VILNIAUS UNIVERSITETAS
MATEMATIKOS IR INFORMATIKOS FAKULTETAS
KOMPIUTERIJOS KATEDRA

Baigiamasis bakalauro darbas

VU MIF Informacinių technologijų atviros prieigos centro apskaitos sistemos kūrimas

Development of Accounting System for VU MIF IT Open Access Centre

Atliko: 4 kurso, 1 grupės studentė:
Ieva Draugelytė (parašas)

Darbo vadovas:
Linas Būtėnas

VILNIUS
2017

Turinys

Anotacija

Šis rašto darbas pristato VU MIF informacinių technologijų atviros prieigos centro apskaitos sistemos kūrimo procesą. Pagrindiniai reikalavimai buvo suteikti galimybę apdoroti duomenis apie atviros prieigos centro užsakovus, kurti apskaitas, generuoti mokėjimus, siųsti bei gauti pranešimus ir sukurti galimybę vartotojams teikti paslaugų užsakymo paraiškas internetu. Visos apsibrėžtos funkcijos buvo įgyvendintos, užtikrintas jų veikimas. Sistema turi atskiras administratoriaus ir paprasto vartotojo dalis. Tai yra unikali, pagal atviros prieigos centro poreikius realiam naudojimui sukurta sistema.

Summary

This report presents the development of VU MIF information technology open access center accounting system. The main requirements of this project were to create abilities to maintain data about open access center clients, create applications, generate payments, send and receive messages as well as provide and ability for users to create service order applications online. All main functions were successfully developed, it was ensured that they work as supposed to. The system has separate parts for administrator and ordinary users. It is a unique system that was created with intention to be used as a real tool and meet all presented open access center requirements.

Įvadas

Prieš pradedant vystyti šį projektą, neegzistavo jokia kita panašaus pobūdžio apskaitos sistema, kuri būtų skirta VU MIF informacinių technologijų atviros prieigos centrui (toliau - ITAPC). Duomenys buvo saugomi *Excel* failuose, nebuvo jokio automatizuoto jų apdorojimo. Tai buvo nepatogi praktika ir buvo išreikštas noras turėti sistemą, kuri padėtų lengviau saugoti, tvarkyti bei apdoroti duomenis, susijusius su ITAPC užsakovais, suteikti prieigą bei galimybę pateikti paslaugų paraiškas internetu. Sistema buvo kuriama bendradarbiaujant su ITAPC administratore, kartu apsibrėžiant reikiamas funkcijas, planus ir kitus įgyvendinimo niuansus. Šios apskaitos sistemos tikslas yra optimizuoti bei palengvinti ITAPC administratoriaus (-ės) darbą, suteikti lengvesnę prieigą prie atviros prieigos centro užsakovų duomenų, automatizuoti duomenų apdorojimą ir pateiktį, suteikti vartotojams internetinę prieigą prie paslaugų užsakymo paraiškų. Pagrindiniai darbo metu iškelti uždaviniai:

- Išskirti svarbiausius reikalavimus sistemai;
- Atlikti esamų technologijų ir galimų sprendimų analizę;
- Sukurti sistemos administratoriaus dalį, kurioje būtų galima atlikti apskaitas, saugoti bei apdoroti duomenis, generuoti reikiamą informaciją, stebėti mokėjimų būsenas;
- Sukurti sistemos vartotojų dalį, kurioje būtų galima internetu teikti paraiškas ITAPC paslaugų užsakymui;
- Testuoti bei tobulinti sistemą pagal iškeltus konkrečius reikalavimus.

Šio darbo metu visi išsikelti uždaviniai buvo įvykdyti – buvo atlikta populiariausių verslo valdymo sistemų ir reikalingų technologijų analizė, sistema turi sukurtas atskiras administratoriaus ir vartotojo dalis. Administratorius gali apdoroti ir dirbti su visais su užsakovais susijusiais duomenis, automatiškai generuoti norimą informaciją, gauti bei siųsti pranešimus, stebėti duomenų statusus. ITAPC vartotojams yra sukurtos dinaminės paslaugų užsakymo paraiškų formos, suteikiančios galimybę teikti paraiškas internetu. Apie vartotojų pateiktus duomenis nuolatos informuojamas administratorius, jis turi prieigą prie visų sistemos ir vartotojo pateiktų duomenų. Sistema yra parengta naudojimui ir tolimesniam vystymui pagal poreikius.

1. Užduoties analizė

Šiam procesui buvo skirta daug laiko, siekiant užtikrinti jog keliami poreikiai bei lūkesčiai sistemai buvo teisingai suprasti ir įgyvendinti. Prieš pradedant tolimesnį kūrimo procesą, reikėjo praplėsti turimą pirminę architektūrą, peržvelgti ir perdaryti sistemos išdėstymą. Tam buvo pasitelktas prototipavimas, įvairių diagramų braižymas, technologijų analizė. Nors ši sistema buvo kuriama unikaliai pagal konkrečius fakulteto poreikius, egzistuoja įvairių jau sukurtų sistemų (ar jų karkasų) bendroms įmonių apskaitų ar valdymo reikmėms, todėl derėjo atlikti ir pastarųjų analizę, įsivertinant panašumus, panaudojimo galimybes arba atvirkščiai – priežastis, kodėl daug vertingiau yra sukurti naują sistemą, nesinaudojant esamomis.

1.1 Sistemos pagrindas

Sistema buvo pradėta vystyti dar praeito semestro kursinio darbo metu. Atliekant kursinį darbą pagrindinis dėmesys buvo skiriamas poreikių analizei, techninėms galimybėms bei būdams, kuriais būtų galima įgyvendinti išsikeltus poreikius. Pagal atliktos analizės rezultatus buvo kuriami pirminiai sistemos karkasai, kurie demonstravo duomenų suvedimo bei atvaizdavimo galimybes. Šis karkasas buvo pasitelktas tolimesniems testavimams, kurių metu ITAPC administratorė suvedinėjo realius duomenis, išsakė savo pastabas bei reikiamas įgyvendinti funkcijas ateityje, kurios pilnai pateisintų ITAPC apskaitos sistemos naudą ir našumą. Šio kursinio darbo metu paaiškėjo, jog tokia sistema yra tikrai reikalinga bei turinti potencialą, tad buvo nuspręsta ją vystyti ir toliau. Bakalaurinio darbo metu pasitelktas karkasas buvo gerokai praplėstas duomenų ir formų atžvilgiu bei jam buvo suteiktas visas apsibrėžtas pagrindinis funkcionalumas. Iš paprasto duomenų suvedimo karkaso buvo sukurta realiai funkcionuojanti sistema, turinti įvairius automatizuotus, duomenis apdorojančius modulius, prieigą ne tik administratoriui, bet ir realiems ITAPC paslaugų užsakovams.

1.2 Verslo valdymo sistemų analizė

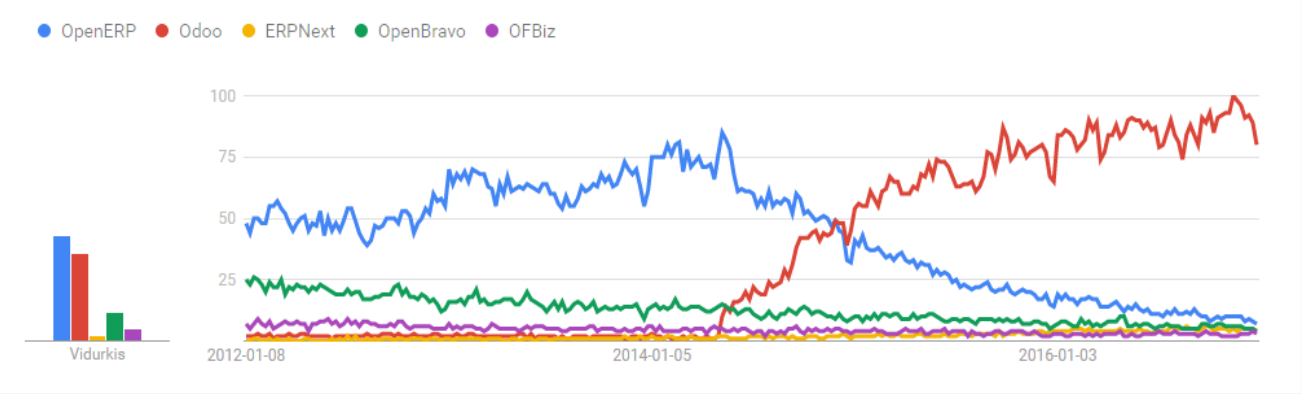
Verslo valdymo sistema (angl. ERP - *Enterprise Resource Planning*) yra speciali programinė įranga, skirta kompiuterizuoti įmonės valdymą, naudojama apskaitos vedimo palengvinimui, efektyviam visų resursų išnaudojimui, analitinės įmonės veiklos ataskaitų sudarymui ir daugeliui kitų įmonės procesų valdyti. Šios sistemos yra populiarios įvairių įmonių ar organizacijų apskaitos (bei kitų modulių) įgyvendinimo projektuose, todėl prieš įgyvendinant ITAPC apskaitos sistemą buvo domimasi, ar yra tinkama pasiūla tarp verslo valdymo sistemų, kuri tenkintų konkrečius ITAPC apskaitos sistemai iškeltus poreikius. Pagrindinis išsikeltas kriterijus buvo šių sistemų kaina, ieškoma buvo nemokamų variantų, nes projektas yra nebiudžetinis, tad buvo vykdoma paieška tarp atvirojo kodo (angl. *Open source*) verslo valdymo sistemų pasiūlymų skirtingose rinkose (Lietuvos ir užsienio). Sekančiuose pavyzdžiuose bus analizuojamos trys (viena – Lietuvos ir kitos - užsienio) verslo valdymo sistemos, pristatant įvairius jų privalumus, trūkumus bei galutinį sprendimą, kodėl jomis nebuvo naudojamasi kuriant ITAPC apskaitos sistemą.

- ***Contour Enterprise* lietuviška verslo valdymo sistema**

Contour Enterprise [1] yra ERP klasės atviro kodo apskaitos ir verslo valdymo sistema, orientuota į lietuviškas stambaus ar vidutinio dydžio įmones. Daugiausiai ši valdymo sistema yra pritaikoma įvairaus pobūdžio buhalterinėms apskaitoms, kas iš pradžių pasirodė kaip tinkamas variantas ITAPC projektui. Kaip pateikta oficialiame puslapyje, šia sistema jau naudojasi tikrai daug garsių ir didelių Lietuvos įmonių, ji yra prieinama ir lengvai atsisiunčiama kiekvienam užsiregistravusiam vartotojui. Pagrindiniai sistemos pliusai yra tokie, jog sistema yra itin plati, naujoviška, apimanti įvairaus pobūdžio apskaitas, logistiką, biudžetą, dokumentų ir net projektų valdymą. Tačiau nemokamoje versijoje pasirinkti norimų paketų nėra leidžiama (teikiami yra visi paketai iš karto su ribotu funkcionalumu), tam tikros papildomos paslaugos (didesnio masto diegimas ir apmokymai) reikalauja apmokėjimų. ITAPC apskaitai tai nebuvo tinkamas variantas, kadangi toks kiekis įvairių paketų tikrai nėra reikalingas, tai nėra didelė įmonė, o vieno apskaitos modulio, tinkamo ITAPC ši sistema taipogi nesiūlo (egzistuoja buhalterinės bei pirkimo ar pardavimo apskaitos). Šiuo metu Lietuvoje nėra daug atvirojo kodo verslo sistemų pasiūlos, galbūt ateityje atsiras mažesnių ir lankstesnių, nedidelius modulius siūlančių sistemų, kurias bus galima pritaikyti tokio masto mažesniems projektams kaip ITAPC apskaitos sistema.

- ***Odoo* ir *OpenBravo* užsienietiškos verslo valdymo sistemos**

Analizuoti *Odoo* [2] ir *OpenBravo* [3] verslo valdymo sistemas buvo pasirinkta dėl pagrindinės priežasties – tai yra itin lanksčios ir populiarios pasauliniu lygmeniu sistemos, kurias galima pritaikyto mažesnės apimties projektams, įmonėms ar organizacijoms.



