TALLINNA TEHNIKAÜLIKOOL
Infotehnoloogia teaduskond

Tanel Tinits 192850IADB

Progressiivne veebirakendus vabade praamipiletite leidmiseks

Bakalaureusetöö

Juhendaja: Kristiina Hakk
PhD

Tallinn 2023

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: Tanel Tinits

05.01.2023

Annotatsioon

Hiiumaa ja Saaremaa praamipiletid on tipptundideks tihti välja müüdud juba mitmeid nädalaid enne reisi toimumist. Reisijad, kellel ei ole võimalik reisi pikalt ette planeerida, ei saa seetõttu eelmüügist piletit osta. Elavas järjekorras pileti ostmine võib tähendada mitme tunni pikkust järjekorras ootamist.

Bakalaureusetöö eesmärk on luua progressiivne veebirakendus, mis võimaldab kasutajal tellida teavitusi, mis saadetakse SMS ja tõuketeavituste teel, kui väljamüüdud praamipiletid uuesti müügile peaksid tulema.

Töö käigus vaadeldakse alternatiivseid võimalusi progressiivsetele veebirakendustele, luuakse nõuded eesmärgiks seatud rakendusele, kirjeldatakse rakenduse loomiseks tehtud tehnoloogiate valikuid ning vaadeldakse erinevaid teavituste saatmise võimalusi.

Praktilises osas läbiviidav arendus toimub iteratiivsel meetodil, kus esimese arendustsükliga luuakse kontseptsioonitõestus ning iga järgneva arendustsükliga lisatakse funktsionaalsust. Bakalaureusetöö lõpuks valmib kasutatav lahendus.

Lõputöö on kirjutatud eesti keeles ning sisaldab teksti 35 leheküljel, seitse peatükki, 16 joonist, viis tabelit.

Abstract

Progressive Web Application for Finding Available Ferry Tickets

On rush hour are Hiiumaa and Saaremaa ferry tickets mostly sold out weeks before the trip. Often can't passengers plan so much ahead to buy the tickets. Waiting time in a live queue might reach to many hours.

Aim of this Bachelor's these is to create a progressive web application, that would make possible for passengers to order a notification for sold out tickets when they should be available again.

In the process of the thesis there will be examined alternate solutions for progressive web applications, how requirements were set for the application, descriptions of how choices were made for technology choices and descriptions for different kind of notification methods.

Practical part describes how the development was done in iterative cycles, where first iteration was the proof of concept and every next iterations adds additional functionality to the initial result. By the end of the Bachelor's theses there will be a fully functional application.

The thesis is in Estonian and contains 35 pages of text, seven chapters, 16 figures, five tables.

Lühendite ja mõistete sõnastik

AD	*Active Directory*, kasutajate kataloogiteenus
Agiilne arendus	*Agile development*, arendusmeetod, kus kasutatakse lühikesi korduvaid arendustsükleid, kus iga kordusega on võimalik vajadusel kohandada töö skoopi
API	*Application Programming Interface*, rakendusliides
CI/CD	*Continuous Integration/Continuous Delivery or Continuous Deployment*, pidev integratsioon/pidev tarnimine
Cron	UNIX-i töövahend ettemääratud tegumite perioodiliseks täitmiseks ettemääratud aegadel
CSS	*Cascading Style Sheet*, keel veebilehtede kujundamiseks
DNS	*Domain Name System*, domeeninimede süsteem
GSM	*Global System for Mobile Communications*, mobiilvõrgu protokoll
HTML	*HyperText Markup Language*, hüperteksti märgistuskeel, veebilehtede loomise vahend
HTTP	*HyperText Transfer Protocol*, protokoll andmete edastamiseks arvutivõrkudes
HTTPS	*HyperText Transfer Protocol Secure*, turvaline protokoll andmete edastamiseks arvutivõrkudes krüpteeritud kujul
IDE	*Integrated Development Environment*, integreeritud arenduskekkond, koodiredaktor
IOS	*iPhone Operating System*, Apple poolt arendatav operasioonisüsteem iPhone, iPad ja iPod seadmetele
JSON	*JavaScript Object Notation*, andmemahu mõttes kerge andmevahetusformaat, mis on ühteaegu nii inim- kui ka masinloetav
JWT	*JSON Web Token*, JSON formaadis ligipääsuvõti, mida kasutatakse turvateabe jagamiseks kahe süsteemi vahel
Konveier	*Pipeline*, järjestikku ühendatud andmetöötluselementide komplekt, kusjuures ühe elemendi väljund on järgmise sisendiks

