VILNIAUS UNIVERSITETAS
MATEMATIKOS IR INFORMATIKOS FAKULTETAS
PROGRAMŲ SISTEMŲ KATEDRA

Interaktyvios pratybos

Interactive lecture

Kursinis darbas

Atliko: 3 kurso 2 grupės studentas
Marijus Siliūnas (parašas)

Darbo vadovas: Dr. Kristina Lapin (parašas)

Vilnius – 2016

TURINYS

Įvadas

Temos aktualumas

Statistika apie technologijas, dabartinė situacija

Darbo tikslas

Sukurti identifikavimo mechanizmą, kuris sudarytų pagrindą tolimesnei sistemos plėtrai, leistų plėtojant sistemą automatizuoti studentų lankomumo žymėjimo procesą.

Šiuo darbu siekiama palengvinti darbą dėstytojams paskaitų metu žymint lankomumą. Sukurti sistemą, kurioje studentai skatinami aktyviai dalyvauti paskaitose apdovanojant sistemoje už kiekvieną veiksmą. Atsiskaitymų metu studentams bei dėstytojui aiškiai pateikti atsiskaitomą darbą. Sukurti sistemą, kuri leistų teikti kokybišką atsaką akivaizdinių studijų užsiėmimų metu, gerina studentų ir dėstytojų bendradarbiavimą.

Siekiami rezultatai

Sukurti studentų identifikavimo sistemą, kuri automatizuotų lankomumo žymėjimo procesą. Šią sistemą praplėsti studentų skatinimo aktyviai dalyvauti užsiėmimuose mechanizmu.

Ištirti dėstytojų pasikartojantys procesai paskaitų metu ir nustatyta kuriuos iš jų galima automatizuoti. Sukurta sistema su automatizuotu studentų identifikavimo procesu bei studentų skaitinimo aktyviai dalyvauti paskaitoje mechanizmu.

1. Studento identifikavimas

Kuo tai svarbu.

1.1. Dabartinės situacijos apžvalga

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. 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, nes nėra bendros sistemos universitete. Reikalinga sistema, kurioje būtų suvedami dėstomi dalykai, dėstantys dėstytojai bei studentų grupės ir studentai. Sistemoje būtų vedamas lankomumo žurnalas.

Lankomumo žymėjimą galima visiškai automatizuoti. Taip būtų supaprastinamas dėstytojų darbas bei taupomas laikas. Lankomumo žymėjimą galima automatizuoti ne vienu būdu, tačiau pirmiausia reikia studentus identifikuoti.

1.2. NFC technologija

Vienas iš būdų yra Lietuvos studento pažymėjimo (toliau - LSP) panaudojimas studento įdentifikavimui. 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. Norint įgyvendinti tokį identifikavimo procesą dar reikalinga integracija su duomenų baze, kurioje būtų susiejamas unikalus LSP indentifikavimo numeris su LSP duomenimis. Gauti duomenys turėtų būti patikrinami ir su vidine universiteto sistema patikrinant, ar toks studentas tikrai egzistuoja, ar jis turi dalyvauti šioje paskaitoje ir t.t. Taip pat universiteto auditorijose turėtų būti įdiegti NFC skaitytuvai. Norint efektyviai atlikti žymėjimą turėtų būti įrengtas ne vienas, nes pažymėti galima tik po vieną studentą.

1.3. Bluetooth technologija

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 informacija. Privalumas prieš LSP skaitymą - nereikia integruoti trečiųjų šalių paslaugų. Identifikaciniai numeriai išsaugomi ir susiejiami su studentu kuriamoje sistemoje.

Universiteto auditorijose taip pat turėtų būti įdiegiami skaitytuvai, kurie nuskaitytų studentų mobiliųjų telefonų unikalius identifikacinius numerius. Tokio skaitytuvo tereikėtų vieno audito-

rijoje, 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.

1.4. NFC ar Bluetooth?

Pateikti būdai turi ir privalumų, ir trūkumų. Visgi tinkamesnis būdas identifikuoti studentus yra pasitelkus Bluetooth technologiją, nes: nepriklausoma nuo trečiųjų šalių paslaugų (services); auditorijose reikia įrengti mažiau skaitytuvų ir / ar greitesnis identifikavimas (nesusidaro eilės prie skaitytuvų).