1 paveikslėlis: *Google Analytics* verslo valdymo sistemų paieškų kiekiai

([*https://www.google.com/trends/explore?q=OpenERP,Odoo,ERPNext,OpenBravo,OFBiz&hl=en-US*](https://www.google.com/trends/explore?q=OpenERP,Odoo,ERPNext,OpenBravo,OFBiz&hl=en-US) **)**

Šių sistemų populiarumą iliustruoja 1 paveikslėlyje pateikta *Google Analytics* diagrama, vaizduojanti didžiausių atvirojo kodo verslo valdymo sistemų paieškų kiekį per pastaruosius penkerius metus. Rezultatuose akivaizdžiai matoma, jog ženkliai pirmaujančios yra *Odoo* bei *OpenERP* sistemos, tačiau *OpenERP* ilgainiui buvo perimtas *Odoo* ir pastaraisiais metais sparčiai populiarėja būtent pasirinktasis *OpenBravo*. Kiekviena iš jų turi milijonus vartotojų, pasižymi itin paprasta vartotojo sąsaja bei siūlo daug įvairių modulių. Pagrindiniai modulių paketai mažai skiriasi nuo analizuotos lietuviškos sistemos siūlomų, tačiau parsisiuntus jų siūlomas nemokamas versijas, naudojimui pateikiamas itin ribotas pasirinkimas. Naudojantis nemokama pradine versija ir praėjus 3 mėn. laikotarpiui tolimesnis nemokamas naudojimas yra stabdomas, taip pat yra pritaikytas limitas vartotojų skaičiui. Nemokama versija negarantuoja nuolatinio palaikymo, yra ribotos galimybės norimų funkcijų modifikacijoms. Tai yra pripažintos ir labai stabilios sistemos, tačiau nemokamos versijos nepatenkintų konkrečių ITAPC apskaitos sistemos poreikių, žiūrint į sistemos ateities vystymo perspektyvas vis vien tektų įsigyti mokamą prenumeratą. *Odoo* ir *OpenBravo* tinka visoms įmonėms, orientuotoms į profesionalumą bei kokybę resursų, finansų valdyme bei apskaitos procese, kurios gali į tai investuoti papildomų lėšų.

Atlikus pasirinktų atviro kodo verslo valdymų sistemų analizę galima teigti, jog pilnai nemokamų variantų nėra. Siūlomos nemokamos versijos yra labiau skirtos pasibandymams, tai marketingo strategija. Galbūt yra projektų, kurių poreikius tenkina ir nemokamos versijos, tačiau ITAPC apskaitos sistemos atveju, dauguma siūlomų modelių yra neaktualūs, netenkinantys konkrečių poreikių arba reikalaujantys didelio pritaikymo.

Analizuojant mokamas versijas – tai labiau didesnio masto įmonėms ir verslams pritaikyti funkcionalumai, padedantys stambaus verslo valdyme įvairiais aspektais. Siekiant naudotis šiomis sistemos ar pritaikyti jų funkcionalumą reikalingi papildomi tiek personalo, tiek ir finansiniai ištekliai. Todėl buvo priimtas galutinis sprendimas nesinaudoti jau esamų verslo valdymo sistemų paslaugomis, taip siekiant ne tik sutaupyti, bet ir sukurti tikrai unikalią sistemą, kuri pilnai tenkintų visus iškeltus ITAPC apskaitos reikalavimus, būtų paprasta naudoti ir neturėtų perteklinių funkcijų.

1.3 Elektroninio parašo galimybės

Kadangi šioje sistemoje buvo siekiama integruoti galimybę vartotojams internetu pateikti ITAPC paslaugų užsakymo paraiškas, reikėjo iš anksto atsižvelgti į šių paraiškų validavimą ir autentifikavimą. Originalios paraiškos reikalauja naudotojų parašų, todėl kuriant internetinę versiją, buvo nuspręstą paanalizuoti elektroninio parašo galimybę pasirašant užpildytas internetines paraiškas.

Remiantis LR elektroninio parašo įstatymo 8 straipsniu [4], el. parašas, „[...]patvirtintas galiojančiu kvalifikuotu sertifikatu, elektroniniams duomenims turi tokią pat teisinę galią kaip ir parašas rašytiniuose dokumentuose ir yra leistinas kaip įrodinėjimo priemonė teisme“, taigi tai yra įstatymiškai patvirtintas oficialių dokumentų pasirašymo būdas, tinkantis ITAPC paslaugų užsakymo paraiškai. Konkrečiai Lietuvoje sutarčių pasirašymo el. parašu galimybę teikia *GoSign.lt* ir *iSign.lt* interneto sistemos. Iš šių dviejų pažangesnė yra *iSign.lt,* kuri suteikia galimybę dokumentus pasirašyti ne tik PDF, bet ir ADOC formatu. Taip pat šioji internetinė sistema turi viešai prieinamus savo *ISIGN* API kodo fragmentus bei dokumentaciją. Tuo tarpu *GoSign.lt* veikia gerokai paprastesniu principu ir norint pakviesti pasirašyti sutartis, visas pasirašymo procesas turi vykti atskirai būtent *GoSign.lt* svetainėje, kas ilgainiui gali pasidaryti nepatogu. Šio darbu metu buvo nuspręsta nediegti elektroninio parašo sistemoje – pats įdiegimo ir adaptavimo procesas reikalauja tiekėjo specialistų pagalbos, taip pat atsižvelgiant į ateities perspektyvas, sistemai reikėtų daugiau nei nemokamos versijos, kuri neribotų saugojamų ir pasirašomų dokumentų kiekio, suteiktų papildomų pasirašytų dokumento apdorojimą palengvinančių funkcijų.
Todėl plečiant sistemą ateityje ir investuojant daugiau lėšų, *iSign.lt* siūlomi sprendimai būtų geriausias pasirinkimas ITAPC paslaugų užsakymo paraiškoms.

1.4 Užduoties reikalavimų analizė

ITAPC apskaitos sistemos kūrimui buvo iškelti konkretūs reikalavimai, kurie darbo eigos metu šiek tiek kito, nuolatos prisidėdavo ir papildomų. Reikalavimai buvo iškelti ITAPC administratorės, buvo išskirtos svarbiausios ir reikalingiausios funkcijos, sistemos ateities galimybės bei prioritetai. Šio bakalaurinio darbo metu keliami uždaviniai buvo skiriami pagrindiniam sistemos funkcionalumo įgyvendinimui, lengvesniam darbui su duomenimis bei automatizuotam informacijos apdorojimui. Pagal prioritetą buvo išskirti šie svarbiausi funkcionalumo reikalavimai:

- Pranešimų siuntimo modulis – priminimai administratoriams ir paslaugų užsakovams, šablonų siuntimas;
- Sistemos nustatymų skiltis su galimybe keisti bei papildyti reikiamus parametrus (paslaugų tipai, pranešimų laiškai ir kt.);
- Suvestinių pateikimas, informacijos skelbimas apie artėjančius bei praleistus mokėjimus;
- Naujas automatinis mokėjimų generavimas, pritaikytas suvestinėms;
- Paslaugų formų skaitmenizavimas bei dinaminis jų atvaizdavimas, užpildytų duomenų pateiktis administratoriui;
- Esamų formų praplėtimas, papildomos duomenų validacijos, patobulinta sistemos pateiktis.

2. Sistemos projektavimas

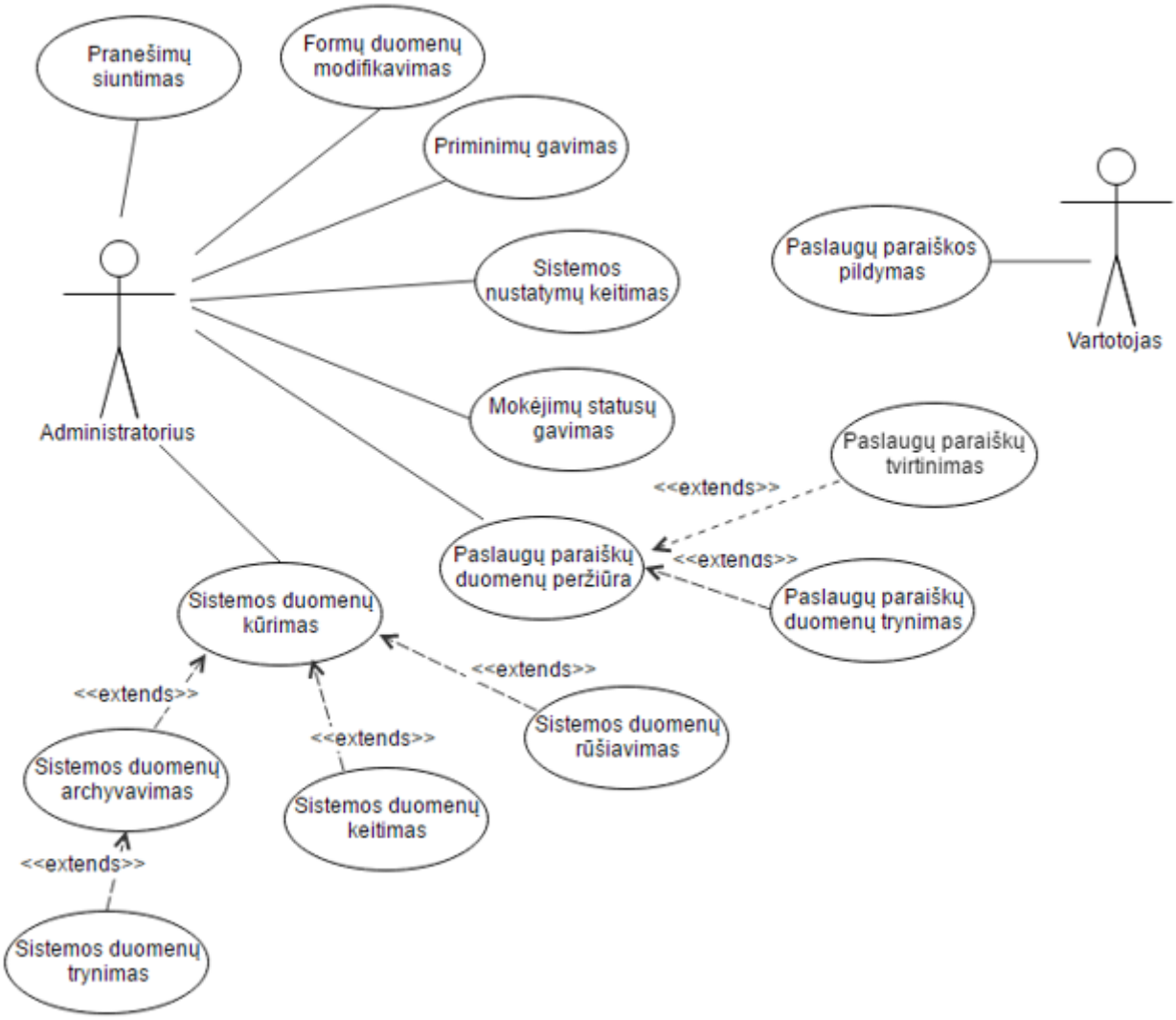
2.1 Vartotojų rolės ir funkcijos

Sistema yra pritaikyta dvejoms pagrindinėms vartotojų grupėms – ITAPC klientams ir administratoriams. Pastarieji turi visas galimas teises sistemoje, turi prieigą prie sistemos apskaitos dalies, didžiausias funkcionalumas yra pritaikytas būtent šiai grupei. Tuo tarpu ITAPC klientams prieiga galima tik prie paslaugų užsakymo formos. Visos galimos sistemos vartotojų funkcijos yra išvardintos 1 lentelėje.

Funkcija	Aprašymas
Paraiškų, mokėjimų, naudotojų ir užsakovų kūrimas	Kiekvienam kūrimui pateikiamos atskiros formos su detalia informacija, visi duomenys yra validuojami prieš išsaugant. Administratorius gali matyti padarytas pildymo klaidas bei formos kūrimo statusą.
Duomenų skaitymas ir rūšiavimas	Prisijungusiam administratoriui yra matomi visi egzistuojantys sukurti duomenys, pateiktys yra lentelėse, kuriose yra galimas duomenų rūšiavimas pagal tam tikrus jų atributus.
Duomenų archyvavimas/trynimas	Apsisaugant, administratorius negali iškart ištrinti duomenų, pirmiausiai jie yra perkeliami į archyvą iš kurio administratorius duomenis gali ištrinti visam laikui arba atstatyti atgal.
Duomenų modifikavimas	Visi sukurti duomenys yra laisvai modifikuojami administratoriaus priklausomai nuo to, kas modifikuojama, atitinkamai gali persiskaičiuoti kiti jau egzistuojantys duomenys.
Nustatymų keitimas	Administratorius turi atskirą nustatymų panelę, kurioje gali modifikuoti tam tikrus nustatymus – slaptažodžio keitimą, el. paštų priskyrimą priminimų ir pranešimų gavimui.
Duomenų pateikties modifikavimas formose	Nustatymų panelėje administratorius turi galimybę modifikuoti tam tikrus duomenis, kurie atsivaizduoja paraiškų formose. Pvz. paslaugos tipai, kuriuos panelėje administratorius gali trinti bei pridėti ir vėliau pasirinkti formose.
Pranešimų siuntimas	Mokėjimų suvestinėje administratorius turi galimybę siųsti pranešimus užsakovams su automatiškai sugeneruotu pranešimu arba redaguoti jį redaktoriaus pagalba prieš išsiunčiant.
Mokėjimų statuso gavimas	Suvestinėje administratorius gali stebėti mokėjimų statusus – kurie mokėjimai vėluoja, yra artimiausi ar turi būti siunčiami šiandien.
Paslaugų paraiškos pildymas	Paprastas sistemos vartotojas gali rinktis tarp dviejų formos tipų, ją pildyti ir taip pateikti užklausą užsakymui.
Paslaugų paraiškos duomenų peržiūra/trynimas	Administratorius gauna pranešimus apie naujas paraiškas, gali peržiūrėti jų duomenis savo panelėje, trinti juos.

