

TARTU ÜLIKOOL
Arvutiteaduse instituut
Informaatika õppekava

Gregor Nõges
Mobiilirakendus Ettevõtluskõrgkoolile Mainor
Bakalaureusetöö (9 EAP)

Juhendaja: Helle Hein

Tartu 2020

Mobiilirakendus Ettevõtluskõrgkoolile Mainor

Lühikokkuvõte:

Bakalaureusetöö käigus loodi Eesti Ettevõtluskõrgkoolile Mainor mobiilirakendus, mida saab kasutada nii Android kui ka IOS platvormil. Mobiilirakendus võimaldab Mainori tudengitel tunniplaani vaadata, vajalikke teateid saada, kooli uudiseid lugeda ning muid kooliga seotud infopäringuid teha. Rakendus integreeriti Mainori olemasoleva õppeinfosüsteemiga, mille kaudu teostati kasutajate autentimist, küsiti loengukava ning teisi kooli ja õpilasega seotud päringuid. Mobiilirakendus loodi raamistikus React Native, mis võimaldab mobiilirakendusi arendada nii Android kui ka IOS platvormile.

Võtmesõnad:

Mobiilirakendus, Eesti Ettevõtluskõrgkool Mainor, React Native, Android, IOS

CERCS: P175

Mobile application for Estonian Entrepreneurship University of Applied Sciences

Abstract:

In this thesis, a mobile application was created for the Estonian University of Applied Sciences Mainor, which can be used on both Android and IOS platforms. The mobile application was developed for Mainor students, which allows them to view lesson plans, get the necessary notifications, read school news and other school related information. The application was integrated with the existing Mainor school information system, through which user authentication, lecture schedule, requesting news and other queries were performed. The application was created with React Native framework, which allows mobile applications to be made on both Android and IOS platforms.

Keywords:

Mobile application, Estonian Entrepreneurship University of Applied Sciences, React Native, Android, IOS

CERCS: P175

Sisukord

Sissejuhatus	5
1. Olemasolev lahendus	7
1.1 Alternatiivsed lahendused	8
1.1.1 Google Calendar	8
1.1.2 Arendada veebileht mobiilisõbralikuks	9
2. Nõuded rakendusele	10
2.1 Funktsionaalsed nõuded	10
2.1.1 Sisselogimine	10
2.1.2 Tunniplaani vaade	11
2.1.3 Kooli uudiste lugemine	11
2.1.4 Tõukemärguannete vastuvõtmine	12
2.1.5 Rakenduse keele muutmine	12
2.1.6 Tagasiside andmine rakendusele	13
2.1.7 Väljalogimine	13
2.2 Mittefunktsionaalsed nõuded	14
2.2.1 Rakenduse stiil ja värvide kasutus	14
2.2.2 Platvormide ühilduvus	14
3. Mobiilirakenduste arendusviisid	15
3.1 Erinevad viisid kuidas mobiilirakendusi arendada	15
3.1.1 Veebipõhised rakendused	15
3.1.2 Klassikalised mobiilirakendused	15
3.1.3 Hübriidsed mobiilirakendused	16
3.2 Kasutatud tehnoloogia React Native	17
3.2.1 React Native arhitektuur	18
4. Rakenduse vaated	19
4.1 Avakuva vaade (<i>Splash screen</i>)	19
4.2 Sisselogimise vaade	20
4.3 Tunniplaani vaade	21
4.4 Uudiste vaade	22
4.5 Teadete vaade	22
4.6 Seadete vaade	23
4.7 Tagasiside vaade	24
5. Rakenduse testimine	26
Kokkuvõte	29
Viidatud kirjandus	30
Lisad	32

I.	Programmi lähtekood	32
II.	Litsents	33

Sissejuhatus

Eesti Ettevõtluskõrgkool Mainor (Mainor) on Eesti suurim erakõrgkool, mis on asutatud 1992. aastal ja keskendub ettevõtlusega seotud erialade õpetamisele. Mainoris saab õppida nii bakalaureuse kui ka magistri tasemel ning õppetöö toimub eesti, inglise ja vene keeles. Hetkel õpib Mainoris 1600 üliõpilast ja kõrgkoolil on üle 5000 vilistlase. Õpingud toimuvad nii Tallinnas kui Tartus. Üliõpilaste ja õpingute haldamiseks on kõrgkoolis õppeinfosüsteem (ÕIS), mis võimaldab vaadata tunniplaani, õppekava täitmist, kooliga seotud uudiseid, õpiplaane, esitatud töid ja nii edasi. Samuti saab vaadata ja muuta kooliga seotud dokumente ja maksta õppemaksu [1].

ÕISI suurim probleem on halb kasutajasõbralikkus mobiilseadmetel ning suur ajakulu triviaalse info pärimisel. Näiteks, kui tudeng tahab vaadata tunniplaani, peab ta avama veebilehitseja, navigeerima Mainori koduleheküljele, sisse logima ja navigeerima tunniplaani vaatesse. Samuti, kui on vajadus tudengitele tähtsaid teateid edasi anda, nagu loengukava muudatus, siis kasutab Mainor SMS sõnumite saatmist, mis ei ole kuluefektiivne.

Lõputöö eesmärk on luua mobiilirakendus, mis lahendaks ülaltoodud probleeme. Mobiilirakendusega on mugavam teostada igapäevaseid kõrgkooliga seotud toiminguid: vaadata loengukava nädalate kaupa, lugeda uudiseid ja teateid. Samuti hoiab rakendus tõukemärguannetega (*push notifications*) üliõpilast paremini ja soodsamalt kursis aktuaalsete sündmustega. Lisaks saab tulevikus rakenduses vaadata ülikooliga seotud dokumente, hindeid ja õppekava täitmist. Samuti on plaanis juurde arendada õppekava maksmise ja märkmete tegemise funktsionaalsus. Rakendus on vaja integreerida Mainori õppeinfosüsteemiga.

Autor valis antud teema, sest Mainor soovis endale mobiilirakendust ning autor oli valmis seda tegema. Samuti kasutab autor mobiilirakenduse tegemisel uusi tehnoloogiaid, millega autor ei olnud varem tuttav.