Käsujada, koodijada	*Script*, väike kompileerimata käskude kogum mida interpreteerib või täidab teine programm ilma kasutaja sekkumiseta
MSAL	*Microsoft Authentication Library*, Microsofti autentimisteek
MoSCoW	*Must, Should, Could, Won't*, nõuete olulisuse alusel grupeerimise metoodika
NPM	*Node Package Manager*, Node paketihaldur
ORM	*Object Relational Mapping*, objekt-relatsiooniline kaardistamine
PHP	*Hypertext Preprocessor*, serveris kasutatav interpreteeritav programmeerimiskeel
PWA	*Progressive Web Application*, progressiivne veebirakendus
SMS	*Short Message Service*, lühisõnumiteenus mobiiltelefonil
SSH	*Secure Shell Protocol*, turvalise ühenduse protokoll
URL	*Uniform Resource Locator*, võrguaadress

Sisukord

Jooniste loetelu

Tabelite loetelu

1 Sissejuhatus

Hiiumaale ja Saaremaale reisides on tihti soovitud päevadel või kellaaegadel praamipiletid välja müüdud. Sagedased on juhtumid, kus võib siiski soovitud ajaks kohti vabaneda, kuid nende avastamiseks on vaja pidevalt teenusepakkuja kodulehte jälgida. Käsitsi iga natukese aja tagant lehe vaatamine on tülikas ning aeganõudev.

Lõputöö eesmärk on luua rakendus, mis võimaldab kasutajal vabanenud praamipiletite kohta teavitusi tellida. Loodav lahendus peab olema kasutatav nii mobiilsetel seadmetel kui arvutis. Autori soov on luua lahendus progressiivse veebirakendusena, mis oleks kasutatav nii veebibrauseris kui erinevatel seadmetel iseseisvalt paigaldatavana.

Progressiivne veebirakendus on hea alternatiiv mobiilirakendustele, kuna need on kasutatavad nii veebibrauseris, kui ka erinevates seadmetes iseseisvalt paigaldades. Platvormipõhiste- ja hübriidmobiilirakenduste loomine on enamasti keerulisem ning ajamahukam progressiivsete veebirakendustega võrreldes. Samal ajal pakuvad progressiivsed veebirakendused rohkem võimalusi kui tavalised veebirakendused, näiteks võrguühenduseta kasutamine ja tõuketeavituste saatmine. [1]

Töö teises peatükis vaadeldakse probleemi olemust ning kirjeldatakse eesmärk – luua progressiivne veebirakendus vabade praamipiletite teavituste saamiseks. Kolmandas peatükis vaadeldakse alternatiivseid võimalusi mobiilirakenduste loomiseks ning kirjeldatakse erinevaid teavituste saatmise viise, nende eeliseid ja puudusi. Neljandas peatükis seatakse valmivale rakendusele nõuded ning valitakse sobilikud tehnoloogiad arenduseks. Viiendas peatükis kirjeldatakse iteratsioonide kaupa töö käiku. Kuuendas peatükis kirjeldatakse valminud lahendust ning autor toob välja võimalused, kuidas piletite teavituse süsteemi oleks võimalik veel edasi arendada.