1 lentelė: Sistemos vartotojų funkcijos

Siekiant geriau suvokti šias vartotojo funkcijas buvo sukurta panaudos atvejų diagrama (2 paveikslėlis), kurioje vizualiai matomas 1 lentelėje aprašytų funkcijų ir vartotojų sąryšis.

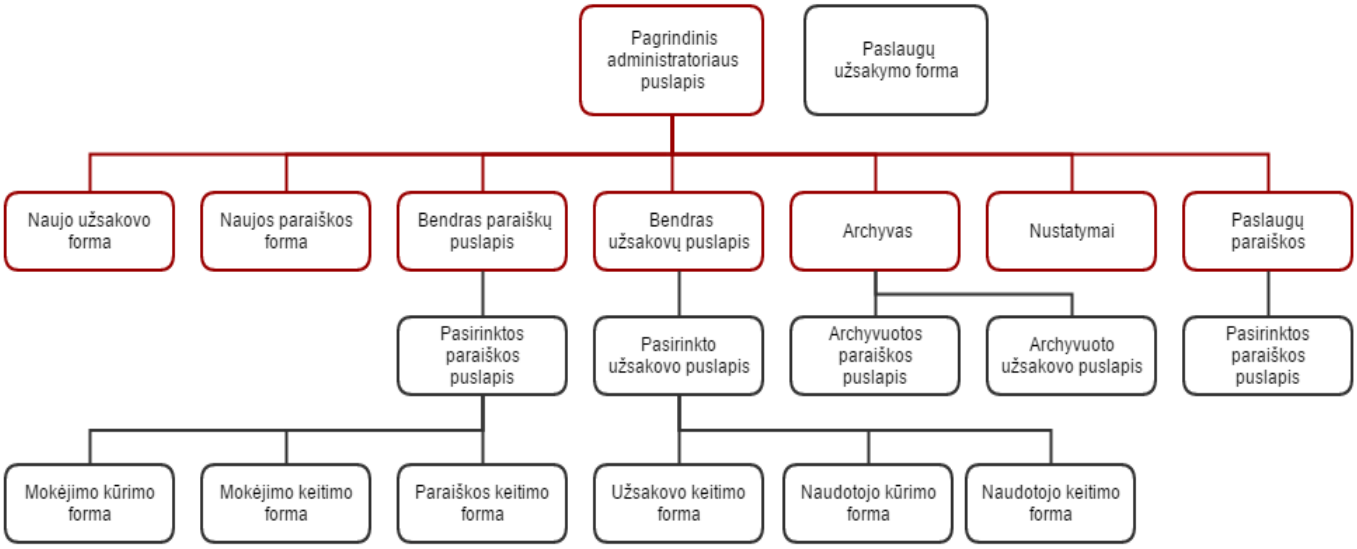


2 paveikslėlis: Panaudos atvejų diagrama

Kaip matoma pagal lentelėje ir diagramoje pateiktus duomenis, pagrindinis dėmesys sistemoje yra skiriamas administratoriaus darbui su duomenimis, egzistuoja pagalbinės funkcijos, kurios administratoriui padeda realiu laiku sekti naujus duomenis, ateinančius mokėjimus, lengviau informuoti užsakovus. Į sistemos duomenų apibrėžimą įeina paraiškų, mokėjimų, užsakovų bei naudotojų duomenys, kuriuos pildo administratorius.

2.2 Sistemos navigacija

Prieš vykdant pirminių prototipų kūrimą buvo apsibrėžti visi planuojamos sistemos puslapių langai bei sukurtas navigacinis žemėlapis. Tai padėjo išlaikyti pastovumą kūrimo metu ir sukurti logiškai susietą bei patogią vartotojo sąsają. 3 paveikslėlyje yra pateikta galutinė šio darbo metu sukurta bei išbaigta ITAPC apskaitos sistemos navigacinė schema.

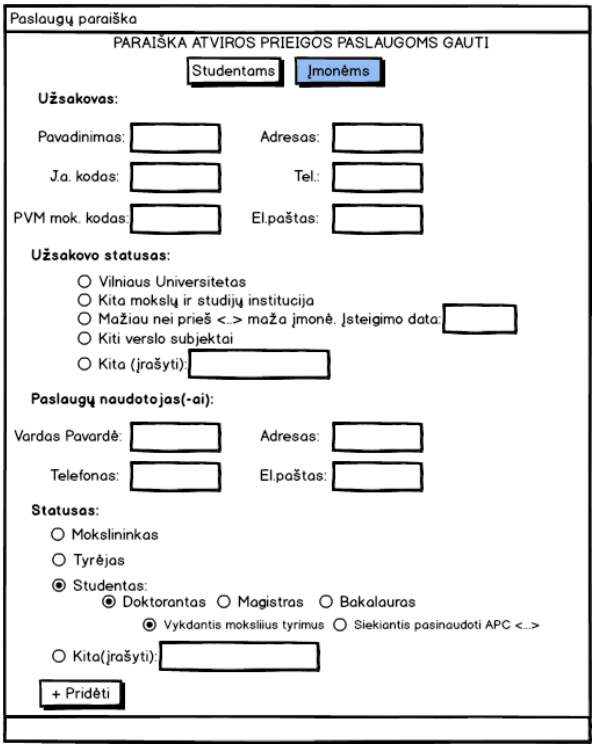


3 paveikslėlis: Sistemos navigacinė schema

Kaip matoma 3 paveikslėlyje esančioje schemoje, visi pagrindiniai sistemos langai gali būti pasiekiami iš pagrindinio puslapio. Diagramoje išskirti pirmojo lygmens puslapiai gali būti pasiekiami iš bet kurio kito sistemos puslapio per visuomet matomą navigacinę juostą. Pagrindinės ir administratoriui aktualiausios formos – užsakovo ir paraiškos – gali būti pasiekiamos patogiai per pagrindinį puslapį. Papildomai šias formas galima pasiekti ir iš kitų puslapių, kuriuose yra dirbama su užsakovo arba paraiškos duomenimis. Pasirinktos paraiškos puslapis papildomai talpina tai paraiškai sugeneruotus mokėjimus, kurių kiekvienas turi savo atitinkamas keitimo formas, taip pat iš čia galima pasiekti ir pačios paraiškos keitimo formos puslapį. Identiška struktūra egzistuoja ir užsakovų puslapyje – čia pasiekiami užsakovo keitimo, naudotojų kūrimo ir jų keitimo formų langai. Archyvas savyje talpina atskirus langus archyvuotiems užsakovams ir paraiškoms, o nustatymų langas talpina visas galima administratoriaus sistemos modifikacijas viename puslapyje. Administratoriui atskirame lange taip pat yra pasiekiamos vartotojų užpildytos paslaugų paraiškos, kiekviena iš jų turi savo atskirą langą su visais vartotojo pateiktais duomenimis. Tuo tarpu pati paslaugų užsakymo paraiška turi savo atskirą nuorodą, kuri nėra susieta su administratoriaus sistemos dalies navigacija, ši paraiška gali būti pasiekiama iš išorinio puslapio.

2.3 Prototipavimas

Prototipavimo esmė buvo sukurti bei apsibrėžti pirminę vartotojo sąsają, kuri tenkintų vartotoją, jam būtų patogi bei nusakytų galimas sistemos funkcijas. Remiantis prototipais, buvo įgyvendinamas ir pats funkcionalumas, brėžiniai padėjo išlaikyti susikurtą pirminę architektūrą. Jų kūrimui buvo pasitelktas *Balsamiq* [5] įrankis, su kurio pagalba buvo sukurti nauji langai, imituotas judėjimas per sistemą bei suteikta galimybė pasitestuoti sistemos navigaciją. Įvykdžius testavimus, prototipai buvo tobulinami pagal pageidavimus, todėl pasiektas galutinis realus sistemos variantas yra itin panašus į prototipinį. Pavyzdinį dalinį paslaugų paraiškos langą, sukurtą pasitelkiant minėtąjį įrankį, galima pamatyti 4 paveikslėlyje.



4 paveikslėlis: Paslaugų paraiškos dalinis prototipas

2.4 Pagrindinės technologijos pasirinkimas

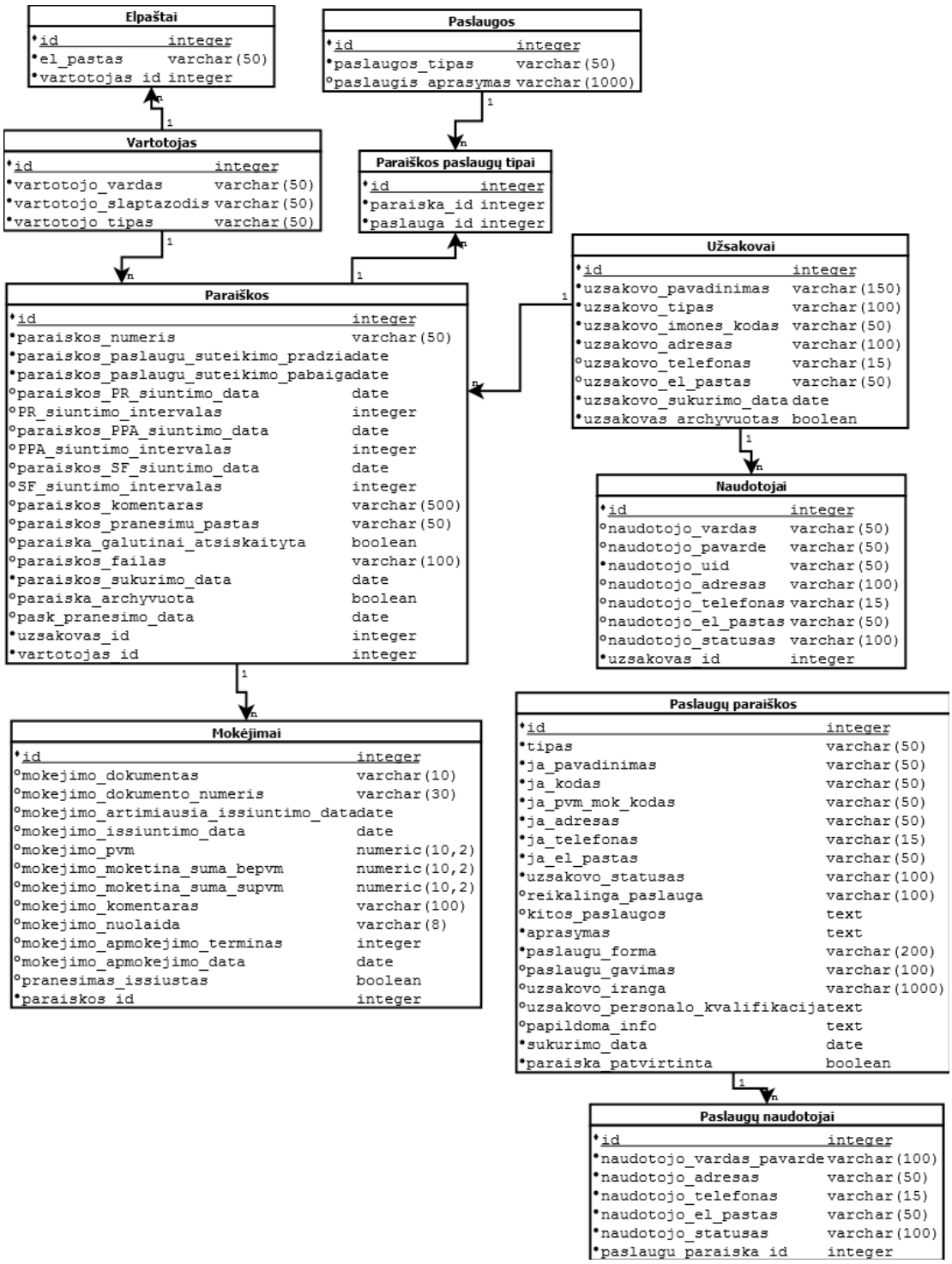
Pagrindine technologija ITAPC apskaitos sistemos kūrimui buvo pasirinktas *Django* karkasas, kuriame yra naudojama *Python* programavimo kalba (konkrečiai *Django* 1.10 ir *Python* 3.5 versijos). Rinktasi tarp daugelio kitų technologijų nebuvo, kadangi vienas iš ITAPC apskaitos sistemos reikalavimų buvo naudotis nurodytuoju *Django* karkasu, kadangi tai yra pagrindinė technologija, naudojama kuriant ir kitas fakulteto sistemas.

