TALLINNA TEHNIKAÜLIKOOL
Infotehnoloogia teaduskond

Kristo Erte 179849IADB

ERR VIKERRAADIO JÄRJEJUTTUDE ÜHTSE RAKENDUSE ARENDAMINE IOSI JA ANDROIDI PLATVORMIDELE

bakalaureusetöö

Juhendaja: Jaanus Pöial
doktor
Madis Lauri
bakalaureus

Tallinn 2021

Autorideklaratsioon

Kinnitan, et olen koostanud antud lõputöö iseseisvalt ning seda ei ole kellegi teise poolt varem kaitsmisele esitatud. Kõik töö koostamisel kasutatud teiste autorite tööd, olulised seisukohad, kirjandusallikatest ja mujalt pärinevad andmed on töös viidatud.

Autor: Kristo Erte

29.04.2021

Annotatsioon

Antud bakalaureusetöö eesmärgiks on luua lihtne ja toimiv rakendus Vikerraadio järjejuttude kuulamiseks. Probleem, mida rakendus lahendaks on ERR'i Vikerraadio järjejuttude lihtsa kuulamise võimalus mobiiliseadmetes. Arvestades nutitelefonide populaarsust on see organisatsiooni jaoks väga oluline.

Töös tuuakse välja mobiilirakenduste arendusviisid ning keskendutakse peamiselt hübriidmobiilirakenduste loomise viisidele. Analüüsi osas minnakse süvitsi kolme autori poolt valitud tehnoloogia võrldusega. Mainitud osa lõpuks valib autor välja ühe arendusvõimaluse ja loob selles planeeritud rakenduse.

Bakalaureusetöö teises pooles kirjeldatakse loodud rakendust, tuuakse välja nii sisemine arhitektuur, kui ka kasutajale nähtavad vaated ning seletatakse nende tööd ja funktsionaalsust.

Lõpuks toob autor välja ka loodud lahenduse puudused, töö ajal valminud uued muudatused ning ka tulevikuplaanid.

Töö on kirjutatud eesti keeles ning sisaldab 6 peatükki 27 leheküljel ning 20 joonist.

Abstract

The aim of current thesis is to create ERR Vikeradio's series application for iOS and Android platforms. The purpose of creating this application is to create clean and working solution for Vikeraadio. The analysis of the existing solution concluded that the application in use is based on outdated technology and difficult to use.

First part of thesis is analysis about three hybrid mobile app development frameworks. Author compared their performance and popularity and chose one of them to write the solution.

.As a result of analysis, an application was created. Second part of thesis consists descriptions of inner architecture and views. Lastly, there is a part about problems yet to solve and future about the application.

This thesis is written in English and is 27 pages long, including six chapters, twenty figures.

Lühendite ja mõistete sõnastik

ERR	Eesti rahvusringhääling
REST	Representational state transfer - - veebiteenusega suhtlemise liidese arhitektuur
JSON	JavaScript Object Notation ehk Javascript'il põhinev andmevahetusvorming
HTTP	Hypertext Transfer Protocol - hüperteksti edastusprotokoll

Sisukord

Jooniste loetelu

1 Sissejuhatus

Tänapäeval on inimeste elus lahutamatu koht mobiiltelefonil. Võib öelda, et tihti saadab nutiseade meid hommikust õhtuni. Käesoleva töö käigus loodud rakendus on loodud mõttel mugavalt ja lihtsalt võimaldada Vikerraadio järjejuttude kuulamist. Ehkki Vikerraadiol on olemas ka veebi- ning veebiraadio, siis tuleb välja, et inimesed veedavad rakendustes 89% nutiseadme kasutusajast. Veebilehtedel kulutatakse kõigest 11% [1].

Esiteks tuleb välja tuua probleem, mis andis tõuke uue arenduse loomiseks. ERR-il puudub hetkel ainult Vikerraadio järjejuttude mängimiseks rakendus. Mainitud kategooria saated on olemas veebis ning segamini teistes vanades rakendustes. Lisaks oli osakonnast töölt lahkunud ka arendaja, kes valdas vana rakenduse arendustehnoloogiat.

Antud probleemi lahendamine on küllaltki aktuaalne, kuna praegusel ajastul on organisatsiooni jaoks oluline omada konkurentsivõimelist mobiilirakendust. Hetkel pole ka ERRi mobiilirakenduste hulgas ka ühtegi hübriid-mobiilirakendust. Selle tõttu soovis bakalaureusetöö autor luua uut väärtust töökoha arendusosakonda. Varasemad loodud mobiilirakendused on töökohas loodud platvormipõhiste lahendustena. Autoril on endal tekkinud varem olukordi, kus ei tea millist hübriidmobiilirakenduse tehnoloogiat õppida või eelistada. Sellest tulenevalt soovis autor tuua välja võrdluse erinevate tehnoloogiate vahel.

Töö lähtetingimusteks on ERR'il oma infosüsteem, mis haldab kõiki organisatsiooni veebirakendusi, mobiilirakendusi ning teisi lahendusi. Autori poolt oli varasemalt valmis kirjutatud REST-ahitektuuriga päringud, mis tagastab andmebaasist JSON vormis infot järjejuttude kohta. Samuti on rakenduses olevad audiofailid pärit samast serverist. Nende vahenditega on eesmärgiks luua disanilt puhas, lihtsasti kasutatav ja ühte funktsiooni hästi täitev rakendus, mis oleks hübriidrakendus nii iOSile kui Androidile.

Töövahenditest kasutati peamiselt Visual Studio Code lähekoodiredaktorit. IOS'i osas muudatuste tegemiseks oli vajalik Xcode 11 ning androidi poolel kasutas autor Android Studio versiooni 4.1.3. Rakenduse disain ei ole autori poolt loodud, vaid selle valmistas arendusmeeskonnas töötav disaner.

Töö koosneb viiest suuremast peatükist. Alustuseks kirjeldatakse sissejuhatuses bakalaureusetöös lahendatavat probleemi ning aktuaalsust. Teises peatükis tuuakse välja olemasoleva rakenduse puudused ja põhjus uue lahenduse loomiseks. Samuti kirjeldatakse põgusalt ka uut rakendust. Järgmine peatükk on lühike ülevaade mobiilirakenduste arendamisvõimalustest, milleks on platvormipõhised- ja hübriidtehnoloogiad. Kuna töö lähtetingimuseks oli hübriidmobiilirakenduse tehnoloogia kasutamine, siis platvormipõhiste viiside kirjeldamisega süvitisi ei minda, vaid neljas peatükk on loodud kirjeldamaks kolme valitud hübriidtehnoloogiat ja nende võrdlust. Plaanis on võrrelda eelkõige populaarsust, kiirust ja tulevikupotensiaale. Peatüki lõpuks tuleb valida üks tehnoloogia, milles rakendus luua. Viies peatükk kirjeldab loodud lahenduse arenduskäiku, tuuakse välja pildid vaadetest ja navigeerimisest. Samuti tuuakse välja probleemsed kohad ning tulevikuplaanid.

2 Varasem rakendus ning uus loodav rakendus

Siin peatükis toob autor välja vana rakenduse kirjelduse ning puudused ning uue loodava rakenduse kirjelduse.

2.1 Varasema rakenduse kirjeldus

Varasem järjejuttude kuulamise rakendus loodi aastal 2017 aastal kasutades Unity arenduskeskkonda ning ülejäänud osa oli arendatud platvormipõhiselt. Valmis lahenduses oli rohkelt erinevaid kategooriaid, mis olid segamini ning aegunud. Lisaks oli suur osa lahendusest Unitys ehitatud mängu kujul, mis ei leidnud enam kasutust. Lisaks oli arendusmeeskonnast lahkunud töötaja, kes valdas eelmainitud arendustehnoloogiaid. Veel tekkisid rakenduses vead aja möödudes mainitud mängudega. Tundus mõistlikum luua uus versioon rakendusest kasutades mõnda arenenumat tehnoloogiat, kui panustada ressurssi arendajal Unity õppimisele.

2.2 Loodava rakenduse kirjeldus

Bakalaureusetöö autorilt organisatsioonis töötades telliti uus rakendus, mis oleks arhitektuurilt puhas ning lihtsasti kasutatav. Lisatingimus oli, et rakenduse kategooriaid ning sisu oleks mugav muuta ERR'i suurest infosüsteemist. Tähtis oli, et rakendus töötaks sujuvalt ning laadiks serverist kiirelt andmeid ja kuvaks kasutajale.
Omalt poolt seadis autor pakkumise, et kirjutada lahendus kasutades mõnda uut tehnoloogiat, et anda ka teistele arendajatele meeskonnas võimalus silmaringi avardada. See oli suuresti ka autori enda huvist, kuna varasemalt on ta olnud dilemma ees. Hübriidmobiilirakenduste loomise tehnoloogiad on mitmeid, kord kiidetakse ühe, kord teist, kuid milline oleks õige valik. Uurimustöö analüüsi osa annab võimaluse selgusele jõuda selles.

3 Mobiilirakenduste arendusviisid

Mobiilirakendusi võib arendada kahel võimalikul viisil. Esiteks platvormipõhiselt ehk kas iOS'i keskkonnas või siis androidi opertasioonisüsteemile vastavaid töövahendeid kasutades. Siin peatükis toob autor välja lühikese kirjelduse mõlemast arendusviisist.

3.1 Platvormipõhised mobiilirakendused

Tänapäevased nutitelefonid töötavad enamuses kahe peamise platvormi peal, milleks on iOS ning Android. Lisaks on veel 2.6% osakaaluga Microsofti platvorm [2]. Platovormipõhiseks mobiilirakenduseks nimetatakse rakendust, mis on arendatud kasutades tehnoloogiat, mis on toetatud selle nutitelefoni operatsioonisüsteemi poolt, kas siis iOSile või Androidile. Antud arendusmetoodika tugevuseks võib pidada paremat rakenduse jõudlust, lihtsamat arendustööd arendajale. Samas, kui soov on luua mõlemale platvormile samasugust rakendust, siis arenduskulu suureneb mõlemale operatsioonisüsteemile arendatava aja võrra [3]. Veel võib välja tuua kahe põhilise platvormi osakaalude suhte, milleks uurimustöö kirjutamise ajal 82.8% Androidile ning 13.9% iOS'ile [2].

