VILNIAUS UNIVERSITETAS
MATEMATIKOS IR INFORMATIKOS FAKULTETAS
PROGRAMŲ SISTEMŲ KATEDRA

Užsiėmimų lankomumo žymėjimas, panaudojant studentų išmaniuosius telefonus

Lecture attendance marking using students' smartphones

Bakalaurinis darbas

Atliko: 4 kurso 2 grupės studentas
Marijus Siliūnas (parašas)

Darbo vadovas: Dr. Kristina Lapin (parašas)

Vilnius – 2017

TURINYS

Įvadas

Tyrimo objektas

Artimojo ryšio technologijų panaudojimas akivaizdinėse studijose.

Temos aktualumas

Užsiėmimų metu yra žymimas studentų lankomumas. Šį darbą atlieka dėstytojas kiekvieno užsiėmimo metu. Studentai kviečiami vardu ir atsiliepę yra pažymimi kaip dalyvaujantys paskaitoje. Taip nereikalingai užimamas paskaitos laikas bei atsiranda žmogiškos klaidos tikimybė – gali būti pažymėtas ne tas studentas ir pan. Tai ypač svarbu, kuomet galutinis įvertinimas priklauso ir nuo lankomumo. Dėstytojams tai papildoma atsakomybė ir jie turi patys pasirūpinti priemone, kurioje galėtų vesti studentų lankomumo žurnalą. Reikalinga sistema, kurioje būtų suvedami dėstomi dalykai, dėstantys dėstytojai bei studentų grupės ir studentai, vedamas lankomumo žurnalas.

Darbo tikslas

Sukurti dalyvaujančių užsiėmime studentų identifikavimo modulį ir juo paremtą studentų lankomumo žymėjimo sistemą, kuri sudarytų pagrindą tolimesnei plėtrai. Išvystyta sistema leis gerinti studentų ir dėstytojų bendradarbiavimą akivaizdinių studijų užsiėmimų metu.

Uždaviniai

Tikslui pasiekti keliami šie uždaviniai:

- Apžvelgti studentų automatizuoto identifikavimo, panaudojant išmaniuosius telefonus, technologines galimybes ir parinkti viena iš jų.
- Apžvelgti išmaniųjų telefonų programėlių technologinius įgyvendinimo būdus.
- Suformuluoti pagrindinius kuriamo modulio naudojimo scenarijus.
- Sukurti studentų identifikavimo modulio vartotojų sąsajos eskizus.
- Apibrėžti studentų lankomumo žymėjimo sistemos svarbiausius komponentus bei apžvelgti jų komunikavimo aspektus.

Siekiami rezultatai

- Sukurtas automatizuotas studentų lankomumo žymėjimo modelis.
- Sukurtos sistemos įgyvendinimo gairės ir sudarytas technologijų rinkinys.
- Sukurtas sistemos diegimo vaizdas.
- Pateikti studentų identifikavimo modulio vartotojo sąsajos eskizai.

1. Užsiėmime dalyvaujančių studentų identifikavimas

Lankomumo žymėjimas yra pasikartojantis ir toks pat procesas, kurį galima visiškai automatizuoti. Norint automatizuoti lankomumo žymėjimo procesą, pirmiausia reikia pasirinkti priemonę, kurią galime unikaliai identifikuoti ir susieti su studento vardu ir pavarde, esančias dėstytojo sąraše. Lengviausiai prieinamos studentams priemonės yra:

- Lietuvos studento pažymėjimas
- Išmanusis telefonas

Automatizavus lankomumo žymėjimo procesą, pasitelkiant vieną iš pateiktų priemonių, būtų supaprastinamas dėstytojų darbas bei taupomas užsiėmimo laikas.

1.1. Lietuvos studento pažymėjimas

Vienas iš būdų yra Lietuvos studento pažymėjimo (toliau - LSP) panaudojimas studento identifikavimui. Naujo pavyzdžio LSP turi integruotą NFC (angl. *Near Field Communication*) technologiją. Jos pagalba galime nuskaityti LSP unikalų identifikavimo numerį ir pagal jį identifikuoti studentą. Identifikavus studentą sistemoje pažymimas jo dalyvavimas paskaitoje.

Visi studentai pradėję studijuoti įsigyja LSP todėl studentams toks procesas nesukeltų jokių nepatogumų. Pagrindinis trūkumas, kad universiteto auditorijose turėtų būti įdiegti *NFC* skaitytuvai arba dėstytojai turi turėti išmaniuosius telefonus su įdiegta *NFC* technologija. Norint efektyviai atlikti žymėjimą turėtų būti įrengtas ne vienas, nes pažymėti galima tik po vieną studentą. Taip pat studentai gali pamiršti pasiimti su savimi LSP.

1.2. Išmanusis telefonas

Kitas būdas galėtų būti panaudojant Bluetooth technologiją. Pati technologija nėra nauja, tačiau vis tobulinama. Beveik visi šiuo metu naudojami mobilieji telefonai turi integruotą šią technologiją. Mobiliuosius telefonus turi visi studentai. Kiekvienas įrenginys, kuriame yra integruota Bluetooth technologija, turi savo unikalų identifikavimo numerį, kurį galima nuskaityti. Kaip ir prieš tai pateiktu būdu - šį numerį reikėtų susieti su studento vardu ir pavarde.

Universiteto auditorijose taip pat turėtų būti įdiegiami skaitytuvai, kurie nuskaitytų studentų mobiliųjų telefonų unikalius identifikacinius numerius. Tokio skaitytuvo tereikėtų vieno auditorijoje, nes mobiliųjų telefonų siunčiamus signalus Bluetooth ryšius galima nuskaityti visus vienu metu. Nors studentai turėtų įjungti šią technologiją telefone, pats procesas vyktų žymiai efektyviau ir paprasčiau.

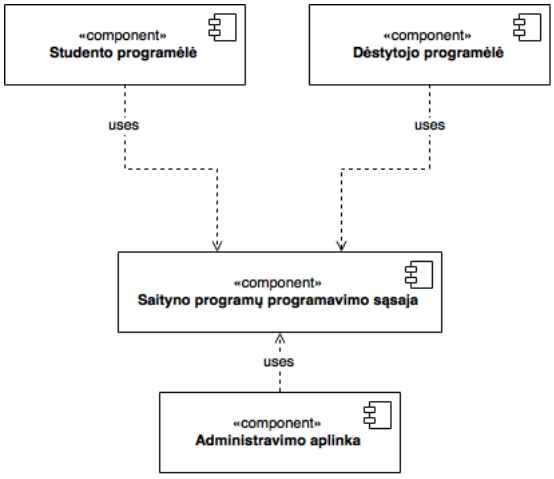
1.3. LSP ar išmanusis telefonas?

Pateikti būdai turi ir privalumų, ir trūkumų. Visgi tinkamesnis būdas identifikuoti studentus yra pasitelkus Bluetooth technologiją, nes:

- Auditorijose nereikia įrengti specialios įrangos
- Greitesnis identifikavimas, nesusidaro eilės prie skaitytuvų

Verta paminėti, kad šie būdai nėra priešinami, o laikomi alternatyvomis. Jei telefonas išsikrovęs, galima būtų identifikuoti su LSP. Tokiu atveju turėtų būti įrengti *NFC* technologijos skaitytuvai.