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

Dažnai poreikis yra palaikyti kuo daugiau įrenginių. Ir šiuo atveju - reikalinga, kad naudojimąsis sistema būtų kuo prieinamesnis. Būtent dėl šios priežasties buvo pradėtos kurti technologijos leisiančios rašyti vieną kodą vienai programėlei ir ji veiktų populiariausiose 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. Sava 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 savas (angl. *native*) programėles. Šiuo atveju programėlės yra rašomos *C#* programavimo kalba, o programinė įranga, reikalinga surinkti bei vystyti programėles, yra nemokama. Vidutiniškai 75 procentai programėlės kodo yra naudojama bendrai tarp operacinių sistemų. Platforma suteikia pilną priėjimą prie įrenginio operacinės sistemos aplikacijų programavimo sąsajos (visose operacinėse sistemose). 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. 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. Dėl to šis variantas nėra tinkamas mūsų atveju - nėra galimybės pasinaudoti įrenginio *Bluetooth* ryšiu.

1.5.2.3. Hibridinė programėlė

Hibridinė programėlė - tai saityno programėlė, tačiau integruota į savą programėlę. Tai suteikia galimybę priėjimą prie įrenginio operacinės sistemos aplikacijų programavimo sąsajos (angl.

Application Programming Interface, API). Tokiu būdu mes galime pasiekti išmaniojo įrenginio resursus, o šiuo atveju - mums reikalingą *Bluetooth* modulį.

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. Hibridinė ar sava programėlė?

Tiek vienas, tiek kitas programėlių kūrimo būdas turi ir pliusų, ir minusų. Hibridines programėles galima kurti greičiau ir paprasčiau, 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. Vis dėlto tai yra geresnis variantas nei rašyti kiekvienai operacinei sistemai atskirą programėlę, laiko ir kainos atžvilgiu.

Kuriamos sistemos realizacijai reikia kuo geresnės integracijos su išmaniuoju įrenginiu. Todėl tinkamiausias būdas kurti programėlę sistemai yra naudojant *Xamarin* platformą. Ji leis sukurti intuityvesnę, vientisesnę bei patikimesnę programėlę.

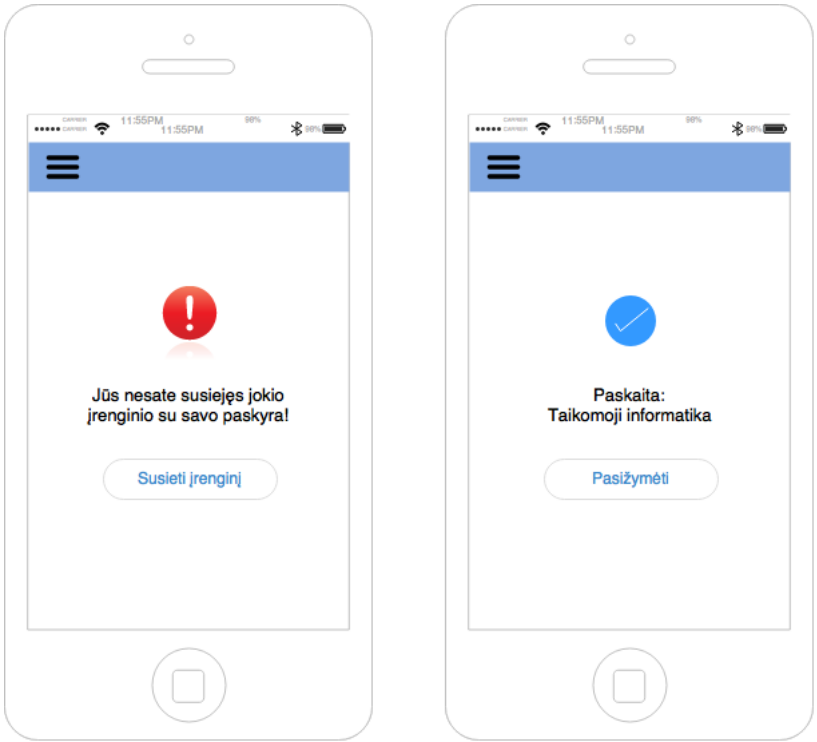
1.5.3. Programėlės vartoto sąsaja

Studentams skirtos programėlės pagrindinės funkcijos - pasižymėti dalyvavimą užsiėmime bei susieti įrenginį su paskyra. Projektuojant vartotojo sąsają pagrindinis dėmesys skiriamas šių veiksmų paprastumui, stengiamasi, kad užduočių atlikimui reikėtų kuo mažiau veiksmų.

