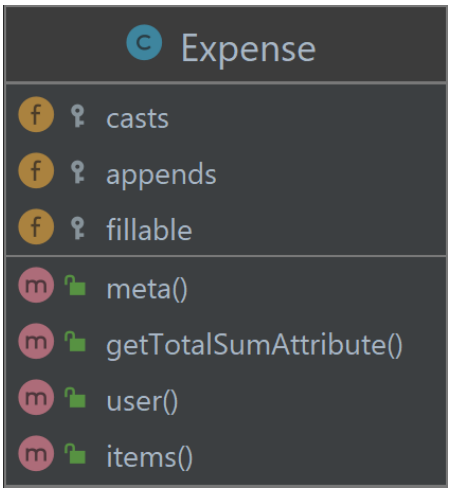
Visi sistemos failai buvo kuriami ir saugomi pagal Laravel numatytą tvarką. Buvo sukurti septyni Eloquent modeliai: *Invoice, Expense, Item, Meta, ImportData, InvoiceMail* ir *User*. Šie modeliai buvo atsakingi už atitinkamą duomenų bazės lentelės tvarkymą.



15 pav. *Invoice* modelis

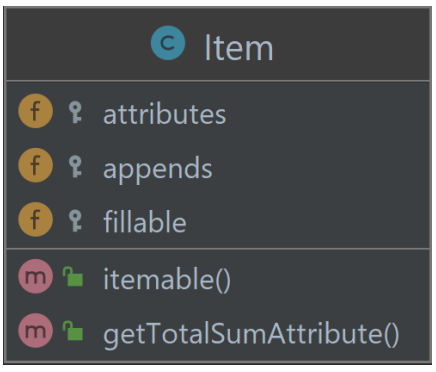
***Invoice* modelis** (žiūrėti 15 pav.), atsakingas už sąskaitų-faktūrų valdymą, sudarytas iš 10 skirtingų metodų ir keturių kintamųjų. *user, meta, email* ir *items* metodai nurodo priklausomybę atitinkamiems modeliams ir leidžia, pasinaudojus objektiniu programavimu, pasiekti šių modelių reikšmes, neatliekant tiesioginės paieškos duomenų bazėje. *getTotalSumAttribute* metodas grąžina bendrą sąskaitos-faktūros paslaugų kainų sumą. *getInvoice* metodas iššaukia *mPDF* modulį, kuriam perduoda visą sąskaitos informaciją, o šis grąžina jau sugeneruotą PDF failą. *downloadInvoice* veikia taip pat kaip ir *getInvoice* metodas, tačiau jau į naršyklę grąžina sugeneruotą PDF failą, kurį vartotojas gali išsisaugoti. *boot* metodas yra automatiškai iškviečiamas tik sukūrus šį modelį ir užregistruojama, kad modelio ištrynimo metu būtų kartu pašalinami ir *invoiceItems* modeliai iš duomenų bazės. *booted* metodas taip pat yra automatiškai iškviečiamas iškart po modelio sukūrimo ir jame aktyvuojamas *generateSfCode* privatus metodas, kuris sugeneruoja ir priskiria modeliui naują sąskaitos-faktūros numerį ir kodą. *fillable* kintamasis, kaip ir visuose kituose modeliuose, nurodo karkasui, kokios reikšmės modeliui gali būti priskirtos jo sukūrimo metu, o *importFillable* nurodo papildomas priskiriamas reikšmes sąskaitos faktūros importavimo metu kuriamoms sąskaitoms. *casts* nurodo karkasui, į kokį tipą automatiškai versti pateiktas reikšmes. *appends* kintamasis nurodo karkasui, kad šis turėtų automatiškai prie modelio informacijos pridėti kintamajame nurodytą laukelį, kuris yra automatiškai apskaičiuojamas

nurodytame metode. Šiuo atveju, *appends* kintamajame nurodyta pridėti *total_sum* laukelį, kuris Laravel karkaso yra automatiškai gaunamas iš jau anksčiau minėto *getTotalSumAttribute* metodo. Šioje vietoje, karkasas automatiškai randa sąsają tarp metodo ir kintamojo reikšmės pagal jų pavadinimų panašumą.



16 pav. *Expense* modelis

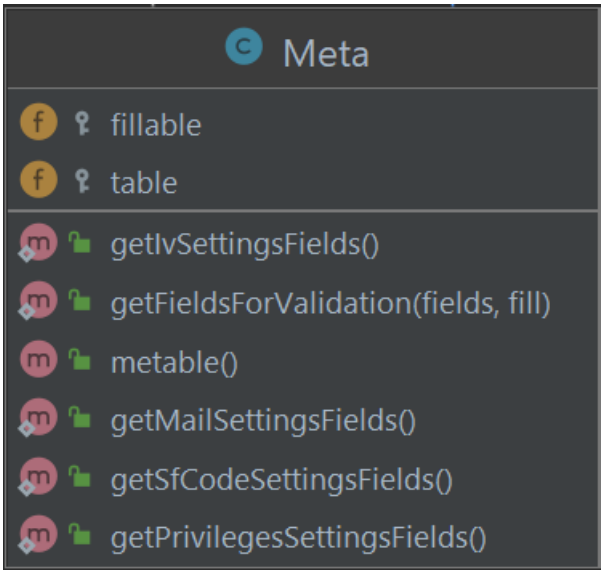
***Expense* modelis** (žiūrėti 16 pav.), atsakingas už išlaidų valdymą, sudarytas iš 4 metodų ir trijų kintamųjų. *Meta*, *user* ir *items* metodai nurodo išlaidų modelio priklausomybę nuo kitų, atitinkamų, modelių. *getTotalSumAttribute* metodas kartu su *appends* kintamuoju, lygiai taip pat kaip ir *Invoice* modelyje, Laravel pagalba prideda modeliui naują reikšmę, kuri yra apskaičiuojama minėtame metode.



17 pav. *Item* modelis

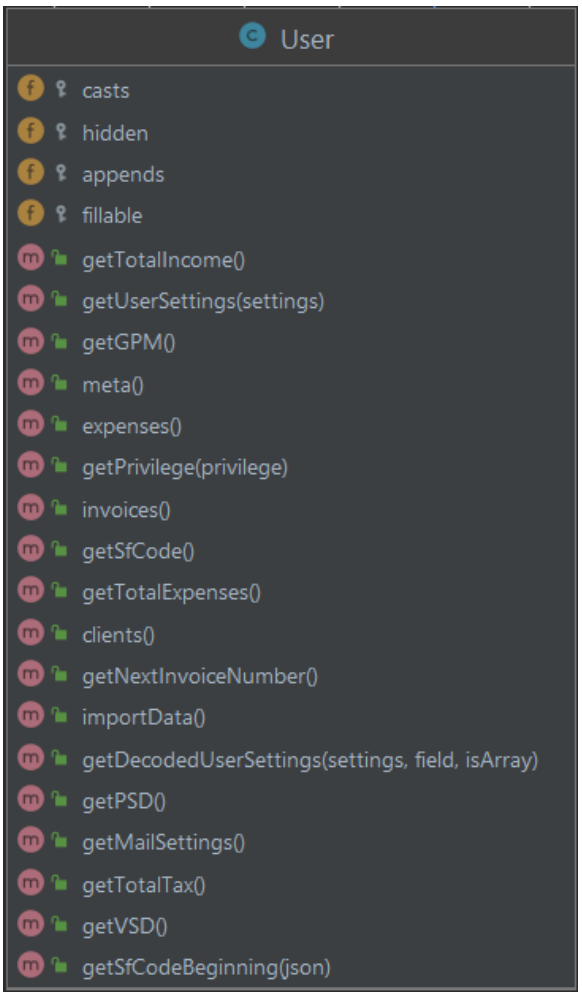
***Item* modelis** (žiūrėti 17 pav.), atsakingas už sąskaitos-faktūros ir išlaidų paslaugų valdymą, yra sudarytas iš dviejų metodų ir trijų kintamųjų. *itemable* metodas nurodo, kad šis metodas yra laikomas morfiniu ir leidžia kitiems metodams, turintiems *morphMany* ar kitą *morph* sąsają su šiuo metodu, išsaugoti ir skaityti savo paslaugos informaciją *item* lentelėje. *getTotalSumAttribute* metodas apskaičiuoja bendrą paslaugos sumą daugindamas paslaugos kiekį

iš paslaugos kainos. *attributes* kintamasis nurodo karkasui, kokios numatytos reikšmės turėtų būti išsaugotos duomenų bazėje, jei nėra nurodytos kitos.



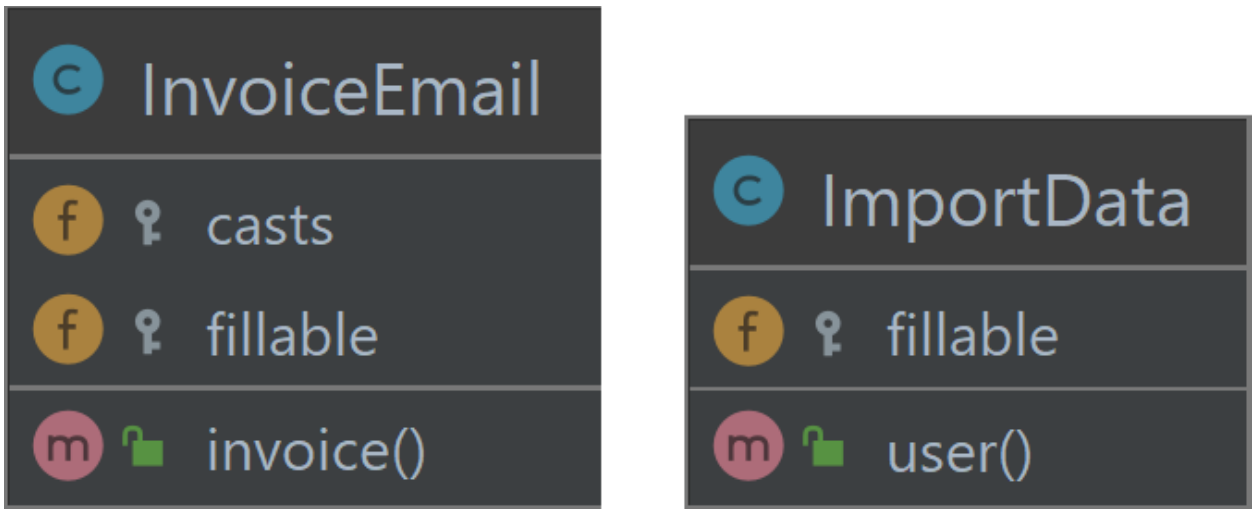
18 pav. *Meta* modelis

***Meta* modelis** (žiūrėti 18 pav.), atsakingas už meta informacijos valdymą, yra sudarytas iš 6 metodų ir dviejų kintamųjų. *metable* metodas nurodo, taip pat kaip ir *Item* modelio metodas *Itemable,* kad šis metodas yra laikomas morfiniu ir leidžia kitiems metodams, turintiems *morphMany* ar kitą *morph* sąsają su šiuo metodu, išsaugoti ir skaityti savo meta informaciją *meta* lentelėje. *getFieldsForValidation* metodas grąžina atitinkamą reikšmių sąrašą autentifikacijai, kurį jam perduoda atitinkamas vienas metodas iš visų likusių metodų. Šį metodą naudoja visi Livewire komponentai, kurie tikrina vartotojo įvestas reikšmes. Žodis *meta* neturi anglų kalboje daugiskaitos varianto su galūne s, tad karkasui, naudojant kintamąjį *table,* reikia nurodyti, kad šio modelio lentelė duomenų bazėje bus vadinama *meta,* o ne *metas.*



19 pav. *User* modelis

User **modelis** (žiūrėti 19 pav.), atsakingas už vartotojo informacijos valdymą, yra sugeneruotas automatiškai karkaso ir JetStream modulio, kurie prideda *fillable, hidden, casts* ir *appends* kintamuosius. Tačiau norint, kad vartotojas turėtų galimybę kurti sąskaitas-faktūras ir sekti mokesčius, reikia pridėti naujų metodų. *meta, expenses, importData* ir *invoices* metodai nurodo priklausomybės ryšį tarp atitinkamų modelių. *getUserSettings, getPrivilege, getMailSettings* ir *getDecodedUserSettings* metodai priima nustatymo ar lengvatos pavadinimą ir grąžina iš *meta* modelio atitinkamą reikšmę. *getSfCode* metodas grąžina vartotojo nustatymuose nurodytą sąskaitos-faktūros numerio priedėlį ir kitos sąskaitos numerį, o *getNextInvoiceNumber* metodas grąžina tik kitos sąskaitos numerį. *getTotalIncome* metodas grąžina visų vartotojo išrašytų sąskaitų bendrą sumą, o *getTotalExpenses* grąžina bendrą visų suvestų išlaidų sumą. *getGPM, getPSD* ir *getVSD* metodai grąžina atitinkamus, jau apskaičiuotus, vartotojo mokesčius.