2 Probleemi taust ja projekti eesmärk

Viimastel aastatel on Hiiumaa ja Saaremaa sõidukiga reisijate arv pigem tõusnud [2], [3]. Enamasti liiguvad inimesed saartele just nädalavahetuseks. Sellega seoses on suvisel ajal nädalavahetustel Hiiumaa ja Saaremaa liinil praamipiletid tihti välja müüdud. Probleemi esineb ka teistel aastaaegadel, kuid siis on seda väiksemal määral.

Igale väljumisele müüakse pileteid kuni 70% praami täituvusest. Ülejäänud pileteid on võimalik osta elava järjekorra alusel [4]. Suuremate sündmuste korral on elav järjekord mitmeid tunde pikk [5].

Pileteid on võimalik muuta ja tagastada kuni 15 minutit enne laeva väljumist. Autor on tähele pannud, et kui praamid.ee veebilehte pidevalt kontrollida, siis vabanenud piletid tulevad uuesti müügile. Nende avastamiseks on aga vaja pidevalt teenusepakkuja kodulehte jälgida. Käsitsi iga natukese aja tagant lehe vaatamine on tülikas ning aega nõudev.

2.1 Eesmärgi püstitus

Käesoleva lõputöö raames soovib autor luua rakenduse, mis kontrollib praamid.ee lehelt vabade praamipiletite saadavust ning teavitab sellest kasutajat. Rakendus on mõeldud kasutamiseks nii arvutis kui mobiilsetes seadmetes. Kasutajale saadetavad teavitused peavad olema võimalikult operatiivsed.

Lõputöö eesmärk on leida optimaalne lahendus ning realiseerida sellise rakenduse loomine. Samuti soovib hoida autor rakenduse käitamisega seotud kulud minimaalsed ning kasutada ära juba olemasolevaid ressursse.

2.2 Lähtetingimused

Käesolevas peatükis määratletakse töö lähtetingimused, milleks on andmete kättesaadavus piletite müügi keskkonnast ning valmiva rakenduse majutamine veebikeskkonda.

2.2.1 Andmete kättesaadavus

Esmalt oli vaja leida võimalus, kuidas praamid.ee lehelt on võimalik vabasid aegasid kontrollida. Autor uuris lehekülge Chrome veebilehitseja arendaja tööriistadega ning leidis, et praamide sõiduajad laetakse asünkroonse päringuga eraldi API-st (*Application Programming Interface*) ehk rakendusliidesest. Päringu parameetriteks on vaja kaasa anda kuupäev ning sõidu suund. Päringu vastuseks antakse JSON (*JavaScript Object Notation*) formaadis planeeritud sõitude kellaajad ning iga sõidu kohta täpsustav informatsioon: parvlaeva kood, hinnagrupp, vabade piletite arvud erinevate gruppide kaupa ja veel muid parameetreid. Päringu vastus JSON formaadis on näidatud joonisel 1. Vaatluse tulemusena arvestame, et rakenduse loomisel on võimalik seda rakendusliidest kasutada vabade praamipiletite saadavuse kontrollimiseks.


```
{
  "totalCount": 2,
  "items": [
    {
        "uid": "366211782305254744844153898388985170054",
        "dtstart": "2023-01-02T12:30:00.000+0000",
        "dtend": "2023-01-02T13:45:00.000+0000",
        "status": "CONFIRMED",
        "capacities": { "pcs": 239, "bc": 10, "sv": 0, "bv": 0, "dc": 0 },
        "pricelist": { "code": "HL" },
        "transportationType": { "code": "R" },
        "ship": { "code": "TI" }
    }, {
        "uid": "181570380239379370144340788741115148996",
        "dtstart": "2023-01-02T15:30:00.000+0000",
        "dtend": "2023-01-02T16:45:00.000+0000",
        "status": "CONFIRMED",
        "capacities": { "pcs": 344, "bc": 300,"sv": 56,"bv": 11,"dc": 0},
        "pricelist": { "code": "HL" },
        "transportationType": { "code": "R" },
        "ship": { "code": "TI" }
    }
  ]
}
```


Joonis 1. praamid.ee JSON vastus praamiaegade kohta.