Vartotojo sąsaja kuriama naudojant *Xamarin Forms* įrankį, todėl visuose įrenginiuose, nepriklausomai nuo operacinės sistemos, vartotojo sąsaja bus atvaizduojama ta pati. 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.

1 pav. Prisijungimo langas

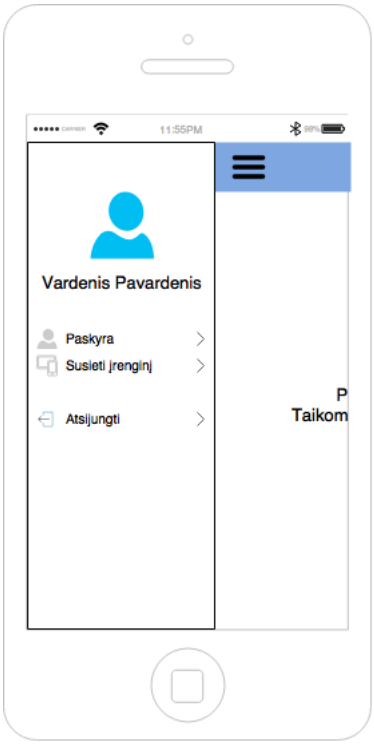
Prisijungimo langas (1 pav.) matomas tik pirmą kartą prisijungiant arba po atsijungimo. Naudotojui reikia suvesti prisijungimo vardą bei slaptažodį ir paspausti mygtuką „Prisijungti“. Jei pateikti duomenys teisingi, atidaromas pagrindinis langas (2 pav.). Kitu atveju rodomas klaidos pranešimas.



2 pav. Pagrindinis langas

Pagrindiniame lange (2 pav.), priklausomai nuo to, ar esame susiejęs įrenginį su mūsų pasky-

ra, matome arba pranešimą, kad turime susieti įrenginį su paskyra arba paskaita, kurioje galime pasižymėti, kad dalyvaujame.



3 pav. Meniu

Meniu (3 pav.) pateikiamas prisijungusio naudotojo vardas ir pavardė bei nedidelė profilio nuotrauka. Meniu atidaryti galima viršutinėje juostoje paspaudus meniu ikoną.

1.5.4. Programėlės naudojimo scenarijai

1.5.4.1. Prisijungimas

Įdiegus ir pirmą kartą programėlę paleidus naudotojui atidaromas prisijungimo langas (1 pav.). Studentas suveda prisijungimo vardą bei slaptažodį. Spaudžia prisijungti. Sistemai grąžinus sėkmingą pranešimą, programėlė išsisaugo prisijungimo duomenis (arba grąžintą access token, priklausomai nuo sistemos įgyvendinimo). Kitą kartą paleidus programėlę prisijungimas vyksta automatiškai.

1.5.4.2. Įrenginio susiejimas

Pagrindiniame programėlės lange (2 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 (3 pav.) paspausdami ant meniu ikonos viršutinėje juostoje. Atsidariusiame meniu pasirenkame „Susieti įrenginį“.

1.5.4.3. Lankomumo pasižymėjimas

Studentas atėjęs į paskaitą įsijungia programėlę bei įrenginio Bluetooth ryšį. Pagrindiniame lange (2 pav.) spaudžia mygtuką „Pasižymėti“. 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.

Alternatyvus scenarijus: Studentui tereikia įjungti Bluetooth ryšį ir būti susiejus įrenginį su paskyra. Skaitytuvas aptikęs įrenginį automatiškai pažymi studento dalyvavimą paskaitoje.

1.5.4.4. Paskyros peržiūra

Programėlėje studentas taip pat gali peržiūrėti savo paskyrą meniu (3 pav.) spausdamas mygtuką „Paskyra“.