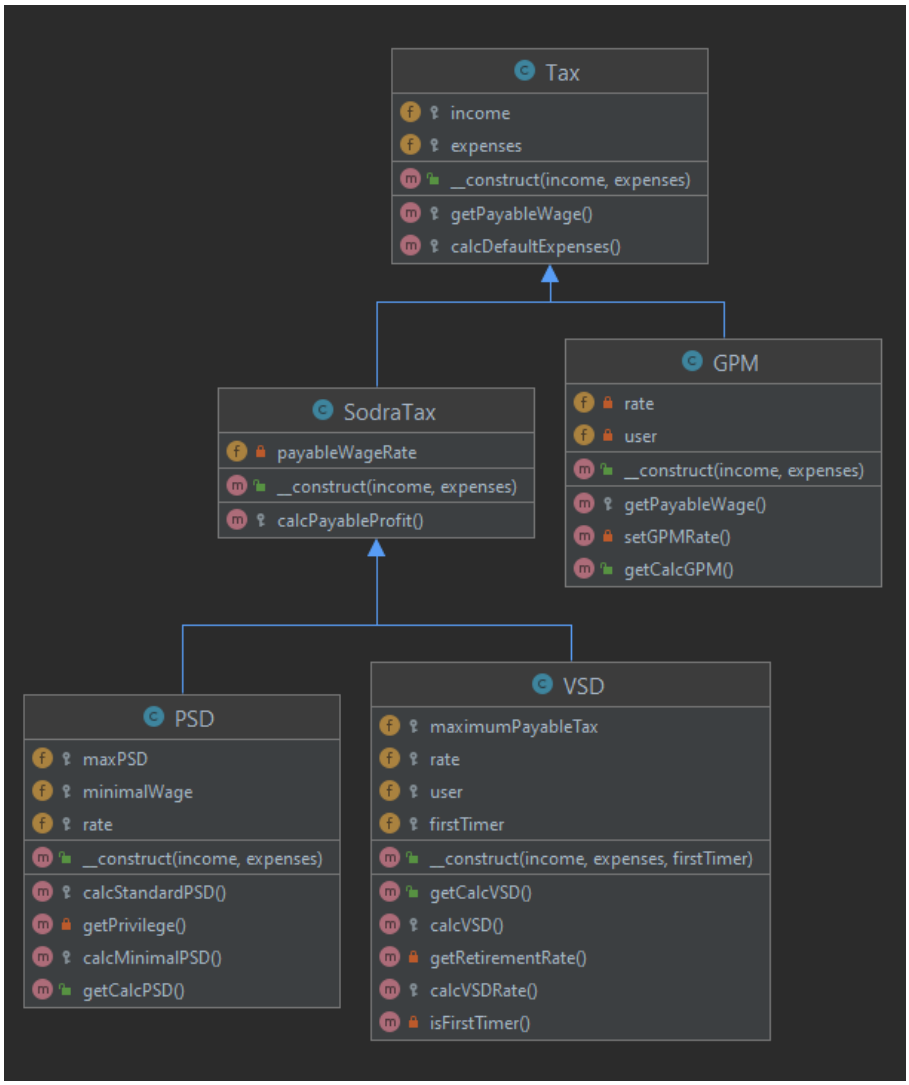


20 pav. *InvoiceEmail* ir *ImportData* modeliai

InvoiceEmail ir ***ImportData*** modeliai (žiūrėti 20 pav.), atitinkamai atsakingi už sąskaitų-faktūrų priminimų el. laiškų ir importuojamų sąskaitų informacijos išsaugojimą ir valdymą duomenų bazėje. Abu modeliai turi tik vieną metodą, atsakingą už modelio priklausomybės ryšį su kitais modeliais. Taip pat, modelių *fillable* kintamajame yra aprašomi duomenų bazės laukeliai, kurie gali būti užpildomi kuriant modelius, o *InvoiceEmail cast* kintamasis papildomai nurodo, į kokio tipo PHP duomenis bus verčiamos duomenų bazės reikšmės.

4.2 Mokesčių apskaičiavimo klasės

Siekiant sklandžiai valdyti gyventojų pajamų mokestį ir valstybinį socialinį bei privalomąjį sveikatos draudimus buvo sukurtos 5 papildomos klasės, kurios tarpusavyje yra susietos: *Tax, SodraTax, GPM, PSD* ir *VSD*.



21 pav. Mokesčių klasių medis

Pagrindinė tėvinė klasė ***Tax,*** kuri yra susijusi su visomis kitomis mokesčių klasėmis (žiūrėti 21 pav.), turi du metodus, du kintamuosius ir konstruktorių. Konstruktorius priskiria reikšmes, perduotas iš paveldimų klasių (angl. *child class*), kintamiesiems. *calcDefaultExpenses* metodas (žiūrėti 22 pav.) apskaičiuoja įprastas išlaidas, kurios gaunamos bendrą išrašytų sąskaitų sumą dauginant iš 30 procentų. *getPayableWage* metodas patikrina, kurios išlaidos didesnės - ar įprastos (30 procentų nuo visų pajamų), ar nurodytos vartotojo nustatymuose, ir grąžina pelną, kurį gauna iš visų pajamų atėmus didesnes išlaidas.

```php
protected function calcDefaultExpenses() {
    return round( num: $this->income * 0.3,  precision: 2);
}
```

22 pav. *calcDefaultExpenses* metodas

Tiesiogiai iš *Tax* klasės yra paveldima ***GPM*** klasė. Ši klasė sudaryta iš trijų metodų, dviejų kintamųjų ir konstruktoriaus. Konstruktorius perduotas reikšmes iškart perduoda tėvinei klasei. *getPayableWage* metodas (žiūrėti 23 pav.) apskaičiuoja GPM mokesčio skaičiavimui reikalingą apmokestinamo pelno sumą, kuri yra gaunama lyginant, ar įprastos išlaidos (atskaičius 30 procentų nuo pajamų), ar realios išlaidos su įskaičiuotomis PSD ir VSD įmokomis yra didesnės. Pasirinkus didesnę reikšmę, ši yra atimama iš bendro pelno ir gauta reikšmė perduodama tolimesniam GPM apskaičiavimui.

```php
protected function getPayableWage() {
    $psd = new PSD($this->income, $this->expenses);
    $vsd = new VSD($this->income, $this->expenses);

    if($this->expenses + $psd->getCalcPSD() + $vsd->getCalcVSD() > $this->calcDefaultExpenses()) {
        return $this->income - $this->expenses - $psd->getCalcPSD() - $vsd->getCalcVSD();
    } else {
        return $this->income - $this->calcDefaultExpenses();
    }
}
```

23 pav. *getPayableWage* metodas

setGPMRate metodas (žiūrėti 24 pav.) tikrina, ar vartotojas veiklos nustatymuose yra pažymėjęs, kad užsiima laisvąja profesija.

```php
private function setGPMRate() {
    $isFreeMarket = $this->user->getPrivilege( privilege: 'isFreeMarketActivity');
    if($isFreeMarket === 'yes' || $this->getPayableWage() > 35000) {
        $this->rate = 0.15;
    } elseif($this->getPayableWage() <= 20000) {
        $this->rate = 0.05;
    } elseif($this->getPayableWage() > 20000 && $this->getPayableWage() <= 35000) {
        $this->rate = 0.15 - (0.1 - 2 / 300000 * ($this->getPayableWage() - 20000));
    }
}
```

24 pav. *setGPMRate* metodas

Jei vartotojas užsiima laisvąja profesija, tuomet GPM tarifas nustatomas kaip 15 procentų, kitu atveju yra tikrinama, ar atskaičius PSD ir VSD įmokas, apmokestinamasis pelnas yra mažesnis

nei 20 tūkstančių – jei taip, tuomet nustatomas 5 procentų tarifas. Jei šis pelnas viršija 20 tūkstančių, bet nesiekia 35 tūkstančių eurų, tuomet, taikant VMI formulę, yra gaunamas GPM tarifas tarp 5 ir 15 procentų, o jei pelnas perkopia 35 tūkstančių eurų ribą, tuomet GPM tarifas prilyginamas 15 procentų. *getCalcGPM* metodas apskaičiuoja gyventojų pajamų mokestį pagal jau nustatytą GPM tarifą ir daugina jį iš apmokestinamo pelno.

SodraTax klasė (žiūrėti 25 pav.) turi vieną metodą, kintamąjį ir konstruktorių. Konstruktorius perduoda reikšmes tėvinei klasei, gautas iš paveldimų klasių, ir nustato Sodros pajamų tarifą. *calcPayableProfit* metodas apskaičiuoja apmokamų Sodros pajamų sumą, kuri yra gaunama tėvinės klasės apskaičiuotą pelną dauginant iš 90 procentų.

```
class SodraTax extends Tax
{
    private $payableWageRate;

    function __construct($income, $expenses)
    {
        parent::__construct($income, $expenses);

        $this->payableWageRate = 0.9;
    }

    protected function calcPayableProfit(){
        return round( num: $this->getPayableWage() * $this->payableWageRate,  precision: 2);
    }
}
```

25 pav. *SodraTax* klasė

PSD klasė tiesiogiai paveldi tėvinės *SodraTax* klasės metodus ir kintamuosius. Ši klasė yra sudaryta iš keturių metodų, trijų kintamųjų ir konstruktoriaus. Konstruktorius perduoda klasei perduotas reikšmes tėvinei klasei ir nustato minimalios algos tarifą, maksimalų PSD įmokos dydį, bei PSD mokesčio tarifą. *getPrivilege* metodas grąžina teigiamą reikšmę, jei vartotojas veiklos nustatymuose yra pažymėjęs, kad turi lengvatą. *calcMinimalPSD* (žiūrėti 26 pav.) metodas apskaičiuoja minimalią PSD mokesčio mokėtiną sumą, kuri yra skaičiuojama minimalią algą dauginant iš 12 mėnesių ir dauginant iš PSD tarifo. *calcStandartPSD* (žiūrėti 26 pav.) metodas apskaičiuoja standartinį PSD mokestį, kuris yra apskaičiuojamas nuo realaus pelno, gauto iš tėvinės klasės *SodraTax*. *getCalcPSD* (žiūrėti 26 pav.) lygina minimalų PSD ir realaus pelno PSD mokesčius ir grąžina tą, kuris yra didesnis, nebent vartotojas turi lengvatą - tuomet visada grąžinamas realaus pelno PSD, net jei jis yra mažesnis už minimalios algos PSD mokestį.

```php
protected function calcMinimalPSD() {
    $monthly = round( num: $this->minimalWage * $this->rate,  precision: 2);
    return $monthly * 12;
}

protected function calcStandardPSD() {
    return $this->calcPayableProfit() * $this->rate;
}

public function getCalcPSD() {
    if($this->calcStandardPSD() >= $this->calcMinimalPSD() || $this->getPrivilege()) {
        return min(round($this->calcStandardPSD(),  precision: 2), $this->maxPSD);
    } else {
        return round($this->calcMinimalPSD(),  precision: 2);
    }
}
```

26 pav. *calcMinimalPSD, calcStandartPSD* ir *getCalcPSD* metodai iš *PSD* klasės

VSD klasė, taip pat kaip ir *PSD*, turi tėvinę *SodraTax* klasę. *VSD* klasė sudaryta iš penkių metodų, keturių kintamųjų ir konstruktoriaus. Konstruktorius perduoda tėvinei klasei šiai klasei perduotas reikšmes ir nustato standartinį VSD tarifą. *getRetirementRate* (žiūrėti 27 pav.) tikrina, ar vartotojas nustatymuose yra pažymėjęs, kad papildomai kaupia pensijai. Pagal tai atitinkamai yra koreguojamas VSD mokesčio tarifas ir maksimali mokėtino mokesčio suma.

```php
private function getRetirementRate(){
    $rate = $this->user->getPrivilege( privilege: 'additionalPension');
    switch($rate){
        case 'pens21':
            $this->maximumPayableTax = 8678.38;
            return 0.024;
        case 'pens3':
            $this->maximumPayableTax = 9027.38;
            return 0.03;
        default:
            $this->maximumPayableTax = 7282.4;
            return 0;
    }
}
```

27 pav. *getRetirementRate* metodas

isFirstTimer metodas tikrina, ar vartotojas yra pažymėjęs, kad veiklą vykdo pirmą kartą. Jei taip, tuomet VSD mokestis vartotojui nėra taikomas. *calcVSDRate* metodas apskaičiuoja

mokėtiną VSD tarifą įvertinus, ar vartotojas papildomai kaupia pensijai. *calcVSD* metodas apskaičiuoja VSD galutinę mokesčio sumą įvertinant, ar ši suma neperkopia maksimalios VSD įmokos sumos. Jei ši suma yra viršijama, tuomet grąžinama maksimali VSD įmoka.