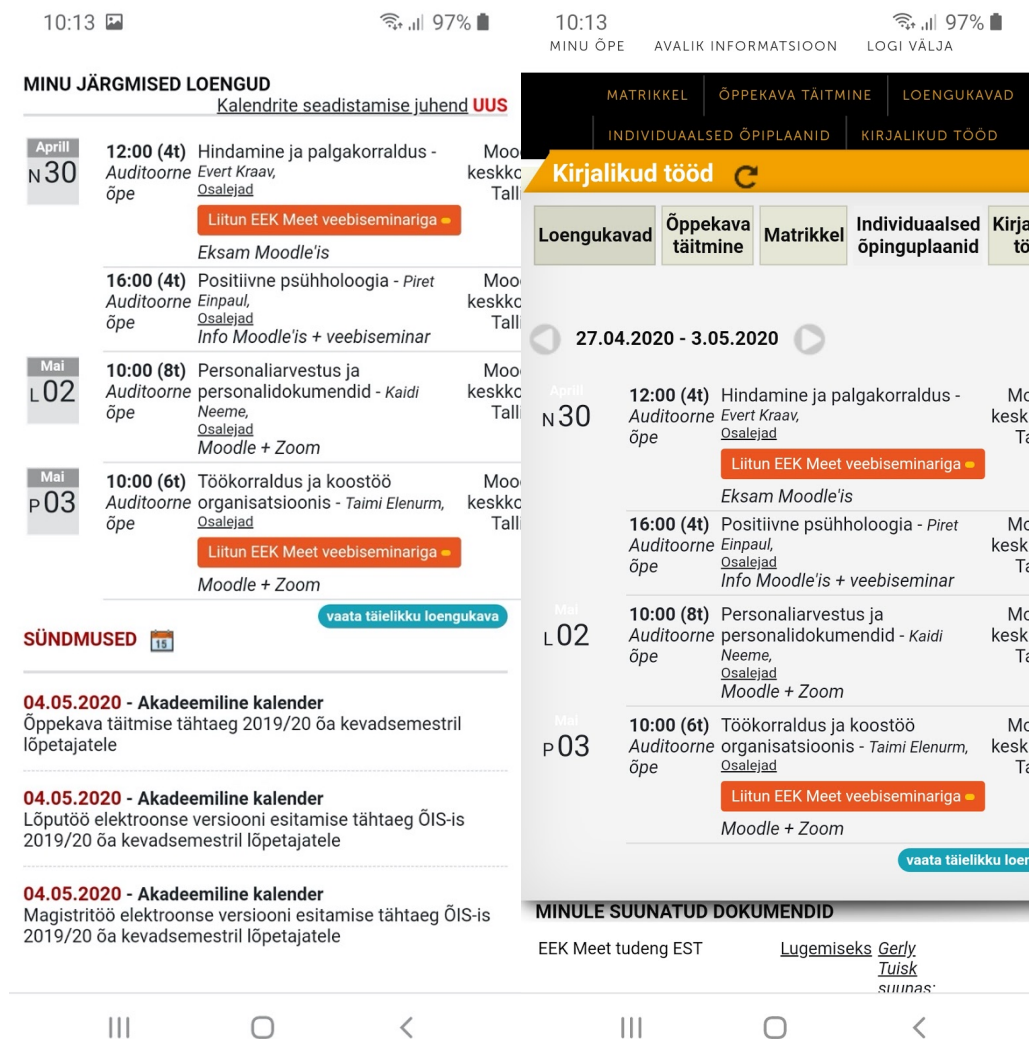
Lõputöö koosneb viiest põhiosast: ülevaade Eesti Ettevõtluskõrgkoolist Mainor ja praegustest ning alternatiivsetest lahendustest, funktsionaalsed ja mittefunktsionaalsed nõuded rakendusele, erinevad mobiilirakenduste arendusviisid, käesoleva rakenduse vaated ning tagasiside. Esimeses peatükis antakse ülevaade Eesti Ettevõtluskõrgkool Mainorist ja uue rakenduse vajalikkusest. Teises peatükis tuuakse välja rakenduse funktsionaalsed ja mittefunktsionaalsed nõuded. Kolmas peatükk räägib erinevatest võimalustest, kuidas

mobiilirakendusi arendada. Neljandas peatükis antakse ülevaade rakenduse erinevatest vaadetest ning kuidas on mobiilirakendus Mainori ÕIS süsteemiga integreeritud. Viies peatükk räägib rakenduse testimise viisidest ja nende tulemustest. Viimases peatükis võetakse lõputöö käigus tehtud töö kokku. Lisadena on esitatud rakenduse lähtekood ja litsents (vt Lisa I ja Lisa II).

1. Olemasolev lahendus

Mainoris on kasutusel Moodle ja õppeinfosüsteem ehk ÕIS, kuhu saab sisse logida nende kodulehel www.eek.ee. ÕISi kaudu saab registreerida ainetele, vaadata tunniplaani ja kooliga seotud dokumente, lugeda ülikooli uudiseid jpm [1].

Mainoril on olemas veebileht, mis on mõeldud peamiselt arvutis kasutamiseks, mistõttu on seal palju funktsionaalsust ja kogu info on ühele lehele kokku pandud. Seetõttu on seal raske navigeerida ja vajalikke asju üles leida. Näiteks, kui tudeng soovib oma tunniplaani vaadata, siis peab ta iga kord sisse logima ja õigele lehele navigeerima. Seejärel näidatakse tudengile kõiki järgmiseid tunde ning on raske eristada tunde, mis toimuvad sellel või järgmisel nädalal. Samuti ei mahu osad lingid ja aknad ekraanile, paljud nupud ja lingid on liiga väikesed, mistõttu võib tudeng tihti sellest mööda vajutada. Üldiselt ei tööta leht sama funktsionaalselt nagu arvuti vaates. Aegkriitiliste ning tähtsate teadete edastamiseks on Mainoris hetkel kasutusel SMS sõnumite saatmine. Joonisel 1 on välja toodud, missugune on Mainori ÕISi kasutamine mobiiltelefonis.



Joonis 1. Mainori veebileht mobiilis (tunniplaani ja kirjalike tööde vaade).

1.1 Alternatiivsed lahendused

Kuna probleem on Mainorile tõsine, siis otsiti viise, kuidas seda lahendada. Järgmisena tuuakse paar alternatiivset lahendust mobiilirakendusele ning selgitatakse, miks need variandid ei sobinud ja miks on vaja ikkagi luua rakendus.

1.1.1 Google Calendar

Google Calendar on Google'i poolt loodud kalendri rakendus, kus saab sündmuseid luua, nendele meeldetuletusi lisada ja inimesi kutsuda. Google Calendar täidab tunniplaani vaatamise ülesande väga hästi ära ning on lisaks kasutatav nii mobiilseadmetel kui ka arvutis. Samuti annab Google Calendar tõukemärguannetega teada, kui sündmusega toimub mingi muudatus.

Kahjuks ei võimalda Google Calendar tasuta versioonis tõukemärguandeid saata, kui kutsutud osalised pole märkinud ennast sündmusel “osalejaks”. On raske nõuda, et tudengid märgiksid ennast igas tunnis “osalejaks”, mistõttu ei saa tasuta versiooni kasutada. Teisalt läheks tasuline versioon Mainorile liiga palju maksma ning samuti ei võimaldaks see teisi vajalikke funktsionaalsusi: vaadata hinded, dokumente, õppekava täitmist jne. Seetõttu jäi Google Calendar valikust välja.

1.1.2 Arendada veebileht mobiilisõbralikuks

Teine võimalus oli arendada olemasolev veebileht mobiilisõbralikuks ning panna kõige populaarsemad infopäringud ja lingid pealehele. Siis oleks tudengitel mugavam tunniplaani ja teateid vaadata ning muid kooliga seotud tegevusi teha. Kuid kuna see variant ei kaota SMS saatmise kulu ära ning tudengid peavad ikkagi veebilehele minema ning sisse logima, siis otsustati ka see alternatiiv kõrvale jätta.

2. Nõuded rakendusele

Rakenduse funktsionaalsed ja mittefunktsionaalsed nõuded töötati välja koostöös Eesti Ettevõtluskõrgkool Mainoriga. Antud peatükis tuuakse välja mõlemad nõuded.

2.1 Funktsionaalsed nõuded

Rakenduse peamised funktsionaalsed nõuded on: rakendusse sisse- ja väljalogimine, tunniplaani vaatamine, Mainori kõrgkooliga seotud uudiste lugemine ja rakenduses tagasiside andmise võimalus. Samuti peaks rakendust olema võimalik kasutada eesti, inglise ja vene keeles. Lõpuks, rakendus peab olema võimeline vastu võtma tõukemärguandeid ning neid kasutajale näitama.

2.1.1 Sisselogimine

Kirjeldus: Kasutaja logib sisse.

Eeldus: Kasutaja omab Mainor ÕISI kasutajakontot.

Põhivoog:

1. Kasutaja avab rakenduse ja jõuab sisselogimise vaatesse.
2. Kasutaja sisestab oma kasutajanime ja parooli ning vajutab nuppu “Logi sisse”.
3. Kasutaja navigeeritakse tunniplaani lehele.

Tulemus: Kasutaja logis rakendusse sisse ning ta sisselogimise andmed jäetakse meelde, et kasutaja ei peaks järgmine kord uuesti sisse logima.

Alternatiivsed vood:

Kirjeldus: Kasutaja on unustanud oma parooli

1. Kasutaja klikib tekstil “Unustasin parooli”.
2. Rakendus viib ta Mainori parooli taastamise lehele, kus kasutaja saab oma parooli uuendada.

Tulemus: Kasutaja saab oma parooli uuendatud ning läbi põhivoo rakendusse sisse logida.

2.1.2 Tunniplaani vaade

Kirjeldus: Kasutaja saab näha tunniplaani.