3.2 Hübriidmobiilirakendused

Hübriidmobiilirakendused on rakendused, mis on arendatud ühe koodibaasi põhjal toetama mõlemat platvormi samaaegselt. Siit tuleb ka peamine eelis võrreldes platvormipõhiste rakendustega. Kui on soov luua samasugune rakendus mõlemale platvormile, siis on suur ajavõit kirjutada ainult ühe korra kood, mis töötab iga operatsioonisüsteemiga seadmetes. Muidugi tuleb arvesse võtta, et mingid eripärad võivad olla erinevatel platvormidel, näiteks erinevate sensorite kasutus, kasutajaliidese eripärad ning Apple, Google ja Windowsi rakenduspoodide erinevus. Mainitud olukord puudutab siiski üldjuhul umbes 20% koodist [3].

4 Hübriidmobiilirakenduste tehnoloogiate võrdlus rakenduse arendmiseks

Rakenduse loomise esimeses faasis oli seatud lähtetingimus, et igale platvormile peab tulema samasugune rakendus. Autoril oli töökogemus olemas iOS platvormil rakenduse loomisel, kuid Androidi osas oli teadmistes puudujääke. Viimase osas oleks pidanud töö autor pidanud kulutama aega selle õppimiseks. Sellest tulenevalt oli mõistlik otsus luua hübriidmobiilirakendus.
Sellest tingituna pidi autor tegema valiku erinevate tehnoloogiate vahel. Töö koostajale oli ajapikku jäänud meelde kolm nime: Windows Xamarin, React Native ning Flutter. Isikliku kogemuse põhjal otsustas autor, et analüüsib just neid kolme tehnoloogiat ja teeb valiku nende seast. Järgnevates alampeatükkides toob autor välja kõigi kolme kirjelduse välja.

4.1 Microsoft Xamarin

Xamarin on aastal 2011 loodud hübriidrakenduse tehnoloogia, mis kasutab Microsofti .NET platvormi ning C# keelt loomaks siduvat kihti üle androidi ning iOSi operatsioonisüsteemide rakendusliidese. Mainitud lahendus annab võimaluse täielikult kontrollida platvormi eripärasid. Kui arendaja on tuttav C# keelega, siis on tal küllaltki lihtne alustada arendust Xamariniga [4]. Visual Studioga saab lisada paketti, kus on koheselt kõik olemas: Xamarini SDK, testimiskeskkond ja analüüsitarvikud. Lubatud on, et 80-90% koodist on jagatav platvormipõhiselt, mis kiirendab arendusprotsessi. Lisaks toimub kogu arendus Visual Studio keskkonnas ehk pole vajadust kasutada iOS'i Xcode ega Android Studiot. Lubatud on ka Xamarini rakendusel platvormipõhise lahendusega ligilähedast kiirust, kuid reaalsus on näidanud, et Xamarin jääb siiski veidi alla. Mainitud hübriidmobiilirakenduse tehnoloogial on ka omad puudused. Androidi ja iOS'i uuendustega järje pidamine võtab aega, kuna see sõltub täielikult Xamarini arendustiimist. Veel oluline punkt on limiteeritud valikuvõimalus kasutada vabavaral põhinevaid teeke. Arendusfaasis oled sunnitud kasutama .NET vabavara lahendusi, mis

võib seada rakendusele piire. Organisatsioonile on ka kallis osta litsentsi Microsofti arendustehnoloogiatele, kulud võivad kujuneda tuhandetesse eurodesse. Tulevikule mõeldes on ka muret tekitav asjaolu, et Microsoft teatas, et Xamarin.Formsi toetamisest loobutakse novembris 2021. [5] See uudis teeb ettevaatlikuks uue rakenduse loomise kasutades mainitud tehnoloogiat. Kõige viimaks on ka Xamarini arendajate arv väiksem kui platvormipõhiste rakenduste ning näiteks React Native kasutajate hulgaga, mis teeb omakorda raskemaks probleemide korral vastuste leidmise [5].

4.2 Facebooki React Native

React Native on aastal 2015 loodud Javascripti keele raamistik, mille eesmärgiks on muuta Javascriptis kirjutatud kood platvormipõhiseks. Tehnoloogia baseerub Facebooki loodud React raamistikul, mida kasutatakse veebirakenduste loomisel. Kood kirjutatakse Javascripti keeles ning kasutades JSX laiendit. Koodi kompileerumisel kasutatakse mobiilirakenduse operatsioonisüsteemi kasutajaliideseid, mille tulemusel saavutatakse lahendus, mis näeb välja ja käitub nagu platvormipõhine mobiilirakendus. Tehnoloogia loomisel arvestati seda, et varasemad tehnoloogiad kompileerudes kasutasid siiski veebivaateid, mis üritasid jäljendada platvormipõhist väljanägemist ja funktsionaalust. Kuid see kulutas palju resurssi ning ei olnud täiesti identne. React Native kompileerib koodi iOS'i puhul Objective-C-ks ning Androidi puhul Java keelde. Arendajale, kes on tuttav React raamistikuga on õppimiskurv eriti väike. Lisaks saab kasutada React Nativega kõiki veebibrauseri jälgkäitust. React Native eeliseks peetakse arenduskiirust [6]. Näiteks teatas Walmart, et nende mobiilirakenduses jagatakse iOS'i ning Androidi vahel 95% koodist kasutades React Native.[7]. Ka jälgjälitus on kiirem kui platvormipõhiste mobiilirakenduste loomisel. Samuti tegeleb Facebook endiselt raamistiku arendamise ning toetamisega, võrreldes näiteks Xamariniga, kus Microsoft teatas mingi osa raamistikust loobumisest. React Native puuduseks võib pidada jõudlust keerulisemate rakenduste loomisel ning piiratud kolmandate osapoolte teekide hulgast [7]. Autor ise ei ole kunagi React raamistikku kasutanud, seega tema jaoks oleks React Native õppimiskurv küllaltki suur ning sellega kaoks arenduskiiruse eelis, mida rõhutatakse.

4.3 Google Flutter

Flutter on hübriidmobiilirakenduste arendamise platvorm, mis on loodud aastal 2018. See on täiesti tasuta kasutatav, Google poolt toetatud ning vaba lähtekoodiga. Flutteri loomisel ei olnud eesmärgiks luua tehnoloogia, millega arendada mobiilirakendusi, vaid luua täiesti igal platvormil töötav lahendus. Lahenduse üks missioon oli teha tehnoloogia, mis võimaldaks luua igale platvormile rakendusi paremini, kui operatsioonisüsteemi enda pakutud võimalusega. Võib öelda, et Flutteri arendustiim on saanud oma eesmärgiga hakkama. Google suutis luua Flutteriga androidi rakenduse, mis näitas mõnes testis kolmekordselt paremat kiirust võrreldes Androidi platvormil loodud rakendusega. Flutteri arenduskiirus on võrreldes kahe eelmisega samuti kiire, kuna kirjutatakse kood üks kord igale platvormile [8]. Autor katsetas Flutteri koodi laadimiskiirust ning tänu *hot reload*'ile ilmusid muudatused seadmes silmapilgselt. Suur tugevus on ka kasutajaliidese loomisel. Flutteriga on võimalus luua lõuendile joonistada täiesti enda disainitud vidinaid ehk disainivalikud on väga rikkalikud. Samas on võimalik kasutada ka olemasolevaid vidinaid, mille hulk on samuti suur. On võrreldud ka Flutteris arendatud rakenduse töötamist vanemates seadmestes ning tuli välja, et rakendus töötab identselt näiteks Android 5.1.1 ja Android 8.1.0 versioonidega. Eelmainitult on lubatud ka platvormipõhite tehnoloogiatega peaaegu identset jõudlust [9]. Google tugi teekidele on ka muljetavaldav. Puudustest võib välja tuua, et mainitud raamistik on küllaltki uus veel ja ei ole veel saavutanud ülal mainitud raamistikega sarnast stabiilsust, pidevalt tuleb uusi uuendusi juurde [10].

4.4 Hübriidmobiilirakenduste tehnoloogiate kiiruse võrdlus

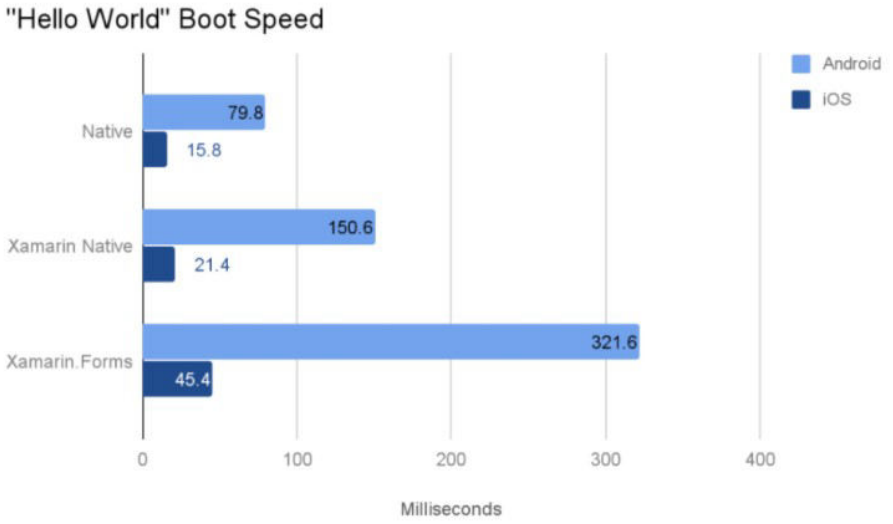
Autor otsustas esimesena võrrelda ülal mainitud valikute töökiirust. Jõudlust võib võrrelda mitmeti, näiteks kui kiiresti suudab raamistik suhelda seadme rakendusliidesega (kaamera, failisüsteem, gps). Teisalt on tähtis kui palju aega kulub kasutajaliidese laadimisele ning kolmandaks ka äriloogika osa ehk mis kiirusega suudab rakendus tegeleda mälu ning matemaatiliste arvutustega.
Autor toob esimesena välja võrdluse Xamarini jõudluse võrreldes platvormipõhiste lahendustega.

4.4.1 Xamarini võrdlus platvormipõhiste rakendustega

Alustuseks otsustas autor võrrelda internetis avalikult olevate materjalide abil Microsoft Xamarini raamistikku platvormipõhiste tehnoloogiatega. Võrdlusesse võeti eelkõige rakenduse käivituskiirus ja REST arhitekuuri päringute kiirus, kuna just seda hakkab loodav rakendus tegema. Seejärel vaadeldi andmebaasi päringute kiiruseid.