4.3 Vartotojo sąsaja ir Livewire komponentai

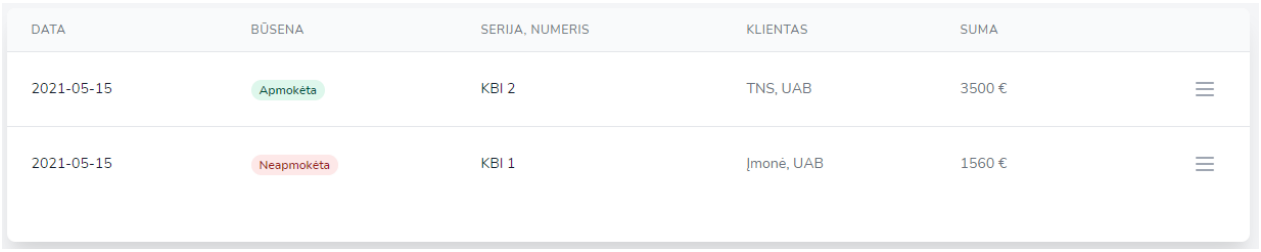
Visa vartoto sąsaja buvo kurta naudojant Livewire komponentus. Šių komponentų iš viso buvo sukurta 18 ir jie atsakingi už šias vartotojo sąsajos dalis:

- pagrindinis puslapis;
- neprisijungusio vartotojo meniu juosta;
- pranešimai apie sėkmingai atliktą veiksmą;
- sąskaitos išrašymo forma;
- išlaidos įvedimo forma;
- išrašytų sąskaitų informacija lentelėje (4 komponentai);
- įvestų išlaidų informacija lentelėje (3 komponentai);
- visi vartotojo veiklos nustatymai (5 komponentai);
- sąskaitų-faktūrų importavimo apdorojimo forma.

Siekiant išlaikyti vienodą ir nuoseklų sistemos dizainą, visi komponentai naudoja tą patį, pagrindinį šabloną, esantį *resources* > *views* > *layouts* > *app.blade.php* faile.

4.3.1 Sąskaitų-faktūrų ir išlaidų lentelės

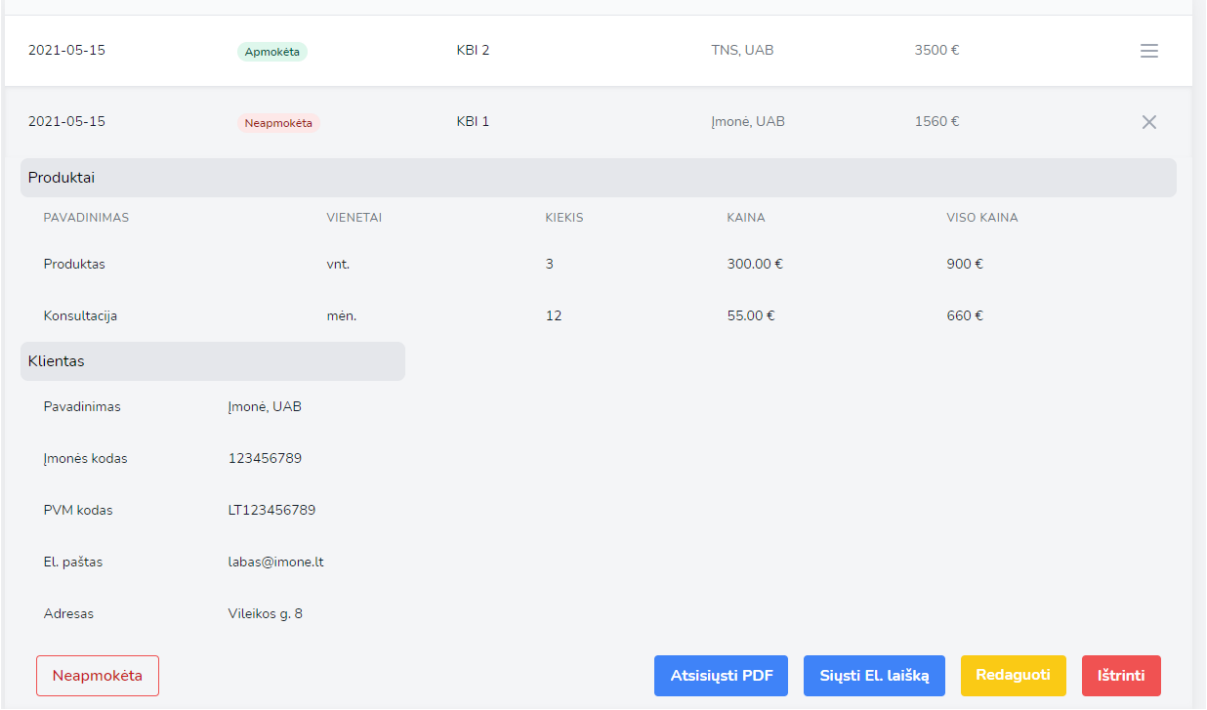
Sudėtingiausias komponentų rinkinys yra išrašytų sąskaitų peržiūros lentelė. Šiame komponente yra nurodomi dveji papildomi komponentai – *invoice-list-item* ir *invoice-info* – iš kurių pastarasis yra paslėptas ir rodomas tik tuomet, kai yra paspaudžiamas daugiau informacijos apie sąskaitą indikuojantis mygtukas (žiūrėti 28 pav. esantį, trimis horizontaliomis linijomis pažymėtą mygtuką). Pasinaudojus Livewire, antras komponentas ir visa jo informacija yra užkraunama puslapyje tik atidarius patį komponentą, tad pirmo užkrovimo metu nėra grąžinama nereikalinga informacija, taip išlaikant puslapį greitu.



28 pav. Neišskleista sąskaitų-faktūrų lentelė

Suskleistame komponente (žiūrėti 28 pav.) yra matoma visa svarbiausia informacija apie išrašytą sąskaitą-faktūrą:

- sąskaitos išrašymo data,
- ar sąskaita apmokėta,
- sąskaitos numeris,
- kliento pavadinimas, kuriam buvo išrašyta sąskaita,
- sąskaitos suma.



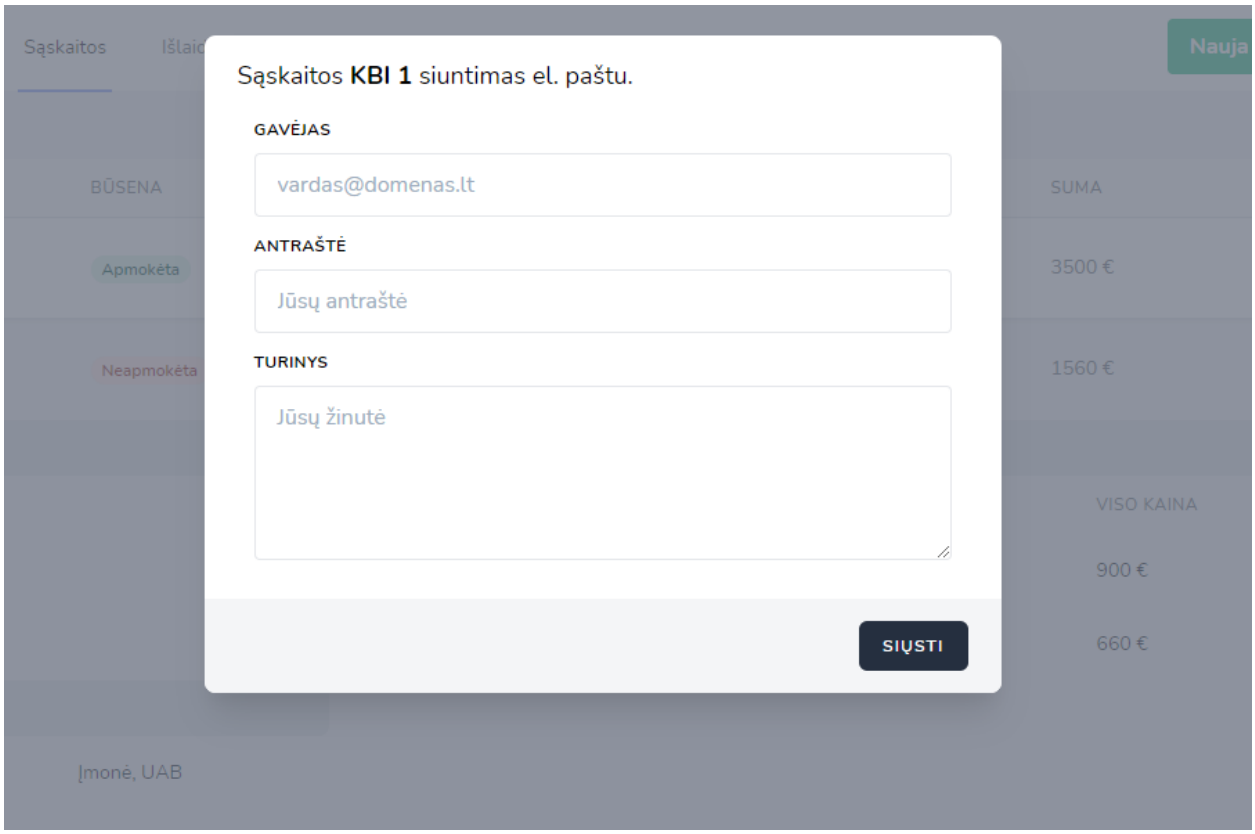
29 pav. Išskleista sąskaitų-faktūrų lentelė

Išskleistame komponente (žiūrėti 29 pav.) galima matyti visą, su sąskaita susijusią, informaciją. Išsiskleidusioje lentelėje galima pamatyti, kokie produktai buvo pridėti prie sąskaitos bei kokios yra jų kainos, kiekis, mato vienetai ir bendra produkto kaina. Komponente taip pat matoma ir detali kliento informacija:

- kliento pavadinimas,
- įmonės kodas,

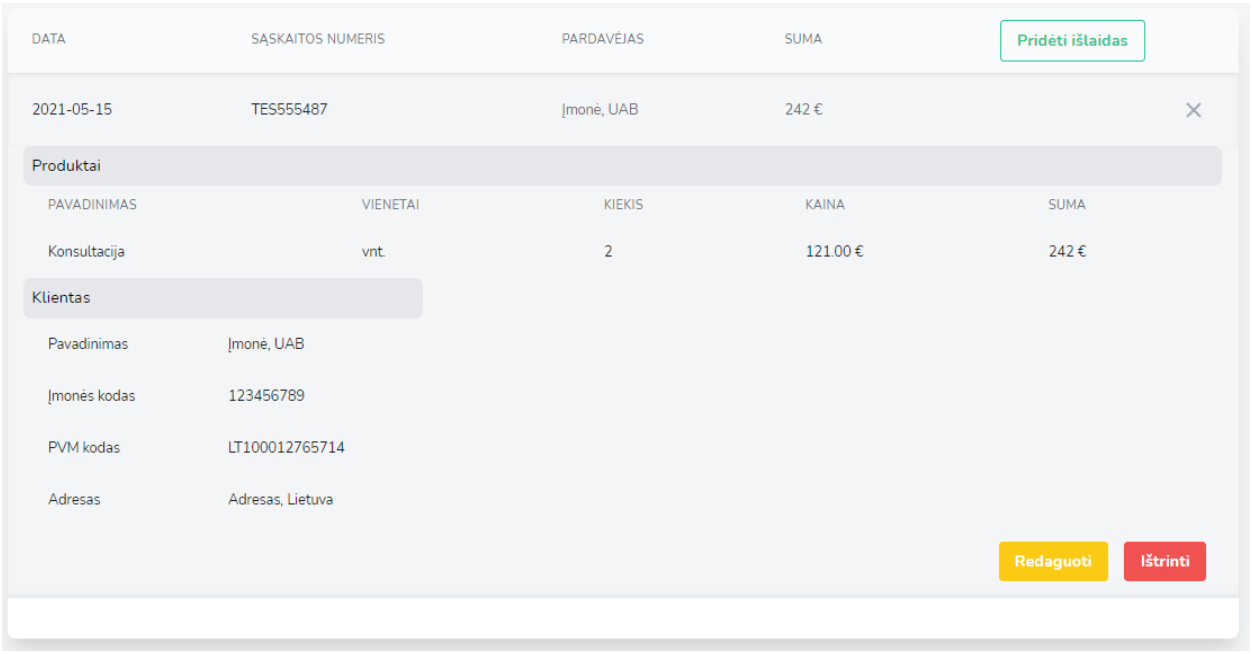
- PVM kodas (jei toks buvo įvestas),
- kliento el. pašto adresas,
- kliento registracijos adresas.