Eeldus: Kasutaja on sisse logitud.

Põhivoog:

1. Kasutaja avab rakenduse ja ta suunatakse kohe tunniplaani vaatesse.

Tulemus: Kasutaja näeb käesoleva nädala tunniplaani.

Alternatiivsed vood:

Kirjeldus: Kasutaja tahab vaadata järgnevate või eelnevate nädala tunniplaane.

1. Kasutaja avab rakenduse ja ta suunatakse kohe tunniplaani vaatesse.
2. Kasutaja klikib nuppu “Järgmine nädal” või “Eelmine nädal”, mille peale näidatakse vastavalt järgmise või eelmise nädala tunniplaani.
3. Kasutaja vajutab nuppu “See nädal”, mille peale näidatakse kasutajale käesoleva nädala tunniplaani.

Tulemus: Kasutaja saab vaadata erinevate nädalate tunniplaane.

2.1.3 Kooli uudiste lugemine

Kirjeldus: Kasutaja tahab kooliga seotud uudiseid lugeda.

Eeldus: Kasutaja on sisse logitud ja on avanud rakenduse.

Põhivoog:

1. Kasutaja avab külgmenüü vajutades üleval vasakul nurgas olevat burgermenüü nuppu ning valib lingi „Uudised“.
2. Kasutaja navigeeritakse uudiste lehele, kus kasutaja saab uudiseid lugeda.

Tulemus: Kasutaja näeb uudiste pealkirju ning nende uudiste algust.

Alternatiivsed vood:

Kirjeldus: Kasutaja tahab tervet uudist lugeda.

1. Kasutaja vajutab uudise peale.
2. Uudis avatakse täismahus.
3. Uudise sulgemiseks vajutab kasutaja uuesti uudise peale.

Tulemus: Kasutaja saab uudiseid lugeda.

2.1.4 Tõukemärguannete vastuvõtmine

Kirjeldus: Rakendus suudab tõukemärguandeid vastu võtta ning neid kasutajale näidata.

Eeldus: Kasutaja on sisse logitud ja on avanud rakenduse.

Põhivoog:

1. Mainorist saadetakse teade, mis jõuab rakendusse.
2. Kasutaja vajutab tõukemärguande või kellukese ikooni peale.
3. Kasutaja navigeeritakse teadete lehele, kus saab lugeda 30 viimati saadetud teadet.

Tulemus: Kasutaja saab Mainorilt teateid ning neid lugeda.

Alternatiivsed vood:

Kirjeldus: Rakendus suudab tõukemärguandeid vastu võtta ka siis, kui rakendus pole avatud.

Eeldus: Kasutaja on sisse logitud.

1. Mainorist saadetakse teade, mis jõuab rakendusse.
2. Kasutaja vajutab tõukemärguande peale.
3. Rakendus avatakse ning kasutaja navigeeritakse teadete lehele, kus saab lugeda 30 viimati saadetud teadet.

Tulemus: Kasutaja saab Mainorilt teateid ning neid lugeda ka siis kui rakendus on kinni.

2.1.5 Rakenduse keele muutmine

Kirjeldus: Kasutaja tahab rakenduses kasutatavat keelt muuta.

Eeldus: Kasutaja on sisse logitud ja on avanud rakenduse.

Põhivoog:

1. Kasutaja avab külghmenüü vajutades üleval vasakul nurgas olevat burgermenüü nuppu ning valib lingi „Seaded“.
2. Kasutaja navigeeritakse seadete lehele.
3. Kasutaja vajutab keelevahetus nupu peale.
4. Rakendus avab valiku kolmest keelest: eesti keel, inglise keel, vene keel.
5. Kasutaja valib keele vajutades keele teksti peale.

Tulemus: Kasutaja saab muuta rakenduses kasutatavat keele.

2.1.6 Tagasiside andmine rakendusele

Kirjeldus: Kasutaja saab anda tagasisidet rakenduse kohta.

Eeldus: Kasutaja on sisse logitud ning on avanud rakenduse.

Põhivoog:

1. Kasutaja avab külghmenüü vajutades üleval vasakul nurgas olevat burgermenüü nuppu ning valib lingi „Tagasiside“.
2. Kasutaja navigeeritakse tagasiside andmise lehele.
3. Kasutaja vajutab alale, kuhu tagasisidet kirjutada.
4. Rakendus avab klaviatuuri ning kasutaja saab tagasisidet kirjutada.
5. Kasutaja sulgeb klaviatuuri ning vajutab nuppu „Saada“.
6. Rakendus avab informatiivse hüpikakna .
7. Kasutaja vajutab nuppu „OK“.

Tulemus: Kasutaja andis rakendusele tagasisidet, mis edastatakse Mainori IT osakonnale.

2.1.7 Väljalogimine

Kirjeldus: Kasutaja soovib rakendusest välja logida.

Eeldus: Kasutaja on sisse logitud.

Põhivoog:

1. Kasutaja avab külgmenüü, vajutades üleval vasakul nurgas olevat burgermenüü nuppu ning valib lingi „Logi välja“.
2. Kasutaja navigeeritakse sisselogimise lehele.

Tulemus: Kasutaja logitakse rakendusest välja ning ta sisselogimise andmed kustutatakse ära.

2.2 Mittefunktsionaalsed nõuded

Eesti Ettevõtluskõrgkool Mainorile olid mittefunktsionaalsed nõuded vähemtähtsamad, mistõttu oli neid vähem ning rohkem rõhku pandi funktsionaalsetele nõuetele. Seetõttu jäi autorile palju otsustada, kuidas erinevaid lehti rakenduses koostada ning missugused need välja peaksid nägema.

2.2.1 Rakenduse stiil ja värvide kasutus

Mainor soovis, et rakendus võiks olla sarnane nende veebilehele, kuid natukene puhtama stiiliga. Seetõttu võttis autor aluseks Mainor veebilehe, mille tegi ümber minimalistliku stiilile. Tänu sellele on rakenduses näha Mainorile iseloomulikku kollast-oranži värvi kasutust koos väheste nuppude ning teksti plokkidega.

2.2.2 Platvormide ühilduvus

Mainor soovis samuti, et rakendus oleks kättesaadav nii Android kui ka IOS platvormide kasutajatele. Seetõttu valis autor Facebooki loodud mobiilirakenduste loomise raamistiku React Native, millega saab arendada mobiilirakendusi just nendele kahele platvormile [2].

3. Mobiilirakenduste arendusviisid

Selles peatükis tuuakse välja populaarsemad viisid, kuidas mobiilirakendusi arendatakse ning kuidas React Native, mida rakenduse arendusel kasutati, nendest erineb. Samuti kirjeldatakse raamistikku React Native'it ning selle arhitektuuri.

3.1 Erinevad viisid kuidas mobiilirakendusi arendada

On olemas palju erinevaid viise kuidas mobiilirakendusi arhitektuuri poolest jagada. Kolm põhilist ja enim levinumat on: veebipõhised (*progressive web app*), klassikalised (*native app*) ning hübriidsed rakendused (*hybrid app*). Lisaks on olemas ka teistsuguse arenduse arhitektuuriga raamistikke nagu React Native ja Flutter, mis paljude arvates ei klassifitseeru eelmainitud liigenduse alla, vaid on segu neist [3].

3.1.1 Veebipõhised rakendused

Veebipõhised rakendused (PWA) on arenduse poolest kõige kergemad, sest sisuliselt on tegemist veebilehtedega, mis vastavad järgmistele nõuetele: veebileht peab kasutama HTTPS veebi protokollit ja *Service worker*-i ning veebilehel peab olema “manifest.json” fail [4].

Samuti on PWA-d tavaliselt üles ehitatud, kasutades erinevate ekraani suurustele reageerivat disaini (*responsive design*). See pole küll PWA nõue, kuid kui veebileht pole kohandatud mobiilis kasutamiseks, siis on ka kasutajatel ebameeldiv rakendust kasutada ning pigem kasutavad seda arvutis või külastavad konkurendi lehte.