1. "Tere tulemast" rakenduse käivituskiirus

Lihtsa "Tere tulemast" rakenduse käivitamisel on näha, et Xamarin on ligi 2 korda aeglasem kui platvormipõhiselt loodud rakendus. Kasutades Xamarin.Formsi on vahe ligi 4 korda.

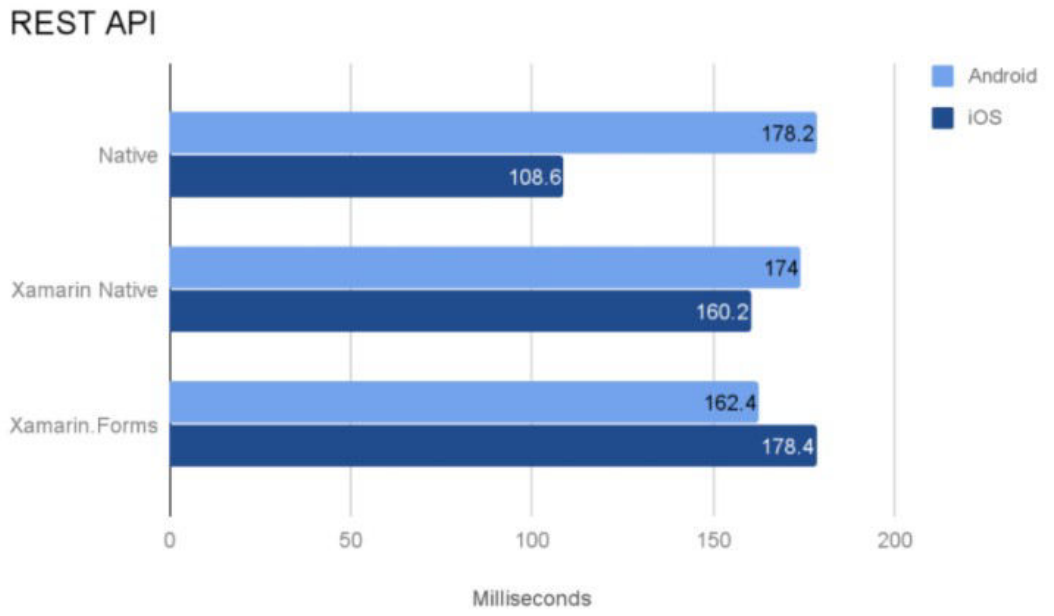


Joonis 1. Xamarini rakenduse käivituskiirus võrreldes platvormipõhiste rakendustega.[11]

Lihtsa "Tere tulemast" rakenduse käivitamisel on näha, et Xamarin on ligi 2 korda aeglasem kui platvormipõhiselt loodud rakendus. Kasutades Xamarin.Formsi on vahe ligi 4 korda.

2. REST tarkvaraarhitektuuri laad

Kui rakendus on käivitunud, siis on sinna vaja laadida andmeid, tihti välisest serverist REST päringuga. Lõputöös loodavas rakenduses tehakse päringuid ka vaadete vahetamisel, seega peaks toimuma see kiirelt. Järgnevas testis vaadeldi Rest päringu laadi Xamarini ning platvormipõhiselt loodud rakenduses.

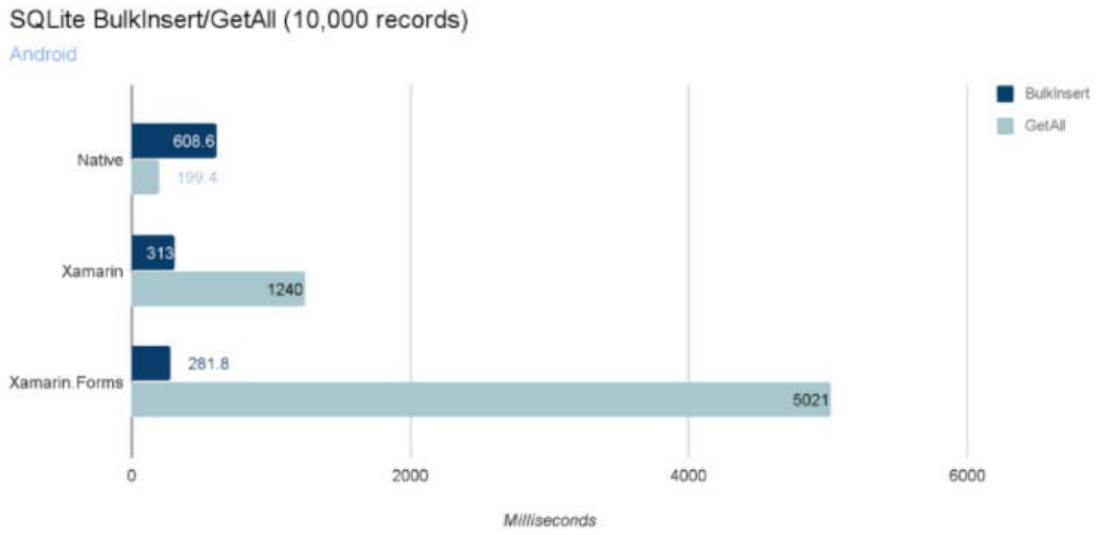


Joonis 2. Xamarini REST päringu laad võrreldes platvormipõhise lahendusega. [11]

Joonisel on näha, et Androidiga suudab mainitud hübriidtehnoloogia sammu pidada, kuid iOS on kiirem.

3. Andmebaasi päringud

Rakenduse loomisel on andmebaasiga suhtlus üks kõige olulisemaid osasid. Tihti kasutaja tegutseb rakenduses ning siis on vajalik andmeid salvestada andmebaasi. Kui suhtlus andmebaasiga on aegalne, siis pole ka rakenduse kasutus sujuv.



Joonis 3. Xamarini andmebaasipäringu kiirus võrreldes platvormipõhise lahendusega [11]

Xamarin suutis andmete sisestamisel platvormipõhiste lahendustega järge pidada, kuid andmete pärimisel kulus Xamarin.Fromsil lausa 5 sekundit, et kõik andmed kätte saada.

4. Rakenduse suurus

Rakenduse loomisel tuleb ka mõelda, kui palju võtab lõplik lahendus kasutaja seadmes mälu. Siin ei suuda Xamarin platvormipõhiste rakendustega võrrelda. Androidi rakendusega võrreldes on mälukasutuse suures kahekümne kordne.
Joonis 4 Xamarini rakenduse suurus võrreldes platvormipõhise lahendusega

Android			**iOS**		
Native	Xamarin.Android	Xamarin.Forms	Native	Xamarin.iOS	Xamarin.Forms
0,4 Mb	8,2 Mb	15,6 Mb	0,4 Mb	3,6 Mb	11,8 Mb

Joonis 4. Xamarini rakenduse suurus võrreldes platvormipõhise lahendusega [11]

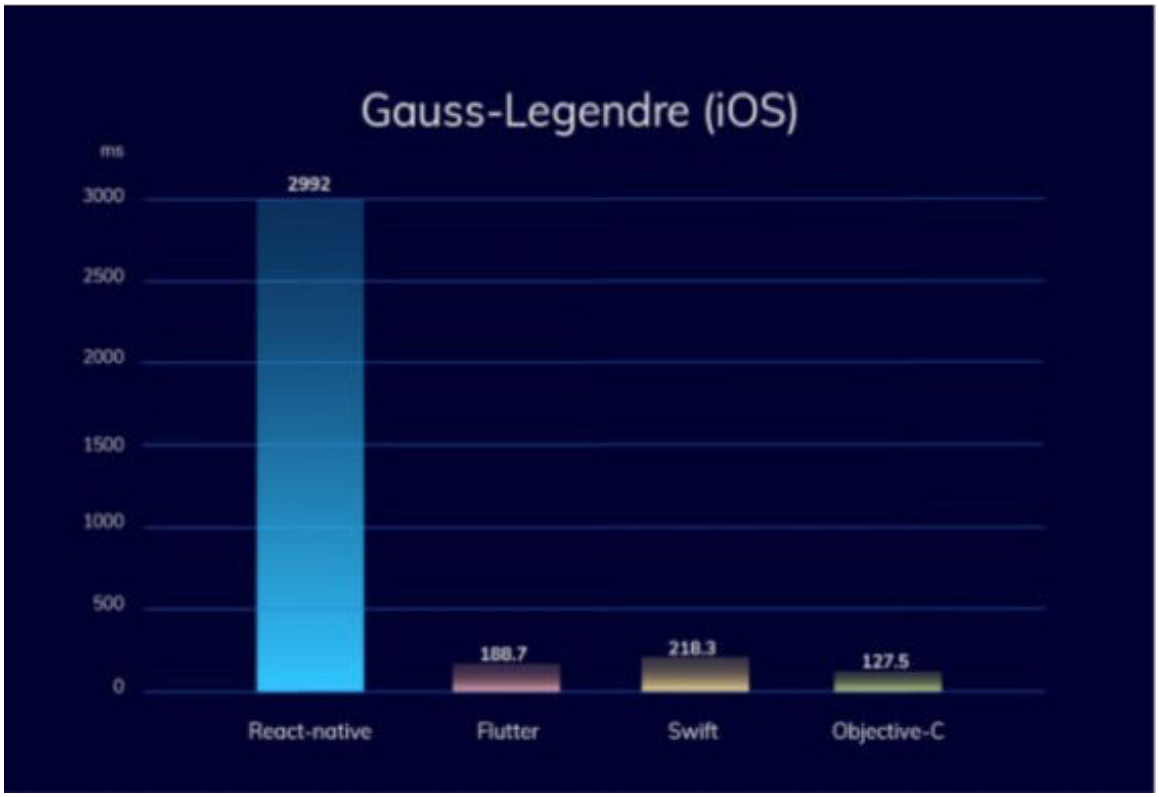
Siin ei suuda Xamarin platvormipõhiste rakendustega võrrelda. Androidi rakendusega võrreldes on mälukasutuse suurus kahekümne kordne.