Išskleistame komponente, taip pat galima atlikti ir kitus veiksmus, pasinaudojant komponento apačioje esančiais mygtukais. Paspaudus kairėje esantį mygtuką, indikuojantį sąskaitos apmokėjimo būseną, galima pažymėti sąskaitą kaip apmokėtą arba neapmokėtą. Kairiausias mygtukas dešinėje pusėje, pažymėtas „Atsisiųsti PDF“, leidžia greitai sugeneruoti ir atsisiųsti sąskaitos faktūros PDF versiją. Tačiau, jei vartotojas nenori naudotis savo el. pašto programa norint išsiųsti sąskaitą savo klientui, jis tai gali padaryti tiesiai iš sistemos paspausdamas „Siųsti el. laišką“ mygtuką, kuris atidaro modalią laiško turinio formą (žiūrėti 30 pav.).



30 pav. Modali el. laiško forma sąskaitos siuntimui

Šioje formoje galima suvesti norimą laiško turinį ir išsiųsti laišką klientui prie kurio automatiškai bus prisegta išrašyta PDF sąskaita-faktūra. Likę mygtukai „Redaguoti“ ir „Ištrinti“ leidžia sąskaitai atlikti atitinkamus veiksmus.

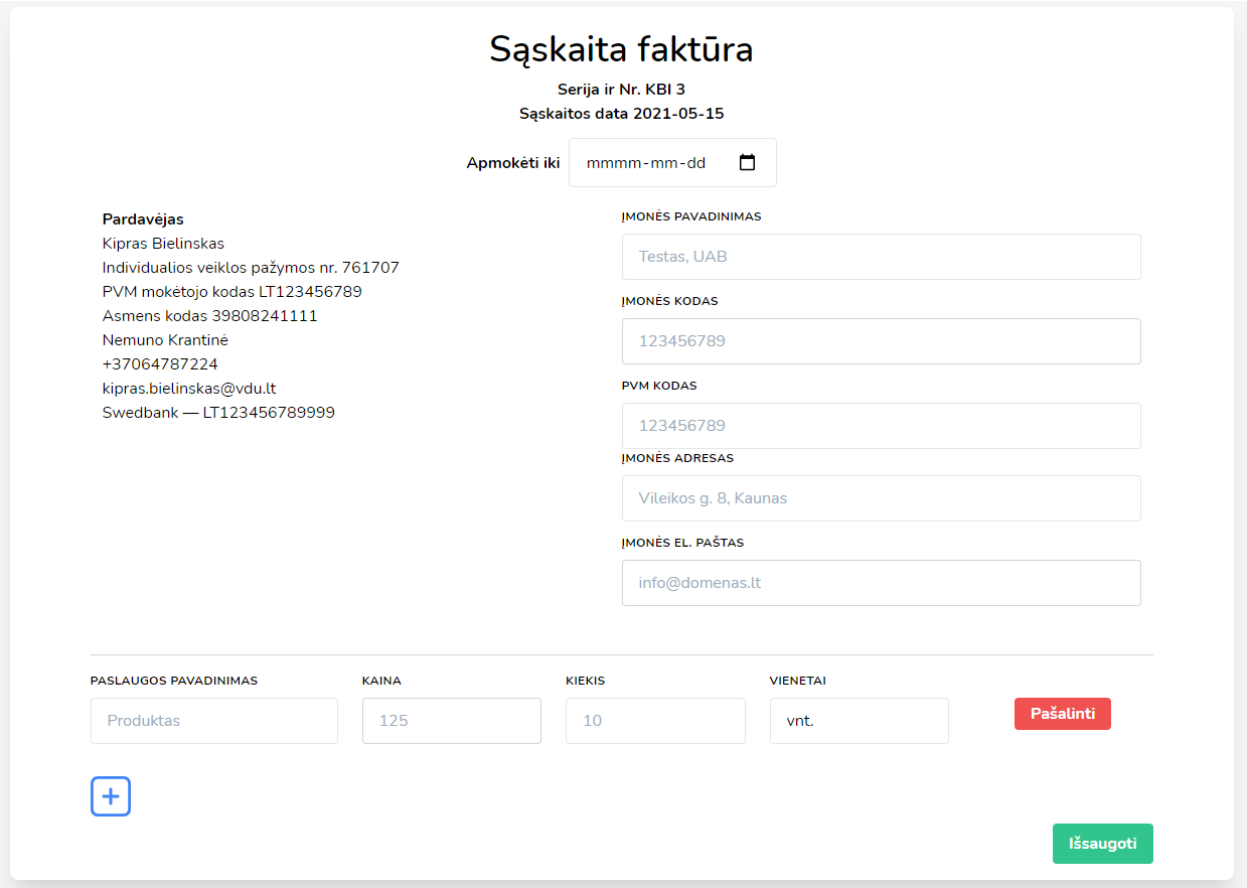


31 pav. Išskleista išlaidų lentelė

Iš esmės vizualiai labai panašūs komponentai naudojami ir išlaidų peržiūrai ir valdymui (žiūrėti 31 pav.). Vienas iš skirtumų – išlaidose išskleistame komponente galima atlikti tik du veiksmus - redaguoti ir ištrinti išlaidą. Toks sukurtų komponentų naudojimas keliose vietose leidžia paspartinti sistemos kūrimo laiką ir išlaikyti vienodą dizainą.

4.3.2 Sąskaitos-faktūros ir išlaidų įvedimo formos

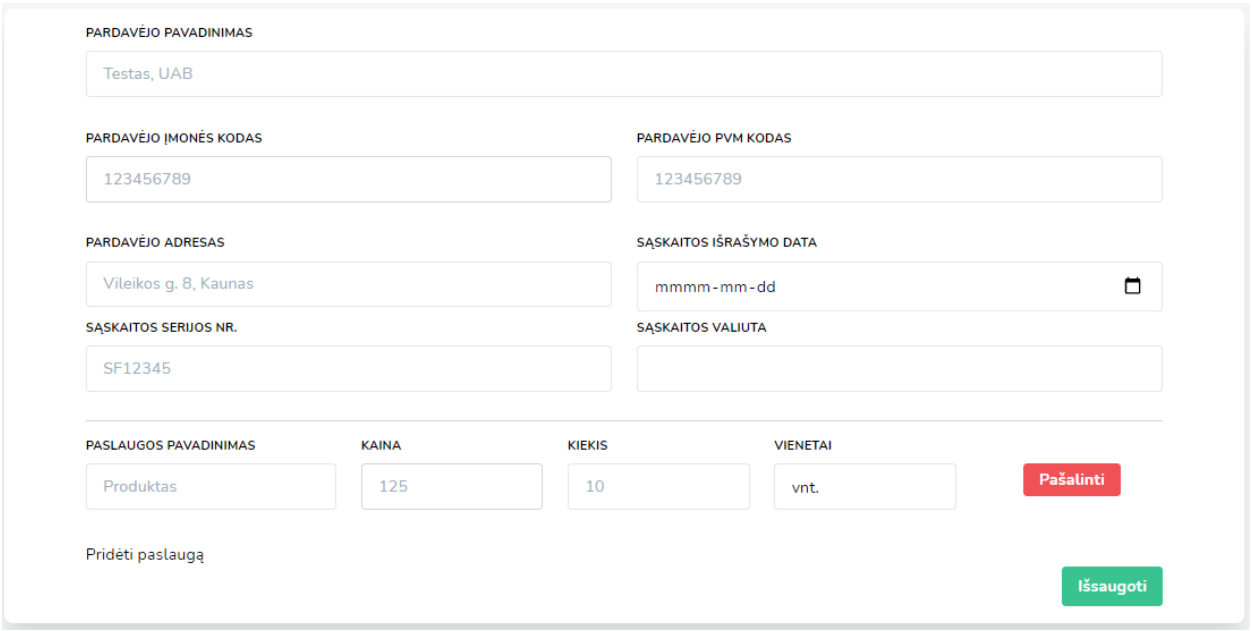
Sąskaitų-faktūrų įvedimo forma (žiūrėti 32 pav.), leidžia įvesti visą su sąskaita susijusią informaciją. Sąskaitos įvedimo formos komponentas sudarytas iš dviejų atskirų komponentų - kliento informacijos formos ir paslaugų įvedimo formos. Prie kliento informacijos galima įvesti įmonės pavadinimą, registracijos kodą, PVM kodą, registracijos adresą ir kontaktinį el. pašto adresą. Po horizontalaus brūkšnio prasideda kitas komponentas – paslaugų įvedimo forma. Čia galima įvesti su paslauga susijusią informaciją: paslaugos pavadinimas, kaina, paslaugos kiekis ir paslaugos vienetai. Paspaudus apačioje kairėje esantį mėlyną mygtuką su pliuso ženklu, galima pridėti papildomų paslaugų laukus. Paslaugų kiekis nėra ribojamas.



32 pav. Sąskaitų-faktūrų įvedimo forma

Formos viršuje esantis „Apmokėti iki“ datos įvedimo laukelis leidžia išrašytoje sąskaitoje nurodyti, iki kada klientas turėtų apmokėti išrašytą sąskaitą. Šiame laukelyje nurodyta apmokėjimo data taip pat yra naudojama ir nurodyti sistemai, kada siųsti automatinius sąskaitos apmokėjimo priminimus.

Kadangi įvedamų išlaidų sąskaitos-faktūros gali būti įvairaus dizaino ir formatų, ši įvedimo forma yra kur kas paprastesnė ir leidžia įvesti dažniausiai sąskaitose esančią, su išlaida susijusią, informaciją (žiūrėti 33 pav.).

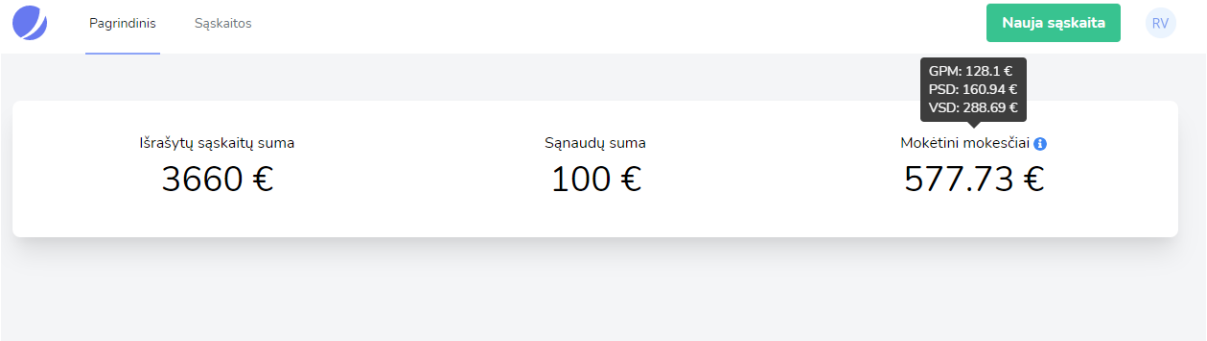


33 pav. Išlaidų įvedimo forma

Tokios sąskaitų-faktūrų ir išlaidų įvedimo formos leidžia patogiai ir suprantamai išrašyti sąskaitas ir suvesti patirtas išlaidas.

4.3.3 Pagrindinis puslapis

Pagrindinio puslapio komponentas aiškiai ir suprantamai atvaizduoja pagrindinę paskyros informaciją: išrašytų sąskaitų sumą, suvestų sąnaudų sumą ir mokėtinų mokesčių sumą.



34 pav. Pagrindinis puslapis su detalia mokamų mokesčių informacija

Užvedus pelę ant prie „Mokėtini mokesčiai“ užrašo esančio, *i* raide pažymėto mygtuko, galima pamatyti detalią mokesčių informaciją (žiūrėti 34 pav.) kurioje mokėtini mokesčiai padalinami į GPM, PSD ir VSD. Toks itin paprastas pagrindinis puslapis leidžia labai greitai ir nesiblaškant pamatyti svarbiausią vykdomos veiklos informaciją.

4.3.4 PDF sąskaita-faktūra

PDF sąskaita-faktūra yra sugeneruoja mPDF modulio pagalba (žiūrėti 35 pav.).