1.4. Sistemos naudojimo vizija



1 pav. Sistemos komponentai

Pagal pateiktą *UML* sistemos komponentų diagramą (1 pav.) matome, kad sistemą sudaro keturi pagrindiniai komponentai – studento ir dėstytojo telefonai, saityno programų programavimo sąsają ir administravimo aplinka.

1.5. Mobilioji programėlė studentams

Toks proceso įgyvendinimas prasideda nuo programėlės sukūrimo išmaniesiems telefonams. Taip pat svarbu, kad programėlė ne tik padėtų identifikuoti studentą, tačiau būtų kuo paprastesnė (reikėtų atlikti kuo mažiau veiksmų) naudotojui.

Kiekvienas įrenginys turintis integruotą Bluetooth technologiją turi ir unikalų MAC (angl. *Media Access Control*) adresą, dar vadinamą fiziniu adresu. Kadangi šis adresas kievienam įrenginiui yra unikalus, tai juo galime naudotis norint identifikuoti studentus. Norint susieti studento mobilųjį telefoną su jo paskyra sistemoje studentas įsidiegia programėlę. Programėlėje studentas prisijungia prie savo paskyros ir susieja mobilųjį telefoną. Programėlė nuskaito telefono Bluetooth MAC adresą ir kartu su studento identifikavimo numeriu (gaunamas prisijungus prie paskyros) siunčia į sistemą. Sistema gautą MAC adresą priskiria tam studentui. Programėlės pagrindinė funkcija yra susieti telefoną su paskyra. Tuo pačiu galima pateikti studento tvarkaraštį, lankomumo ataskaitą.

Norint, kad sistema galėtų naudotis kuo daugiau studentų, programėlė turi būti palaikoma skirtingose įrenginiuose bei operacinėse sistemose. Pačios populiariausios išmaniųjų įrenginių operacinės sistemos yra *Android* ir *iOS*. Taip pat gana plačiai naudojama *Windows Phone* operacinė sistema [IDC15].

1.5.1. Programėlių kūrimas pagal operacinę sistemą

1.5.1.1. Android

Tai yra populiariausia, apie 80 procentų rinkos užimanti [IDC15], operacinė sistema. Ši platforma yra atviro kodo ir prižiūrima *Google* kompanijos. *Android* operacinė sistema diegiama į tokių kaip *LG*, *Sony* ir kitų gamintojų išmaniuosius telefonus. Šiai platformai programėlės kuriamos *Java* programavimo kalba. Programavimo aplinkai kūrėjai siūlo nemokamą įrankį *Android Studio* bei pamokų ciklą pirmąjai programėlei sukurti.

1.5.1.2. iOS

Antra pagal populiarumą, apie 15 procentus rinkos turinti [IDC15], operacinė sistema. Ši platforma yra palaikoma ir sukurta *Apple* korporacijos. Programėlės kuriamos *Objective-C* arba *Swift* programavimo kalbomis.

1.5.1.3. Windows Phone

Trečia pagal populiarumą, apie 3 procentus išmaniųjų įrenginių rinkos užimanti [IDC15], operacinė sistema. Tai yra *Microsoft* korporacijos produktas. Operacinė sistema diegiama *Nokia* korporacijos *Lumia* išmaniųjų telefonų linijoje. Programavimo kalba – *C#*.

1.5.2. Viena programėlė – visoms operacinėms sistemoms

Studentai turi daug ir skirtingų išmaniųjų telefonų, todėl sistema turi palaikyti kuo daugiau įrenginių. Mūsų atveju taip pat reikalinga, kad naudojimąsis sistema būtų kuo prieinamesnis. Neseniai pradėtos kurti technologijos leisiančios rašyti kodą vienai programėlei ir ji veiktų populiariausiose [IDC15] išmaniųjų įrenginių operacinėse sistemose. Kol kas nėra tiek išvystyta technologija, tačiau šiuo metu galime su nedidelėmis korekcijomis sukurti programėlę tinkamą populiariausioms išmaniųjų irenginių operacinėms sistemoms.

1.5.2.1. Savoji programėlė

Sparčiai tobulėjanti platforma, skirta kurti programėles *Android*, *iOS* ir *Windows Phone* operacinėms sistemoms, tai *Xamarin* [Xam16]. Naudojantis šia platforma galima kurti visoms trims mobiliosioms operacinėms sistemoms savąsias (angl. *native*) programėles. Šiuo atveju programėlės yra rašomos *C#* programavimo kalba, o programinė įranga, reikalinga kurti programėles, yra nemokama. Vidutiniškai 75 procentai programėlės kodo yra naudojama bendrai tarp operacinių

sistemų. Platforma suteikia pilną priėjimą prie įrenginio operacinės sistemos programų programavimo sąsajos (angl. *Application Programming Interface, API*). Vartotojo sąsają galima kurti pagal operacinės sistemos standartus arba naudojant platformos kūrėjų įrankį *Xamarin.Forms*.

1.5.2.2. Saityno programėlė

Priklausomai nuo poreikių, galima sukurti tinklalapį pritaikytą mobiliesiems įrenginiams. Kuriant galima pasitelkti karkasus, skirtus kurti būtent tokias saityno programėles, pavyzdžiui: *Bootstrap*, *Foundation* [aww16]. Tai yra greitas ir pigus būdas, prieinamas iš visų operacinių sistemų. Tačiau tinkamas tik tuomet, kai nereikia arba reikalingos elementarios (garsas, failai įrenginio atmintyje) sąsajos su įrenginiu. Taip pat programėlė laikoma serveryje ir yra parsiunčiama į telefoną kiekvienos užklausos metu, todėl gali sukelti nepatogumų, nes naudojant mobilųjį internetą dažniausiai duomenų kiekis yra ribojamas.

1.5.2.3. Hibridinė programėlė

Hibridinė programėlė – tai saityno programėlė, tačiau integruota į savą programėlę. Tai suteikia galimybę priėjimą prie įrenginio operacinės sistemos programų programavimo sąsajos. Tokiu būdu mes galime pasiekti išmaniojo įrenginio resursus.

Vienas iš labiausiai paplitusių karkasų yra *Ionic* [Mar15]. Naudojantis šiuo karkasu galime programėlę kurti kaip tinklalapį, programuojant *JavaScript* programavimo kalba bei rašant *HTML* ir *CSS* kodą vartotojo sąsajai kurti.

1.5.2.4. Tinkamiausio būdo parinkimas

Mūsų kuriamos sistemos atveju svarbiausia gera programėlės integracija su įrenginio komponentais, o konkrečiau – įrenginio *Bluetooth* ryšio moduliu. Taip pat labai svarbu, kad programėlė būtų kuo prieinamesnė – turi būti palaikomos populiariausios išmaniųjų telefonų operacinės sistemos.