4.4.2 React Native, Flutteri ning platvormipõhiste rakenduste võrdlus

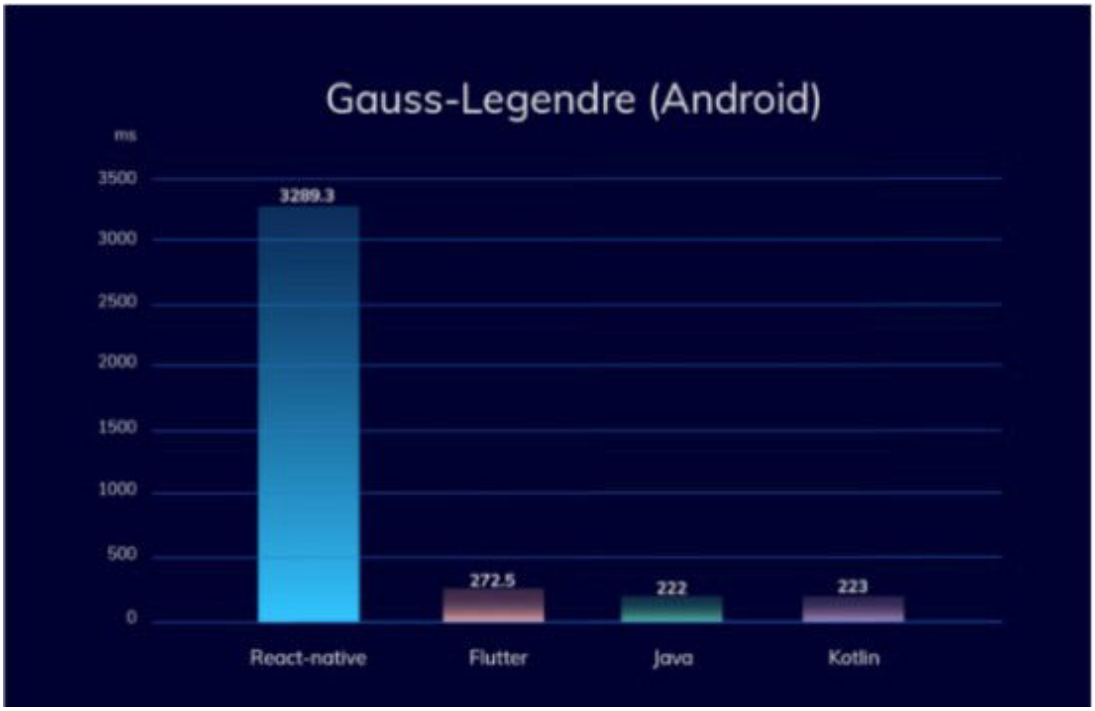
Järgmiseks otsustas autor võrrelda React Native, Flutteri ning platvormipõhiste rakenduste jõudlust. Järgevalt välja toodud testide põhjal võib öelda, et React Native kasutab oluliselt rohkem mälu ning seadme protsessori töövõimet. Parima tulemuse andis platvormipõhine lahendus, kuid Flutter ei jäänud palju alla.

1. Tehnoloogia arvutusjõudlus mobiiliseadmes

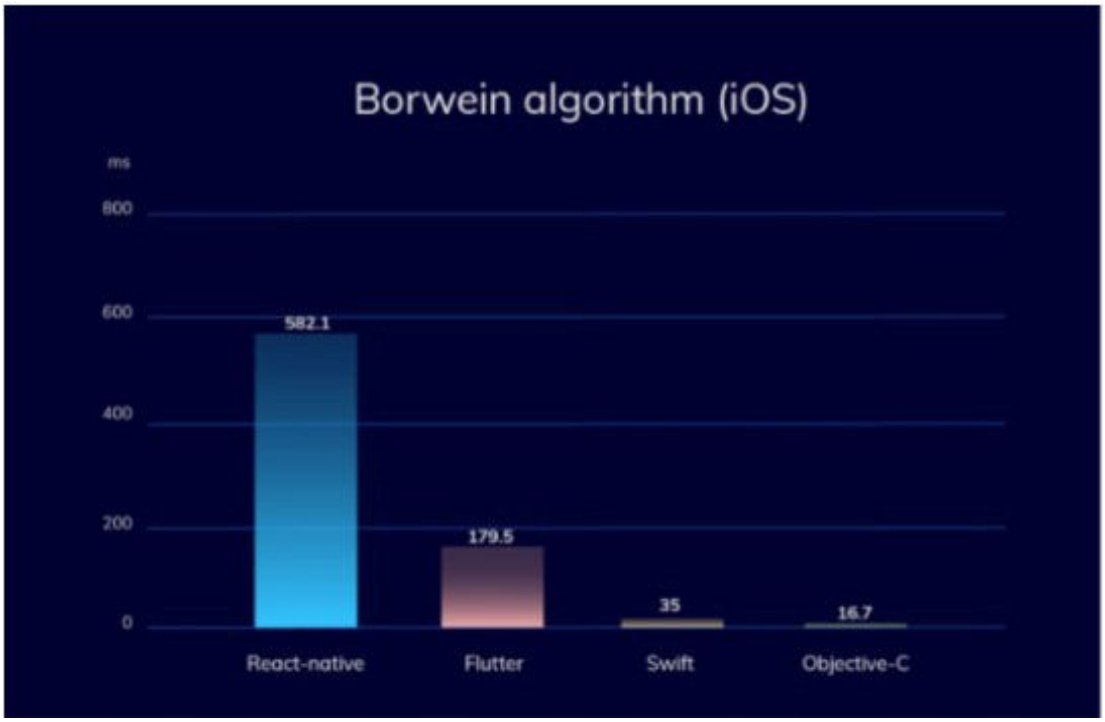
Alustuseks võrreldi React Native, Flutter ja platvormipõhiste rakenduste arvutuskiiruseid. Loodavas rakenduses ei ole küll vaja lahendada keerulisi matemaatilisi võrrandeid, kuid antud testid annavad hea ülevaate tehnoloogia võimekusest seadmes.



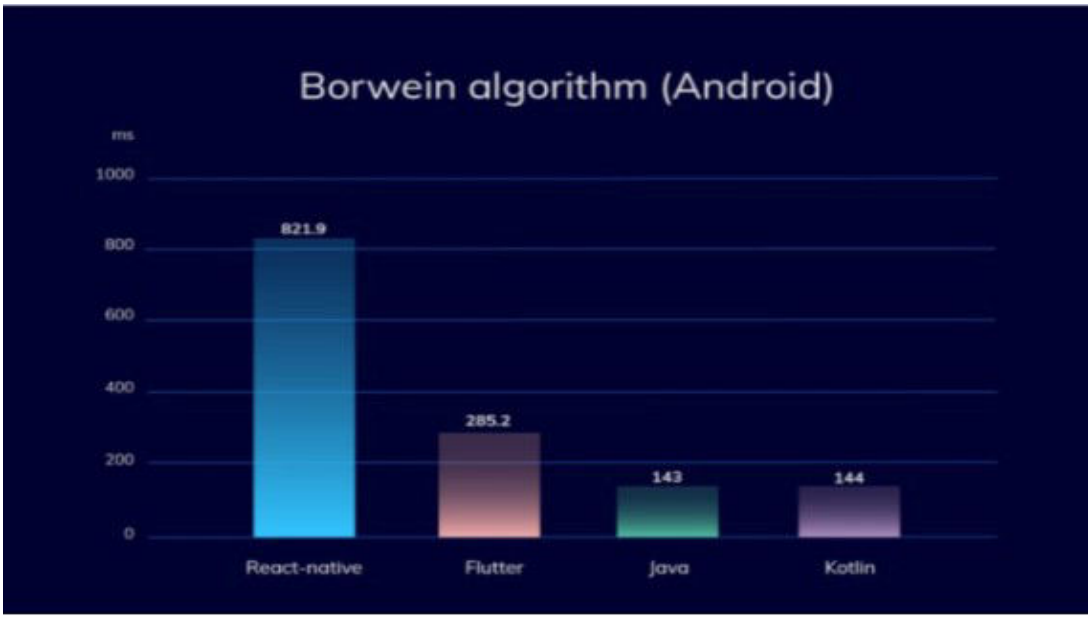
Joonis 5. React Native, Flutteri ning iOS'i programmeerimiskeelte ajakulu Gauss-Legendre algoritmiga [12]



Joonis 6. React Native, Flutteri ning Androidi programmeerimiskeelte ajakulu Gauss-Legendre algoritmiga [12]



Joonis 7. React Native, Flutteri ning iOS'i programmeerimiskeelte ajakulu Borwein'i algoritmiga [12]

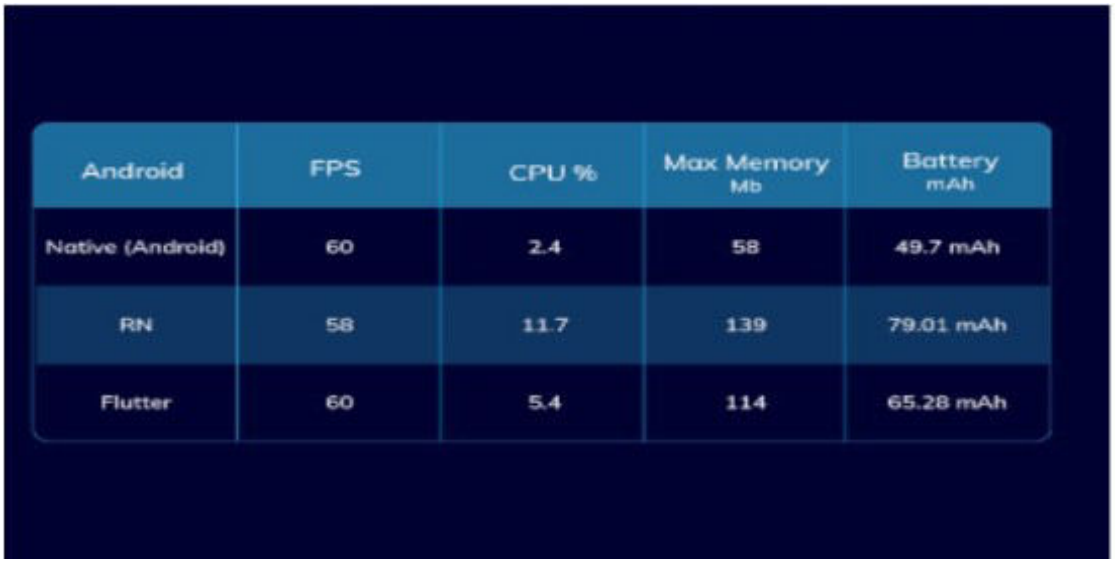


Joonis 8. React Native, Flutteri ning Androidi programmeerimiskeelte ajakulu Borwein'i algoritmiga [12]

Testi tulemustest on näha, et React Native ei suuda operatsioonipõhiste rakendustega sammu pidada. Samas Flutter üllatas kohati isegi parema arvutusjõudlusega kui Swift keeles kirjutatud rakendus.