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 vartotojo sąsaja

Projektuojant skaitytuvo programėlės vartotojo sąsają stengtąsi pateikti taip, kad reikėtų kuo mažiau veiksmų norint atlikti pagrindines užduotis. Svarbiausios programėlės funkcijos yra paskaitos pasirinkimas bei studentų įrenginių aptikimas, todėl šioms užduotims skirtas pagrindinis dėmesys.

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

4 pav. Prisijungimo langas

Prisijungimo langas (4 pav.) toks pat kaip ir studentų programėlėje.

1.6.2. Skaitytuvo programėlės funkcijos

1.6.2.1. Prisijungimas

Įdiegus ir pirmą kartą programėlę paleidus naudotojui atidaromas prisijungimo langas. 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.

1.6.2.2. Paskaitos pradžia

Paleidus programėlę ir prisijungus (automatiškai arba suvedus prisijungimo duomenis) pagrindiniame lange spaudžia „Pradėti paskaitą“. Atidaromas paskaitų sąrašas, rikiuotas pagal paskaitos laiką (nurodoma administravimo dalyje). Paspaudžia ant paskaitos, atsiranda patvirtinimo langas, spaudžia „Pradėti“. Programėlė pradeda skleisti informaciją *Bluetooth* ryšiu. Rodomas paskaitos langas su pasižymėjusiais studentais.

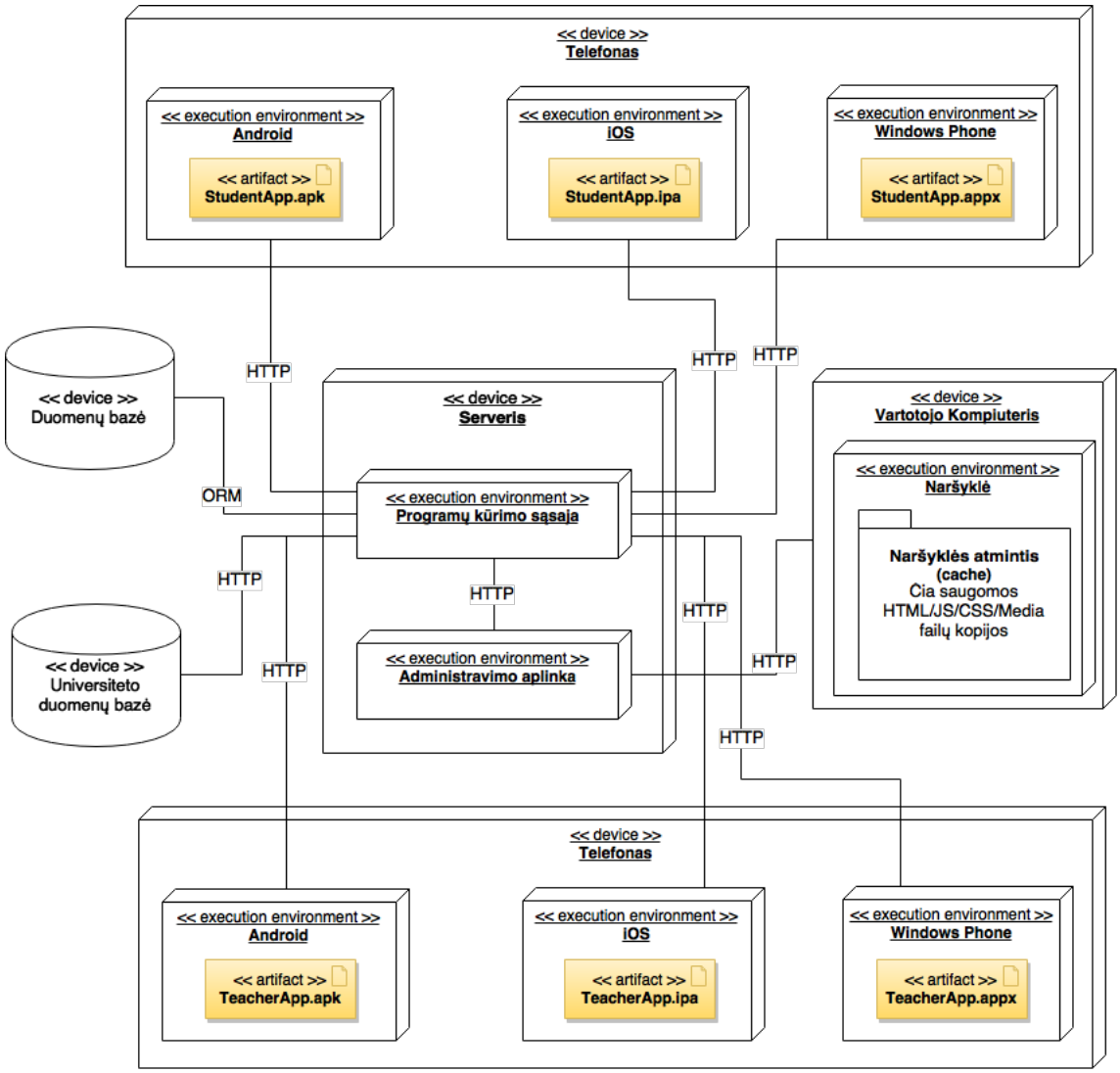
Alternatyvus scenarijus: Pradėjus paskaitą periodiškai ieškoma įrenginių su įjungtu *Bluetooth* ryšiu. Rastų įrenginių *Bluetooth MAC* adresai siunčiami į sistemą.