Viimaste aastatega on PWA-d palju populaarsust kogunud ja üha enam suuremad ettevõtted nagu Starbucks ja Twitter on PWA tehnoloogia peale üle läinud. Populaarsuse kasv tuleneb asjaolust, et PWA-d on aastatega uusi funktsionaalsusi juurde saanud ning võimaldavad juba ligilähedast kasutajakogemust klassikaliste mobiilirakendustega [5, 6].

3.1.2 Klassikalised mobiilirakendused

Klassikalised mobiilirakendused on kõige levinumad ja populaarsemad mobiiltelefonis. Klassikalisi mobiilirakendusi omavad ettevõtted nagu Spotify, Snapchat, YouTube, Waze

ja paljud muud. Klassikaliste mobiilirakenduste puhul kirjutatakse rakendus igale mobiilplatvormile eraldi. IOS platvormile kirjutatakse enamasti Swift keeles, kuid toetatakse ka Objective C või C programmeerimiskeelt. Androidi mobiilplatvormile kirjutatakse Java, Kotlini või C++ keeltes [7, 8].

Klassikalistel mobiilirakendustel on palju eeliseid teiste lähenemistega võrreldes. Nad võimaldavad kõige paremat jõudlust, mistõttu on need kasutajale intuiitiivsed, interaktiivsed ja sujuvad. Samuti, kuna rakendus kirjutatakse spetsiaalselt ühele platvormile, siis on tagatud parim rakenduse töötamine platvormi erinevate versioonide vahel. Lisaks saab kasutada ka kõiki funktsionaalsusi, mida mobiiltelefon pakub [3].

Klassikalise mobiilirakenduste arenduse üks suurim miinus on suurem arenduse aeg ja maksumus. Kuna klassikaliste rakenduste puhul peab mõlema mobiilplatvormi jaoks eraldiseisva rakenduse üles ehitama, siis tuleb leida arendustiim mõlema mobiili platvormi jaoks, mistõttu mõlema platvormi arenduse ja hooldamise peale kulub palju aega ja raha. Samuti võib spekuloida, et veebiarendus on kergem, kui mobiilirakenduste arendus, mistõttu on raskem leida arendajaid kindla mobiilplatvormi kohta [3, 10].

3.1.3 Hübriidsed mobiilirakendused

Hübriidne mobiilirakenduste arendus on kolmas suurim viis, kuidas mobiilirakendusi arendada ning selliselt arendatud rakendusi nimetatakse hübriidrakendusteks. Kuigi ei ole olemas selget definitsiooni, mis on hübriidrakendus, on IT kogukondades ja foorumites kõige rohkem levinud järgmine arusaam hübriidsetest rakendustest ja arendusest. Hübriidne arendus on segu klassikalisest ja veebipõhisest arendusest, kus rakendustes kasutatakse mobiilplatvormi *native* veebi näitamise komponenti WebView (Androidi süsteemi peal WebView ja IOS süsteemi peal UIWebView). WebView komponendi funktsiooniks on renderdada veebilehte rakenduses. Arendaja saab Swifti või Java keele asemel kasutada hoopis näiteks HTML, CSS ja JavaScripti programmeerimiskeeli et mobiilirakendust luua. Samuti lubavad mõned raamistikud kasutada ka teisi veebiarenduse raamistikke nagu React.js või Angular. Seega hübriidrakendused on sisuliselt veebilehed, mis on pandud Webview kontaineri sisse, mida näidatakse kasutajale. Populaarsemad raamistikud hübriidsete mobiilirakenduste arendamiseks on Cordova (varem Phonegap), Ionic, Framework 7 ja palju teisi [9, 11, 12].

Hübriidse arenduse suurim eelis on, et ühe lähtekoodiga saab kirjutada rakenduse mitmele mobiilplatvormile. See tähendab, et arendaja võib kirjutada ainult ühe veebilehe ning kasutades näiteks Cordovat, paneb raamistik veebilehe WebView konteinerisse ning viib rakenduse sobivasse formaati, et seda oleks võimalik üles panna ja alla laadida erinevate platvormide poodidest (Androidis on Google Play ning IOS peal on App Store). Ühe lähtekoodi kasutamine vähendab arenduse maksumust ja arendusele kuluvat aega. Sellest tulenevalt on ka rakenduse arendus odavam. [13, 14]

Hübriidse arenduse suurim miinus on rakenduste jõudlus. Kuna hübriidrakendustel on arhitektuuriliselt rohkem kihte ning kuna kogu rakendus pannakse WebView komponendi sisse, on hübriidsed rakendused tavaliselt aeglasemad ja vähem interaktiivsed. Samuti on hübriidisel arendusel suur miinus valitud raamistiku piirangud. Vähem kogenud arendaja saab arendada vaid sellist rakendust, mida raamistik lubab ning kui tal on vaja sellist funktsionaalsust, mida raamistik ei paku, peab ta teise raamistiku valima [3, 9].

3.2 Kasutatud tehnoloogia React Native

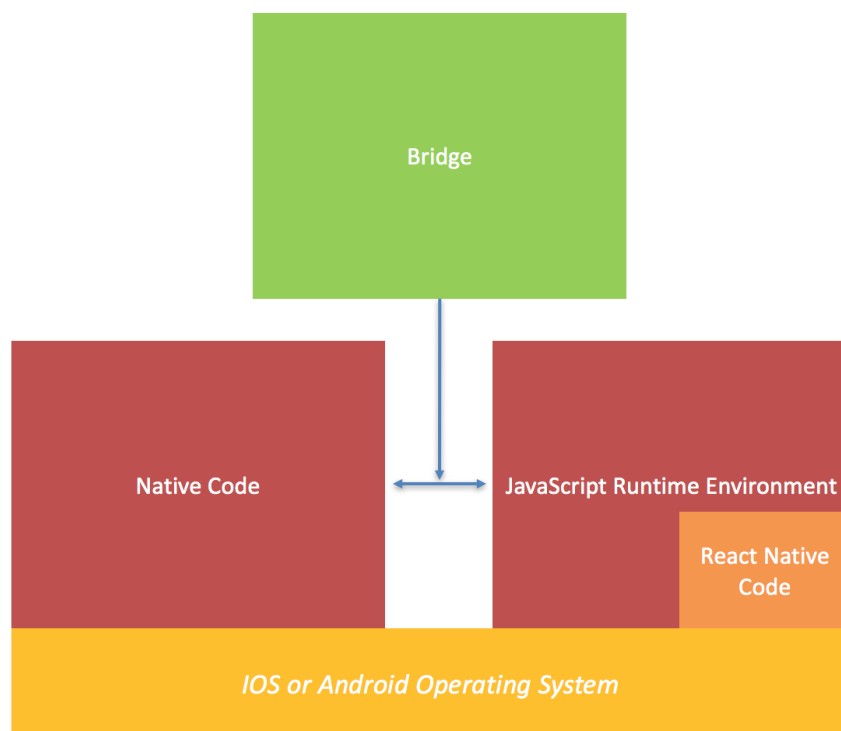
React Native (RN) on vabavaraline raamistik, mis võimaldab mobiiliarendust, mis on segu hübriidsest ja klassikalisest arendusest. RN-ga saab arendada mobiilirakendusi nii IOS ja Androidi platvormidele. RN loodi 2015 aastal Facebooki poolt ning seda on järjepidevalt uuendatud ning edasi arendatud. RN-i teeb eriliseks nende lähenemisviis, kuidas mobiilirakendusi arendada. Tegemist on raamistikuga, kus ühe lähtekoodiga saab teha mobiilirakendusi IOS ja Androidi platvormidele, mistõttu sarnaneb see hübriidrakenduste arendamisele. Sellegipoolest ei ole React Native ei “klassikaline” ega ka hübriidne lähenemine, sest RN kasutab platvormi spetsiifilisi *native* vaateid ja komponente, kuid rakenduse äriloogika arvutatakse eraldi lõime (*thread*) peal. Seetõttu on RN raamistiku poolt arendatavad rakendused parema jõudlusega ja intuitiivsemad, kui teised tavapärased hübriidsed rakendused.