Bendros technologijos naudojimas apsaugoja nuo ateityje galinčių kilti integracijos problemų bei užtikrina lengvesnį darbų perdavimą kitiems specialistams, dirbantiems tuo pačiu karkasu. Papildomos priežastys, lėmusios pasirinkti būtent šį karkasą - dėka *Python* kalbos, kuri išsiskiria savo paprasta sintakse ir yra lengvai suprantama [6], išmokti dirbti karkasu, vykdyti užduotis ir perduoti kodą vystyti kitiems yra gerokai paprasčiau. Taip pat, turint omenyje jog internetinių sistemų kūrimas reikalauja daug darbo su SQL užklausomis, *Django* tampa labai pravartus, nes šių užklausų neteks maišyti su programiniu kodu. Taip yra todėl, nes *Django* pagal sukurtus modelius, rūšiavimo užklausas ar kitas komandas, skirtas duomenų bazei, automatiškai sugeneruoja lentelės, modifikuoja bei atnaujina duomenis jose. Paprastesnis darbas su duomenų bazėmis yra itin svarbus, kadangi šis projektas būtent yra orientuotas į tinkamą duomenų saugojimą bei apdorojimą, egzistuoja daug sąsajų tarp skirtingų duomenų tipų.

2.5 Duomenų bazė

Duomenų bazių technologija buvo pasirinkta *PostgreSQL,* kuri yra populiariausia *Django* karkasuose, puikiai padeda susitvarkyti su užklausomis, bei turi valdymo įrankį *pgAdmin,* kuris vizualizuoja duomenų bazės lenteles, padeda jas rūšiuoti bei modifikuoti. *Django* karkasas turi specialų modulį, skirtą būtent šiai duomenų bazių technologijai [7], kuris padeda daug paprasčiau dirbti su duomenimis - tai dar viena priežastis kodėl buvo priimtas šis technologinis sprendimas.

Pasirinkus pagrindines technologijas buvo pradėtas duomenų bazės projektavimo etapas. Šis etapas buvo pradėtas dar kursinio darbo metu, buvo sukurtos iš viso keturios lentelės – paraiškų, mokėjimų, užsakovų ir naudotojų. Jos savyje talpino dalį pagrindinių administratoriaus suvedamų duomenų. Galutiniame variante šios lentelės buvo gerokai praplėstos, priskirta daug papildomų atributų, dalyvaujančių sistemoje sukurtose automatizavimo funkcijose (duomenų pateikties, suvesties, pranešimų ir kt.). Taip pat buvo skurtos papildomos naujos lentelės tiek administratoriui, tiek naujai sukurtai ITAPC užsakovų pusei. Darbo eigoje schema nuolatos kito, kaskart įvykdžius didesnius architektūrinius sprendimus sistemoje schema būdavo perbraižoma, drauge su administratore diskutuojama, kokie papildomi duomenys dar galėtų atsirasti ateityje. Galutinis šios schemos variantas gali būti matomas 5 paveikslėlyje pateiktoje schemoje.



5 paveikslėlis: Duomenų bazės schema

Kaip matoma pateiktoje schemoje, duomenys susideda iš dviejų pagrindinių dalių – administratoriaus dalies bei paslaugų užsakovų (nesusietos ryšiais). Jos tarpusavyje neturi jokio sąryšio, taip pat administratoriaus dalis turi gerokai daugiau duomenų, tad ir pagrindinis dėmesys buvo skiriamas šiai daliai. Administratoriaus duomenys susideda iš keturių pagrindinių lentelių – paraiškų, mokėjimų, užsakovų ir naudotojų. Šios lentelės saugo paties administratoriaus suvedamus duomenis ir yra tarpusavyje priklausomos. Sukurtiems užsakovams egzistuoja vieno su daugelio sąryšis su paraiškų ir naudotojų lentelėmis.

Tai žymi, jog kiekvienam sistemoje sukurtam užsakovui galima priskirti *n* kiekį paraiškų ir naudotojų, o kiekviena paraiška ir naudotojas gali turėti sau priskirtą tik vieną unikalų užsakovą. Identiškas sąryšis egzistuoja ir tarp paraiškų bei mokėjimų duomenų. Kiekvienai paraiškai yra automatiškai sugeneruojami (arba pridedami rankomis) mokėjimai iš kurių kiekvienas išoriniu raktu yra susietas su viena unikalia paraiška. Paraiška papildomai turi daugelis su daugeliu sąryšį su paslaugomis, kadangi paslaugos yra sukuriamos atskirai administratoriaus panelėje ir jos gali būti priskiriamos neribotam kiekiui skirtingų paraiškų. Todėl šiam sąryšiui yra sukurta atskira paraiškos paslaugų tipų lentelė, siejanti paraiškas ir paslaugas.

Paties administratoriaus prisijungimo duomenys yra saugomi atskiroje vartotojo lentelėje, o šioji yra siejama vieno su daugeliu sąryšiu su elektroninių paštų ir paraiškų lentelėmis. Elektroniniai paštai yra išskirta kaip atskira lentelė, nes tai yra atskiras masyvas, kuris yra kuriamas pasitelkiant bibliotekos pagalbą, tai pranešimams skirti paštai, kurių administratorius gali prisiskirti norimą kiekį. Paslaugų paraiškų ir paslaugų naudotojų lentelės saugo išorinio sistemos naudotojo užpildytus duomenis, jos nėra siejamos su administratoriaus dalimi, joje yra tik atvaizduojamos. Saugant paraiškų formą, vienu metu yra išsaugojami duomenys į dvi lenteles, kadangi tai yra jungtinė forma, kurioje pildant galima dinamiškai pridėti norimą kiekį naudotojų. Taip pat dėl tos pačios priežasties šios lentelės yra siejamos vieno su daugeliu ryšiu, kiekvienas pridėtas naudotojas turi jam priskirtą paslaugų paraiškos identifikacinį numerį kaip išorinį raktą.

3. Sistemos įgyvendinimas

3.1 Serverio konfigūracija

Kūrimo metu sistema buvo talpinama lokaliame serveryje, šia konfigūracija bei sistemos paleidimu automatiškai pasirūpina *Django* karkasas, pakanka pasinaudoti tinkamomis komandomis bei susieti sistemą su duomenų baze (šiame projekte tai *PostgreSQL*). Norint tinkamai susieti projektą su duomenų baze bet kuriuo atveju reikalinga tinkamai pakeisti projekto nustatymo faile esančius duomenis, nustatyti esamą duomenų bazės adresą, pavadinimą bei prisijungimo duomenis prie jos. Šiai konfigūracijai bei duomenų bazės susiejimui buvo panaudotas specialus *psycopg2* paketas [8].

Kadangi lokaliai paleista sistema nėra pasiekiama kitiems, buvo pasinaudota *OpenNebula* VU MIF debesijos teikiamomis galimybėmis ir projektas darbo metu buvo paleistas virtualioje mašinoje iš kurios galėjo būti pasiekiamas. Konfigūracija virtualioje mašinoje yra identiška lokaliai, visas siejimas su duomenų bazėmis, paketų diegimas bei sistemos paleidimas yra toks pats. Vieninteliai papildomi žingsniai – pačios virtualios aplinkos tinkamas paleidimas.

Projektui naudojamos virtualios mašinos techniniai duomenys:

- **Disko talpa: 5GB;**
- **Operatyvioji atmintis: 512MB;**
- **Operacinė: Debian 8;**

Šie duomenys yra tikrai pakankami, dabartinė projekto stadija sunaudoja mažai resursų (užima apie 1,5/5GB), tačiau esant projekto plėtimosi planų bei galimybių, buvo pasirinkti kiek didesni resursai paliekant vietos plėstis. Taip pat virtualioje mašinoje yra įdiegti visi reikalingi paketai darbui su projektu. Ši mašina yra prieinama nepastoviai tik labai trumpam periodui, tad buvo naudojama tik testavimo tikslais, norint jog sistema būtų pasiekiama nuolatos, ateityje reikėtų sukurti pastovią virtualią mašiną su tais pačiais resursais.

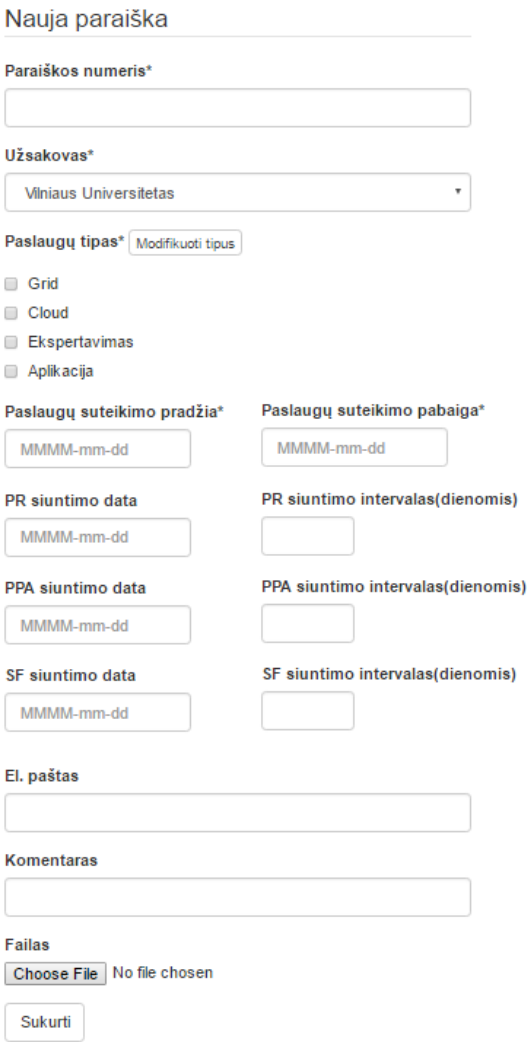
3.2 Puslapių kūrimas

Įgyvendinus planavimo dalį – nubraižius galutinius prototipus bei susikūrus navigacinę schemą, pagal tai buvo kuriami realūs sistemos svetainės langai. Pasitelkiant HTML, CSS, *Javascript* bei *jQuery* technologijas buvo sukurti visi svetainės administratoriaus ir užsakovų dalies puslapiai, navigacija, formos bei lentelės. Kiekvienas puslapis yra laikomas *Django* karkaso šablonu ir yra laikomas kiekvienai aplikacijai sukurtame atitinkamame šablonų aplankale.
Šioje fazėje pasitarnavo pasirinktas karkasas – dažnai arba kiekviename puslapyje pasikartojantys elementai (pvz. navigacinė juosta, stiliai ir skriptai) neturėjo būti perrašomi iš naujo kiekvienam puslapiui, jie yra sukelti baziniam šablone, kurį karkaso *include* metodu praplečia kiti šablonai ir taip paveldi visas bazinio šablono savybes bei elementus. Tai padėjo ir įgyvendinant dinamines formas, susidedančias iš kelių atskirų šablonų, kurie būdavo modifikuojami pagal parinktis to vartotojui nepastebint. Visi sukurti puslapiai bei formos buvo testuojami drauge su ITAPC administratore, juos nuolatos papildant bei tobulinant pagal išsakytus prašymus bei pastebėjimus.

Šio darbo metu dizainas nebuvo laikomas prioritetu, svarbiausia buvo užtikrinti, jog sistema veiktų sklandžiai ir įgyvendintų visus poreikius. Esamam pradiniam dizainui įgyvendinti buvo pasitelkta *Bootstrap* [9] biblioteka, kuri leido tam tikriems elementams suteikti labai paprastą, bet tvarkingą dizainą, taip sutaupant daug laiko. Šioje sistemoje orientuojamasi į paprastą bei minimalistinį dizainą – informacija yra išskaidyta, siekta kuo mažiau apkrauti puslapius, aiškūs bei ryškūs mygtukai, tvarkingos formos bei lentelės. Sistema yra skirta darbui, tad ji neturėtų būti perkrauta ar kelianti nuovargį dirbant su ja. Ateityje dizainas gali būti labiau pritaikomas ir derinamas su kitomis fakulteto sistemomis.