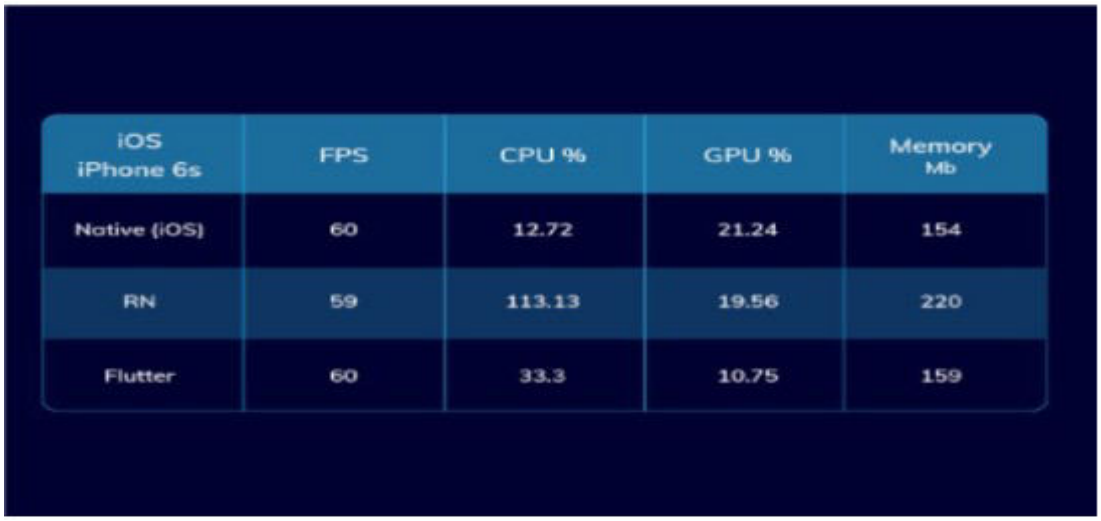
2. Listivaadete ning meediate laadimiskiirus ning ressursi kasutus

Alati ei ole vajalik rakenduse loomisel sooritada keerulisi matemaatilisi tehteid, vaid põhiline on silmale nähtavate vaadete ning meediate laadimine. Järgnevas testis võrreldi Flutteri, React Native ning platvormipõhiste rakenduste vaadete ning meediate laadimiskiirust.



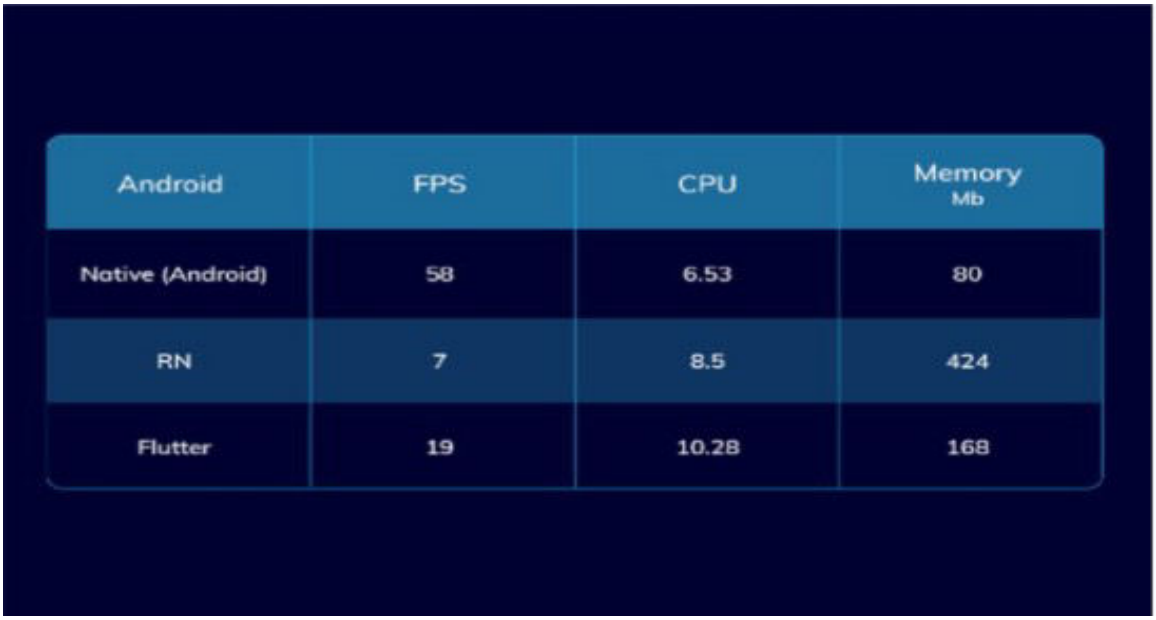
Android	FPS	CPU %	Max Memory Mb	Battery mAh
Native (Android)	60	2.4	58	49.7 mAh
RN	58	11.7	139	79.01 mAh
Flutter	60	5.4	114	65.28 mAh

Joonis 9. Listivaate laadimise ressurssikasutuse võrdlus React Native, Flutteri ja Androidi vahel [13]



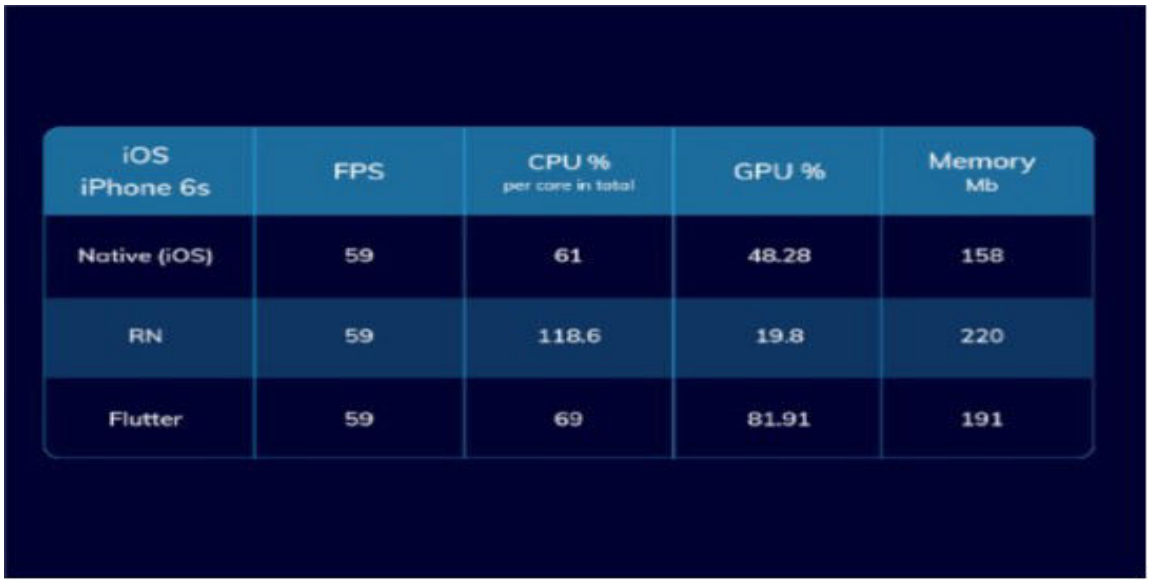
iOS iPhone 6s	FPS	CPU %	GPU %	Memory Mb
Native (iOS)	60	12.72	21.24	154
RN	59	113.13	19.56	220
Flutter	60	33.3	10.75	159

Joonis 10. Listivaate laadimise ressurssikasutuse võrdlus React Native, Flutteri ja iOS'i rakenduse vahel [13]



Android	FPS	CPU	Memory Mb
Native (Android)	58	6.53	80
RN	7	8.5	424
Flutter	19	10.28	168

Joonis 11. Animatsioonide laadimise ressurssikasutuse võrdlus React Native, Flutteri ja Androidi rakenduse vahel [13]



iOS iPhone 6s	FPS	CPU % per core in total	GPU %	Memory Mb
Native (iOS)	59	61	48.28	158
RN	59	118.6	19.8	220
Flutter	59	69	81.91	191

Joonis 12. Animatsioonide laadimise ressurssikasutuse võrdlus React Native, Flutteri ja iOS'i rakenduse vahel [13]

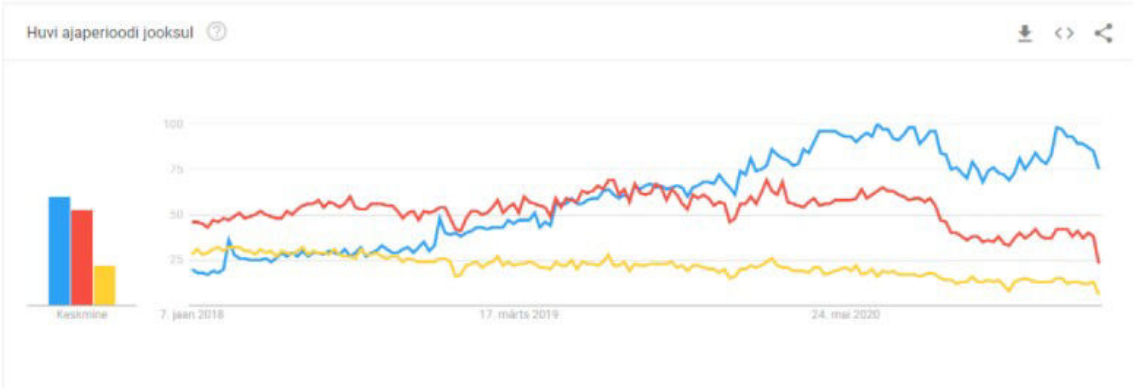
Tulemustest on näha kõige paremat võimekust näitasid platvormipõhised tehnoloogiad. Neile järgnes Flutter ja peale seda React native. Tuleb mainida, et joonisel 11 väga halb jõudlus mõlema hübriidmobiilirakenduse tehnoloogia poolt on ühe kindla animatsiooni tõttu.