Sąskaita faktūra

Serija ir Nr. KBI 1
Sąskaitos data 2021-05-15
Apmokėti iki 2021-05-22

Pardavėjas
Kipras Bielinskas
Individualios veiklos pažymos nr. 761707
PVM mokėtojo kodas LT123456789
Asmens kodas 39808241111
Nemuno Krantinė
+37064787224
kipras.bielinskas@vdu.lt
Swedbank — LT123456789999

Pirkėjas
Įmonė, UAB
Įm. kodas 123456789
PVM mokėtojo kodas LT123456789
Vileikos g. 8

Pavadinimas	Kiekis	Matas	Kaina	Iš viso
Produktas	3	vnt.	300.00 €	900 €
Konsultacija	12	mėn.	55.00 €	660 €

Bendra suma 1560 €

Sąskaitą išrašė: Kipras Bielinskas

35 pav. Sugeneruota PDF sąskaita-faktūra

Sąskaitos informacija yra gaunama iš įvairių sistemos šaltinių. Pardavėjo informacija yra gaunama iš vartotojo veiklos nustatymuose įvestos veiklos informacijos. Pirkėjo ir produktų informacija gaunama iš sąskaitos kūrimo metu įvestos aktualios informacijos sąskaitos formoje. Sistema automatiškai įvertina, koks buvo paskutinės išrašytos sąskaitos numeris ir serija, todėl naujai išrašytos sąskaitos serija bus tokia pati, o numeris bus vienu didesnis už praeitos. Tačiau sistema taip pat atsižvelgia ir į vartotojo veiklos nustatymuose įvestą informaciją. Jei prie sąskaitos-faktūros nustatymų nurodytas sąskaitos serijos priešdėlis ar / ir numeris nesutampa su automatiškai sugeneruotu priešdėliu ir numeriu, tuomet naujai išrašyta sąskaita atsižvelgia į nustatymuose nurodytą kodą, o sekančios sąskaitos kodas generuojamas atsižvelgiant į praeitos sąskaitos seriją ir numerį.

Surinkus visą reikiamą sąskaitos informaciją, ši yra surašoma į *invoice.blade.php* sąskaitos-faktūros šablono failą, kuriame yra papildomai tikrinama, kokie sąskaitos laukeliai yra užpildyti, o neužpildyti nėra įtraukiami į sugeneruotą PDF sąskaitą (pvz.: jei sąskaitos įvedimo

formoje nėra įvesta „Apmokėti iki“ data, tuomet ir sugeneruotame PDF faile nėra rodoma „Apmokėti iki“ eilutė). Tačiau mPDF modulis nesugeba naudoti *blade.php* failų, tad jam reikia perduoti jau į HTML konvertuotus failus su supaprastintais CSS stiliais. Laimei, Laravel leidžia labai nesudėtingai atlikti šią konversiją ir grąžinti paprastą HTML failą. Kadangi modulis supranta nepilną ir labai ribotą CSS kalbos interpretaciją, labai sudėtinga suformuoti sudėtingų stilių sąskaitas-faktūras.

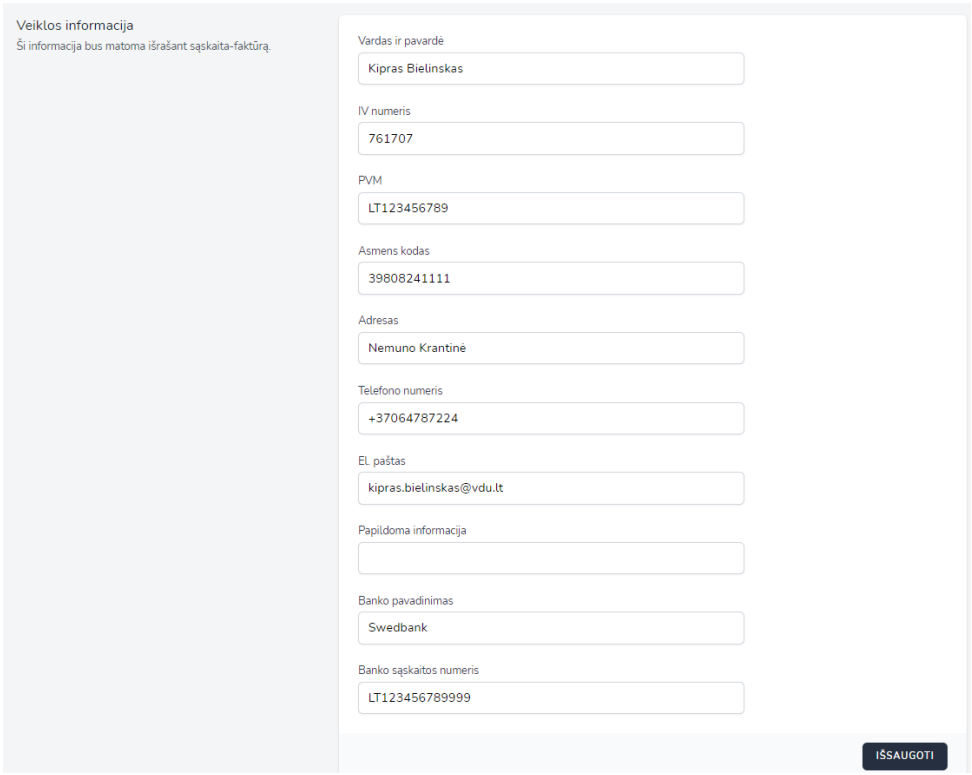
4.3.5 Vartotojo veiklos nustatymai

Vartotojo bendri veiklos nustatymai yra sugrupuoti pagal tokias logines kategorijas:

- veiklos informacija,
- sąskaitos-faktūros informacija,
- automatinių el. laiškų informacija,
- lengvatos ir mokesčių nustatymai,
- sąskaitų CSV importavimas.

4.3.5.1 Veiklos informacija

Veiklos informacijos nustatymuose (žiūrėti 36 pav.) galima įvesti daug, su vykdoma veikla susijusios, informacijos.



36 pav. Veiklos informacijos nustatymai

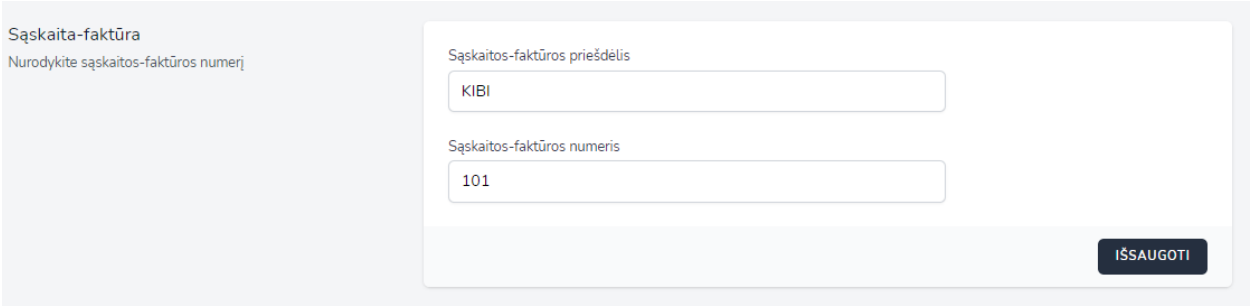
Galima įvesti tokią informaciją, kaip:

- vardas ir pavardė,
- individualios veiklos numeris,
- PVM kodas,
- asmens kodas,
- registracijos adresas,
- kontaktinis telefono numeris bei el. pašto adresas,
- banko pavadinimas bei banko sąskaitos numeris,
- bet kokia norima papildoma informacija.

Visi šie įvesti laukeliai bus matomi kuriant sąskaitą-faktūrą ar ją sugeneravus PDF formatu. Siekiant sumažinti informacijos dubliavimą ir pagerinti vartotojo patirtį, vartotojui užsiregistravus sistemoje, registracijos formoje įvesti duomenys - vardas, pavardė ir el. pašto adresas – yra iškart automatiškai pridedami ir veiklos nustatymuose prie atitinkamų laukelių.

4.3.5.2 Sąskaitos-faktūros nustatymai

Vartotojas gali personalizuoti išrašomų sąskaitų faktūrų seriją ir numerį (žiūrėti 37 pav.).



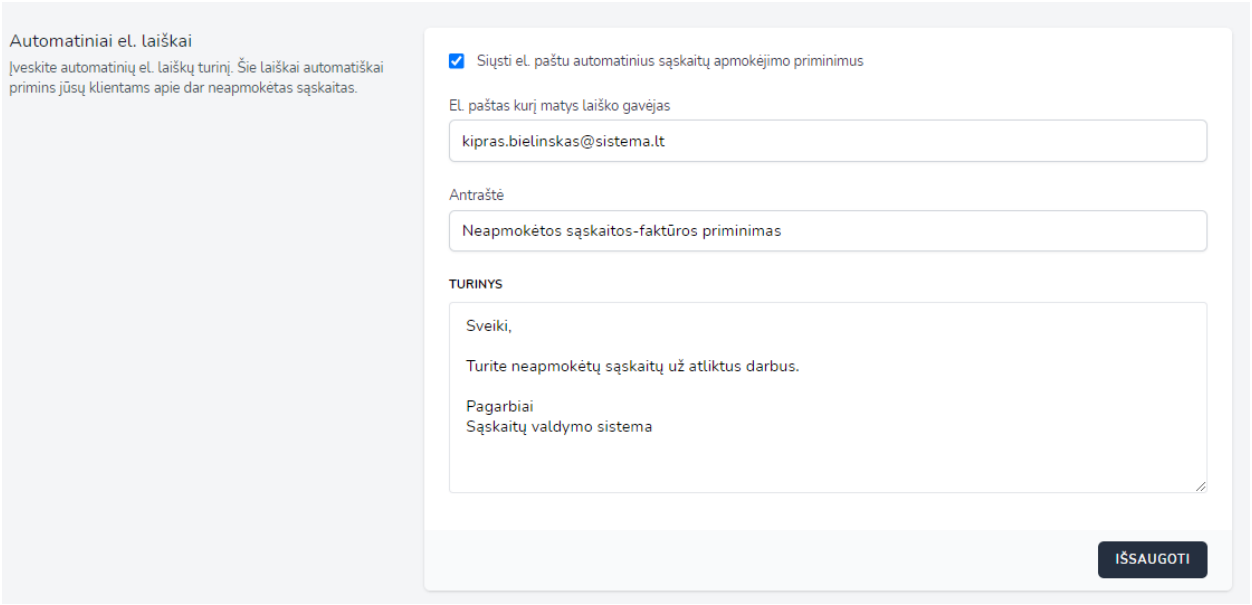
37 pav. Sąskaitos-faktūros nustatymai

Šie nustatymai visuomet tiksliai nurodo, koks sąskaitos numeris bus priskirtas kitai naujai sąskaitai. Pakoregavus ir išsaugojus šiuos nustatymus, sistema naujai sąskaitai priskirs būtent šiuos, naujai išsaugoto priešdėlio ir numerio, nustatymus.

Siekiant vartotojui po sėkmingos registracijos kaip įmanoma labiau sumažinti privalomos papildomos konfigūracijos kiekį, naujam vartotojui automatiškai priskiriami numatytieji sąskaitos nustatymai, kuriuose sąskaitos priešdėlis būną lygus „SF“ , o sąskaitos numeris lygus 1.

4.3.5.3 Automatiniai el. laiškų priminimai

Veiklos nustatymuose vartotojas taip pat gali pažymėti, ar nori, kad sistema automatiškai siųstu neapmokėtų sąskaitų priminimus.



38 pav. Automatinių el. laiškų nustatymai

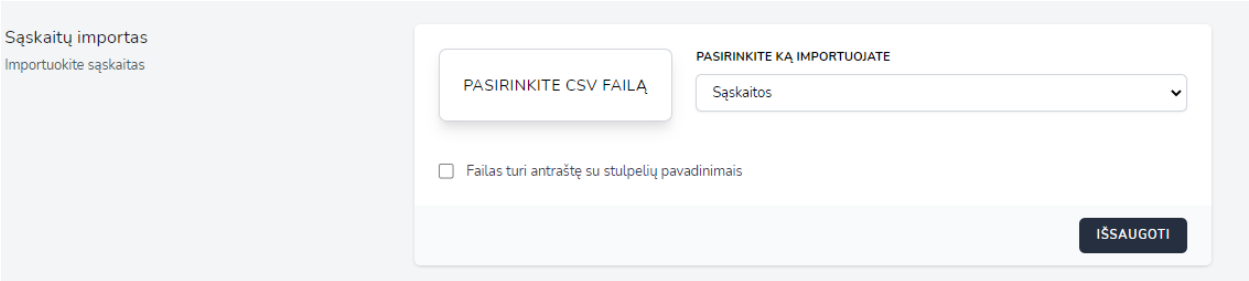
Pažymėjus varnelę (žiūrėti 38 pav.), siunčiamo el. laiško informacija yra sudaroma iš toliau nurodytos, teksto laukeliuose įvestos informacijos. Šis automatinis pranešimas bus siunčiamas sąskaitoje nurodytą apmokėjimo termino pabaigos datą. Klientas gaus laišką, kuris bus siunčiamas iš el. pašto adreso, nurodyto pirmajame teksto laukelyje. Laiško antraštėje matysis antrojo laukelio tekstas, o laiško turinyje matysis trečio teksto laukelio informacija. Taip pat, atitinkama sugeneruota PDF sąskaita-faktūra bus automatiškai prisegama prie laiško.