3.3 Formų kūrimas

Formos yra svarbiausias komponentas ITAPC apskaitos sistemoje, su jomis yra atliekamas pagrindinis darbas, per jas yra saugojama bei validuojama visa svarbiausia informacija. Kuriant paraiškas, užsakovus, naudotojus ar teikiant paslaugų paraiškas yra kaskart pateikiamos formos užpildymui. Kiekviena iš jų yra apibrėžta atitinkamoje aplikacijoje specialiame *forms.py* faile, jame yra tiksliai nusakoma kokie formos laukai bus sukurti, kokie egzistuos prierašai ar papildomos funkcijos prie laukų. Tam yra pasitelkiama atitinkama karkaso *Forms* klasė, kuri suteikia specialias funkcijas laukelių bei formų kūrimui, jų apribojimui, validacijoms. Formos yra atitinkamų objektų modelių atitikmenys, jose gali būti atvaizduojami ir saugomi tik egzistuojantys objekto atributai. Pagrindinės administratoriaus formos – paraiškų formos – pavyzdį galima pamatyti pateiktame 6 paveikslėlyje. Tokio paties pobūdžio formos yra sukurtos ir užsakovų, naudotojų bei mokėtojų administratoriaus pildomiems duomenims.



6 paveikslėlis: Administratoriaus paraiškos forma

3.3.1 Formų validacija

Apsirašius formą, norimame HTML šablone įdedama sukurtos formos žymė ir karkasas sugeneruoja pilną formą, kuri yra pateikiama sistemos vartotojui. Formos saugojimą, atvaizdavimą ir validavimą atlieka viena funkcija. Siunčiant užpildytą formą yra naudojamas POST metodas, jeigu sistema neatpažįsta šio metodo (t.y. vartotojas nesiunčia formos) yra atvaizduojama tuščia forma, kitu atveju kuomet naudojamas POST metodas yra validuojami visi įvesti duomenys ir forma arba yra išsaugojama į duomenų bazę, arba yra grąžinamos atitinkamos klaidos ir forma neišsaugoma. Kiekviena forma, priklausomai nuo joje esančios informacijos tipo bei kiekio, turi skirtingus validacijos metodus ir galimas klaidas.

Klaidų pranešimai taipogi yra aprašyti *forms.py* faile, nurodant koks klaidos pranešimas turi pasirodyti pažeidus tam tikrą modeliuose nustatytą atributo taisyklę. Taip pat tam tikroms sistemos formoms yra sukurtos nestandartinės validacijos, kurias norint įgyvendinti buvo perrašomi atskirų formos laukų saugojimo metodai (*Django* karkase tam naudojami *clean_field_name* metodai), nurodantys kaip turi būti nuskaitomi duomenys ir kurioje nuskaitymo vietoje gali kilti klaida.

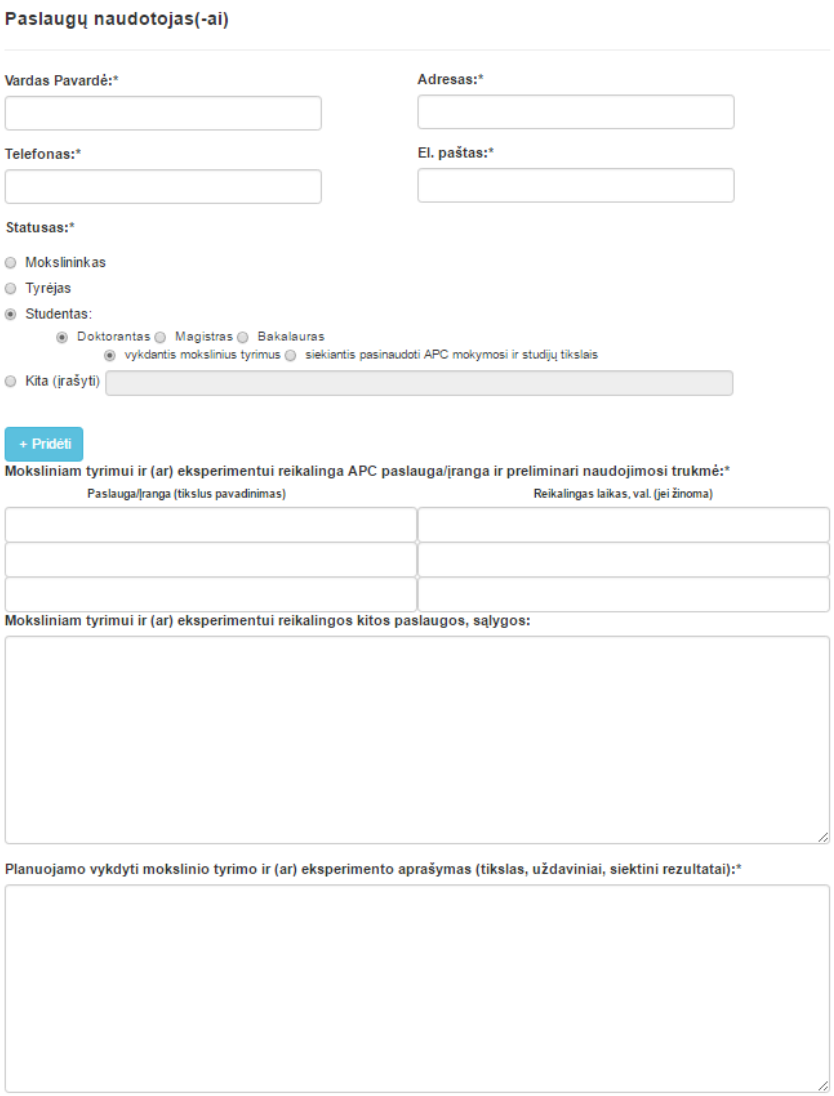
3.3.2 Paslaugų užsakymo forma

Paslaugų užsakymo forma buvo kuriama pagal konkretų egzistuojančių ITAPC paslaugų paraiškų popierinį pavyzdį [10]. Šią forma paversti skaitmenine buvo nuspręsta dėl kelių svarbių priežasčių – esama popierinė buvo itin nepatogi ir nepritaikyta didesniam kiekiui duomenų, taip pat joje egzistavo daug neaktualios informacijos, pateiktis buvo sunkiai suprantama vartotojams. Be to, kuriant šią sistemą buvo siekiama palengvinti pačių paslaugų užsakymą, modernizuoti visą procesą.

Ši forma yra visiškai kitokia nei sistemos administratoriaus dalyje egzistuojančios formos (kaip pvz. 6 paveikslėlyje pateikta forma). Paslaugų užsakymo forma yra dinaminė, susidedanti iš kelių formų rinkinių. 7 paveikslėlyje yra matoma pirmoji šios ilgos formos dalis, kurioje atsispindi formos tipo pasirinkimo galimybė ir keletas įvesties laukų. Renkantis formos tipą, tame pačiame formos šablone yra keičiamasi tarp dviejų atskirų formų – studento ir įmonės, viena iš pasirinktų yra užkraunama tame pačiame puslapyje. Šių formų laukai iš esmės nesiskiria, pasirinkus studento tipą, tam tikri laukai yra automatiškai užpildomi ir parenkami vartotojui taip palengvinant formos pildymo procesą. Tėvine forma yra laikoma įmonės forma, kuri apibrėžia visus formų laukelius, jų validacijas bei papildomus įskiepius. Studento forma paveldi viską iš įmonės formos, tik papildomai turi priskirtus tam tikrus studentui būdingus atributus prie konkrečių formos laukelių.

7 paveikslėlis: Pirmoji paslaugų paraiškos formos dalis

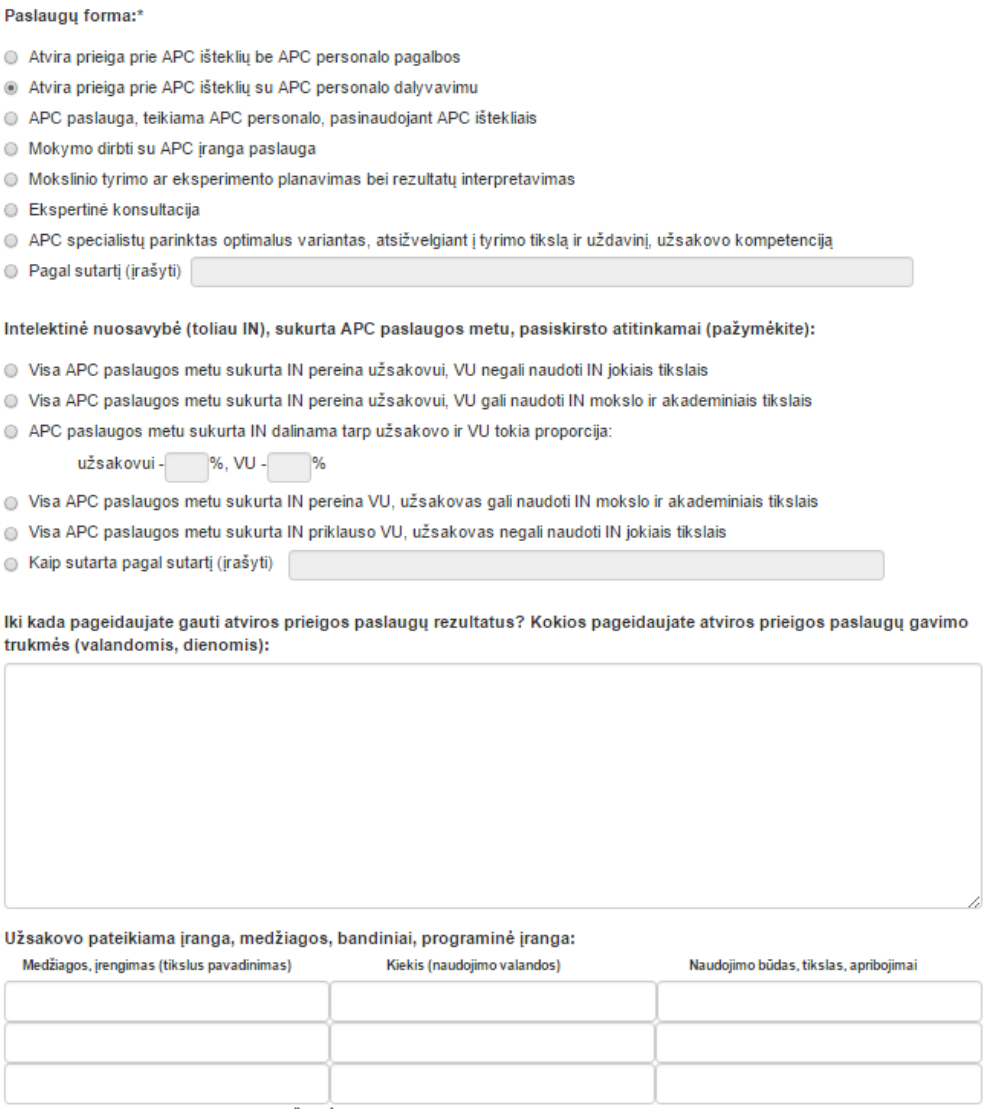
Pateiktame 7 paveikslėlyje taip pat yra matomas užsakovo statuso pasirinkimo laukas, turintis žymėjimo bei teksto įvedimo kombinaciją. Kombinuoti formos laukų *Django* karkasas lengvai neleidžia, tai nėra standartinė funkcija. Siekiant užpildyti šį trūkumą bei sukurti kitokio pobūdžio laukus buvo pasitelkti nestandartiniai laukų valdikliai (angl. *custom field widgets*), kurių dėka galima buvo susikurti nestandartinius laukų tipus savarankiškai (pvz. trigubi ar dvigubi laukai, pasirinkimo ir įvesties laukai viename). Susikurtiems nestandartiniams laukams prisireikė ir nestandartinio duomenų apdorojimo saugojimo metu – priklausomai nuo parinkties (angl. *radio*) rakto, išsaugojamos yra viena arba kelios reikšmės. Jeigu vartotojas formos užsakovo statuse pasirenka opciją, kuri leidžia pačiam vartotojui įvesti reikšmę, į duomenų bazę yra išsaugomas masyvas, kurio pirmasis elementas yra parinkties raktas (pvz. raktas *A* žymi reikšmę *Vilniaus Universitetas*), o antrasis – vartotojo įvesties tekstas. Kitu atveju yra išsaugomas tik raktas, pagal kurį vėliau administratoriaus panelėje yra paimama ir atvaizduojama atitinkama reikšmė. Taip pat šioje vietoje buvo pasinaudota *jQuery* technologija pačiame formos šablone, siekiant dinamiškai aktyvuoti ir deaktyvuoti įvesties laukus pagal vartotojo pasirinkimą.