	Savoji	Saityno	Hibridinė
Prieinamas *Bluetooth* ryšys	+		+
Veikia visose populiariausiose operacinėse sistemose	+	+	+
Kūrimo įrankiai yra nemokami	+	+	+

1 lentelė. Programėlių kūrimo būdų palyginimas

Norint identifikuoti studentus panaudojant jų išmaniuosius telefonus programėlei būtinas priėjimas prie įrenginio *Bluetooth* ryšio modulio, todėl saityno programėlė mums netinka. Hibridines programėles galima kurti greičiau ir paprasčiau nei savąsias, tačiau jų veikimo sparta bei vientisumas gali kelti nepatogumų naudotojams. Taip pat yra sudėtingiau išnaudoti įrenginio ir jo operacinės sistemos teikiamas funkcijas, pasiekiamas per aplikacijų programavimo sąsaja. Todėl tinkamiausias būdas kurti programėlę sistemai yra naudojant *Xamarin* platformą. Ji leis sukurti intuityvesnę, vientisesnę bei patikimesnę programėlę.

1.5.3. Savosios programėlės detalieji naudojimo scenarijai

1.5.3.1. Prisijungimas

Įdiegus ir pirmą kartą programėlę paleidus naudotojui atidaromas prisijungimo langas (2 pav.). Studentas suveda prisijungimo vardą bei slaptažodį. Spaudžia prisijungti. Sistemai grąžinus sėkmingą pranešimą, programėlėje išsaugomi prisijungimo duomenis. Kitą kartą paleidus programėlę prisijungimas vyksta automatiškai.

1.5.3.2. Įrenginio susiejimas

Pagrindiniame programėlės lange (3 pav.) matome pranešimą, kad nėra susietas įrenginys su paskyra. Spaudžiame „Susieti įrenginį“. Programėlė surenka reikalingą informaciją apie įrenginį (*Bluetooth MAC* adresą) ir siunčia užklausą į sistemos aplikacijų programavimo sąsają. Grąžinama žinutė apie sėkmingą arba nesėkmingą įrenginio susiejimą su paskyra.

Alternatyvus scenarijus: Prisijungus programėlėje, atidarome meniu (4 pav.) paspausdami ant meniu piktogramos viršutinėje juostoje. Atsidariusiame meniu pasirenkame „Susieti įrenginį“.

1.5.3.3. Lankomumo pažymėjimas

Rankinis scenarijus: Studentas atėjęs į paskaitą įsijungia programėlę bei įrenginio Bluetooth ryšį. Pagrindiniame lange (3 pav.) spaudžia mygtuką „Dalyvauju“. Programėlė ieško skaitytuvo ir nuskaito jo siunčiamą informaciją: paskaitos pavadinimą, dėstytojo vardą ir pavardę. Rodomas patvirtinimo langas su šia informacija. Studentas spaudžia „Taip“. Patvirtinimas siunčiamas į sistemą ir pažymimas studento dalyvavimas paskaitoje.

Automatizuotas scenarijus: Jei studentas išmaniajame telefone įjungęs *Bluetooth* ryšį ir yra susiejęs įrenginį su paskyra, skaitytuvas aptikęs įrenginį automatiškai pažymi studento dalyvavimą užsiėmime.

1.5.3.4. Paskyros peržiūra

Programėlėje studentas taip pat gali peržiūrėti savo paskyrą meniu (4 pav.) spausdamas mygtuką „Paskyra“.

1.5.4. Savosios programėlės vartotojo sąsaja

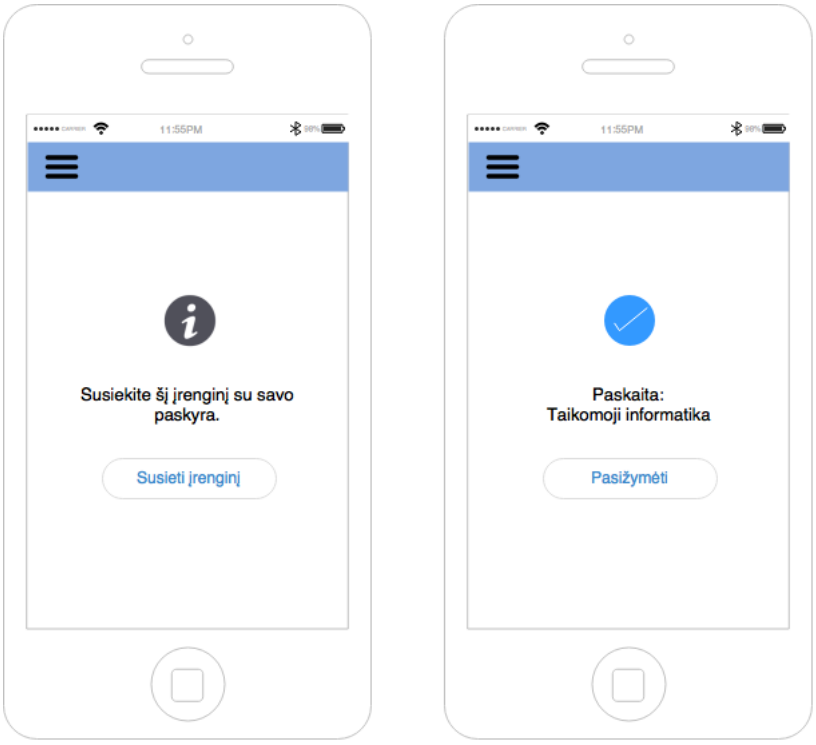
Studentams skirtos programėlės pagrindinės funkcijos – pasižymėti dalyvavimą užsiėmime bei susieti įrenginį su studento vardu ir pavarde. Projektuojant vartotojo sąsają pagrindinis dėmesys skiriamas šių veiksmų paprastumui, stengiamasi, kad užduočių atlikimui reikėtų kuo mažiau veiksmų. Vartotojo sąsają, naudojant *Xamarin* platformą, galima kurti dviem būdais:

- Kiekvienos operacinės sistemos vartotojo sąsajos kūrimo įrankiu
- *Xamarin* platformos kūrėjų kurtu įrankiu *Xamarin Forms*

Kuriant vartotojo sąsają kiekvienos operacinės sistemos vartotojo sąsajos kūrimo įrankiu kyla nepatogumas, kad reikia tuos pačius langus aprašyti tris kartus. Vartotojo sąsaja kuriant *Xamarin Forms* įrankiu, visuose įrenginiuose, nepriklausomai nuo operacinės sistemos, vartotojo sąsaja bus atvaizduojama ta pati ir aprašoma vieną kartą. Šio įrankio naudojimas pagreitina įgyvendinimą, nes nebereikia atskirai kurti kiekvienai operacinei sistemai vartotojo sąsajos atskirai. Tuo pačiu reikia atsižvelgti į tai, jog dėl tos pačios priežasties įrenginių ekrano dydžiai labai skirtingi.

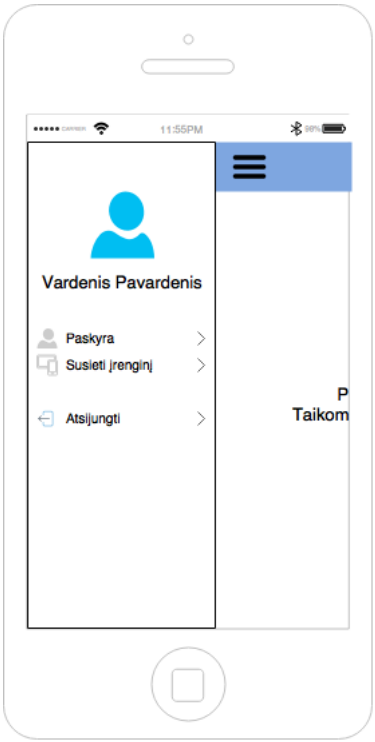
2 pav. Prisijungimo langas

Prisijungimo langas (2 pav.) matomas tik pirmą kartą prisijungiant arba po atsijungimo. Naudotojui reikia suvesti prisijungimo vardą bei slaptažodį ir paspausti mygtuką „Prisijungti“. Jei pateikti duomenys teisingi, atidaromas pagrindinis langas (3 pav.). Kitu atveju rodomas klaidos pranešimas.