Sistema, pasinaudodama Laravel *Schedule* moduliu, kiekvieną dieną 10 valandą ryto tikrina, ar yra neapmokėtų sąskaitų, kurių apmokėjimo terminas baigiasi šiandien. Tokius laiškus su prisegtomis sąskaitomis sistema automatiškai išsiunčia vėluojantiems apmokėti klientams.

4.3.5.4 Sąskaitų importavimas

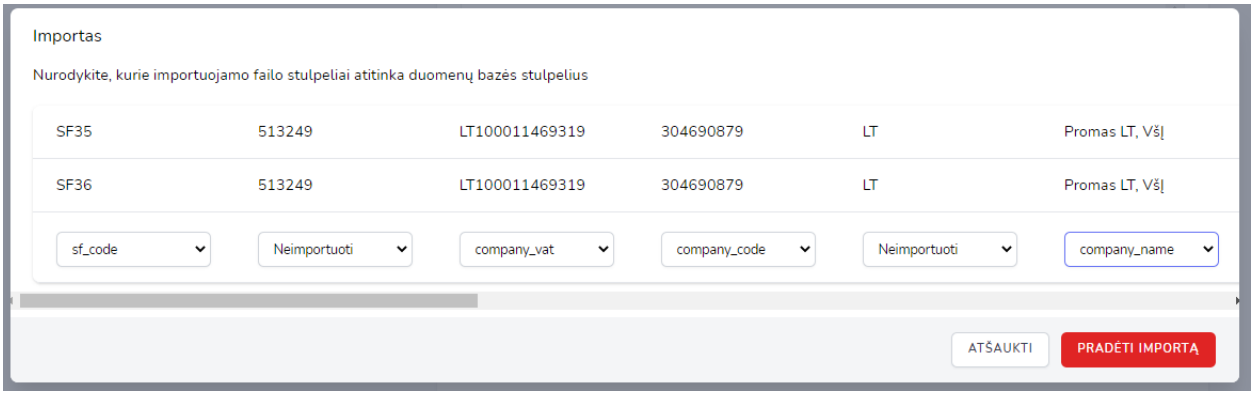
Įvertinus realistiškas situacijas, kuomet vartotojai rinktųsi naudotis kuriama sistema, tik maža dalis būtų pirmą kartą vedantys individualios veiklos apskaitą. Didžioji dalis vartotojų jau ne pirmus metus vykdytų veiklą ir jau būtų pasirinkę apskaitos sistemą finansų vedimui. Siekiant palengvinti tokių vartotojų migraciją iš kitos apskaitos sistemą į šią kuriamą sistemą, reikia suteikti galimybę vartotojui importuoti savo jau seniau išrašytas sąskaitas-faktūras. Tam buvo sukurtas

modulis (žiūrėti 39 pav.), kuris leidžia įkelti populiarų eksportavimui ir importavimui naudojamo CSV formato failą.



39 pav. Sąskaitų importavimo modulis

Tačiau, nėra galimybės užtikrinti, kad kitos sistemos eksportuojamus duomenis į CSV failą surašys vienodai išdėstydamos juos stulpeliuose. Gali taip atsitikti, kad kitos sistemos iš viso nekaupia vienokių ar kitokių su sąskaitomis susijusių duomenų, o galbūt kaip tik kaupia ir išeksportuoja papildomus duomenis, kurių kuriama sistema nebuvo suprojektuota valdyti. Šiai problemai išspręsti, buvo integruota galimybė pasirinkti, kaip tiksliai bus apdorotas įkeltas CSV failas.



40 pav. Importuojamų sąskaitų apdorojimo forma

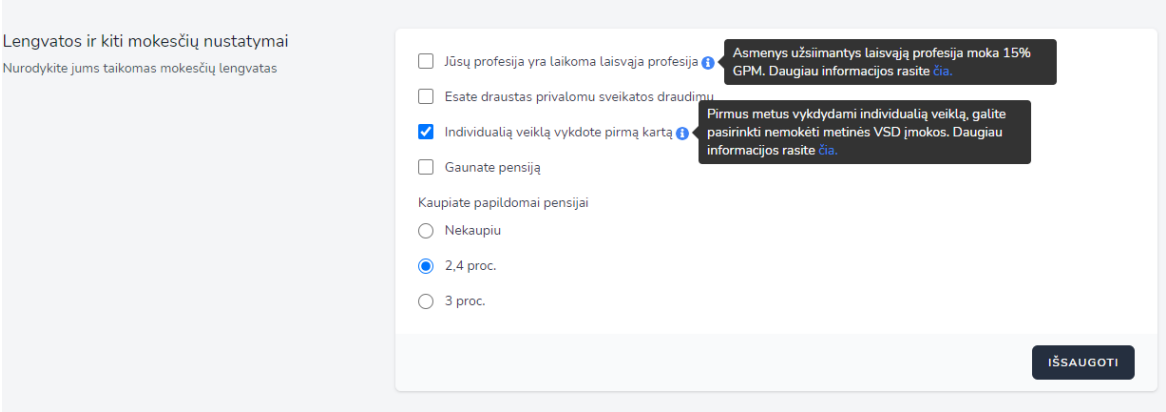
Sėkmingai įkėlus CSV failą į sistemą, vartotojui yra atidaroma modale esanti importo apdorojimo forma (žiūrėti 40 pav.). Šioje formoje yra atvaizduojami visi importuojamo failo stulpeliai ir dvi eilutės su CSV failo turiniu. Vartotojas gali pasirinkti, kokią informaciją importuoti į sistemą prilyginant kiekvieną stulpelį atitinkamai sistemos duomenų bazės stulpelio reikšmei. Vartotojas taip pat gali pasirinkti stulpelius, kuriuos sistema ignoruos ir neimportuos į sistemą.

Sistemai apdorojant didelio dydžio CSV failus kyla ir kita problema – negana to, kad vartotojas būtų verčiamas laukti, kol sistema apdoros visas faile esančias sąskaitas ir per tą laiką negalės toliau dirbti sistemoje, bet ir kitiems vartotojams gali pasijusti sistemos sulėtėjimas. Tokios sistemų problemos gali labai greitai atbaidyti vartotojus, kurie mano, jog sistema yra

nepatikima. Šiai problemai išspręsti buvo pasitelktas Laravel *Horizon* modulis, kuriame galima importavimo darbą užregistruoti į bendrą darbų eilę. Tuomet, asinchroniniu būdu, *Horizon* paskiria laisvus sistemos resursus – darbuotojus - atlikti darbus iš eilės. Tokiu būdu, sistemos darbas nesutrinka net importuojant ir itin didelio dydžio CSV failus, o vartotojas gali tęsti darbą sistemoje, kol fone toliau importuojamos sąskaitos.

4.3.5.5 Lengvatos ir mokesčių nustatymai

Realiame pasaulyje, individualią veiklą vykdantys žmonės turi įvairių skirtumų, susijusių su savo veiklai būdingų mokėtinų mokesčių apskaičiavimu. Šie skirtumai gali atsirasti kai vartotojai pasirenka skirtingas pensijų kaupimo programas ar šiems priklauso įvairios lengvatos. Siekiant sistemoje integruoti kaip įmanoma tikslesnę mokesčių apskaičiavimo metodiką, kuri prisitaiko prie įvairių asmenų poreikių, reikia atsižvelgti ir į lengvatų bei pensijos nustatymų integravimą.



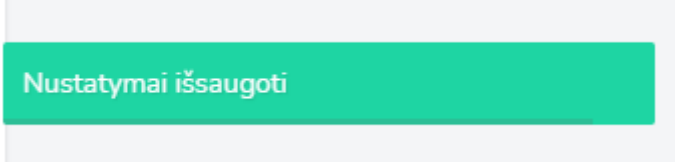
41 pav. Lengvatų ir pensijų nustatymai

Vartotojai, turintys tam tikrų lengvatų, jas gali iškart nurodyti sistemos nustatymuose (žiūrėti 41 pav.). Taip pat, asmenys, papildomai kaupiantys pensijai, gali pasirinkti jiems taikomą pensijos procentinę išraišką. Šie nustatymai daro įtaką vartotojo apskaičiuojamiems mokesčiams.

Siekiant geriau paaiškinti, kokias lengvatas vartotojas turėtų pasirinkti, buvo integruoti papildomi paaiškinimai prie laisvosios profesijos ir pirmą kartą vykdomos veiklos pasirinkimų. Šie paaiškinimai automatiškai pasirodo užvedus pelę ant pasirinkimų tekstų ar ant papildomą informaciją indikuojančios *i* raidės. Jei vartotojui nepakanka pateiktos informacijos, jis gali paspausti nuorodą, kuri jį nukreips į daugiau informacijos turintį Sodros ar VMI puslapius.

4.3.6 Sistemos pranešimai

Visi elementai sistemoje yra atnaujinami neperkraunant puslapio. Tai, įprastu atveju, sukeltų naudojimosi neaiškumų, nes vartotojui nebūtų suteikiama jokia indikacija, kad jo įvesti duomenys buvo sėkmingai išsaugoti. Šiai problemai išspręsti buvo panaudota JavaScript pranešimų biblioteka *Noty*.



42 pav. *Noty* pranešimas

Po kiekvieno sėkmingo veiksmo, vartotojui iššoka dinaminis animuotas pranešimas viršutiniame dešiniajame kampe su užrašu, kad veiksmas buvo atliktas sėkmingai (žiūrėti 42 pav.). Šis pranešimas gali būti iškart išjungtas paspaudus ant jo. Jeigu vartotojas nepaspaudžia išjungimo mygtuko, laukelis automatiškai pradingsta po trijų sekundžių.

4.4 Vartotojo vadovas

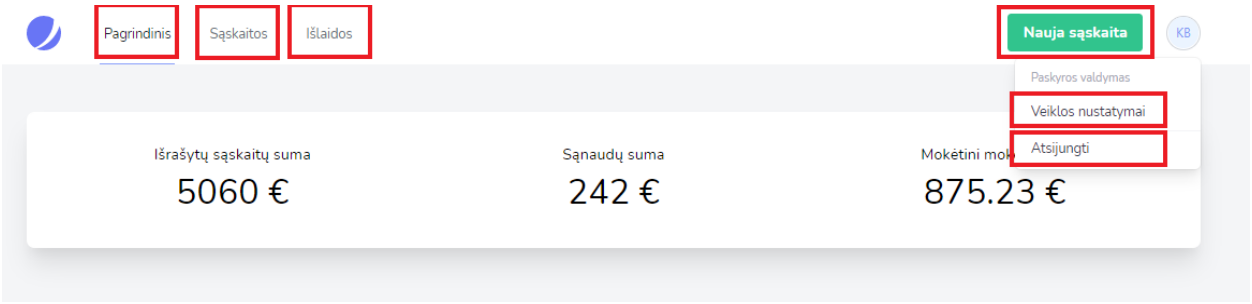
Sukurtai sistemai galima nurodyti tokį vartotojo vadovą. Norint pradėti darbą sistemoje, pirmiausia reikia joje užsiregistruoti suvedant prašomą informaciją registracijos lange (žiūrėti 13 pav.). Sėkmingai užsiregistravus, registracijos formoje pateiktą informaciją reikia suvesti prisijungimo lange (žiūrėti 12 pav.). Sėkmingai prisijungus prie sistemos, galima pradėti ja naudotis. Prie sistemos prisijungus pirmą kartą, reikia įvesti visą aktualią vartotojo veiklos informaciją veiklos nustatymų puslapyje (žiūrėti 36, 37, 38, 39 ir 41 pav.). Tuomet vartotojas jau gali išrašyti sąskaitą-faktūrą spustelėjęs ant mygtuko „Nauja sąskaita“, esančio meniu juostos dešiniajame kampe. Išrašydamas sąskaitą, vartotojas turi užpildyti sąskaitos-faktūros lange esančią informaciją (žiūrėti 32 pav.). Norėdamas įvesti išlaidą, vartotojas turi paspausti meniu juostos kairiajame kampe esantį mygtuką „Išlaidos“, o atsidariusiame lange dešinėje – „Pridėti išlaidas“. Atsidariusiame išlaidų įvedimo lange, vartotojui reikia užpildyti su išlaida susijusius laukelius (žiūrėti 33 pav.). Vartotojui grįžus į pagrindinį puslapį bus matoma visa svarbiausia veiklos informacija – išrašytų sąskaitų, sąnaudų ir mokėtinų mokesčių sumos.