8 paveikslėlis: Antroji paslaugų paraiškos formos dalis

8 paveikslėlyje pateiktoje antrojoje paslaugų formos dalyje matyti keli standartinės teksto įvesties laukai, valdiklio pagalba sukurti dvigubi laukai (sudarantys lentelę) bei paslaugų naudotojų skiltis. Būtent paslaugų naudotojai ankščiau būdavo surašomi į visiškai atskirą formą ir prisegami prie pagrindinės, taip siekiant apeiti prieš tai paminėtą problemą, jog pagrindinė forma nėra pritaikyta didesniam duomenų kiekiui, šiuo atveju – daugiau negu vienam naudotojui. Todėl sprendžiant šią problemą buvo sukurtas specialus naudotojų formų rinkinys (angl. *formset*) su kuriuo formos pildymo metu dinamiškai galima pridėti ir atimti norimą kiekį naudotojų. Tačiau formų rinkiniai suteikia funkcionalumą tik iš vidinės karkaso pusės, norint tai pritaikyti dinamiškai pačioje formoje, reikėjo savarankiškai sukurti vartotojo pateikties logiką išorinėje sistemos dalyje. Pirmiausiai šablone yra išskiriamas atskiras elementas, kuris ir bus kaskart dubliuojamas vartotojui pridedant naują naudotojo formą. Šis elementas savyje talpina tą pati naudotojų formų rinkinį, tačiau su specialiu *empty_form* atributu, kuris leidžia manipuliuojant prefiksais dubliuoti tuščias formas.

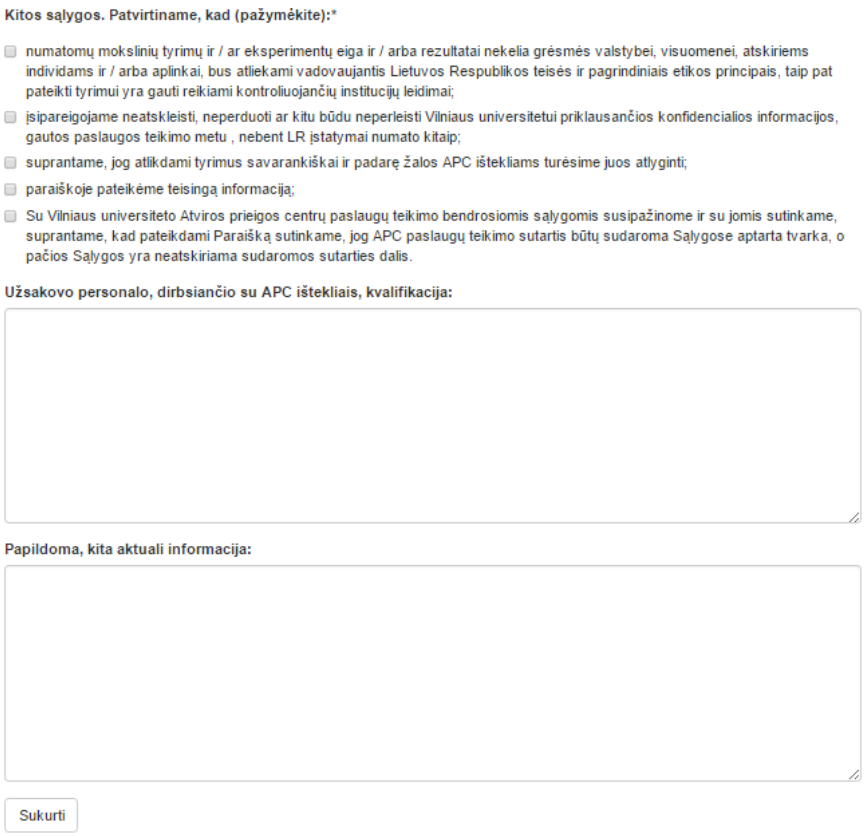
Vartotojui pasirinkus pridėti papildomą naudotoją, *jQuery* pagalba tuščia forma yra kopijuojama, jai ir jos visiems laukams suskaičiuojamas bei priskiriamas atitinkamas prefiksas (prefiksas masyvo pagrindu žymi kelinta tai forma formų rinkinyje) taip pat pakeičiamas ir bendras naudotojų formų skaičius (šis skaičius perduodamas saugojimo metu į vidines karkaso funkcijas). Nukopijuota forma su atitinkamais prefiksais yra įkraunama jau į vartotojui matomą naudotojų formų rinkinio elementą šablone. Tas pats galioja ir trinant naudotojus – prefiksų skaičius mažinamas, o trinamo naudotojo forma yra pašalinama iš rinkinio.



9 paveikslėlis: Trečioji paslaugų paraiškos formos dalis

9 paveikslėlyje pateiktoje trečiojoje paraiškos formos dalyje, matomi tie patys nestandartiniai laukeliai. Siekiant spręsti egzistavusios formos problemas dėl nepatogumo, vartotojui renkantis paslaugų formą, pagal pasirinkimą atitinkamai paslepiama arba pateikiama intelektinės nuosavybės formos dalis, taip forma labiau prisitaiko prie paties vartotojo.

Taip pat vidinėje karkaso dalyje validuojami yra tik konkretūs vartotojo pasirinkti laukai (nors visos vieno tipo parinktys yra laikomos bendru vienetu) drauge su vartotojo įvestimis, grąžinami konkretūs klaidų pranešimai.



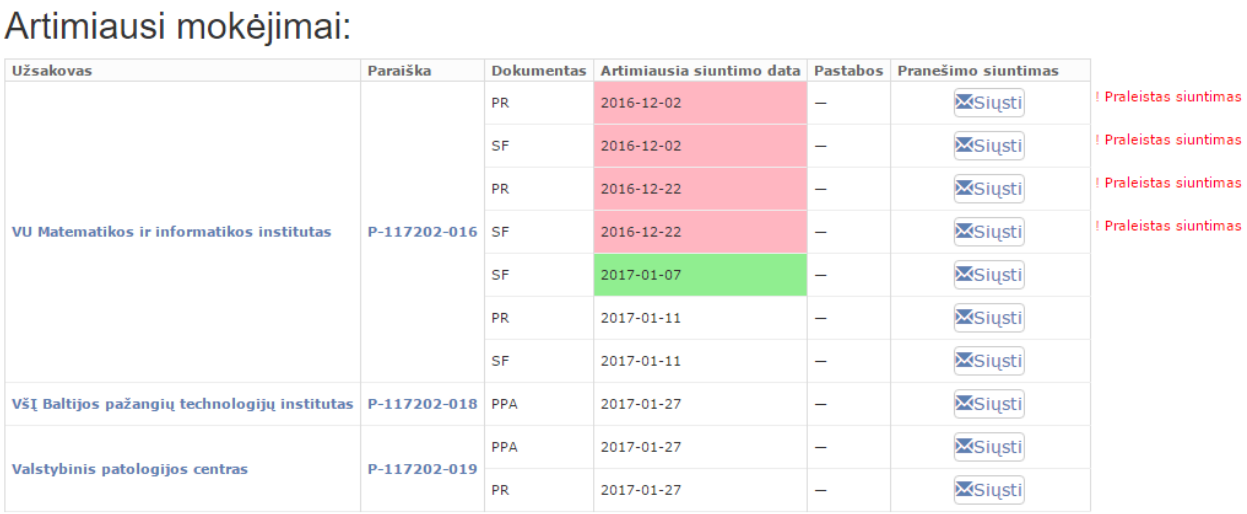
10 paveikslėlis: Ketvirtoji paslaugų paraiškos formos dalis

Paskutinėje 10 paveikslėlyje pateiktoje paslaugų paraiškos formos dalyje be standartinių teksto laukelių yra pateikti ir sąlygų patvirtinimo punktai, kurie duomenų bazėje nėra saugomi, tik yra atliekama papildoma validacija, užtikrinanti, jog vartotojas yra pažymėjęs bei sutinka su visais išvardintais punktais. Paspaudus sukūrimo mygtuką ir sėkmingai praėjus tiek pagrindinės formos tiek ir naudotojo formų rinkinio validacijas abi formos yra išsaugomos duomenų bazėje bei susiejamos paraiškos identifikaciniu numeriu kaip išoriniu raktu.

3.4 Mokėjimų suvestinės kūrimas

Viena aktualiausių administratoriui sistemos funkcijų – paprasta ir patogi mokėjimų suvestinė. Šis funkcionalumas buvo vienas pagrindinių sistemos pageidavimų, kurio labai trūko ITAPC administratorei dirbant su ITAPC duomenimis. Tai pagrindiniame administratoriaus puslapyje pateikiama lentelė, kuri surenka visose paraiškose egzistuojančius mokėjimus bei pateikia juos administratoriui atitinkama tvarka pagal artimiausią siuntimo datą. Šios datos nurodo kada ir kokio pobūdžio (pagal mokėjimo dokumentą) turi būti išsiųstas pranešimas paslaugų užsakovui.

Taip pat suvestinė fiksuoja ir žymi praleistus arba šią dieną siunčiamus mokėjimus, skelbia įspėjimus, suteikia galimybę greitai išsiųsti pranešimus užsakovams. Galutinio suvestinės varianto pavyzdys gali būti matomas pateiktame 11 paveikslėlyje.

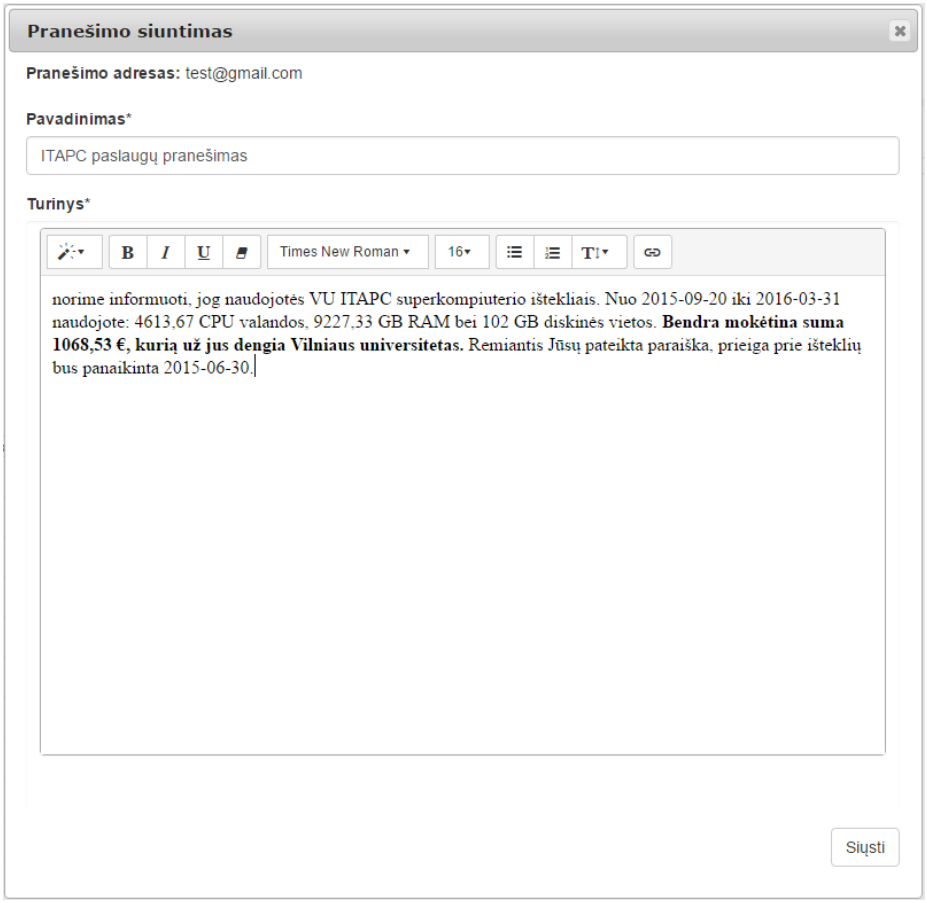


11 paveikslėlis: Administratoriaus mokėjimų suvestinė

Siekiant sukurti tokią suvestinę buvo pasitelkta *django-tables2* [11] biblioteka, kuri padeda lengviau sukurti lentelių struktūras, apsibrėžti atributus, pateikiamų duomenų rinkinius, stilius bei suteikia galimybę lengvai įgyvendinti norimą rūšiavimą. Atskirame *tables.py* faile esančiai suvestinės lentelei yra perduodami du objektai – paraiškos ir mokėjimai. Visi mokėjimų objektai yra filtruojami pagal artimiausią siuntimo datą, tuomet papildomai atliekami palyginimai su esama data ir taip priskiriamų arba šiandien siunčiamų mokėjimų klasės bei įspėjimai. Papildomai lentelėje kiekvienam mokėjimui yra priskiriama jo paraiška ir užsakovas su nuorodomis, nukreipiančiomis į konkrečius objektus. Suvestinė kas kartą užkrovus puslapį atsinaujina, duomenys yra pakartotinai nuskaitomi, todėl pakitus esamai datai ar išsiuntus mokėjimo pranešimą, lentelėje duomenys dinamiškai atsinaujins ir visuomet atitiks realią situaciją.