3 pav. Pagrindinis langas

Pagrindiniame lange (3 pav.), priklausomai nuo to, ar esame susiejęę įrenginį su mūsų paskyra, matome arba pranešimą, kad turime susieti įrenginį su paskyra arba paskaita, kurioje galime pasižymėti, kad dalyvaujame.



4 pav. Meniu

Meniu (4 pav.) pateikiamas prisijungusio naudotojo vardas ir pavardė bei nedidelė profilio nuotrauka. Meniu atidaryti galima viršutinėje juostoje paspaudus meniu ikoną.

1.6. Skaitytuvas – išmanusis telefonas

Kitas svarbus komponentas, norint identifikuoti studentus, naudojant *Bluetooth* ryšį – skaitytuvas. Tai galėtų būti atskiras prietaisas arba tiesiog išmanusis telefonas. Reikalinga sukurti atskirą programėlę, kuri būtų įdiegta dėstytojo išmaniajame telefone.

Paskaitos metu programėlė nuskaito įrenginius esančius patalpoje ir siunčia įrenginių sąrašą su jų *Bluetooth MAC* adresais į sistemą, kur gautas unikalus identifikatorius susiejamas su studento paskyra ir pažymimas jo dalyvavimas paskaitoje. Norint tai įgyvendint, dėstytojas taip pat turi turėti savo paskyrą sistemoje. Paskyroje pridedamos paskaitos, priskiriamos grupės bei studentai.

Kaip ir programėlė skirta studentams, taip ir skirta dėstytojams turi būti kuo prieinamesnė. Todėl jos kūrimui taip pat pasirinkta *Xamarin* platforma.

1.6.1. Skaitytuvo programėlės funkcijos

1.6.1.1. Prisijungimas

Įdiegus ir pirmą kartą programėlę paleidus naudotojui atidaromas prisijungimo langas (5 pav.). Dėstytojas suveda prisijungimo vardą bei slaptažodį. Spaudžia prisijungti. Sistemai grąžinus sėkmingą pranešimą, programėlė išsisaugo prisijungimo duomenis arba prieigos raktą (priklausomai nuo sistemos vidinės pusės įgyvendinimo). Kitą kartą paleidus programėlę prisijungimas vyksta automatiškai.

1.6.1.2. Paskaitos pradžia

Paleidus programėlę ir prisijungus (automatiškai arba suvedus prisijungimo duomenis) pagrindiniame lange (6 pav.) naudotojas paspaudžia ant mygtuko „Paskaita“, atidaromas paskaitos pasirinkimo langas (6 pav.), išsirenka paskaitą ir ant jos paspaudžia. Atidaromas pagrindinis langas (6 pav.) ir rodoma pasirinkta paskaita. Taip pat pasirenkama grupė. Parinkus paskaitą ir grupę naudotojas spaudžia „Pradėti paskaitą“. Programėlė pradeda skleisti informaciją *Bluetooth* ryšiu. Rodomas paskaitos langas su pasižymėjusiais studentais.

Alternatyvus scenarijus: Pradėjus paskaitą periodiškai ieškoma įrenginių su įjungtu *Bluetooth* ryšiu. Rastų įrenginių *Bluetooth MAC* adresai siunčiami į sistemą.

1.6.1.3. Paskaitos pabaiga

Paskaitos lange (7 pav.) naudotojas spaudžia „Užbaigti paskaitą“. Rodomas patvirtinimo pranešimas, spaudžia „Užbaigti“. Programėlė nusiunčia užklausą į sistemą dėl paskaitos užbaigimo. Rodomas pranešimas su sėkmingo arba nesėkmingo užbaigimo tekstu. Atidaromas pagrindinis langas (6 pav.).

1.6.2. Skaitytuvo programėlės vartotojo sąsaja

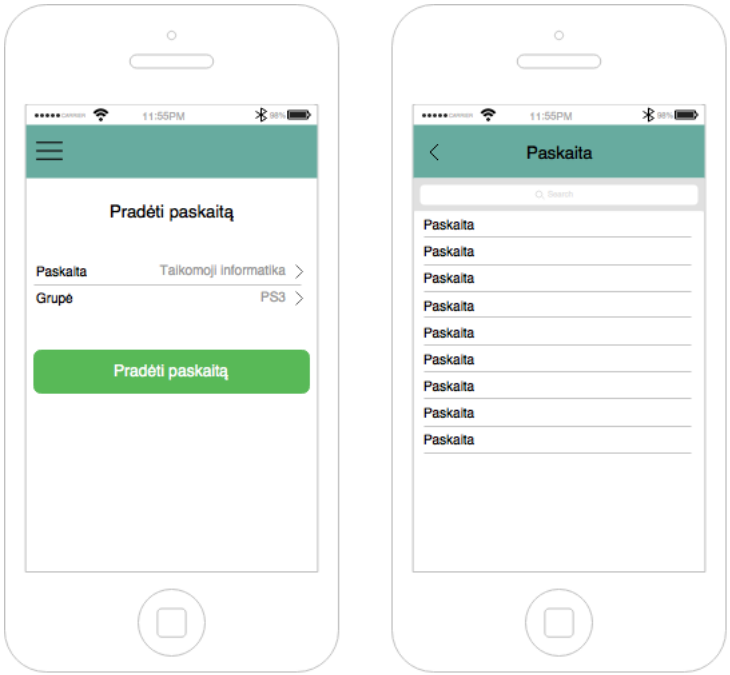
Projektuojant skaitytuvo programėlės vartotojo sąsają stengtąsi pateikti taip, kad reikėtų kuo mažiau veiksmų norint atlikti pagrindines užduotis. Svarbiausios programėlės funkcijos yra paskai-

tos pasirinkimas bei studentų įrenginių aptikimas, todėl šioms užduotims skirtas pagrindinis dėmesys.

Programėlėje išlaikomas ta pati struktūra, kaip ir studentų programėlėje. Kad nebūtų painiojamos tarpusavyje, naudojamos kitos spalvos.

5 pav. Prisijungimo langas

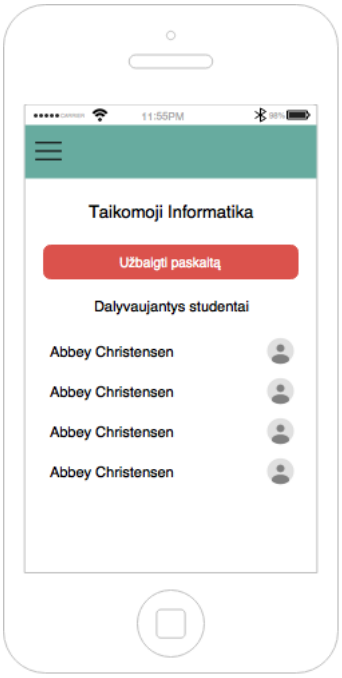
Prisijungimo langas (5 pav.) toks pat kaip ir studentų programėlėje.