2.2.2 Rakenduse veebimajutus

Rakenduse majutamiseks uurib autor erinevaid võimalusi, näiteks Azure App Service, DigitalOcean Droplets ning lihtsamaid veebimajutuse pakkujaid nagu Zone.ee. Kuna planeeritud rakendus kasutab ka andmebaasi, siis on vaja ka lisaressursi andmebaasi jaoks. Azure App Service puhul on võimalik kasutada tasuta veebiserverit veebirakenduse jaoks, kuid puudub tasuta andmebaasisüsteem [6]. DigitalOcean Droplets puhul on võimalik kasutada ise hallatavaid virtuaalservereid, kuid neil on antud töö jaoks sobivad teenused tasulised [7]. Zone.ee puhul on võimalik kasutada teenusepakkuja poolt hallatavat veebiserverit ning MariaDB andmebaasi. Lisaks on Zone.ee pakettides võimalik kasutada Cron töid käsurealt käsujadade perioodiliseks käivitamiseks – näiteks iga viie minuti tagant, mis teeb võimalikuks teenusepakkuja kodulehelt andmete perioodilise kontrollimise. [8]

Kuna autoril on eelnev suurem kogemus Zone.ee veebimajutuse teenusega ning ka juba olemasolev pakett, siis sai praamipileti kontrollimise rakenduse jaoks valitud just see. Zone.ee veebimajutus kodulehe andmetel on tugi MariaDB andmebaasimootorile ning kolmele programmeerimiskeelele – PHP (*Hypertext Preprocessor*), Node.JS ning Go [8]. Rakenduse loomisel on vaja nüüd arvesse võtta, et andmebaasiks saab kasutada ainult MariaDB, ning tagarakenduse programmeerimiskeeled on piiratud PHP, Node.JS ja Go keeltega.

2.3 Olemasolevad ja sarnased lahendused

Planeeritud rakenduse eesmärk on väga spetsiifiline ja täpselt sama eesmärgiga rakendusi autor ei leidnud. Seepärast uurib autor sarnaseid lahendusi, mis on loodud muude teavituste saatmiseks.

Ühe sarnase näitena on olemas teenus Maanteeameti B-kategooria vabade sõidueksamiaegade teavitamiseks. Veebilehelt https://eksamile.ee on võimalik tellida teenus ning vaba eksamiaja tekkimise korral saadetakse e-mail. Võrreldes autori poolt planeeritud lahendusega, ei paku eksamile.ee leht SMS (*Short Message Service*) ega tõuketeavituste saatmist. [9]

Paljudel e-poodidel on teavitusteenus süsteemidesse integreeritud. Näiteks https://www.reserved.com e-pood võimaldab tellida teavituse, juhul kui kaupa ei ole

soovitud koguses laoseisus, kohe toote valimise vaates. Antud juhul on samuti võimalik ainult e-posti teel teavitusi tellida. [10]

Veel leidis autor mitmeid sarnase põhimõttega lahendusi, üheks selliseks oli näiteks kinopiletite otsimise süsteem Movie Ticket Availability Notifier. Antud juhul on tegemist ainult Pythoni lähtekoodiga, mis otsib Google otsingumootorist sobilikke pileteid ning leidmise korral saadab e-posti teavituse. Võrreldes autori kavandatava lahendusega puudub kinopiletite otsimise rakendusel kasutajaliides, SMS- ja tõuketeavituste saatmine. Oluline sarnasus on regulaarne kontrollimine Cron tööde abil. [11]

3 Veebi- ja mobiilirakenduste tüübid ning teavituste saatmise viisid

Autor soovib luua rakenduse, mis oleks kasutatav nii arvutis kui mobiilsetes seadmetes. Käesolevas peatükis vaadeldakse veebirakendusi ning erinevaid mobiilirakenduste tüüpe. Samuti kirjeldatakse erinevaid võimalusi kasutajale teavituste saatmiseks.