4.5 Hübriidmobiilirakenduste tehnoloogiate populaarsuse võrdlus

Eelmises punktis hübriidmobiilirakenduste tehnoloogiate jõudluse võrdluses jäi autorile silma Flutter, kuid vahe teistega ei olnud nii suur, et kohe otsust teha. Arendusprotsessis tekib tihti erinevaid küsimusi ja olukordi, kus on vajalik interneti abi. Sellest tulenevalt peaks olema tehnoloogia kohta veebis piisavalt materjali saadavalt ning kasutajate hulk piisavalt suur. Tihti võib juhtuda nii, et loodaval rakendusel on spetsiifilised eripärad ja vajadus raamistikku ühendada erinevate liidestega. Võib juhtuda, et ametlikust dokumentatsioonist kõiki vastuseid ei leia.

4.5.1 Google Trends

Esiteks otsustas autor vaadelda Google Trends abil ülal mainitud kolme raamistiku otsingupopulaarsust.



Joonis 13. Flutteri, Xamarini ja React Native Google otsinguhuvi. Sinise värviga on kirjeldatud Flutterit, punasega React Native ning kollasega Xamarini [14]

Autor valmis ajavahemikuks aasta 2018 alguse ning graafiku lõpuks on valitud aasta 2021 aprill. Jooniselt on näha, et inimesed olid enim huvi tundnud React native vastu, millele järgnes Xamarin. Tuleb välja, et huvi Flutteri vastu hakkas suurenema aasta 2019 alguses ning aastaks 2020 oli Flutter juba enim otsitud raamistik mainitud kolme võrdluses. Sellest võib oletada, et kasutades Flutterit, leiab suure tõenäosusega oma küsimusele vastuse veebist [14].

4.5.2 Github repositooriumite arv

Eelmises peatükis tuli Google Trendsi kasutades välja, et enim otsitud hübriidmobiilirakenduse tehnoloogia valitud kolmest on Flutter. Lisaks otustas autor, et analüüsib ka GitHub'i repositooriumite arvu, milles on kasutatud mainitud raamistikke. See on oluline, kuna loodav rakendus peab hakkama mängima järjejutte audiot ning tihti on avatud lähtekoodiga olemas valmis lahendused, mis kiirendavad veel arendusprotsessi. Pleier peaks suutma reageerida ka sisse tulevatele telefonikõndele ja teadetele. Kõige selle algusest kirjutamine oleks ebamõistlik ajakasutus. Autor kasutas Githubi otsingumootorit, et leida repositooriumite arv, milles on kasutatud igat töös välja valitud tehnoloogiat. Esimeseks otsinguks oli Xamarin, mille kohta leidis töö autor 42541 repositoorumit. React Native ning Flutter olid mitmekordselt esimesest eest, vastavalt 239161 React Native kohta ning 237037 repositooriumit Flutteri kohta. Siit võib julgelt väita, et hetkel on kõige vähem populaarne Xamarin ning teised kaks on võrdsel tasemel [15].

4.6 Kokkuvõte ja autori tehnoloogiavalik rakenduse loomiseks

Lõputöö autor analüüsis kolme ülal valitud hübriidmobiilirakenduse loomise tehnoloogiaid. Kuna loodav rakendus peab kuvama nimekirja kuulatavatest järjejuttudest ning kuvama mitmeid logosid, siis valitav tehnoloogia peab olema võimeline tegema seda kiirelt ning minimaalse ressursikasutusega. Autor otsustas alustuseks välistada Windowsi Xamarini. Viimane on küll küllaltki kiire võrreldes paltvormipõhiste mobiilirakendustega, kuid puuduseid oli rohkem kui teistel. Esiteks on raamistiku populaarsus veebiotsingute mõistes madal ning sama võib öelda ka GitHubi repositooriumite arvu kohta. Kõige suurem argument, mis muutis töö autori Xamarini osas ettevaatlikuks oli Microsofti teade, et 2021 novembris loobutakse Xamarin.Forms'i toetamisest [16]. Julgelt võib väita, et loodavale rakendusele tuleb peale esimest versiooni veel uuendusi ning muudatusi teha ning vananeva tehnoloogia õppimisele kulutatav aeg ei ole mõistlik. Autoril jäi nüüd valida React Native ning Flutteri vahel. Populaarsuse võrdluses on kaks viimast peaaegu võrdset, Google otsingute osas võidab küll Flutter. Tehnoloogiate jõudluse osas ületas Flutter React Native küllaltki suure

edumaaga. Autor otsustas valida valmiva rakenduse arendustehnoloogiaks Flutteri. Seda põhjusel, et React Native eelis on tihti arendajatel, kellel on kogemus React raamistiku kasutamisega, kuid autoril see puudub. Lisaks mängis rolli ka asjaolu, et praegune disain oli loodud küllaltki lihtne, millega saaks mõlemad raamistikud hakkama, kuid tulevikus võiks kasutada ära Flutteri vidinate loomise võimalust ning teha mitmekesisem disain rakendusele.

5 Valminud rakenduse arenduskäik

Selles peatükis toob autor välja lõputöös loodud rakenduse arenduskäigu. Põgus ülevaade tuuakse rakenduse sisesest arhitektuurist ning andmemudelitest. Pikemalt kirjeldatakse mobiilirakenduse vaateid ja nendes navigeerimist. Kõige lõpuks toob autor välja loodud lahenduse tulevikuvaated ning mida saaks paremaks teha.

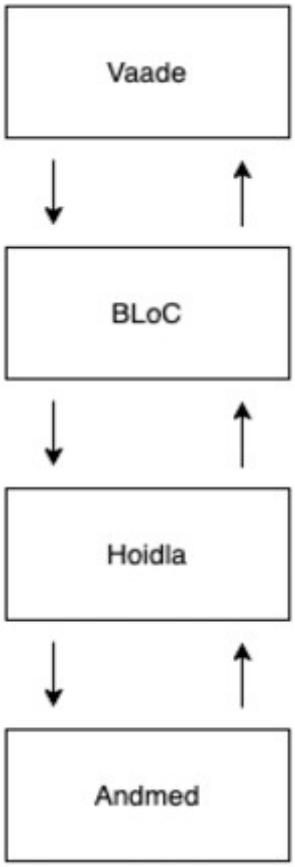
5.1 Andmemudelid

Rakenduses on kasutusel kolm põhist andmemudelit: episood, seriaal, audio. Episood kirjeldab ühte järjejuttu episoodi, see mudel hoiab infot loo id, pealkirja, pildi ning linki välisele striimile, kust mängitakse heli. On olemas üksikepisoode, millel pole sarja, kuid ülejäänute jaoks on mudel nimega Seriaal. Antud tükk hoiab infot sarja id, pealkirja ning pildi kohta. Kuna rakenduse pleier tuleb välisest teegist, siis viimane kasutusel olev mudel on Audio. Selleks, et pleier suudaks lugu mängida, tuleb episood ümber teisendada Audio mudeliks, mis hoiab striimi linki, metaandmeid ning muid pleieriga seonduvaid andmeid.

5.2 Andmete pärimine serverist

Varasemalt on välja toodud, et rakenduse andmed pärinevad serverist, kust tuleb need mingil viisil kätte saada. Lisaks tuli välja mõelda arhitektuur, kuidas saadud andmeid jagada rakenduse erinevate vaadete vahel. Google arendajate, kes on Flutteri raamistiku taga, soovitavad kasutada BLoC tarkvaraahitektuuri.Mainitud arhitekuuri eesmärk on olla andmete allika ning vaadete vahel kasutades andmeneelu ja andmevoogusid, mida omakorda kontrollib sisend-väljundkontroller. Kõige suurem eelis mainitud meetodil on rakenduse oleku kontrollimine. See võimaldab hoida lihtsasti ülevaadet, millal andmed laadivad sisse ning sellel ajal kuvada kasutajale laadimislehte. Kui olek muutub ning andmed on laetud, saab edasi suunata soovitud lehele. Lisaks on iga kiht eraldi väikse tükina ning see võimaldab vea korral kergesti testida rakenduse mingit osa vea suhtes [17].

Internetis on valmis paketid antud arhitektuurimustri kasutamiseks, kuid töö autor otsustas luua selle nullist ise.
Alustuseks tuli luua kõige madalam kiht ehk osa, mis suhtleks otse serveri REST api päringutega. Selleks kasutas autor Flutteri enda HTTP teeki. Selles kõige madalamas kihis tehakse HTTP päring serverisse ning eduka päringu korral edastatakse vastus tarkvarahoidlasse, kus eraldatakse vastusest JSON kujul andmed, mis omakorda teisaldatakse ümber ülal mainitud objektideks. Peale seda saadab hoidla valmis andmed edasi järgmisesse kihti, mis suhtleb vaadetega. Viimasena mainitud kiht hoiab meeles rakenduse oleku, mida on kolm tükki: laadimine, ebaõnnestunud laadimine, edukas laadimine. Eelnevalt mainitult on laadimise ajal kasutajal ees vaade “laadimine”, ebaõnnestumise korral kuvatakse veateade ning soovitus rakendus uuesti käivitada ning protsessi õnnestumise korral suunatakse kasutaja edasi järjekorras olevataesse vaadetesse.



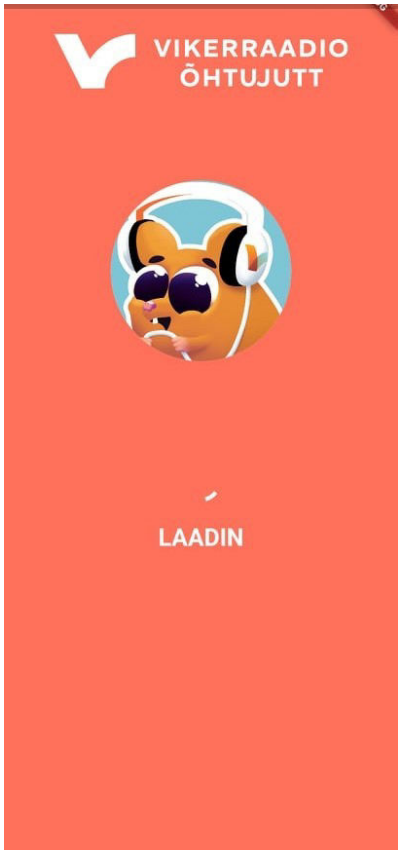
Joonis 14. Joonisel on kujutatud *BloC* tarkvaraahitekuurimustrit.

5.3 Vaated

Flutteri raamistiku eripäraks on, et iga silmale nähtavat objekti nimetatakse jubinaks. Viimane kehtib nii ühe nupu, kui ka terve vaate kohta. Lisaks on iga loodav jubin defineeritud kas olekuga või ilma. Ilma olekute jubinad on muutumatud ehk on võimalik kaasa anda andmeid, mida kuvada. Samas hiljem peale laadimist andmeid juurde lisada ei saa. Vastupidiselt olekuga jubinates saab andmeid dünaamiliselt juurde laadida ning muudatusi teha rakenduse töötamise ajal.