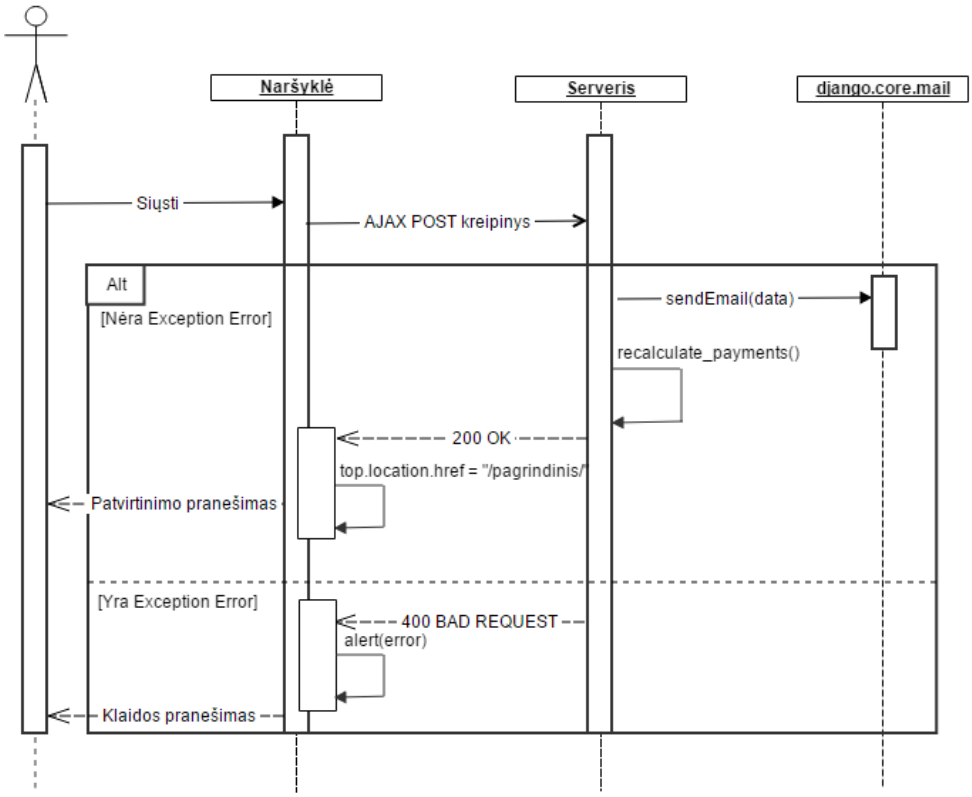
3.4.1 Mokėjimų suvestinės pranešimai

ITAPC apskaitos sistemos administratoriui buvo išreikštas poreikis turėti galimybę lengvai informuoti užsakovą apie reikiamus mokėjimus, greitai ir paprastai surinkti informaciją. Todėl įgyvendinant šį poreikį mokėjimų suvestinėje buvo sukurta galimybė vieno mygtuko paspaudimu sukurti pranešimą ir išsiųsti jį konkrečiam užsakovui, priskirtam pasirinktai paraiškai. Pasirinkus siųsti pranešimą, administratoriui yra atidaromas modulinis langas, kuris yra pateiktas 12 paveikslėlyje.



12 paveikslėlis: Pranešimo siuntimo langas

Šiame lange automatiškai yra paimami gavėjo adreso ir žinutės duomenys. Pateiktame redaktoriuje matomas pranešimas yra tik redaguotina dalis, kuri galutiniame variante yra apjungiama su pilnu pranešimo šablonu ir siunčiama kaip vienas vientisas pranešimas. Pranešimo siuntimo lange pritaikytas WYSIWYG (liet. Kaip Matai Taip Ir Bus (KMTIB)) redaktorius, naudojant *django-summernote* [12] redaktoriaus įskiepį. Modulinio lango, pranešimo siuntimo ir apdorojimo logika buvo sukurta naudojantis *Ajax* įvykių sistemą [13], integruojant ją drauge su *jQuery* bei *Django* karkaso elektroninių laiškų siuntimo moduliu *django.core.mail* [14]. Kuriant šį elektroninių laiškų siuntimo funkcionalumą buvo iškeltas specifinis poreikis turėti galimybę tai daryti paprastai viename puslapyje. Tokiu atveju *Django* karkasas yra ribotas, nes nesuteikia galimybės asinchroniškai užkrauti URL adresų. Siekiant užpildyti šį trūkumą ir įgyvendinti apsibrėžtą funkcionalumą buvo pasinaudota *Ajax* suteikiama asinchroninių užklausų galimybe, kuri sukuria papildomą sluoksnį tarp naršyklės ir serverio [15]. Galutinį pranešimų siuntimo procesą iliustruoja 4 diagramoje pateikta sekos diagrama.



13 paveikslėlis: Pranešimų siuntimo sekos diagrama

Pateiktoje diagramoje dominuoja standartinis užklausų gavimo ir atsakymų siuntimo modelis, kuris yra papildytas ITAPC apskaitos sistemoje egzistuojančiomis papildomomis funkcijomis. Pranešimų siuntimų langas yra užkraunamas *iframe* elemente, kuriame vartotojui apsirinkus siųsti pranešimą yra pateikiamas asinchroninis *Ajax* POST kreipinys į serverį. Gavus šį kreipinį egzistuoja du alternatyvūs keliai, kuriuos lemia duomenų validacijos baigtis – sėkmingai validavus duomenis, *django.core.mail* gauna duomenis per *sendEmail()* funkciją su reikiamais pranešimo duomenimis (žinutės šablonas, adresato duomenys) ir sėkmingai išsiunčia laišką (laiškų siuntimui naudojamas *Google* SMTP protokolas). Tuomet papildomai iškviečiama mokėjimų pergeneravimo funkcija (detaliau aprašyta 3.5 skyriuje), grąžinamas teigiamas atsakymas naršyklei, pakeičiamas pagrindinis URL ir pateikiamas sėkmingo išsiuntimo patvirtinimas vartotojui. Kuomet validacija būna nesėkminga iš serverio iškart yra grąžinamas netinkamos užklausos atsakymas ir vartotojui pateikiama esanti klaida. Tokiu būdu administratorius gali lengvai siųsti pranešimus iš pagrindiniame puslapyje esančios suvestinės, kuri tuo pačiu automatiškai atlieka ir paraiškų mokėjimų pergeneravimą pagal įvykusio išsiuntimo datą.

3.5 Automatinio mokėjimų generavimo kūrimas

Automatinis mokėjimų generavimas yra svarbiausias sistemoje sukurtas algoritmas, apimantis kelis modelius ir jų duomenis, suteikiantis galimybę sistemos administratoriui vykdyti realią apskaitą kuriant, sekant bei vykdant mokėjimus. Saugant kiekvieną paraišką yra tikrinami įvesti duomenų siuntimo intervalai, siuntimo bei paslaugų suteikimo datos. Esant įvestiems bent vieno dokumento duomenims ir esant validžiai paraiškai vykdomas 14 paveikslėlyje aprašytas pseudo-kodas.

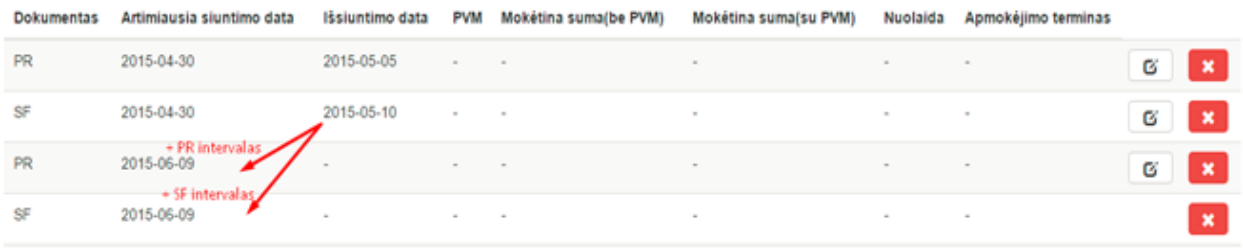
```
Paimamas esamas dokumentų sąrašas, jų datos, intervalai bei apskaičiuoti kiekvieno dokumento
siuntimo dažniai.
For dokumentas in dokumentų sąrašas:
    Priskiriama konkretaus dokumento data ir intervalas;
    Pagal intervalą apskaičiuojami kartai, kiek bus siunčiamas konkretus dokumentas.
    For x in range(0, kartai +1):
        Nurodytą  kiekį kartų yra generuojamas mokėjimas pagal dokumentą;
        Sukuriamas ir išsaugojamas mokėjimo objektas;
        Pastumiamas ir perrašomas intervalas sekančiam dokumentui.
```

14 paveikslėlis: Mokėjimų generavimo išsaugojus paraišką pseudo-kodas

Ši funkcija yra vykdoma paraiškos saugojimo metu, generavimui reikalingi kintamieji yra pasiimami saugomos paraiškos modelio funkcijų pagalba.

Bendra mokėjimų generavimo logika yra tokia – atėmus iš paslaugų suteikimo pabaigos dokumento siuntimo datą yra gaunamas laiko tarpas, kuris padalinamas iš dokumento siuntimo intervalo ir taip gaunamas kiekis kartų kiek bus siunčiamas/sukuriamas mokėjimas pagal dokumentą iki paslaugų suteikimo pabaigos. Visi reikalingi generavimo duomenys yra sutikrinami dar prieš pradedant generavimo funkciją, tad kintamieji, dalyvaujantys generavime, visada turės reikšmes ir bus validūs.

Daug sudėtingesnis ir įmantresnis yra mokėjimų perskaičiavimo algoritmas, kuriame yra atsižvelgiama kokio ir kaip buvo pakeisti mokėjimo duomenys ir pagal tai yra perskaičiuojami visi toliau einantys mokėjimai. Šis algoritmas viso darbo metu buvo perdarytas kelis kartus, siekiant surasti geriausiai poreikius tenkinantį sprendimą – sukurti globalią ir pernaudojamą funkciją. Tai yra viena globali funkcija, egzistuojanti paraiškų aplikacijoje, kuriai gali būti perduodamas keičiamo mokėjimo (siunčiant mokėjimą, pakinta jo faktinė išsiuntimo data) objektas, likusieji funkcijai reikalingi duomenys yra paimami pagal paraiškų modelyje apsibrėžtas funkcijas. Paraiškų modelyje esančių funkcijų rezultatai yra pasiekiami joms nusiunčiant konkretų paraiškos objektą, kuriam priklauso apdorojami mokėjimai. 15 paveikslėlyje pateiktoje sistemos iškarpoje matyti dalis pavyzdinių paraiškos mokėjimų.