3.1 Veebi- ja mobiilirakenduste tüübid

Mobiilsetes seadmetes kasutamiseks mõeldud rakenduste loomiseks on mitmeid erinevaid võimalusi. Näiteks on erinevat tüüpi mobiilirakendused, mis on kasutatavad eraldiseisva rakendusena, kuid ainult mobiilsetes seadmetes. Sellised on näiteks platvormipõhised-, platvormiülesed- ning hübriidmobiilrakendused. Lisaks on veebirakendused, mis on kasutatavad ainult veebibrauseris. On veel progressiivsed veebirakendused, mis on kasutatavad nii veebilehena kui ka paigaldatavad eraldiseisva rakendusena.

3.1.1 Veebirakendused

Veebirakendus on serveritarkvara, mida on võimalik kasutada ainult veebibrauseri kaudu. Kaasaegsed veebirakendused luuakse kohanduva kujundusega ning seetõttu on kasutatavad väga paljudes erinevates seadmetes – arvutites, mobiiltelefonides, tahvelarvutites jne. Tänu sellele on veebirakendused lihtsasti kättesaadavad paljudele kasutajatele. [1]

Kuna veebirakendus asub serveris ning on kasutatav ainult läbi veebibrauseri, siis on selle kasutamiseks vajalik võrguühendus. Ilma serverile ligipääsuta ei ole võimalik rakendust kasutada. Võrreldes mobiilirakendustega on see üks veebirakenduste puuduseid. Lisaks ei ole võimalik veebirakendusega kasutada kõiki seadme- ja operatsioonisüsteemi komponente. [12]

3.1.2 Progressiivsed veebirakendused

PWA (*Progressive Web Application*) on rakendus, kus on ühendatud veebitehnoloogiate ja mobiilirakenduste parimad omadused. See on ühest küljest kasutatav kui tavaline veebilehitsejas avatav veebileht. Teisalt on teda võimalik paigaldada seadmesse täiesti

iseseisva rakendusena ning olenevalt rakenduse iseloomust võib see olla kasutatav ka ilma võrguühenduseta. Üldiselt iseloomustavad PWA-d järgmised omadused [13]:

- **Leitavus** – Sarnaselt veebilehtedele on progressiivsed veebirakendused otsimootoritele kättesaadavad ning tänu sellele on kasutajatele kergesti leitavad.
- **Paigaldatavus** – Sarnaselt mobiilirakendustele on progressiivne veebirakendus paigaldatav töölauale. Paigaldamine ei toimu läbi rakenduste e-poe vaid otse veebilehelt ning töölauale luuakse otsetee rakenduse avamiseks.
- **Lingitavus** – Progressiivne veebirakendus on veebis leitav ning sellele on võimalik luua unikaalne veebiaadress.
- **Võrgust sõltumatus** – Mobiilirakendustele sarnaselt töötab progressiivne mobiilirakendus ka ilma võrguühenduseta. Rakenduse sisu salvestatakse brauseri vahemällu ning on sealt kättesaadav. Võrguühendust vajavad teenused suudavad siluda ühenduse puudumisest tekkivad vead ning informeerida sellest kasutajat.
- **Taashaaratavus** – Progressiivsed veebirakendused võimaldavad taustal töötades andmete uuendamist ning tõuketeavituste edastamist.
- **Kohanduv kujundus** – Progressiivse veebirakenduse disain on kohanduv erineva suurusega seadmetega.
- **Turvalisus** – Progressiivsed veebirakendused töötavad ainult üle turvalise HTTPS (*HyperText Transfer Protocol Secure*) kanali.