5.3.1 Laadimisvaade

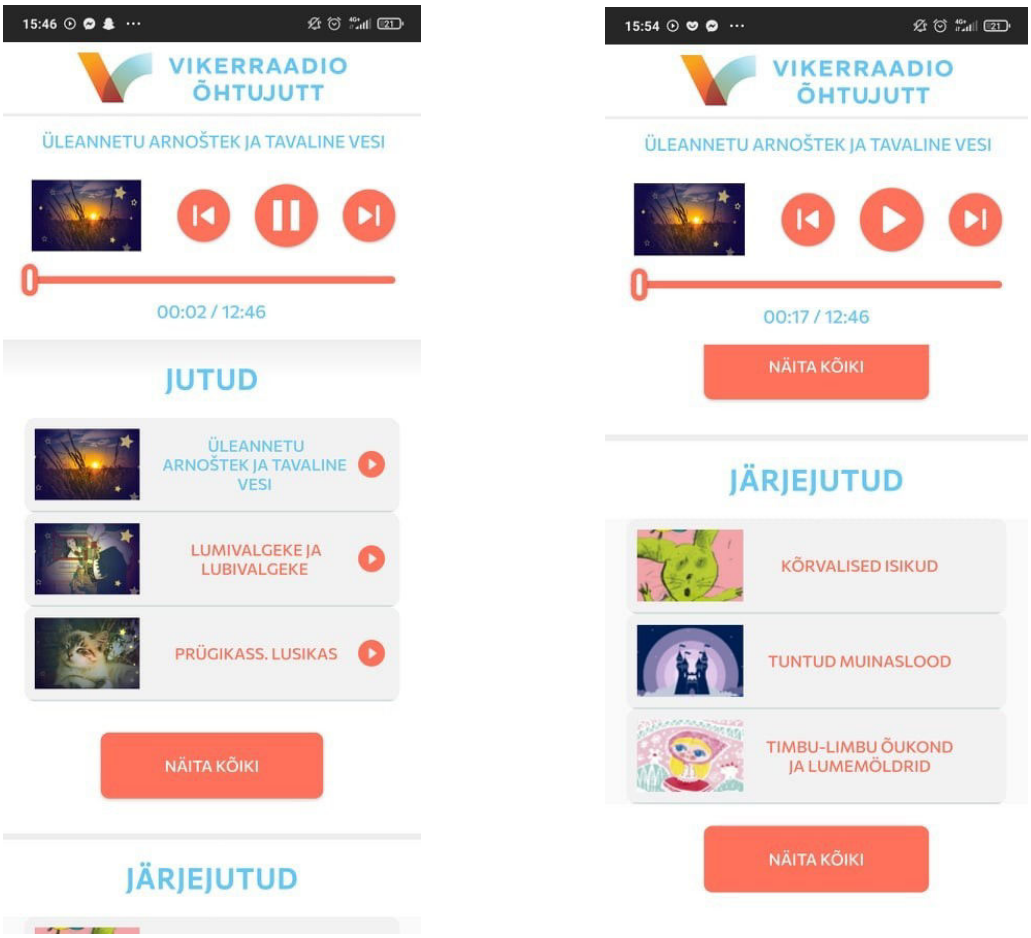
Tegemist on vaatega, mis kuvatakse kasutajale rakendus avades. Selle vaate taustal toimub peatükis 6.2 mainitud andmete pärimine serverist ning teisendus pleierile ja vaadetele sobilikuks. Kuna Flutter saab andmete laadimisega nii kiirelt hakkama, siis kasutaja seda vaadet tihti ei näegi.



Joonis 15. Joonisel on rakenduse laadimisvaade.

5.3.2 Avaleht

Kui andmed on sisse loetud ning teisendatud pleierile ja vaadetele sobivasse formaati, siis tagastab laadimisvidin andmed avalehe vidinale ning laetakse viimane. Rakenduse loomisel on tähtis, et piltide ja audio laadimine oleks piisavalt kiire, et ei tekiks avalehe kerimisel kiilumist ega laadimata pilte. Flutter saab sellega suurepäraselt hakkama ning kõik liigub sujuvalt. Avalehe avamisel hakkab mängima “Jutud” lahtrist viimasena lisatud kirje. Avalehel on kaks lahtrit, esimeses lahtis on viimati lisatud Vikerraadio õhtujutud lastele. Vajutades nuppu “NÄITA KÕIKI”, kuvatakse uus vaade, kus on nimekiri kolmekümnest viimati lisatud õhtujutust. Alla kerides ilmub alumine lahter, kus on Vikerraadio järjejuttude sarjadele. Avalehel kuvatakse kolm sarja, kuid vajutades nupule “NÄITA KÕIKI” suunatakse kasutaja sarja vaatesse, kus on nimekiri kõigist olemasolevatest järjejuttude sarjadest.



Joonis 16. Rakenduse esilehe vaade

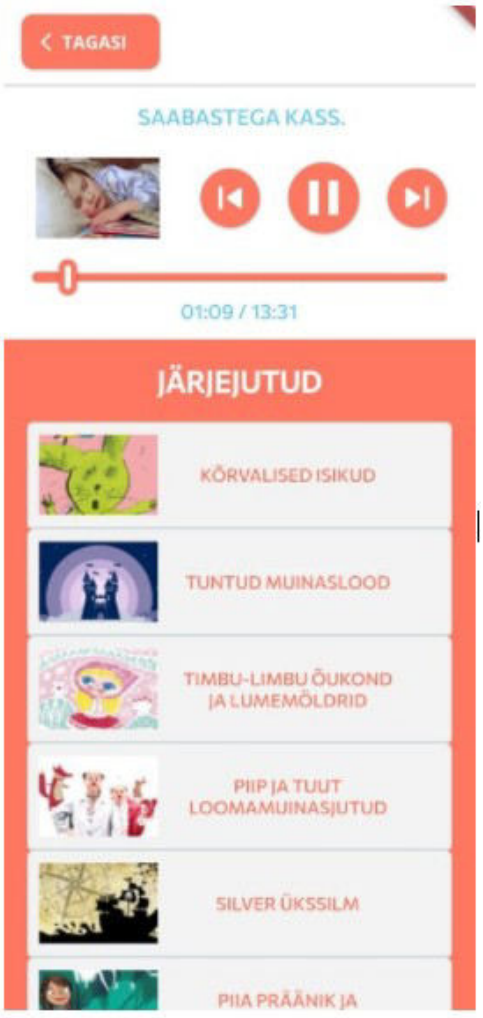
5.3.3 Juttude nimekirja vaade

Antud vaatesse jõuab kasutaja, kui vajutab pealehel olevale nupule “NÄITA KÕIKI” lahtri “JUTUD” juures. Kasutajale kuvatakse hetkel kolmkümmend viimast Vikerraadio õhtujuttu. Mainitud numbrit saab muuta ERR’i infosüsteemis vastavalt vajadusele. Sarnaselt esilehe vaatele on ka siin tänu Flutterile laadimine kiire ning lehe kerimine sujuv. Tagasi esilehele saab kas vajutades nupule “TAGASI” või kasutades seadme libista funktsionaalsust.

Joonis 17. Rakenduse juttude vaade

5.3.4 Järjejuttude vaade

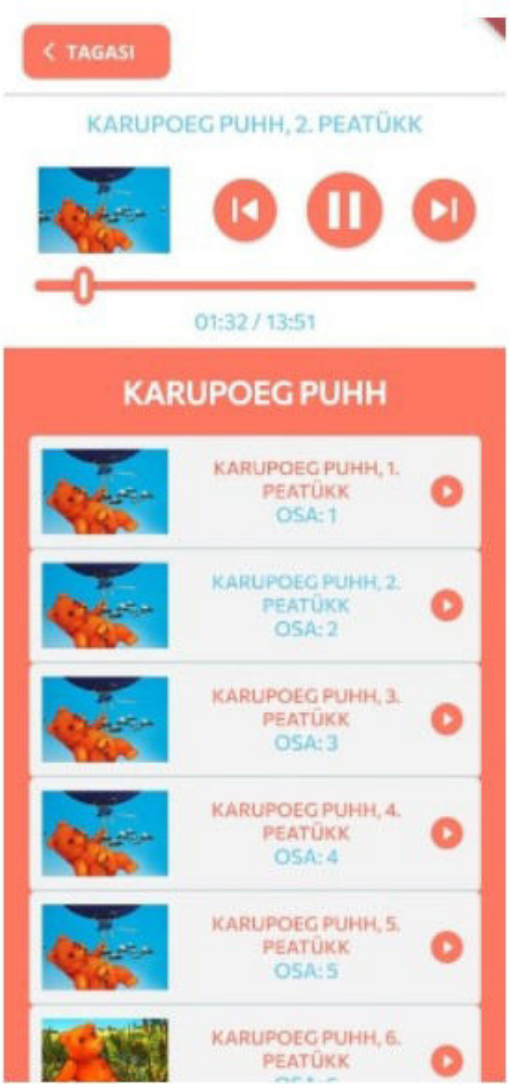
Kui kasutaja on järjejuttude sarjavaates valinud endale meelepärase sarja välja ning vajutanud sellele, siis toimub päring serverisse ja kasutajale laetakse valitud sarja episoodide nimikiri. Osad on märgistatud järjekorranumbritega, mida saab seadistada ERR'i infosüsteemist valides järjekorra kas käsitsi või automaatselt lisamisaja järgi vastavalt kas vanemast uuemaks või uuemast vanemaks. Vaatest saab väljuda vajutades kas nupule "TAGASI" või kasutades seadme libistamisfunktsionaalsust. Rakenduse navigeerimine on ehitatud magasini põhimõttel. Tagasi liikumine viib kasutaja eelmisesse vaatesse, mitte täiesti algusesse.



Joonis 18. Rakenduse järjejuttude vaade

5.3.5 Järjejuttude episoodi vaade

Kui kasutaja on järjejuttude sarjavaates valinud endale meelepärase sarja välja ning vajutanud sellele, siis toimub päring serverisse ja kasutajale laetakse valitud sarja episoodide nimikiri. Osad on märgistatud järjekorranumbritega, mida saab seadistada ERR'i infosüsteemist valides järjekorra kas käsitsi või automaatselt lisamisaja järgi vastavalt kas vanemast uuemaks või uuemast vanemaks. Vaatest saab väljuda vajutades kas nupule "TAGASI" või kasutades seadme libistamisfunktsionaalsust. Rakenduse navigeerimine on ehitatud magasini põhimõttel. Tagasi liikumine viib kasutaja eelmisesse vaatesse, mitte täiesti algusesse.