React Native sarnaneb oma süntaksi tõttu väga palju populaarsele veebiarenduse raamistikule React.js, kus mõlemas raamistikus kasutatakse komponente, et mobiilirakendusi või veebilehti arendada. Raamistikude komponendid on siiski erinevad üksteisest, sest React.js-s kasutatakse HTML programmeerimiskeelt, kuid React Native’is saab kasutada vaid React Native’i poolt toetatud komponente. React Native’s kasutatakse äriloogika jaoks Javascripti ning stiili jaoks CSS programmeerimiskeeli. Samuti saab React

Native'i projekti lisada ka teisi arhitektuurilisi komponente nagu andmebaase, seisundi haldussüsteeme (Redux) jpm. [15]

3.2.1 React Native arhitektuur

Kui tavaliselt kasutatakse hübriidsetel lähenemistel WebView ehk veebivaate komponenti, mille sisse pannakse kogu rakendus, siis React Native kasutab mobiiliplatvormide sisse ehitatud komponente ja vaateid ehk *native* funktsionaalsusi. Täpsemalt, React Native'i rakendus jaguneb suuremas jaotuses kaheks: *native* pool ja Javascripti pool. Javascripti pool hõlmab rakenduse ärilooget ja see jooksub JRE-s (*JavaScript Runtime Environment*), milleks on JavaScriptCore. Et React Native mobiilirakendused saaksid kasutada mobiiliplatvormide sisse ehitatud komponente ning vaateid, on vaja nende poole pöörduda ning selle tagab ära *native* pool. *Native* pool on samuti RN rakenduse osa, kuid see on React Native raamistiku poolt valmis tehtud. Samuti on vaja liidestust *native* ja JRE vahel, sest tegemist on erinevate käituskeskondadega, siis on nende kahe vahele ehitatud sild (*Bridge*), mis pakub suhtlust kahe keskkonna vahel [16]. React Native arhitektuur, lihtsustatud kujul, on esitatud joonisel 2.



Joonis 2. React Native arhitektuur [16].

4. Rakenduse vaated

Selles peatükis tuuakse välja erinevaid vaateid, millest rakendus koosneb ning kirjeldatakse, kuidas on mobiilirakendus Mainori ÕISI süsteemiga integreeritud.

Rakendus koosneb mitmest vaatest, mille vahel saab navigeerida vasakul üleval oleva bürgermenüü abil. Eesti Ettevõtluskõrgkool Mainoril on endal õppeinfosüsteem olemas, mistõttu piisas, et antud mobiilirakendus liidestada olemasoleva ÕISiga. Läbi ÕISI toimub kasutaja autentimine ja sisselogimine, vastava kasutaja kohta tunniplaani pärimine, kooli õppetööga seotud üldiste uudiste ning teadete küsimine, tagasiside andmine rakendusele ning tõuketeadete vastuvõtmine.

4.1 Avakuva vaade (*Splash screen*)

Avakuva vaade on alati rakenduse esimene vaade peale rakendus avamist. Seal kontrollitakse, kas kasutaja on sisse logitud või mitte ning sellest lähtuvalt suunatakse ta sisselogimise vaatesse või tunniplaani vaatesse. Senikaua kui kasutaja sisselogimise staatust kontrollitakse, näidatakse avakuva vaates ühekordset animatsiooni, mis jääb lõpuks paigale. Avakuva vaade on välja toodud Joonisel 3.

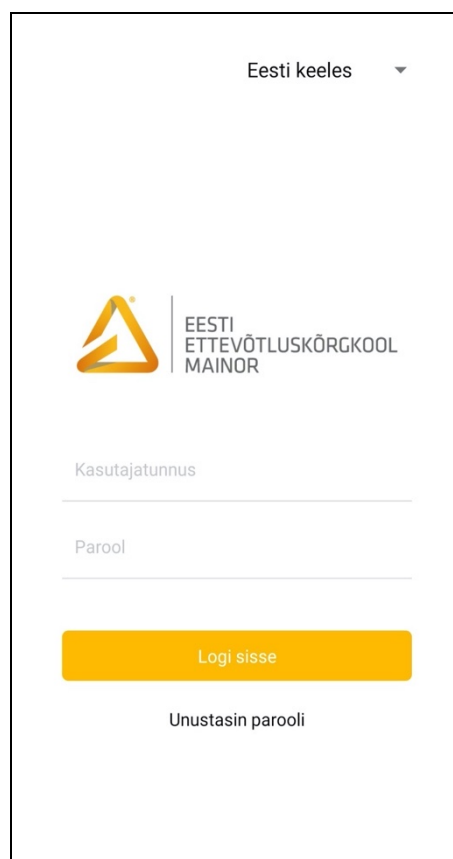


Joonis 3. Avakuva vaade.

4.2 Sisselogimise vaade

Sisselogimise vaatesse navigeeritakse kasutaja siis, kui ta pole rakendusse varem sisse loginud või on välja loginud. Sisselogimise vaates saab kasutaja ennast rakendusse sisse logida, kasutades Mainori õppeinfosüsteemi nime ja parooli. Sisselogimisel kasutab rakendus Mainor ÕISI autentimise teenust, mille põhjal navigeeritakse kasutaja tunniplaani vaatesse või näidatakse veateadet. Samuti saab sisselogimise vaates valida, millist keelt rakenduses kasutada. Valida saab eesti, vene ja inglise keele vahel.

Rakenduses on kasutatud sisselogimist, sest tunniplaani küsimine ja muu õppeinfoga seotud päringud on sensitiivne info, mis ei ole mõeldud avalikuks kasutuseks. Samuti on vaja teada, kelle kohta infot päritakse, mistõttu on sisselogimine vajalik. Sisselogimise vaade on esitatud Joonisel 4.

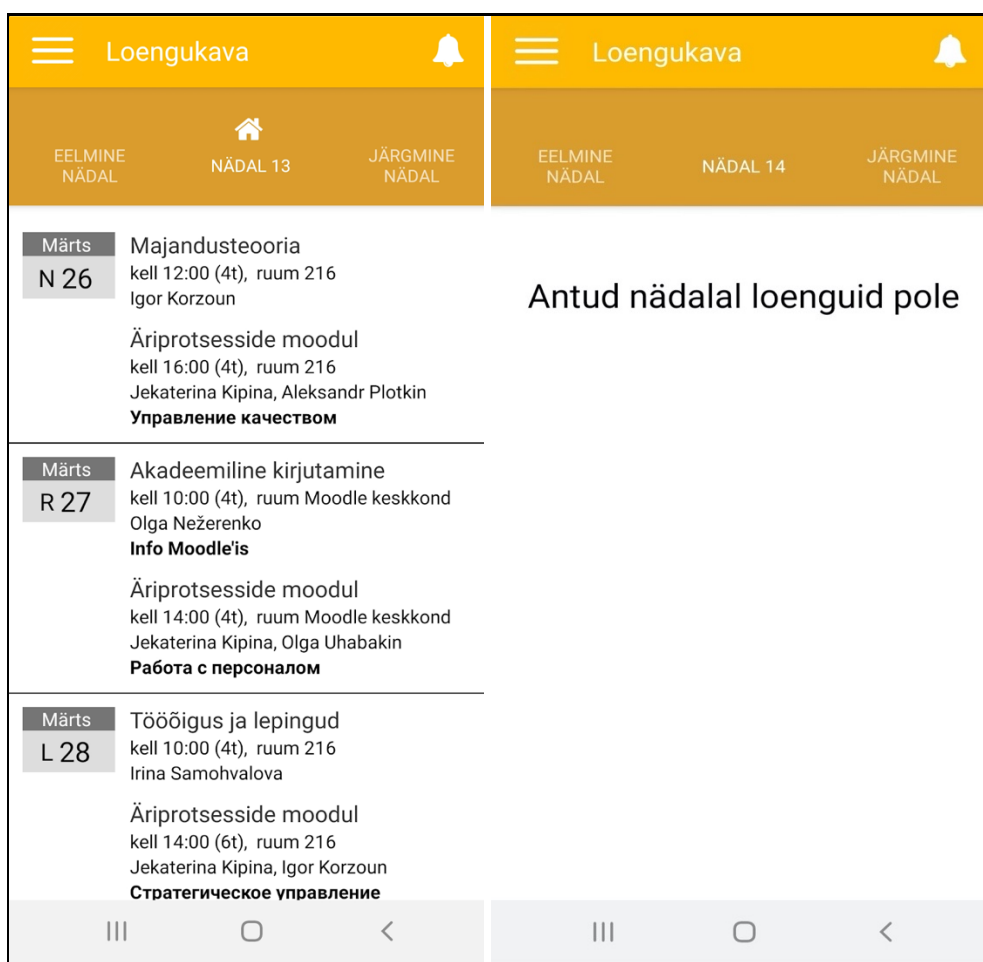


Joonis 4. Sisselogimise vaade.