15 paveikslėlis: Mokėjimų generavimo pavyzdinė iškarpa

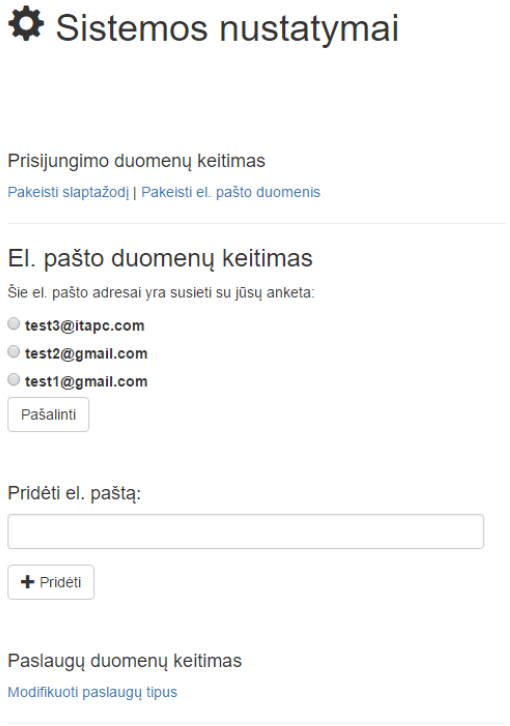
Kaip matyti pagal nubrėžtas rodykles paveikslėlyje, buvo pakeista vieno mokėjimo turinčio sąskaitą faktūrą reali išsiuntimo data. Algoritmas atpažino, kuris mokėjimas buvo pakeistas ir atitinkamai visiems po jo einantiems mokėjimams perskaičiavo artimiausias siuntimo datas. Perskaičiavimai vyksta pasikeitus tik tam tikriems dokumentams, tarkime, jeigu paraiškoje yra mokėjimų turinčių sąskaitą faktūrą, visada bus vykdomi perskaičiavimai būtent pagal mokėjimus su šiuo dokumentu, kitu atveju – pagal pirmą dokumentą masyve. Tai buvo ITAPC administratorės prašymas, kuri pateikė galimus tris mokėjimų dokumentų variantus – gali egzistuoti SF (sąskaita faktūra) su kitų tipų dokumentais, tik PPA (priėmimo-perdavimo aktas) arba tik PR (pranešimas). Atpažinus kokio tipo dokumentas yra pakeistame mokėjime ir nustačius, kad pagal jį bus vykdomas perskaičiavimas, imami paraiškoje nustatyti dokumentų siuntimo intervalai ir jie atitinkamai pridedami prie pakeisto mokėjimo išsiuntimo datos ir taip užpildomi iki pat paraiškos pabaigos (intervalų pridėjimas iliustruotas 15 paveikslėlyje). Kadangi realybėje mokėjimų dokumentai yra siunčiami nuosekliai vienas po kito yra labai svarbu, kad ir išsiuntimo/artimiausio siuntimo datos prisitaikytų bei išlaikytų nuoseklumą. Tačiau visas funkcionalumas nėra paliktas tik automatiniam generavimui – administratorius turi galimybę trinti bei sukurti mokėjimus pats, jie bus sėkmingai įtraukti ir perrūšiuoti algoritmo drauge su kitais esamais mokėjimais.

Kadangi mokėjimų perskaičiavime dalyvauja daug žingsnių, egzistuoja nemažai situacijų, kurias buvo privaloma apžvelgti ir pagal tai pritaikyti algoritmą. Kaskart vis pridedant intervalus yra tikrinama ar paskutinių mokėjimų datos neviršija paslaugų suteikimo pabaigos (kitu atveju ištrinami nereikalingi mokėjimai), koks yra skirtumas tarp artimiausios siuntimo ir išsiuntimo datų (ar nebuvo išsiųsta gerokai anksčiau nei praeiti dokumentai), ar keičiant išsiuntimo datą po to einantys mokėjimai neturi jau pakeistos datos, taip pat uždėtas apribojimas leidžiantis redaguoti tik sekančių arba jau išsiųstų mokėjimų duomenis. Visi šie apribojimai buvo atrasti ir tobulinti testavimo sesijų su fakulteto studijų koordinatore metu.

3.6 Administratoriaus nustatymų ir paslaugų panelė

Sistemos administratoriaus dalyje be duomenų kūrimo formų ir jų pateikčių, egzistuoja papildomos sistemos nustatymų ir paslaugų duomenų panelės.

Nustatymuose administratorius gali pasikeisti prisijungimo duomenis, modifikuoti susietų elektroninių paštų masyvą bei keisti tam tikrus formose pateikiamus duomenis. Pavyzdinį panelės nustatymų panelės langą galima pamatyti 16 paveikslėlyje. Elektroninių paštų ir prisijungimo duomenų funkcijoms kurti buvo panaudota *django-allauth* [16] biblioteka, kuri turi integruotas autentifikavimų, registracijos, vartotojų valdymo ir trečių šalių vartotojų autentifikavimo funkcijas.



16 paveikslėlis: Administratoriaus sistemos nustatymų panelė

Keičiant vartotojo duomenis yra reikalaujama papildomo patvirtinimo iš administratoriaus pašto, taip užtikrinant didesnį sistemos saugumą. Vartotojui priskirtų elektroninių paštų masyvas (skirtas priminimų gavimui) bibliotekos pagalba yra išsaugomas automatiškai sukurtoje duomenų bazės lentelėje ir priskiriamas vartotojui, pridedant elektroninį paštą yra drauge integruota ir jo validacija.

17 paveikslėlis: Administratoriaus paslaugų paraiškų lentelė

Administratorius taip pat turi galimybę savo panelėje stebėti ir tvirtinti naujai pateiktų paslaugų paraiškų duomenis lentelės pavidalu, kuri yra matoma 17 paveikslėlyje. Vartotojui sėkmingai užpildžius paslaugų paraiškos formą, administratoriaus sukurtam elektroninių paštų masyvui yra išsiunčiami pranešimai ir išsaugoti objektai yra iškart atvaizduojami panelėje. Tai yra nesudėtingas objekto paėmimas ir jo duomenų pateikimas konkrečiam šablonui, tačiau šioje vietoje problemiškiausia dalis buvo duomenų bazėje esančių paslaugų paraiškos duomenų nuskaitymas. Taip yra todėl, kad tam tikri atributai gali būti saugomi ir kaip paprasti masyvai ir kaip dvigubi masyvai, ir kaip paprastas vienas duomuo. Todėl sprendžiant šią problemą reikėjo sudėtingesniems atributams sukurti nestandartinius duomenų gavimo metodus. Visų pirma, visi karkaso funkcijomis paimami duomenys iš *PostgreSQL* duomenų bazės yra gražinami vienodai tekstiniu pavidalu (nors ir buvo saugomi kaip masyvas, sąrašas ar kt.), todėl šie tekstiniai duomenys buvo paverčiami iš tekstinio tipo į realų, nustatoma kiek masyve yra elementų ar egzistuoja papildomas masyvas/sąrašas jo viduje. Pagal gautus rezultatus šablonui grąžinamam kontekstui prisiskiriamos masyve gautos reikšmės (jeigu masyve rastas raktas – paimama reikšmė pagal raktą, jeigu raktas nerastas – paimam tiesioginė reikšmė esanti masyve) ir atitinkamai atvaizduojama šablone, kuris visuomet vykdo gaunamo konteksto atpažinimą.

3.7 Pranešimų gavimas

Siekiant, jog sistemos administratorius visados būtų informuotas apie svarbiausius duomenų pokyčius, be mokėjimų suvestinės buvo sukurti ir specialūs pranešimai. Šie pranešimai yra atsakingi už administratoriaus informavimą kuomet užpildoma nauja paslaugų paraiška, praleidžiamas mokėjimas arba yra atėjusi mokėjimo data. Pranešimai bus visuomet siunčiami nustatymuose pateiktam elektroninių paštų masyvui (aprašyta 3.6 skyriuje). Siunčiant pranešimus dėl naujai užpildytų paslaugų paraiškų naudoti papildomų technologijų nereikėjo – sėkmingai validavus ir į duomenų bazę išsaugojus paraišką, pasitelkiant *django.core.mail* laiškų siuntimo funkciją, sugeneruojamas standartinis pranešimas su tikslia nuoroda, vedančia į administratoriaus panelėje esančius paraiškos duomenis. Mokėjimų pranešimai nėra iškviečiami kažkokių konkrečių funkcijų, tam turi vykti nuolatinis duomenų patikrinimas, todėl buvo pasitelkti *cron jobs,* veikiantys *Unix* sistemų pagrindu. Konkrečiai šiam atvejui buvo panaudota *django-crontab* [17] biblioteka, kuri suteikia galimybę itin paprastai nustatymuose sukurti reikiamų pasikartojančių funkcijų tvarkaraščius (pagal *cron* kodą). Pranešimams buvo sukurtas vienas tvarkaraštis – kiekvienos dienos 6h ryto iškviesti funkciją *check_payments_dates()*, kuri patikrintų ar kurių duomenų bazėje esančių mokėjimų datos sutampa su atėjusia arba patapo ankstesnės nei atėjusi. Tokiais atvejais pasitelkiant tą patį *django.core.mail* išsiunčiami pranešimai nurodytiems adresams, apie sistemoje esančius praleistus arba reikiamus tą dieną išsiųsti mokėjimus.

Išvados ir ateities darbai

Šio darbo metu sėkmingai pavyko sukurti sistemą, kuri tenkina darbo pradžioje iškeltus poreikius. Galutinė ITAPC apskaitos sistema palengvina administratoriaus darbą, suteikdama galimybę lengvai vesti ir pilnai apdoroti norimus duomenis, gauti suvestines, siųsti pranešimus, generuoti mokėjimus, valdyti duomenis ir nustatymus. Pats paslaugų užsakymo procesas vartotojams taip pat tapo paprastesnis – sukurta internetinė paslaugų paraiškų forma, kuri yra dinaminė, lengvai suprantama ir prisitaikanti prie vartotojo tipo. Sistemos dėka, administratorius yra nuolatos informuojamas apie mokėjimų pokyčius, užpildytas paraiškas ir jų duomenis. Tai padeda sistemingiau planuoti ir atlikti darbus, išvengti klaidų, lengviau pateikti informaciją užsakovams. Poreikiams įgyvendinti nebuvo pasitelktos jokios jau egzistuojančios verslo valdymo sistemos, kuriant unikalią sistemą buvo ne tik sutaupyti kaštai, bet ir paliekama galimybė lanksčiai vystyti ir modifikuoti sistemą, kuri būtų specialiai pritaikyta ITAPC.

Sukurta ITAPC apskaitos sistema yra pritaikyta praktiniam naudojimui, administratoriaus dalies funkcijos yra išbaigtos. Ateityje paslaugų paraiškų teikimui rekomenduojama implementuoti šio rašto darbo 1.3 skyriuje aprašytas elektroninio parašo technologijas, taip pilnai užtikrinant paraiškose pateikiamų duomenų autentiškumą. Sistemą būtų galima vystyti toliau plečiant vartotojo dalį (pateikiant daugiau informacijos apie ITAPC resursus), pridedant papildomų funkcijų administratoriaus panelėje – sunaudotų ITAPC resursų skaičiuokles, papildomą darbuotojų apskaitą.

Šaltinių sąrašas

[1] *Countour Enterprise* VVS, URL: http://contourenterprise.lt/ , 2017-01-03.

[2] *Odoo* Documentation, v10.0, URL: http://www.odoo.com/documentation/10.0/index.html, 2017-01-03.

[3] *OpenBravo* Documentation, v3.0, URL:http://wiki.openbravo.com/wiki/Main_Page, 2017-01-03.

[4] LR Teisės Aktų Registras, LR elektroninio parašo įstatymas, URL: https://www.e-tar.lt/portal/lt/legalAct/TAR.382345294FBF/TAIS_463802, 2014-01-01.

[5] *myBalsamiq* Documentation, URL: https://docs.balsamiq.com/mybalsamiq/, 2017-01-03.

[6] S. Holderness , *Why Pyhon?*, Code School Blogs, URL: http://codeschool.com/blog/2016/01/27/why-python/, 2016-01-27.

[7] Django documentation, django.contrib.postgres module, v1.9, URL: https://docs.djangoproject.com/en/1.9/ref/contrib/postgres/, 2017-01-03.

[8] Federico Di Gregorio, psycopg2 - Python-PostgreSQL Database Adapter, v2.6.2 Python Package Index, URL: https://pypi.python.org/pypi/psycopg2/, 2017-01-03.

[9] Bootstrap, Global CSS Settings, v3.3.7, URL: http://getbootstrap.com/css/, 2017-01-03.

[10] VU MIF ITAPC paslaugų užsakymas, URL: http://mif.vu.lt/lt3/struktura/itapc#paslaugų-užsakymas, 2017-01-03.

[11] Bradley Ayers, Table/data-grid framework for Django, v1.2.9, Python Package Index, URL: https://pypi.python.org/pypi/django-tables2, 2017-01-04.

[12] Park Hyunwoo, WYSIWYG plugin for Django, v0.8.6, Python Package Index, URL: https://pypi.python.org/pypi/django-summernote/0.8.6, 2017-01-04.

[13] Ajax jQuery API documentation, v3.1, URL: https://api.jquery.com/category/ajax/, 2017-01-04.

[14] Django official documentation, django.core.mail module, v1.10, URL: https://docs.djangoproject.com/en/1.10/topics/email/, 2017-01-04.

[15] Jesse James Garrett, *Ajax: A New Approach to Web Applications*, URL: https://courses.cs.washington.edu/courses/cse490h/07sp/readings/ajax_adaptive_path.pdf, 2007-03-23.

[16] Raymond Penners, Integrated set of Django applications addressing authentication and registration, v0.30.0, Python Package Index, URL: https://pypi.python.org/pypi/django-allauth, 2017-01-01.

[17] Lars Kreisz, crontab powered job scheduling for Django, v0.7.1,
Python Package Index, URL: https://pypi.python.org/pypi/django-crontab, 2017-01-03.