Joonis 19. Rakenduse järjejuttude vaade

5.3.6 Rakenduse pleier

Rakenduses kasutusolev pleier on ehitatud kasutades välist teeki assets_audio_player versiooni 2.0.13. Mainitud teek toetab Androidi, iOSi, veebi ning MacOS rakendusi. Antud pleieri funktsionaalsus on väga lai, toetatud on üksiku audio mängimine, terve pleilistiloendi mängimine. Lisaks testis autor arendusfaasis ka, kas pleier reageerib vastavalt, kui seadmesse tuleb sisse telefonikõne või muud teated. Teek oskab õiges kohas pausile jääda ning hiljem samast kohast mängimist jätkata. Suur tugevus teegil on ka tugi kasutaja tegevuste jälgimise kohta. Salvestatakse millal mingi lugu pausile pandi, kui kaua kuulati ning millal lugu vahetati. See on tähtis, kuna organisatsiooni teistele rakendusele on vastav statistika juba loodud ning saab lihtsasti ka loodav rakendus ühendada statistika programmiga.

Joonis 20. Rakenduse pleier Androidi suletud ekraani vaates

5.4 Rakenduse tulevikuplaanid ning uuendused

Lõputöö valmise ajaks on rakendus olnud Google Play poes ning Apple AppStores kuu aega. Juba töö kirjutamise ajal on autor teinud kolm versiooniuuendust. Esimene muudatus tekkis järjejuttude episoodide vaatega, kuhu autor lisas osade numbrid ning tegi selle ERR'i infosüsteemist kontrollitavaks. Hetkel töötab autor protsessi kallal, millega saaks valminud rakenduse ühendada organisatsiooni statistikaliidesega. Veel on plaanis tegeleda disainimuudatusega. Esimese versiooni disain tuli ajapiirangu tõttu liialt lihtne. Autoril on arendusmeeskonna disaineri poolt loodud uued tükid, millega saaks kasutada Flutteri lõuendile joonistamise võimalust. Mainitu käib eelkõige

tekstiväljade “Jutud” ja “Järjejutud” kohta. Lisaks on palju tegemist veel listivaadete lahtrite kaunistamisega ning varjude lisamisega. Töös välja toodud esimeses versioonis ei ole kasutatud kogu Flutteri kasutajaliidese loomise potensiaali.

6 Kokkuvõte

Bakalaurusetöö teema sai valitud autori töökohas tekkinud vajadusest luua rakendus järjejuttude kuulamiseks. Lähtetingimusteks oli luua disanilt puhas, lihtsasti kasutatav ja ühte funktsiooni hästi täitev rakendus. Lähtetingimustena oli autoril kasutada tema enda loodud REST-ahitektuuriga päringud, mis tagastab andmebaasist JSON vormis infot järjejuttude kohta. Samuti olid olemas audio kujul järjejutud, mida rakendus mängis.

Kuna autori soov oli luua väärtust arendusosakonda esimese hübriidmobiilirakenduse näol, siis otustas ta valida kolm tehnoloogiat, millest ta varem kuulnud oli ning neid analüüsida. Alustuseks luges ta iga raamistiku kohta raamatu läbi. Seejärel kasutas autor Google Trends keskkonda, et võrrelda tehnoloogiate otsingupopulaarust. Samuti võrreldi Githubi repositooriumite arvu. Peale seda analüüsis autor kõigi kolme välja valitud tehnoloogia võimekust erinevates olukordades. Võrreldi seadme mälu - ja protsessorikasutust, animatsioonide kuvamise kiirust ning ka kogu rakenduse seadmes hoiustamisele kuluvat mälumahtu. Analüüsi tulemusel otsustas autor valida välja Google Flutteri raamistikku rakenduse loomiseks.

Töö teises pooles kirjeldati rakenduse arenduskäiku ja kirjeldati valminud rakendust. Esiteks tõi välja autor rakenduse sisese arhitektuuri, kuidas liiguvad andmed serverist sisse, kuidas neid töödeltakse ning seejärel jagatakse vaadetele edasi. Edasi toodi välja iga loodud vaade ning kirjeldus vastava vaate kohta.

Viimaks tõi autor välja bakalaureusetöö tulemusel valminud lahenduse tulevikuplaanid ning praegused puudused. Kuna töös kirjeldatud rakendus on esimene versioon, siis juba kirjutamise ajal tegi autor palju parandusi ning uuendusi. Välja tõi autor puudused disainis. Uuendustest tähtsamaks, mis töö ajal valmis on kasutaja tegevuste kohta statistika tegemisega Gemius teeki kasutades.

Kasutatud kirjandus

[1] „Mobile marketing statistics compilation 2021“, [Võrgumaterjal]. Available: https://www.smartinsights.com/mobile-marketing/mobile-marketing-analytics/mobile-marketing-statistics/ [Kasutatud 28.04.2021]

[2] „Smartphone Market Share“, [Võrgumaterjal] . Available: https://www.idc.com/promo/smartphone-market-share/os [Kasutatud 28.04.2021]

[3] Mahesh Panhale, „ Beginning Hybird Mobile Application Development“, 2015.

[4] Daniel Hindrikes, Johan Karlsson, „Xamarin.Forms Projects: Build seven real-world cross-platform mobile apps with C# and Xamarin.Forms“, 2018.

[5] „The Good and The Bad of Xamarin Mobile Development“, [Võrgumaterjal]. Available: https://www.altexsoft.com/blog/mobile/pros-and-cons-of-xamarin-vs-native/ [Kasutatud 28.04.2020]

[6] Bonnie Eisenman, „Learning React Native: Building Native Mobile Apps with JavaScript“, 2015.

[7] „Pros and cons of React Native for cross-platform app development , [Võrgumaterjal]. Available: https://www.mindk.com/blog/react-native-pros-and-cons/“ [Kasutatud 28.04.2020]

[8] Rap Payne „Beginning App Development with Flutter: Create Cross-Platform Mobile Apps“, 2019.

[9] „Flutter Pros & Cons for Mobile App Owners“, [Võrgumaterjal]. Available: https://www.thedroidsonroids.com/blog/flutter-in-mobile-app-development-pros-and-cons-for-app-owners [Kasutatud 27.04.2020]

[10] „Pros & Cons of Flutter Mobile Development, [Võrgumaterjal]. Available: https://www.futuremind.com/blog/pros-cons-flutter-mobile-development“ [Kasutatud 28.04.2020]

[11] „Performance Comparison: Xamarin.Forms, Xamarin.iOS, Xamarin.Android vs Android and iOS Native Applications“ , [Võrgumaterjal]. Available: https://www.altexsoft.com/blog/engineering/performance-comparison-xamarin-forms- xamarin-ios-xamarin-android-vs-android-and-ios-native-applications/ [Kasutatud 27.04.2020]

[12] „Flutter vs Native vs React-Native: Examining performance“, [Võrgumaterjal]. Available: https://medium.com/swlh/flutter-vs-native-vs-react-native-examining-performance-31338f081980 [Kasutatud 28.04.2020]

[13] „Flutter vs React Native vs Native: Deep Performance Comparison“ [Võrgumaterjal]. Available: https://medium.com/swlh/flutter-vs-react-native-vs-native-deep- performance-comparison-990b90c11433 [Kasutatud 28.04.2020]

[14] Google Trends, [Võrgumaterjal]. Available: https://trends.google.com/ [Kasutatud 28.04.2020]

[15] GitHub [Võrgumaterjal]. Available: https://github.com/ [Kasutatud 28.04.2020]

[16] „Is Xamarin dead?“ [Võrgumaterjal]. Available: https://foresightmobile.com/blog/2020/09/15/isxamarindead [Kasutatud 28.04.2020]

[17] „Architect your Flutter project using BLOC pattern“, [Võrgumaterjal]. Available: https://medium.com/codechai/architecting-your-flutter-project-bd04e144a8f1 [Kasutatud 28.04.2020]

Lisa 1– Lihtlitsents lõputöö reprodutseerimiseks ja lõputöö üldsusele kättesaadavaks tegemiseks[1]

Mina, Kristo Erte

1 Annan Tallinna Tehnikaülikoolile tasuta loa (lihtlitsentsi) enda loodud teose „ERR VIKERRAADIO JÄRJEJUTTUDE ÜHTSE RAKENDUSE ARENDAMINE IOSI JA ANDROIDI PLATVORMIDELE”, mille juhendajad on Madis Lauri ja Jaanus Pöial.

1.1 reprodutseerimiseks lõputöö säilitamise ja elektroonse avaldamise eesmärgil, sh Tallinna Tehnikaülikooli raamatukogu digikogusse lisamise eesmärgil kuni autoriõiguse kehtivuse tähtaja lõppemiseni;

1.2 üldsusele kättesaadavaks tegemiseks Tallinna Tehnikaülikooli veebikeskkonna kaudu, sealhulgas Tallinna Tehnikaülikooli raamatukogu digikogu kaudu kuni autoriõiguse kehtivuse tähtaja lõppemiseni.

2 Olen teadlik, et käesoleva lihtlitsentsi punktis 1 nimetatud õigused jäävad alles ka autorile.

3 Kinnitan, et lihtlitsentsi andmisega ei rikuta teiste isikute intellektuaalomandi ega isikuandmete kaitse seadusest ning muudest õigusaktidest tulenevaid õigusi.

29.04.2021

1 Lihtlitsents ei kehti juurdepääsupiirangu kehtivuse ajal vastavalt üliõpilase taotlusele lõputööle juurdepääsupiirangu kehtestamiseks, mis on allkirjastatud teaduskonna dekaani poolt, välja arvatud ülikooli õigus lõputööd reprodutseerida üksnes säilitamise eesmärgil. Kui lõputöö on loonud kaks või enam isikut oma ühise loomingulise tegevusega ning lõputöö kaas- või ühisautor(id) ei ole andnud lõputööd kaitsvale üliõpilasele kindlaksmääratud tähtajaks nõusolekut lõputöö reprodutseerimiseks ja avalikustamiseks vastavalt lihtlitsentsi punktidele 1.1. ja 1.2, siis lihtlitsents nimetatud tähtaja jooksul ei kehti.