1.6.2.3. Paskaitos pabaiga

Paskaitos lange 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.

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.



5 pav. Sistemos diegimo vaizdas

Pagal pateiktą diegimo diagramą (5 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. Saityno programų programavimo sąsaja

Visas bendravimas tarp programėlių ir pagrindinio sistemos modulio vyksta per aplikacijų programavimo sąsają. Šis sistemos modulis turi būti sukurtas taip, kad būtų galima nesunkiai įtraukti naują modulį į sistemos architektūrą. 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.

2.1.1. Saityno programų programavimo sąsajų tipai

2.1.1.1. 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.

2.1.1.2. REST

REST metodologija 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 analizuoti programavimo kalboje su kuria yra kuriama programa [Mue13].

Naudojant *REST* saityno išteklius yra lengviau įsisavinti metodologiją 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.

2.1.2. Saugumas

Kaip ir kiekvienoje sistemoje labai svarbu, kad prie išteklių neprieitų asmenys, kuriems nėra suteika teisė naudotis saityno programų programavimo sąsaja. Sagumas šiame modulyje įgyvendinamas dviem lygiais: sąsajos prieigos taškų apsauga ir sistemos duomenų apsauga.

2.1.2.1. 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.

2.1.2.2. 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.

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* nebereikia kiekvienos užklausos metu kreiptis į duomenų bazę, todėl atsakymas yra išsiunčiamas greičiau. Taip pat yra taupomi serverio resursai.

2.2. Administravimo aplinka

Sistemai taip pat reikalinga administravimo aplinka, kurioje galima būtų valdyti naudotojus, įrenginius ar suteikti prieigą naujiems sistemos komponentams. Kaip jau minėta 2.1 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 lape. 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.

2.2.1. Vartotojo sąsaja

2.2.2. Naudojimo scenarijai

2.2.2.1. Naudotojo sukūrimas

2.2.2.2. Naudotojo redagavimas ir ištrynimas

2.2.2.3. Studentų grupės sukūrimas

2.2.2.4. Studentų grupės redagavimas ir ištrynimas

2.2.2.5. Paskaitos sukūrimas

2.2.2.6. Paskaitos redagavimas ir ištrynimas

Rezultatai ir išvados

Rezultatų ir išvadų dalyje turi būti aiškiai išdėstomi pagrindiniai darbo rezultatai (kažkas išanalizuota, kažkas sukurta, kažkas įdiegta) ir pateikiamos išvados (daromi nagrinėtų problemų sprendimo metodų palyginimai, teikiamos rekomendacijos, akcentuojamos naujovės).

Literatūra

[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.

Santrumpos

Sąvokų apibrėžimai ir santrumpų sąrašas sudaromas tada, kai darbo tekste vartojami specialūs paaiškinimo reikalaujantys terminai ir rečiau sutinkamos santrumpos.