6 pav. Pagrindinis langas (kairėje) ir paskaitos pasirinkimo langas (dešinėje)

Pagrindiniame lange (6 pav.) galime pasirinkti paskaitą ir grupę. Grupės pasirinkimo langas

yra analogiškas paskaitos pasirinkimo langui (6 pav.) – taip pat pateikiama paieška ir dėstytojų grupių sąrašas. Verta pažymėti, kad grupės pasirinkimo lange pateikiamas sąrašas grupių, kurios yra priskirtos pasirinktai paskaitai.

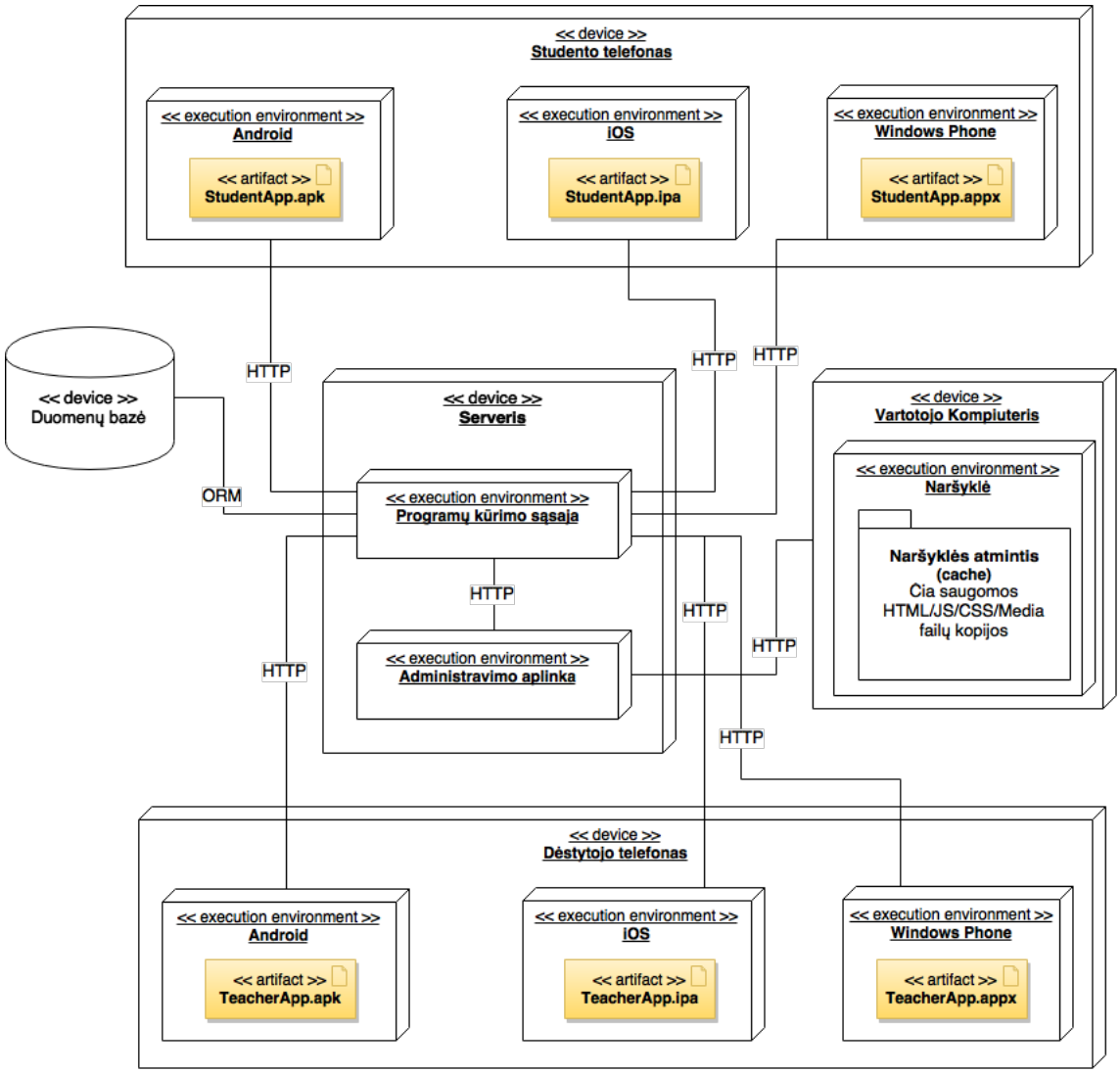


7 pav. Paskaitos langas

Iš pagrindinio lango (6 pav.) patenkama į paskaitos langą (7 pav.). Šiame lange matome paskaitos pavadinimą, paskaitos užbaigimo mygtuką bei dalyvaujančių (pažymėtų) studentų sąrašą.

2. Sistemos vidinė pusė

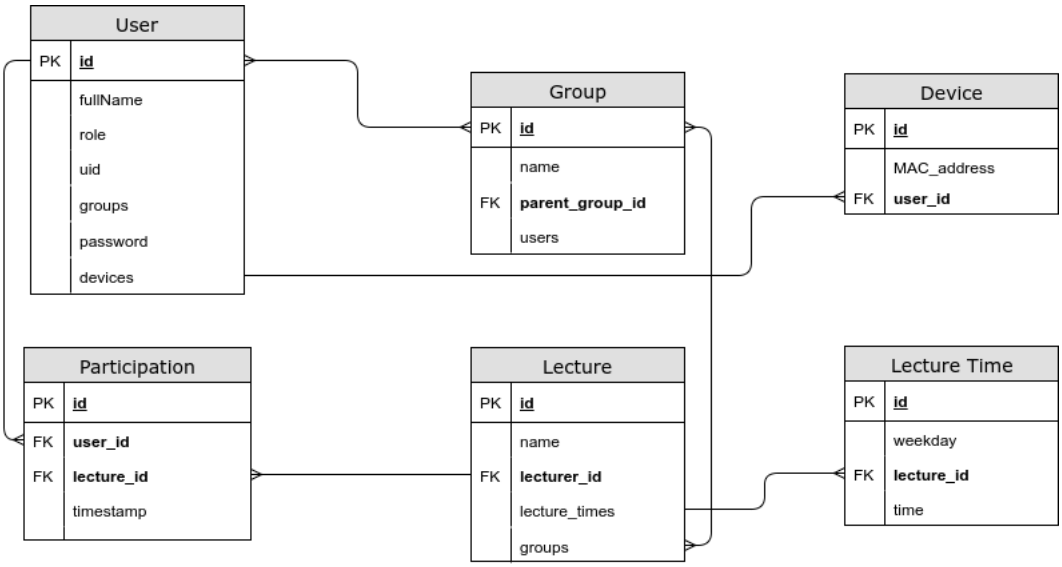
Apžvelgėme, kokiu būdu ir kokiomis priemonėmis sistemos naudotojai galės naudotis sistema. Bet visa tai neveiks be pagrindinės sistemos dalies, atsakingos už duomenų saugojimą, apdorojimą ir pateikimą.