5. TESTAVIMAS

Siekiant užtikrinti, kad kuriama sistema veiktų taip, kaip ji buvo suprojektuota veikti, reikia atlikti platų ir itin kruopštų visos sistemos testavimą. Tai galima atlikti įvairiais būdais, o vienas iš populiariausių ir lengviausių – sistemos kūrėjo rankinis sistemos testavimas. Tokio testavimo metu galima puikiai įsitikinti, kad sistema vizualiai atrodo teisingai ir panaudoti moduliai funkcionuoja tinkamai. Kuriama sistema, palyginus su bet kokiomis kitomis komercinėmis sistemomis, turi sąlyginai labai nedaug skirtingų puslapių, tiksliau, aštuonis (žiūrėti 43 pav.):

- pagrindinį,
- prisijungimo ir registracijos,
- sąskaitų peržiūros ir redagavimo,
- išlaidų peržiūros ir redagavimo,
- veiklos nustatymų puslapius.

Tad tokio dydžio sistemai puikiai tinka vizualių elementų testavimą atlikti rankiniu būdu.



43 pav. Paryškinti atskiri sistemos puslapiai

Tačiau rankinis vizualinis testavimas turi rimtų trūkumų. Viena iš jų – žmogiška klaida. Vykdant testavimą tik vizualiai, gali susidaryti įspūdis, kad sistema veikia teisingai, nors iš tiesų gali būti įsivėlusi nežymi klaida. Tokios problemos tampa itin skaudžiomis, kuomet klaidos įsivelia mokesčių apskaičiavime. Įvedus mažas pajamas, kelių centų netikslumo klaida gali būti lengvai nepastebėta, o paleistoje sistemoje realiems vartotojams tokia klaida gali sukelti labai rimtų nemalonumų, o gal net ir nuostolių. Siekiant užkirsti kelią tokioms problemoms, viso sistemos kūrimo metu buvo taikomas *Unit* testavimo metodas. Šis metodas yra itin plačiai paplitęs tarp sėkmingų komercinių sistemų, nes teisingai įgyvendinus šią programavimo ir testavimo metodiką, galima drastiškai sumažinti kylančių klaidų kiekį. Skirtingai nei rankinis testavimas, kurio metu yra testuojami dideli komponentų ir modulių rinkiniai puslapiuose, *Unit* (lietuviškai, vienetas) testai leidžia taikyti automatizuotus testavimus labai specifinėms sistemos dalims. Kuriamos sistemos atžvilgiu, tai yra labai aktualu, nes leidžia ištestuoti kiekvieną mokesčio

formulę visais įmanomais būdais, pritaikant visas lengvatų kombinacijas ir gautus rezultatus sulyginant su Sodros oficialia individualios veiklos skaičiuokle [9] (žiūrėti 44 pav.).

```
public function testGPMCalc0() {
    $gpm = new GPM( income: 0,  expenses: 0);
    $this->assertEquals( expected: 0, $gpm->getCalcGPM());
}

public function testGPMCalc10000_100() {
    $gpm = new GPM( income: 10000,  expenses: 100);
    $this->assertEquals( expected: 350, $gpm->getCalcGPM());
}

public function testGPMCalc10000_4000() {
    $gpm = new GPM( income: 10000,  expenses: 4000);
    $this->assertEquals( expected: 239.31, $gpm->getCalcGPM());
}

public function testGPMCalc24000_0() {
    $gpm = new GPM( income: 24000,  expenses: 0);
    $this->assertEquals( expected: 840.00, $gpm->getCalcGPM());
}
```

44 pav. GPM mokesčio apskaičiavimo formulės dalis *Unit* testų

Unit testas sudaromas iš dviejų paprastų dalių: reikiamų kintamųjų ar reikiamos situacijos sistemoje sukūrimas ir perdavimas vykdyti, bei gautų rezultatų palyginimas su nurodytais teisingais rezultatais. Įvedus reikšmes iš Sodros skaičiuoklės visiems įvairiems atvejams, galima būti užtikrintam, kad mokesčiai skaičiuojami teisingai ir jais galima pasitikėti.

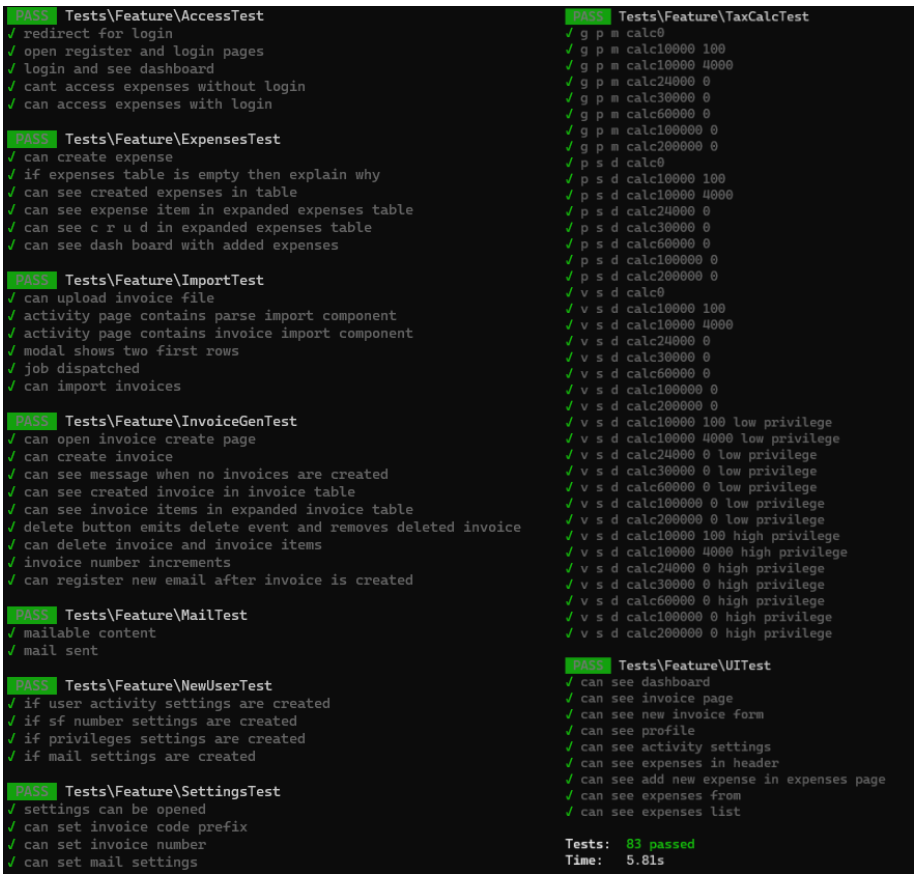
Sistemoje *Unit* testai rašomi ne vien tik mokesčių apskaičiavimui tikrinti, tačiau ir įvairių kitų operacijų teisingam vykdymui užtikrinti. Sistemoje yra parašyti 83 skirtingi testai, kurie pilnai ištestuoja visus sistemos komponentus ir užtikrina komponentų teisingą funkcionavimą. Didžioji dalis testų simuliuoja įvairių sistemoje randamų formų pildymą ir tuomet tikrina, ar duomenys buvo teisingai išsaugoti duomenų bazėje (žiūrėti 45 pav.).

45 pav. *Unit* testas, skirtas patikrinti, ar naujai sukurta sąskaita yra teisingai išsaugojama duomenų bazėje

Kiti testai užtikrina, kad:

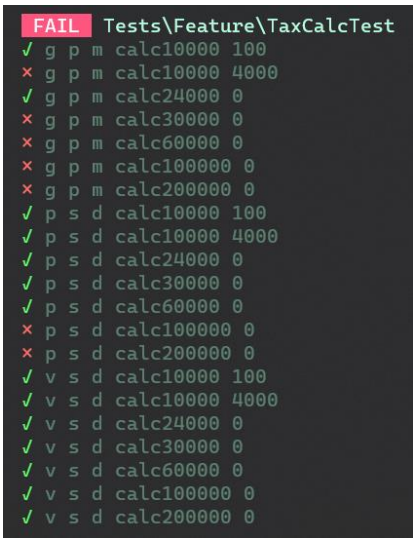
- Tik prisijungę vartotojai gali matyti savo paskyras;
- El. laiškai yra sėkmingai išsiunčiami;
- Reikalingi vartotojo sąsajos elementai yra rodomi tik reikiamais momentais (pvz.: tekstas „Neturite sukūrę sąskaitų-faktūrų“ yra rodomas tik tuomet, kada nėra sukurta nei viena sąskaita. Kitu atveju, šis tekstas turi būti slepiamas);
- Importuojami sąskaitų duomenys yra teisingai suvedami į duomenų bazę;
- Pirmą kartą vartotojui prisiregistravus prie sistemos, tikrinama, ar automatiškai yra sukuriami reikiami veiklos nustatymų įrašai duomenų bazėje.

Tačiau *Unit* testai ne tik užtikrina, kad užbaigta sistema veikia nepriekaištingai, tačiau ir kūrimo procese padeda užtikrinti, kad kuriami nauji komponentai netrukdys teisingai funkcionuoti sistemoje jau integruotiems komponentams. Visas sistemos tikrinimo procesas vyksta komandinėje eilutėje paleidus komandą *php artisan test* (žiūrėti 46 pav.). Tuomet sistema paleidžia visus suprogramuotus testus ir parodo, kurie iš šių komponentų veikia teisingai, o kurie grąžina rezultatą, kuris nesutampa su reikalaujamu.



46 pav. Paleista *php artisan test* komanda ištestavo visą sistemą ir nerado klaidų

Toks programavimo būdas integruojant *Unit* testus labai palengvino mokesčių apskaičiavimo integravimą, nes buvo galima labai aiškiai matyti, kuriuos mokesčių variantus sistema skaičiuoja neteisingai (žiūrėti 47 pav.).



47 pav. Mokesčių apskaičiavimo problemos, užfiksuotos *Unit* testuose

Deja, bet tokias problemas būtų labai sunku aptikti testuojant rankiniu būdu, mat gauti rezultatai nesutapo tik keliais centais ar keliais eurais, o didžiojoje dalyje mokesčių variantų sistema skaičiuodavo teisingai.

Taikant *Unit* testavimą, galima būti užtikrintu, kad sistema veikia teisingai ir paleidus sistemą naudotis realiems vartotojams nekils nenumatytų problemų. Mokesčiai yra apskaičiuojami tiksliai, kaip kad yra numatyta įstatymuose, ir patvirtina sutikrinant gautus rezultatus su oficialioje Sodros individualiai veiklai skirtoje skaičiuoklėje gaunamais rezultatais.

6. REZULTATAI IR IŠVADOS

Bakalauro darbo metu buvo atlikta panašių apskaitos sistemų analizė, asmenų, užsiimančių individualia veikla, mokėtinų mokesčių analizė, programavimo įrankių analizė ir sistemos projektavimo darbai. Atsižvelgus į kiekvieno iš darbo žingsnių išvadas, buvo suprogramuota apskaitos sistema, kuri buvo nuosekliai ištestuota.

6.1 Rezultatai

1. Pagal iškeltus funkcinius reikalavimus buvo suprojektuota duomenų bazės struktūra, kartu su ryšiais tarp skirtingų lentelių.
2. Sukurti sistemos modeliai, reprezentuojantys duomenų bazės lenteles.
3. Aktualiems mokesčiams apskaičiuoti suprogramuota penkių klasių grupė, pateikianti tikslius rezultatus, sutampančius su Sodros puslapyje pateiktos skaičiuoklės rezultatais.
4. Integruotas sąskaitų-faktūrų PDF generavimas ir atsisiuntimas.
5. Suprogramuota vartotojo sąsaja, kurioje naudotojas gali nesudėtingai išrašyti sąskaitą faktūrą, sugeneruoti ir atsisiųsti jos PDF versiją, tiesiai iš sistemos išsiųsti el. paštu sugeneruotą PDF sąskaitą, pažymėti sąskaitą kaip apmokėtą, ar dar laukiančią apmokėjimo.
6. Vartotojas veiklos nustatymuose gali įjungti automatinius sąskaitų apmokėjimo priminimus, įvesti detalią savo veiklos informaciją kartu su lengvatomis, nurodyti, koks turinys bus siunčiamas automatiniais el. laiškais, bei matyti ir redaguoti kitos išrašomos sąskaitos-faktūros kodą.
7. Integruoti dalies lengvatų paaiškinimai su naudingais papildomais informacijos šaltiniais.
8. Vartotojas gali lengvai sekti savo veiklos išlaidas, jas suvesdamas sistemoje.
9. Pagrindiniame puslapyje vartotojas gali pamatyti aiškią informaciją apie savo veiklą: išrašytų sąskaitų sumą, išlaidų sumą, mokėtinų mokesčių sumą bei atskirų mokesčių sumas.
10. Vartotojas gali perkelti savo sąskaitų duomenis iš kitų sistemų į sukurtą apskaitos sistemą pasinaudodamas CSV failu.
11. Sistemoje integruoti interaktyvūs pranešimai, leidžiantys vartotojui suprasti, kada informacija yra sėkmingai išsaugota.