4.3 Tunniplaani vaade

Tunniplaani vaade on esimene vaade, mida kasutaja näeb, kui ta on rakendusse sisse loginud. Rakendus näitab vaikimisi käesoleva nädala tunniplaani, kuid võimaldab ka eelnevate ning järgnevate nädalate tunniplaani vaadata. Enne kasutajale tunniplaani näitamist küsitakse ÕIS-st kasutaja kohta eelmise, käimasoleva ning järgmise nädala tunniplaani. Seda tehakse põhjusel, et kasutaja saaks kiiremini erinevate nädalate tunniplaanide vaadata. Samuti nädala muutmisel küsib rakendus alati kas eelneva või järgneva (vastavalt kummale poole navigeeritakse) nädala tunniplaani ette, et kasutajal oleks vähem ootamist.

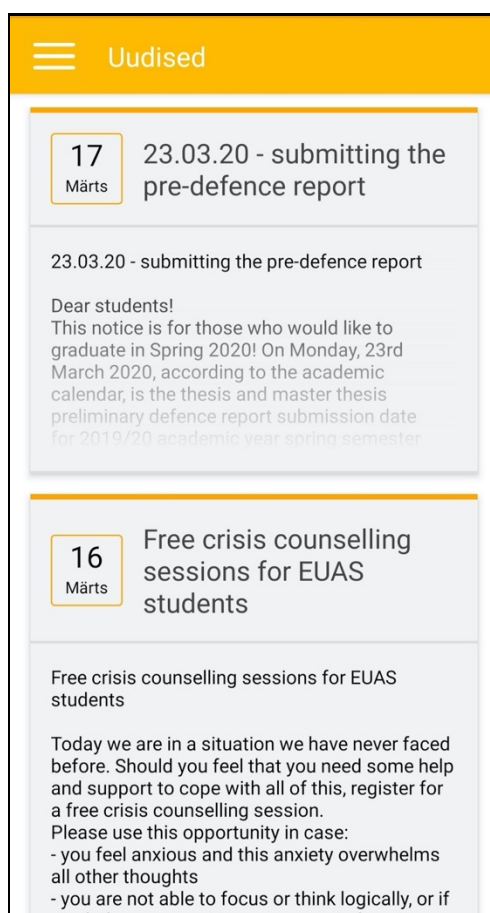
Vaade on jagatud päevadeks, kus on välja toodud kõik tunnid, kuhu üliõpilane on registreeritud. Samuti näidatakse tunniga seotud vajalikku informatsiooni: nimi, kellaaeg, ruum, kestus ning õppejõud. Tunniplaani vaade on esitatud Joonisel 5 ja Joonisel 6.



Joonis 5. Tunniplaani vaade käesoleval nädalal. Joonis 6. Tunniplaani vaade järgmise nädalal.

4.4 Uudiste vaade

Rakendusest saab lugeda ka kooliga seotud uudiseid, nt kuidas kool töötab koroonaviiruse ajal, praktikavõimalused ja palju muud. Uudiste vaatele saab navigeerida, avades külghüppelüü ning valides “Uudised”. Rakendus näitab “Uudiste” vaates 30 hiljutisemat kooliga seotud uudist. Uudised on vaikes olekus suletud ehk näha on ainult uudise pealkirja ja mõned read uudise sisust. Ülejäänud sisu on ära peidetud, et pikemad uudised ei võtaks ekraani peal liiga palju ruumi ära. Uudise lugemiseks tuleb selle peal klikkida, mille peale avaneb uudis täies mahus. Vajutades uuesti uudise peale sulgub uudis. Uudiste vaade on välja toodud Joonis 7 peal.

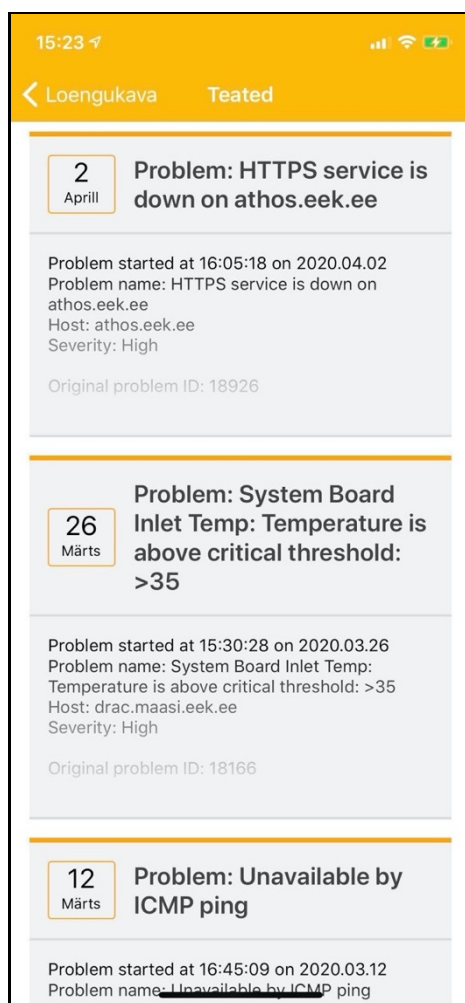


Joonis 7. Uudiste vaade.

4.5 Teadete vaade

Rakenduse teadete lehel on näha kõik teated ehk tõukemärguanded, mida Mainor on tudengile saatnud. Teated on iseloomult tähtsad ja pakilised uudised, nagu tunniplaani muudatus või õppelaenu maksmise tähtaeg. Teadete leht on samamoodi üles ehitatud nagu

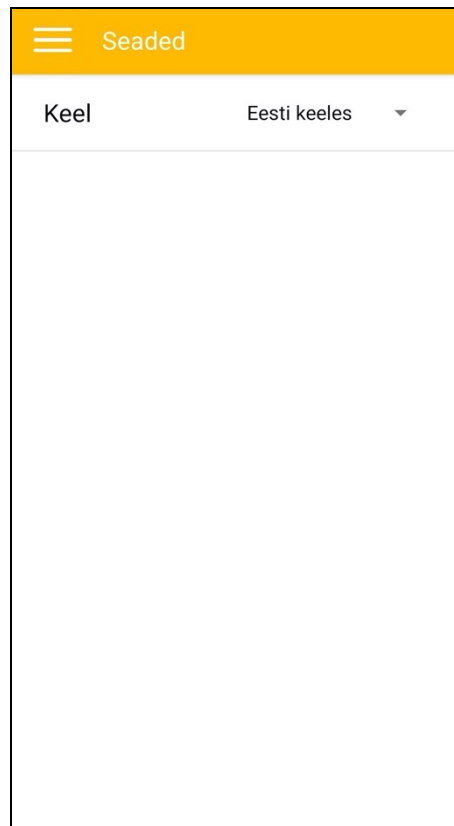
uudiste leht, kus teated on vaikinisi suletud ning avamiseks tuleb nende peale vajutada. Teadete lehele saab navigeerida tunniplaani lehelt, vajutades kellukese ikooni peale. Joonisel 8 on esitatud teadete leht.