8 pav. Sistemos diegimo vaizdas

Pagal pateiktą diegimo diagramą (8 pav.) matome, kad visą sistemą sudaro trys pagrindiniai moduliai: mobiliosios programėlės (aprašytos 1.5 ir 1.6 poskyriuose), programų kūrimo sąsaja bei administravimo aplinka.

2.1. Sistemos esybės



9 pav. Sistemos esybių ryšiai

Kaip matome pateiktoje esybių ryšių diagramoje (9 pav.) sistemos numatytoms funkcijos įgyvendinti reikalingos išvardintos esybės su nurodytais laukais:

- Naudotojas
  - Identifikatorius
  - Pilnas vardas
  - Rolė (studentas, dėstytojas)
  - Unikalus identifikatorius
  - Grupių, kurioms priklauso, sąrašas
  - Slaptažodis
  - Įrenginiai
- Grupė
  - Identifikatorius
  - Pavadinimas
  - Tėvinė grupė
  - Naudotojai
- Įrenginys
  - Identifikatorius
  - *MAC* adresas

  - Naudotojas
- Paskaita
  - Identifikatorius
  - Pavadinimas
  - Dėstytojas
  - Paskaitos laikas
  - Grupių sąrašas
- Paskaitos laikas
  - Identifikatorius
  - Paskaita
  - Savaitės diena
  - Paskaitos laikas
- Dalyvavimo įrašas
  - Identifikatorius
  - Naudotojas
  - Paskaita
  - Įrašo laikas

Naudotojas turi roles – *studento* arba *dėstytojo*. Jei naudotojo rolė *studentas*, tai lietuvos studento pažymėjimo numeris įvedamas kaip unikalus identifikatorius, kitu atveju – įvedamas dėstytojo darbuotojo numeris.

Grupės gali turėti tėvines grupes, todėl tai gali atstoti universiteto kursus ir kurso grupes. Paskaita gali būti sukurta visam kursui ar net keliems kursams arba kurso grupei ar kelioms grupėms.

Paskaitai priskiriamos grupės, dėstytojas. Taip pat sukuriami ir priskiriami paskaitos laikai – nurodoma savaitės diena bei dienos laikas. Ši informacija padės automatizuoti paskaitų lankomumo sekimo aktyvavimo automatizavimą.

Dalyvavimo įrašas sukuriamas, kuomet dėstytojo programėlė arba skaitytuvas aptinka studento telefoną ir atsiunčią užklausą serveriui. Priskiriamas naudotojas (su *studentas* role), paskaita bei pažymima, kokiu laiku studentas atvyko į paskaitą.

2.2. Saityno programų programavimo sąsaja

Visas bendravimas tarp programėlių ir pagrindinio sistemos modulio vyksta per programų programavimo sąsają. Šis sistemos modulis turi būti sukurtas taip, kad būtų galima nesunkiai įtraukti naują modulį į sistemos architektūrą. Taip pat svarbūs kriterijai renkantis sprendimą:

- Maža duomenų apsikeitimo apimtis – turi būti taupomi tinklo ištekliai
- Užklausų apdorojimo ir atsakymo siuntimo sparta

Saityno programų programavimo sąsaja (angl. *Web Application Programming Interface*) yra dažnas sprendimas tokioms sistemoms įgyvendinti. Tai yra sąsaja, susidedanti iš vieno ar daugiau viešai pasiekiamų prieigos taškų (angl. *endpoint*), su apibrėžta užklausos – atsakymo žinučių sistema. Yra du pagrindiniai saityno programų programavimo sąsajų tipai: *SOAP (Simple Object Access Protocol)* saityno paslaugos ir *REST (Representational State Transfer)* saityno ištekliai.

2.2.1. Saityno programų programavimo sąsajų tipai

2.2.1.1. SOAP

Tai supaprastintas protokolas skirtas apsikeisti struktūrizuota informacija decentralizuotoje aplinkoje. Šis protokolas naudoja *XML (EXtended Markup Language)* technologiją apibrėžti žinučių karkasą leidžiantį kurti žinutes, kuriomis galima apsikeisti su daugybe kitų protokolų. *SOAP* projektuotas taip, kad būtų nepriklausomas nuo jokios programavimo kalbos ar modelio [GHM+07].

Kaip rašoma straipsnyje [Mue13] – nors modelis kurtas, kad bendravimas tarp sistemų būtų paprastesnis, tačiau naudojant *XML* technologiją užklausų siuntimas gali tapti labai sudėtingas. Priklausomai nuo programavimo kalbos, gali tekti visą užklausą sukurti rankiniu būdu. Kita vertus, protokolas yra sukurtas *Microsoft* korporacijos, todėl implementuojant *SOAP C#* programavimo kalboje, *XML* kodo beveik netenka matyti.

Viena svarbiausių *SOAP* funkcijų ir privalumų – įtaisytas klaidų apdorojimas. Jei siunčiamoje užklausoje yra klaida, gautame atsakyme pateikiama informacija, kuri gali padėti greitai rasti ir ištaisyti klaidą.

2.2.1.2. REST

REST metodas leidžia dar paprasčiau apsikeisti žinutėmis tarp programų. Dažnai užtenka pagal taisykles sudaryto universalaus adreso. Kuomet reikia pateikti daugiau informacijos dažnai naudojamas *JSON (JavaScript Object Notation)* formatas. Šis būdas yra patogus tuo, kad galima siųsti ir gauti žinutes forma, kurią lengva sintaksiškai analizuoti programavimo kalboje su kuria yra kuriama programa [Mue13].

Naudojant *REST* saityno išteklius yra lengviau įsisavinti metodą nei *SOAP* atveju, taip pat yra efektyvesnis veikimas (galima siųsti mažesnes žinutes) bei ištekliai yra prieinamesni, nereikalingos papildomos programinės priemonės naudojimui. Dėl šių priežasčių tai yra tinkamesnis variantas sistemos saityno programų programavimo sąsajai sukurti.

2.2.2. Programų programavimo sąsajos valdikliai

Kuriant programų programavimo sąsają vadovaujantis *REST* nesunku numatyti reikalingus valdiklius kartu su prieigos taškais. Kiekviena esybė turi savo atskirą valdiklį. Šios sistemos atveju

atskiras valdiklis nekuriamas paskaitos laiko esybei, nes ji neturės atskiros logikos ir yra glaudžiai susijusi su paskaitos esybe.

// TODO: DIAGRAMA kontrolerių (klasių arba šiaip entities – controllers)

2.2.2.1. Naudotojo valdiklis

Naudotojo valdiklis skirtas naudotojų sukūrimui, ištrynimui, redagavimui bei skaitymui. Taip pat šis valdiklis atsakingas už naudotojo įrenginių valdymą. Naudotojai su role *studentas* gali tik skaityti duomenis (ne visus) apie kitus naudotojus ir valdyti savo įrenginius.

Dar viena svarbi šio valdiklio atsakomybė – duomenų įkėlimas. *Dėstytojai* gali įkelti duomenis apie studentus ir taip iškart sukurti *studentų* paketą. Duomenys gali būti įkeliami iš failo arba per integraciją su universiteto informacine sistema. Apie tai plačiau 2.3.1 skyriuje.