6.2 Išvados

1. Atlikus kelių populiariausių ir skirtingų vartotojų grupių apskaitos sistemų analizę, buvo išsiaiškinta, kad nėra vieno, visus kriterijus apimančio apskaitos sistemos pasirinkimo vartotojui, kuris verčiasi individualia veikla.
2. Atlikus mokesčių analizę, išsiaiškinta, kad Sodra ir VMI pelną mokesčiams skaičiuoti apskaičiuoja skirtingai.
3. Išsiaiškinta, kad Laravel yra didžiausias PHP kalbos programavimo karkasas, turintis daugiausiai aktyvių svetainių.
4. Įsitikinta, kad Livewire turi itin daug naudojimo panašumų su JavaScript kalbos populiariausiais karkasais.
5. Programuojant vartotojo sąsają buvo pastebėta, kad kodą dalinant į kuo daugiau Livewire komponentų, ryšiai tarp komponentų tampa lengviau panaudojami.
6. Suprogramavus interaktyvią vartotojo sąsają, kurioje puslapiai nėra perkraunami po kiekvienos sėkmingos operacijos, buvo pastebėta, kad vartotojai nesupranta, ar informacija buvo sėkmingai išsaugota.
7. Naudojant Eloquent buvo pastebėta, kad nenaudojant papildomų resursų užkrovimo (angl. *eager loading*) buvo susidurta su „N+1“ užklausų problema, kuomet pagrindiniam puslapiui užkrauti, sistema įvykdydavo virš 700 užklausų į serverį. Iškart užkraunant papildomus resursus, užklausų kiekis sumažintas iki 33, o puslapio greitis žymiai padidėjo.
8. Norint, kad realistiškai vartotojai norėtų naudotis sukurta sistema, reikėjo integruoti būdą, kaip vartotojams paprastai persikelti jau sukurtas sąskaitas iš kitos sistemos į sukurtą apskaitos sistemą.

7. LITERATŪROS SĄRAŠAS

1] K. Yakal, „www.pcmag.com,“ 18 05 2017. [Tinkle]. Available: https://www.pcmag.com/picks/the-best-online-accounting-services-for-freelancers?test_uuid=01jrZgWNXhmA3ocG7ZHXevj&test_variant=b. [Kreiptasi 21 10 2020].

2] Freshbooks, „FreshBooks,“ 06 05 2020. [Tinkle]. Available: https://www.freshbooks.com/en-ie/about/ourstory. [Kreiptasi 22 10 2020].

3] Freshbooksw, „Freshbooks,“ 20 10 2020. [Tinkle]. Available: https://www.freshbooks.com/. [Kreiptasi 22 10 2020].

4] FreshBooks, „Freshbooks,“ Freshbooks, 20 10 2020. [Tinkle]. Available: https://www.freshbooks.com/pricing. [Kreiptasi 22 10 2020].

5] SimilarWeb, „SimilarWeb,“ 20 10 2020. [Tinkle]. Available: https://www.similarweb.com/website/saskaita123.lt/. [Kreiptasi 22 10 2020].

6] Sąskaita123, „Sąskaita123,“ Sąskaita123, N/A. [Tinkle]. Available: https://saskaita123.lt/. [Kreiptasi 22 10 2020].

7] V. m. inspekcija, „VMI,“ N/A. [Tinkle]. Available: https://www.vmi.lt/cms/virtualus-buhalteris-i.aps. [Kreiptasi 22 10 2020].

8] Sąskaita123, Sąskaita123, 26 01 2019. [Tinkle]. Available: https://saskaita123.lt/blog/kaip-reikia-israsyti-saskaita-faktura-fiziniam-asmeniui/. [Kreiptasi 29 05 2021].

9] Sodra, „Sodra,“ N/A. [Tinkle]. Available: https://www.sodra.lt/skaiciuokles/individualios_veiklos_skaiciuokle. [Kreiptasi 23 10 2020].

10] V. m. inspekcija, „VMI.lt,“ N/A. [Tinkle]. Available: https://www.vmi.lt/cms/gyventoju-pajamu-mokestis9. [Kreiptasi 23 10 2020].

11] Sostinės auditoriai, „Sostinės auditoriai,“ 2018. [Tinkle]. Available: https://auditors.lt/gpm-formule-2018m/. [Kreiptasi 12 04 2020].

12] VMI, „vmi.lt,“ 02 09 2010. [Tinkle]. Available: https://www.vmi.lt/cms/web/kmdb/1.10.7.1/-/asset_publisher/0OhS/content/kaip-apskaiciuojamas-lietuvos-vieneto-apmokestinamasis-pelnas-/10174?redirect=https%3A%2F%2Fwww.vmi.lt%2Fcms%2Fweb%2Fkmdb%2F1.10.7.

1%3Fp_p_id%3D101_INSTANCE_0OhS%26p_p_lifecycle%3D0%2. [Kreiptasi 23 10 2020].

[13] Sodra, 2020. [Tinkle]. Available: https://www.sodra.lt/lt/situacijos/vykdau-individualia-veikla?el_id=1144. [Kreiptasi 24 10 2020].

[14] R. Vainienė, „zodynas.vz.lt," Verslo žinios, N/A. [Tinkle]. Available: http://zodynas.vz.lt/Valstybinis-socialinis-draudimas. [Kreiptasi 25 10 2020].

[15] Sodra, 2020. [Tinkle]. Available: https://www.sodra.lt/lt/situacijos/vykdau-individualia-veikla?el_id=1143. [Kreiptasi 25 10 2020].

[16] Sodra, 2020. [Tinkle]. Available: https://www.sodra.lt/lt/situacijos/vykdau-individualia-veikla?el_id=2045. [Kreiptasi 25 10 2020].

[17] Sodra, „Svarbu savarankiškai dirbantiems: kas keičiasi kitais metais," 17 12 2020. [Tinkle]. Available: https://www.sodra.lt/lt/naujienos/svarbu-savarankiskai-dirbantiems-kas-keiciasi-kitais-metais. [Kreiptasi 12 04 2021].

[18] The PHP Group, „php.net," 2020. [Tinkle]. Available: https://www.php.net/manual/en/intro-whatis.php. [Kreiptasi 17 12 2020].

[19] W3Techs , „w3techs.com," 2020. [Tinkle]. Available: https://w3techs.com/technologies/details/pl-php. [Kreiptasi 17 12 2020].

[20] BuiltWith® Pty Ltd, „Laravel Usage Statistics," BuiltWith® Pty Ltd, 17 12 2020. [Tinkle]. Available: https://trends.builtwith.com/framework/Laravel. [Kreiptasi 17 12 2020].

[21] BuiltWith® Pty Ltd, „symfony PHP Framework Usage Statistics," BuiltWith® Pty Ltd, 17 12 2020. [Tinkle]. Available: https://trends.builtwith.com/framework/symfony-PHP-Framework. [Kreiptasi 17 12 2020].

[22] Symfony SAS, „The MVC Pattern," [Tinkle]. Available: https://symfony.com/legacy/doc/gentle-introduction/1_4/en/02-Exploring-Symfony-s-Code. [Kreiptasi 18 12 2020].

[23] N. Ighodaro, „Pusher," 24 5 2018. [Tinkle]. Available: https://blog.pusher.com/laravel-mvc-use/. [Kreiptasi 18 12 2020].

[24] Laravel LLC, „Eloquent: Getting Started," 2020. [Tinkle]. Available: https://laravel.com/docs/8.x/eloquent#introduction. [Kreiptasi 18 12 2020].

[25] P. REDMOND, „Laravel 8 is Now Released!," Laravel News, 08 09 2020. [Tinkle]. Available: https://laravel-news.com/laravel8. [Kreiptasi 18 12 2020].

J. Freeman, „InfoWorld,“ 08 08 2019. [Tinkle]. Available:
26] https://www.infoworld.com/article/3269878/what-is-an-api-application-programming-interfaces-explained.html. [Kreiptasi 25 10 2020].

B. Turner, „TechRadar,“ 15 12 2019. [Tinkle]. Available:
27] https://www.techradar.com/news/what-is-saas. [Kreiptasi 25 10 2020].

„Cloudflare,“ N/A. [Tinkle]. Available:
28] https://www.cloudflare.com/learning/cloud/what-is-the-cloud/. [Kreiptasi 25 10 2020].

„verslilietuva.lt,“ N/A. [Tinkle]. Available:
29] https://www.verslilietuva.lt/verslauk/verslo-formos-ir-mokesciai/individuali-veikla-pagal-pazyma/. [Kreiptasi 25 10 2020].

VMI, N/A. [Tinkle]. Available: https://www.vmi.lt/cms/pridetines-vertes-
30] mokestis. [Kreiptasi 25 10 2020].

„MDN web docs,“ Mozilla, N/A. [Tinkle]. Available:
31] https://developer.mozilla.org/en-US/docs/Web/HTTP/Status/500. [Kreiptasi 25 10 2020].

„juris24.lt,“ N/A. [Tinkle]. Available: https://www.juris24.lt/zodynas/laisvoji-
32] profesija/. [Kreiptasi 25 10 2020].

„infolex.lt,“ 01 01 2019. [Tinkle]. Available: http://www.infolex.lt/ta/12868:str17.
33] [Kreiptasi 25 10 2020].

Techopedia, „Application Framework,“ [Tinkle]. Available:
34] https://www.techopedia.com/definition/6005/application-framework. [Kreiptasi 18 12 2020].

W3schools, „What is AJAX?,“ [Tinkle]. Available:
35] https://www.w3schools.com/whatis/whatis_ajax.asp. [Kreiptasi 18 12 2020].

W3schools, „CSS .class Selector,“ [Tinkle]. Available:
36] https://www.w3schools.com/cssref/sel_class.asp. [Kreiptasi 18 12 2020].

Tutorialspoint, „MVC Framework - Introduction,“ [Tinkle]. Available:
37] https://www.tutorialspoint.com/mvc_framework/mvc_framework_introduction.htm. [Kreiptasi 18 12 2020].

Educative, „What is Object Oriented Programming? OOP Explained in Depth,“
38] [Tinkle]. Available: https://www.educative.io/blog/object-oriented-programming. [Kreiptasi 18 12 2020].

PHP Group, „Final Keyword,“ [Tinkle]. Available:
39] https://www.php.net/manual/en/language.oop5.final.php. [Kreiptasi 18 12 2020].

W3schools, „PHP OOP - Constructor,“ [Tinkle]. Available:
40] https://www.w3schools.com/php/php_oop_constructor.asp. [Kreiptasi 18 12 2020].

D. Horning, „Zesty.io,“ 07 05 2018. [Tinkle]. Available:
41] https://www.zesty.io/mindshare/marketing-technology/what-is-wysiwyg-what-you-see-is-what-you-get/. [Kreiptasi 16 05 2021].

8. PRIEDAI

8.1 Kursinio darbo santrauka

Autorius	Kipras Bielinskas
Pavadinimas	Apskaitos sistema individualia veikla užsiimantiems vartotojams
Vadovas	Rita Valterytė
Darbas pristatytas	Vytauto Didžiojo Universitetas, Informatikos fakultetas, Kaunas 2020-12-18
Puslapių skaičius	46
Priedų skaičius	2

Individualia veikla besiverčiantys asmenys yra patys atsakingi už savo veiklos buhalterinę apskaitą. Dėl šios priežasties egzistuoja kelios viešai prieinamos sistemos, kurios bando palengvinti apskaitos vedimą, naujų sąskaitų generavimą ir finansų sekimą.

Kursinio darbo metu pristatomos ir palyginamos kelios individualia veikla besiverčiantiems asmenims pritaikytos buhalterijos sistemos. Išnagrinėjami jų trūkumai ir pateikiami funkcinių bei dizaino problemų sprendimo būdai. Pristatomi naujai kuriamos apskaitos sistemos funkciniai reikalavimai, sistemoje įdiegti įvairūs mokesčių tarifai. Pateikti argumentuotai pasirinkti sistemos kūrimo darbams skirti įrankiai ir technologijos. Projektavimo etape pateikiamos veiklos diagramos, duomenų bazės modelis, vartotojų vaidmenys. Realizavimo etape aiškinamas programavimo procesas, numatytų funkcinių reikalavimų įgyvendinimas ir naudojamų įrankių savitumas. Pateikti rezultatai ir išvados.

8.2 Apskaitos sistema

Nuoroda: http://158.129.53.9/

8.3 Apskaitos sistemos kodo saugykla

Nuoroda: https://github.com/kiprasbiel/invoice-generator.git