Joonis 8. Teadete vaade

4.6 Seadete vaade

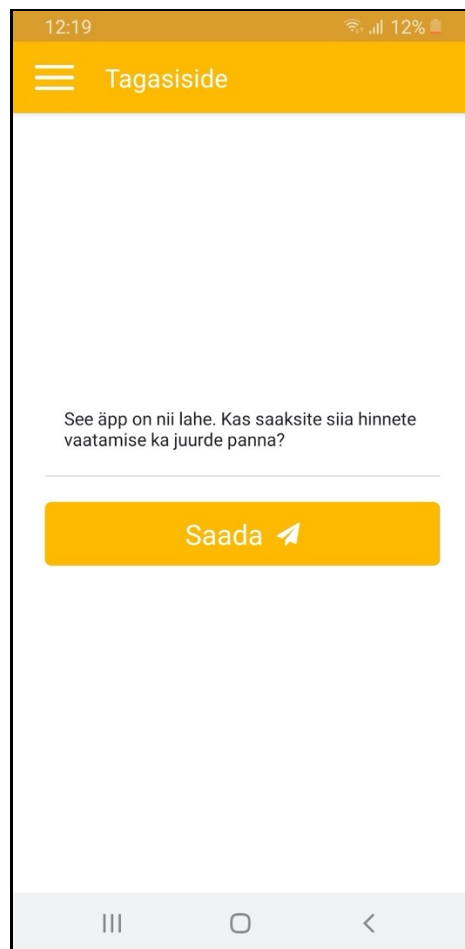
Seadete vaates saab rakenduse kasutaja muuta rakenduse seadeid. Hetkel saab kasutaja ainult muuta rakenduses kasutatavat keelt, kuid tulevikus on plaan lisada sinna juurde tume režiim (*dark mode*), tõuketõmmiste lubamine, muuta tunniplaani vaadet ning lugeda rakenduse informatsiooni ja versiooni. Seadete vaade on esitatud Joonisel 9.



Joonis 9. Seadete vaade.

4.7 Tagasiside vaade

Rakenduses saab anda rakendusele ka tagasisidet. Tagasiside andmise funktsionaalsus on kõige tähtsam rakenduse esimestel kuudel, kus rakenduse toimimises võib esineda vigu ja erinevate telefonide versioonide kokkusobimatust rakendusega. Samuti saab tagasiside vaates anda soovitusi ja uute funktsionaalsuste soovet, mida rakendusse lisada. Tagasiside vaatest antavad teated jõuavad otse arendajate tiimile, mis kiirendab rakendusest vigade leidmist ja nende parandamisele kuluvat aega. Joonisel 10 on esitatud rakenduse tagasiside vaade.



Joonis 10. Tagasiside vaade

5. Rakenduse testimine

Käesolevas peatükis kirjeldatakse rakenduse testimist, selle tulemusi ning antakse ülevaade tehtud parandustest.

Rakenduse testimise etapp oli kiire - rakendus sai valmis 2019 novembri algul ning läks avalikuks 18.11.2019, mis tähendab, et rakendust testiti umbes kaks nädalat. Rakendust testiti väikses suletud grupis, kuhu kuulus autor, Mainori IT osakond ning autori tutvusringkond. Kokku osales testimise protsessis 10 inimest.

Testimine toimus vabas vormis, mis tähendab, et testijal oli voli teha mida testija soovis, kuid pidi oma mõtetest kõva häälega rääkima. Samuti andis testimise läbiviija mõnikord testijale ülesandeid mida teha, nagu nädala vahetamine, keele muutmine, välja logimine ja nii edasi, kui testija ei olnud neid ise juba teinud.

Suletud grupiga testimisel avastati järgmiseid vead ja ebakõlad:

- Avakuva vaade ei töötanud vanemate Androidi mobiilide peal
- Tunniplaani vaates võis nädala number minna lõpmatuseni
- Teksti sisestamisel kattis klaviatuur sisestuslahtri ära
- Uudiste lehel pilt asendada tekstiga
- Polnud vaikeväärtused, mida näidata, kui infot pole.
- Tõukemärguannete salvestamine
- “Pull to refresh” funktsionaalsus ehk et infot värskendada, tuleb vaadet ülevalt alla tõmmata.

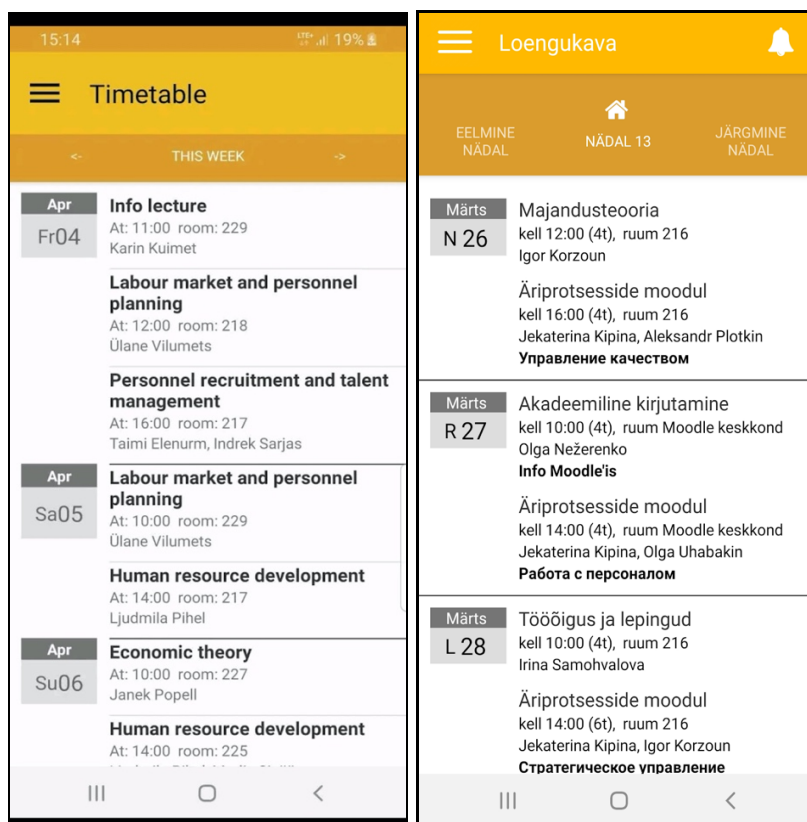
Pärast suletud grupiga testimist otsustati koos Mainori IT osakonnaga teha rakendus avalikuks ning koguda tagasisidet, mida parandada ja muuta. Rakendus on kättesaadav Google Play poes ja Apple App Store-s [17]. Tudengid andsid avaliku versiooni kohta tagasisideks peamiselt rakenduse stiili kohta:

- Lisati rohkem infot tundide kohta
- Rakenduse ikooni muudatused
- Üldise stiili parandused: igal pool kollasel taustal tekst muudeti valgeks, joonduse parandused, parem reageerivus erinevatele ekraani suurustele, teksti suuruse parandused jpm)