2.2.2.2. Grupės valdiklis

Grupės valdiklis leidžia naudotojams, kurių rolė yra *dėstytojas*, kurti naujas grupes, jas redaguoti ir trinti. Skaityti grupės informaciją gali ir naudotojai su role *studentas*, tačiau šie naudotojai gali matyti tik tų grupių, kurioms priklauso, duomenis. *Dėstytojai* gali sukurti pogrupius (priskirti grupei tėvinę grupę) ir pridėti arba išimti naudotojus iš grupės.

2.2.2.3. Paskaitos valdiklis

Šis valdiklis naudotojams su role *dėstytojas* leidžia sukurti naujas paskaitas, jas modifikuoti ir ištrinti. Šis valdiklis dar atsakingas už paskaitos laikų kūrimą – kuriant paskaitą sukuriami ir nurodyti paskaitos vykimo laikai. *Dėstytojai* taip pat gali valdyti užsiėmimui priskirtas grupes. *Studentai* gali tik skaityti informaciją apie paskaitą – koks dėstytojas veda užsiėmimą bei kokiu laiku vyks užsiėmimas.

2.2.2.4. Dalyvavimo įrašo valdiklis

Dalyvavimo įrašo valdiklis *studentams* leidžia tik gauti duomenis apie įrašus, kurie yra susiję su jais. Kurti naujus įrašus gali tik naudotojai su *dėstytojo* role. Pagal sistemos dizainą dėstytojo telefonas nuskaito studento telefono *MAC* adresą ir jį siunčia į programų programavimo sąsają naudojantis savo naudotoju.

Valdiklis gavęs užklausą pirmiausia patikrina, ar užklausos gavimo metu yra vykstanti nurodyta paskaita – patikrinami ar duotojo laiko savaitės diena bei laikas sutampa su bent vienu užsiėmimo vykimo laiku. Naujas dalyvavimo įrašas sukuriamas tik tada, kai nurodyta paskaita vyksta užklausos gavimo metu ir studentas, kurio *MAC* adresas yra gautas priklauso vienai iš užsiėmimui priskirtų grupių.

2.2.3. Saugumas

Kaip ir kiekvienoje sistemoje labai svarbu, kad prie išteklių neprieitų asmenys, kuriems nėra suteika teisė naudotis saityno programų programavimo sąsaja. Sagumas šiame modulyje įgyven-

dinamas dviem lygiais: sąsajos prieigos taškų apsauga ir sistemos duomenų apsauga.

2.2.3.1. Prieigos taškų apsauga

Tai yra išorinės apsaugos sluoksnis – suteikti priėjimą prie saityno programų programavimo sąsajos prieigos taškų tik šios sistemos komponentams. Dažnai šiam tikslui naudojami programų programavimo sąsajos raktas (angl. *API key*).

Sistemos komponentui, kuriam norime suteikti prieigą sugeneruojame unikalų raktą. Norint užtikrinti unikalumą galima pasinaudoti universaliai unikalaus identifikatoriaus kūrimo standartu (angl. *universally unique identifier, UUID*) RFC 4122 aprašytu [LMS05]. Kadangi sistemos komponentai į saityno programų programavimo sąsają kreipsis *HTTP* protokolu, įgyvendinant tokią apsaugą yra priimta šį raktą dėti į užklausos *Authentication* antraštę. Raktas tikrinamas kiekvienos užklausos metu todėl yra kreipiamasi į sistemos duomenų bazę.

2.2.3.2. Sistemos duomenų apsauga

Kuomet pasirūpinta, kad prieigos taškai yra prieinami tik šios sistemos komponentams, dar svarbu užtikrinti, kad skirtingos naudotojų grupės (mūsų atveju – dėstytojai ir studentai) galėtų pasiekti tik jiems skirtus išteklius ir atvirkščiai – negalėtų pasiekti išteklių, kurie yra neskirti tai naudotojų grupei.

Vienas iš būdų prisijungimo duomenis siųsti užklausos antraštėjė, panašiai kaip aprašyta 2.2.3.1 poskyryje. Saityno programų programavimo sąsaja gavusi užklausą kreipiasi į duomenų bazę, kad patikrintų, ar toks naudotojas egzistuoja. Tai kartojama kiekvienos užklausos metu.

Naujas būdas tokioms užduotims spręsti – *JSON Web Token (JWT)* apibrėžtas RFC 7519 standartu [JBS15]. Tai yra savarankiškas ir kompaktiškas būdas perduoti informaciją tarp sistemos komponentų. Dėl nedidelio naudingos apkrovos (angl. *payload*) dydžio, ją galima perduoti tiesiog universaliu adresu ar įdėti į *HTTP* užklausos antraštę.

JWT susideda iš trijų dalių: antraštės, naudingos apkrovos bei parašo. Antraštėje įprastai nurodomas prieigos rakto tipas bei maišymo (angl. *hashing*) algoritmas. Naudingą apkrovą sudaro informacija apie naudotoją: asmeniniai duomenys, naudotojų grupė ir pan. Paraše yra *HMAC SHA256* arba *RSA* algoritmu užšifruojama maišos reikšmė, kuri susideda iš *Base64* būdu užkoduotos antraštės ir naudingos apkrovos.

Naudojant *JWT*, priešingai nei pridedant tik prisijungimo duomenis į užklausos antraštę, nebereikia gavus kiekvieną užklausą kreiptis į duomenų bazę patikrinti, ar toks naudotojas egzistuoja, todėl atsakymas yra išsiunčiamas greičiau. Taip pat yra taupomi serverio resursai.

2.3. Administravimo aplinka

Sistemai taip pat reikalinga administravimo aplinka, kurioje galima būtų valdyti naudotojus, įrenginius ar suteikti prieigą naujiems sistemos komponentams. Kaip jau minėta 2.2 poskyryje – visi sistemos komponentai gali prieiti prie duomenų ir juos valdyti tik per saityno programų programavimo sąsają. Administravimo modulis ne išimtis. Taip pat svarbu, kad administravimo

aplinka būtų pasiekiama iš įvairių įrenginių. Kaip jau 1.5.2.2 poskyryje buvo apžvelgta, jei mums nereikia prieigos prie įrenginio funkcijų, galime kurti saityno programą.

Sparčiai populiarėjantis saityno programų kūrimo būdas yra vieno puslapio programa (angl. *Single Page Application, SPA*). Tai kūrimo metodas, kuomet visas turinys sutalpinamas viename saityno puslapyje. Reikalinga informacija, kodas yra užkraunamas viena užklausa arba dinamiškai, kai to reikia. Tai pagerina naudotojo patirtį, naudojimas yra vientisesnis, nes nereikia perkrauti naršyklės. Taip pat vieno puslapio programa tinka mūsų atveju, nes yra sukurti karkasai labai palengvinantys integravimą su saityno programų programavimo sąsaja [Tak13].

Atsižvelgiant į tai, kad administravimo moduliu nebus naudojamąsi kasdien, galima kurti klasikinę saityno programą. Šiuo būdu modulis bus sukurtas greičiau. Kadangi sistemos saityno programų programavimo sąsajai kurti naudojame plačiai naudojamą *REST* metodą, nepriklausomai nuo pasirinktos technologijos ar karkaso administravimo modulio įgyvendinimui, yra sukurtas pagalbinis įrankis užklausų siuntimui bei apdorojimui.

2.3.1. Duomenų apie studentus įkėlimas

// TODO:

2.3.1.1. Įkėlimas iš failo

// TODO:

2.3.1.2. Įkėlimas per integraciją su universiteto informacine sistema

// TODO:

2.3.2. Administravimo aplinkos naudojimo scenarijai

2.3.2.1. Naudotojo sukūrimas, redagavimas ir ištrynimas

Galimybė sukurti sistemos naudotojus. Sistemoje turime dviejų tipų agentus – sistemos komponentus (programėlė, administravimo aplinka) ir naudotojus (studentai, dėstytojai). Kuriant naudotoją, kaip sistemos komponentą, būtina nurodyti arba sugeneruoti priegos raktą, kuris bus naudojamas jungiantis prie saityno programų programavimo sąsajos. Naudotojai, kaip asmenys, skirstomi į dvi grupes: studentai ir dėstytojai. Pagal šias grupes bus suteikiamos teisės sistemos ištekliu. Taip pat sistemoje turi būti galimybė sukurtus naudotojus redaguoti.

Sistema turi saugoti istoriją, todėl turi būti realizuotas *Soft Delete* duomenų trynimo metodas. Tai trynimo būdas, kuomet vietoj to, kad būtų fiziškai pašalinamas įrašas iš duomenų bazės, yra uždedamas poyžmis, kad tas įrašas yra panaikintas ir nurodoma panaikinimo data. Šiuo metodu trinami įrašai išlaiko ryšius su kitomis esybėmis ir tuo pačiu išlaikome istorinius duomenis.

2.3.2.2. Studentų grupės sukūrimas, redagavimas ir ištrynimas

Kuriant studentų grupes turi būti galimybė iškart priskirti studentus į grupę. Reikalinga pateikti studentų sąrašą su galimybe ieškoti pagal pateiktą tekstą. Studentų grupės negalima ištrinti, jei yra priskirtas bent vienas studentas. Studentus pridėti prie grupės galima ir po studentų grupės sukūrimo.

2.3.2.3. Paskaitos sukūrimas, redagavimas ir ištrynimas

Galimybė kurti bei redaguoti paskaitas. Svarbu nurodyti paskaitos laiką – dieną, valandą, minutę, taip pat priskirti grupes ar grupę, kuriai vedama paskaita. Pagal priskirtas grupes paskaitai bus žymimas lankomumas. Jei studentas nepriklauso grupei, į kurios paskaitą atėjo – jo lankomumas nėra pažymimas. Paskaitos trynimui taip pat taikomas *Soft Delete* metodas.

2.3.3. Administravimo aplinkos naudotojo sąsaja

// TODO:

2.4. Technologinio rinkinio sudarymas

// TODO: Serverio stack'o parinkimas remiantis anskčiau aprašytais dalykais – reikalavimai, modulių (JWT, etc) prieinamumas, scalability; Administravimo aplinkos karkasų parinkimas (tikriausiai React).

Rezultatai ir išvados

Šio darbo metu pasiekti šie rezultatai:

- Sukurtas koncepcinis automatizuoto lankomumo žymėjimo naudojimo scenarijus, pagrįstas *Bluetooth* ryšio technologija
- Sukurtos sistemos įgyvendinimo gairės ir sudarytas technologijų rinkinys
- Suprojektuotas komponentų išsidėstymas sistemoje
- Sukurti mobiliosios programėlės skirtos studentams vartotojo sąsajos eskizai bei sukurti dėstytojams skirtos skaitytuvo programėlės vartotojo sąsajos eskizai

Analizuojant studentų automatizuoto identifikavimo, panaudojant išmaniuosius telefonus, technologines galimybes pastebėta, jog patogus būdas identifikuoti studentus yra pasitelkiant studentų išmaniuosius telefonus su *Bluetooth* ryšio technologija. Įgyvendinant šį būdą nereikia įrengti specialios įrangos auditorijose. Taip pat toks identifikavimas yra greitesnis nei naudojant LSP – nesusidaro eilės prie skaitytuvų.

Apžvelgiant išmaniųjų telefonų programėlių technologinius įgyvendinimo būdus nustatyta, kad savoji programėlė tenkina keliamus kriterijus, todėl yra tinkamas būdas įgyvendinimui. Šis būdas leidžia lengvai panaudoti išmaniojo įrenginio operacinės sistemos programų programavimo sąsają, kurios pagalba galima naudoti įrenginyje įdiegtą *Bluetooth* ryšio technologiją.

Aptariant sistemos komponentų komunikavimo būdus nustatyta, kad tinkamas būdas vykdyti komunikavimą tarp komponentų yra per apibrėžtą saityno programų programavimo sąsają. Taip pat išskirtas *REST* sąsajos kūrimo metodas kaip tinkamas kuriamai sistemai. Šiuo metodu sukurta saityno programų programavimo sąsaja leidžia siųsti ir gauti mažos apimties užklausas, tuo pačiu – greičiau apdorojamos užklausos.

Darbo plėtojimo galimybės

- Sukurti duomenų apsikeitimo sąsają su universiteto informacine sistema. Sąsajos pagalba galima būtų importuoti duomenis apie studentus, kuriant naudotojus nereikėtų informacijos suvedinėti rankomis.
- Pritaikyti žaidimizavimo skatinimo modelį sistemoje, skatinantį studentus aktyviai dalyvauti užsiėmime. Modulio tikslas – skatinti studentų dalyvavimą apdovanojant sistemoje už kiekvieną veiksmą.
- Apžvelgti galimybę įgyvendinti studentų darbų recenzavimo užsiėmimų metu modulį sistemoje. Studentų darbus, šio modulio pagalba, būtų lengviau sekti, teikti pastabas ir įvertinti.

Literatūra

[aww16] awwwards-team. What are frameworks? 22 best responsive css frameworks for web design. http://www.awwwards.com/what-are-frameworks-22-best-responsive-css-frameworks-for-web-design.html, 2016.

[GHM+07] Martin Gudgin, Marc Hadley, Noah Mendelsohn, Jean-Jacques Moreau, Henrik Frystyk Nielsen, Anish Karmarkar ir Yves Lafon. Soap version 1.2 part 1: messaging framework (second edition). http://www.w3.org/TR/soap12-part1/, 2007.

[IDC15] Inc. IDC Research. Smartphone os market share, 2015 q2. http://www.idc.com/prodserv/smartphone-os-market-share.jsp, 2015.

[JBS15] M. Jones, J. Bradley ir N. Sakimura. Json web token (jwt). https://tools.ietf.org/html/rfc7519, 2015.

[LMS05] P. Leach, M. Mealling ir R. Salz. A universally unique identifier (uuid) urn namespace. https://tools.ietf.org/html/rfc4122, 2005.

[Mar15] Danny Markov. Comparing the top frameworks for building hybrid mobile apps. http://tutorialzine.com/2015/10/comparing-the-top-frameworks-for-building-hybrid-mobile-apps/, 2015.

[Mue13] John Mueller. Understanding soap and rest basics and differences. http://blog.smartbear.com/apis/understanding-soap-and-rest-basics/, 2013.

[Tak13] Mikito Takada. *Single page apps in depth a.k.a Mixu's single page app book*. 2013.

[Xam16] Xamarin. Xamarin platform. https://www.xamarin.com/platform, 2016.