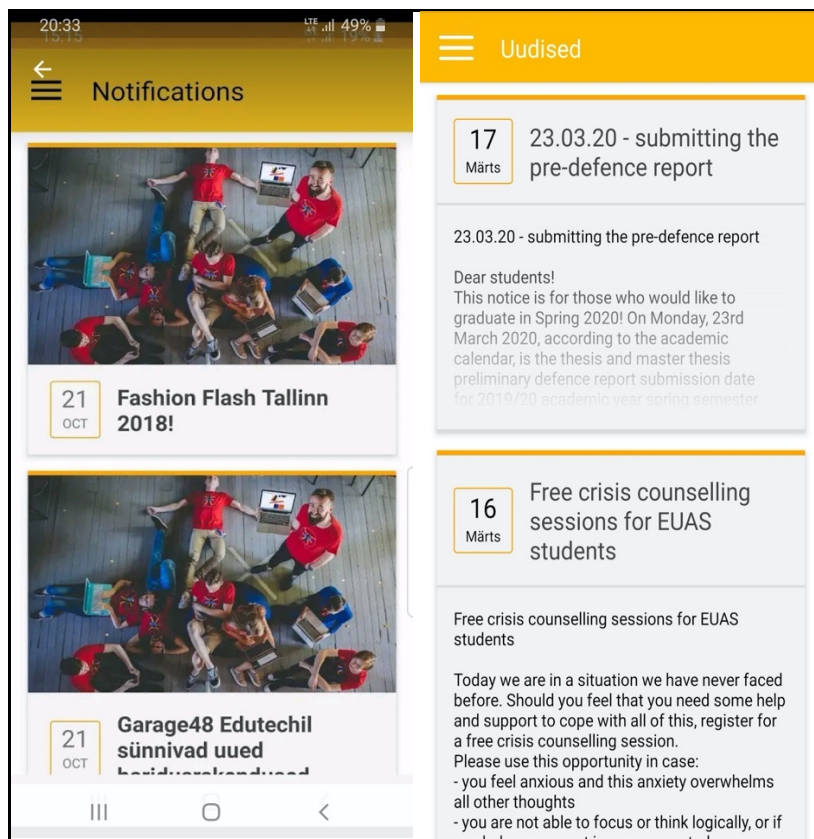
Õnneks ei olnud kogu tagasiside ka ainult negatiivne või konstruktiivne. Samuti anti ka tunnustavat tagasisidet nagu “Imeline, et selline äpp tehti. Loodan, et funktsionaalsus on ka

veel arenemas (loengukava kuuvaade, õppekava/matrikli vaatamise võimalus). Aitäh ja jõudu-jaksu!” ja “This is really nice and I am patiently awaiting more features for sure” ning palju muid.

Rakenduse tunniplaani ja uudiste lehe erinevus enne testimist ja pärast kõikide muudatuste ja vigade parandust on välja toodud Joonisel 11 ja 12.



Joonis 11. Tunniplaani vaate erinevus enne ja pärast testimist



Joonis 12. Uudiste vaate erinevus enne ja pärast testimist

Kokkuvõte

Lõputöö eesmärk oli luua mobiilirakendus Eesti Ettevõtluskõrgkoolile Mainor. Rakendus tegi Mainori tudengitel kõrgkooliga seotud info pärimise, nt tunniplaani vaatamise ja kooli uudiste lugemise kiiremaks ja mugavamaks. Samuti muutis rakendus Mainori tudengitele teavituste saatmise kuluefektiivsemaks, kus SMS sõnumite saatmise asemel saab rakendusse saata tõukemärguandeid. Lisaks on tulevikus plaanis rakendusele juurde arendada hinnete, õppekava, kooli dokumentide vaatamine, õppemaksu maksmine ja palju muud.

Mobiilirakenduste kommuunides eristatakse kolme suuremat mobiilirakenduste arenduse viisi: veebipõhine, klassikaline ning hübriidne arendus. Lõputöö käigus valminud rakendus loodi raamistikuga React Native, mis on segu hübriidsest ning klassikalisest arendusest, võttes head omadused mõlemast arenduse viisist.

Rakendus võeti kasutusele 2019 novembris ja rakendust testiti suletud grupiga. Pärast suletud testimist läks rakendus avalikuks ning tudengid võtsid selle kohe kasutusse. Esimese kuu jooksul anti rakenduse kohta palju tagasisidet ja kõik vead parandati ära ning rakendust täiustati. Järeleulatavat küsitlust ei tehtud, kuid Mainori IT osakonnaga suheldes on teada antud, et Mainori kõrgkool ja tudengid on rakendusega väga rahul ning soovivad rakendusele funktsioone lisada. Rakendus on nimega “EEK Mainor” ning on allalaaditav Google Play poest ja Apple App Store’st.

Viidatud kirjandus

- [1] Eesti Ettevõtluskõrgkool Mainor. (03.12.2019)
<https://www.eek.ee/k%C3%B5rgkoolist/?setlang=est>
- [2] React Native (10.04.2020), <https://reactnative.dev/help>
- [3] Annie Dossey, “A Guide to Mobile App Development: Web vs. Native vs. Hybrid”,
<https://clearbridgemobile.com/mobile-app-development-native-vs-web-vs-hybrid/>
- [4] MDN dokumentatsioon progressiivsetest veebirakendustest
https://developer.mozilla.org/en-US/docs/Web/Progressive_web_apps
- [5] Leht, kus leidub PWA-dega seotu uudiseid <https://www.pwastats.com/>
- [6] Dan Dascalescu, “Why ‘Progressive Web Apps vs. native’ is the wrong question to ask”, <https://medium.com/dev-channel/why-progressive-web-apps-vs-native-is-the-wrong-question-to-ask-fb8555addebb>
- [7] Androidi arenduse kodulehe dokumentatsioon
<https://developer.android.com/guide/components/fundamentals>
- [8] IOS arenduse koduleht programmeerimiskeele Swifti kohta
<https://developer.apple.com/swift/>
- [9] Gigvy, “What’s the Difference between Native vs. Web vs. Hybrid Apps?”,
<https://getgist.com/difference-between-native-vs-web-vs-hybrid-apps/>
- [10] Sam Richard, Pete LePage, “What are Progressive Web Apps?”,
<https://web.dev/what-are-pwas/>
- [11] Dokumentatsioon Adroidi komponendi WebView kohta
<https://developer.android.com/reference/android/webkit/WebView#basic-usage>
- [12] Maximiliano Firtman, “Web-native mobile app frameworks: How to sort through the choices”, <https://techbeacon.com/app-dev-testing/web-native-mobile-app-frameworks-how-sort-through-choices>
- [13] Epifany Bojanowska, “Three Different Ways to Develop a Mobile App”,
<https://naturaily.com/blog/three-ways-to-develop-mobile-apps>
- [14] Apache Cordova dokumentatsioon
<https://cordova.apache.org/docs/en/latest/guide/overview/index.html>

- [15] Tom Occhino, “React Native: Bringing modern web techniques to mobile”,
<https://engineering.fb.com/android/react-native-bringing-modern-web-techniques-to-mobile/>
- [16] Aman Maharjan, “Getting started with React Native: Core Architecture of React Native”, <http://jyaasa.com/blog/getting-started-with-react-native-core-architecture-of-react-native>
- [17] Rakendus “EEK Mainor” Google play poes
<https://play.google.com/store/apps/details?id=com.eekproject&hl=en>

Lisad

I. Programmi lähtekood

1. Mobiilirakenduse lähtekoodi repositoorium – viimati uuendatud 03.20.2020

<https://github.com/Kaljuk/mainorApp>

II. Litsents

Lihtlitsents lõputöö reprodutseerimiseks ja üldsusele kättesaadavaks tegemiseks

Mina, Gregor Nõges,

(autori nimi)

1. annan Tartu Ülikoolile tasuta loa (lihtlitsentsi) minu loodud teose **Mobiilirakendus Ettevõtluskõrgkoolile Mainor**,
(lõputöö pealkiri)

mille juhendaja on Helle Hein,

(juhendaja nimi)

reprodutseerimiseks eesmärgiga seda säilitada, sealhulgas lisada digitaalarhiivi DSpace kuni autoriõiguse kehtivuse lõppemiseni.

2. Annan Tartu Ülikoolile loa teha punktis 1 nimetatud teos üldsusele kättesaadavaks Tartu Ülikooli veebikeskkonna, sealhulgas digitaalarhiivi DSpace kaudu Creative Commons'i litsentsiga CC BY NC ND 3.0, mis lubab autorile viidates teost reprodutseerida, levitada ja üldsusele suunata ning keelab luua tuletatud teost ja kasutada teost ärieesmärgil, kuni autoriõiguse kehtivuse lõppemiseni.
3. Olen teadlik, et punktides 1 ja 2 nimetatud õigused jäävad alles ka autorile.
4. Kinnitan, et lihtlitsentsi andmisega ei riku ma teiste isikute intellektuaalomandi ega isikuandmete kaitse õigusaktidest tulenevaid õigusi.

Gregor Nõges

08.05.2020