3.1.3 Platvormipõhised mobiilirakendused

Platvormipõhised mobiilirakendused (*native applications*) on konkreetse mobiilse seadme operatsioonisüsteemi jaoks kirjutatud rakendused. Näiteks IOS (*iPhone Operating System*) jaoks kasutatakse Swift programmeerimiskeelt ning Androidi rakenduste jaoks Java. Platvormipõhistel rakendustel on mitmeid eeliseid teist tüüpi rakenduste ees [1]:

- Rakendused töötavad kiirelt, usaldusväärselt (*reliable*) ning kiiresti reageerivalt (*responsive*).

- Võimaldavad lihtsamalt kasutada seadme riistvara – kaamerat, mikrofoni, kompassi, erinevaid andureid ning püüda hõlpsalt ekraanil pühkimisliigutusi.
- Ühilduvad seadme enda kujundusmallide ning kogemuskujunduse mustritega.

Teisest küljest on platvormipõhistel rakendustel ka negatiivne pool – iga operatsioonisüsteemi jaoks on vaja luua eraldi koodibaas. See omakorda tähendab, et kõikidele platvormidele rakenduse loomine on ajamahukas ning rahaliselt väga kulukas [1].

3.1.4 Platvormiülesed mobiilirakendused

Platvormiülesed mobiilirakendused (*cross-platform applications*) jagavad erinevate operatsioonisüsteemide jaoks loomisel ühist koodibaasi. Selliselt on võimalik tootmise kulusid olulisel määral vähendada, kuid seadmete platvormide erinevusest tulenevalt on siiski vaja eri kujundusmalle kasutada ning eri seadmetega testida. [1]

Platvormiüleseid mobiilirakendusi võimaldavad luua näiteks Microsoft Xamarin, Google Flutter, React Native ja mitmed teised platvormid. [1]

3.1.5 Hübriidmobiilrakendused

Hübriidmobiilirakendused on sarnaselt platvormipõhistele ja platvormiülestele rakendustele rakenduste poest alla laetavad, kuid tegelikult on nad pigem HTML (*HyperText Markup Language*), Javascript ja CSS abil ehitatud veebilehed, mis on ümbritsetud riistvara realiseeriva kihiga ning käivitatakse lihtsustatud veebibrauseri vaates [1]. Selline lahendus võimaldab küll platvormide üleselt jagada suurt osa koodibaasi, kuid keerulisem on ligi pääseda riistvaralistele komponentidele, nagu kaamera, kompass jmt.

3.1.6 Mobiilirakenduste tüüpide kokkuvõte

Autor soovib luua rakenduse, mis oleks kasutatav nii veebibrauseris kui mobiilirakendusena, kuid soovib töömahtu hoida võimalikult väiksena ning minimaalselt koodi dubleerida. Loodava rakenduse puhul ei ole vajalik riistvaraliste komponentide kasutamine, mida lihtsustaks platvormipõhiselt- või platvormiüleselt rakenduse loomine. Seetõttu valib autor rakenduse tüübiks progressiivse veebirakenduse.

3.2 Teavituste saatmise võimalused

Kasutajale teavituste saatmiseks on võimalik kasutada erinevaid kanaleid, näiteks e-kiri, SMS ja tõuketeavitused. Käesolevas peatükis vaadeldakse neid võimalusi lähemalt ning võrreldakse nende eripärasid.

3.2.1 E-kiri

E-kirja saatmine kasutajale on väga lihtne, kuid enamasti ei sobi kiireloomuliste teavituste jaoks. Tihti võib peale e-kirja saatmist mööda minna mitu tundi enne, kui kasutaja selle avab. Ühe vanema uuringu andmetel on e-maili keskmine avamise aeg peale selle saatmist umbes kuus ja pool tundi [14]. E-kirja eeliseks võib pidada, et sellel ei ole tähemärkidega piiratud mahupiirangut ning sinna võib ka lisada pilte. Kuid tulenevalt e-kirja avamisega seotud viivitustest ei pea autor praamipiletite saadavuse teavitamist e-kirja teel mõistlikuks.

3.2.2 SMS

SMS on lühisõnumiteenus, mis võeti kasutusele juba 1990 aastal kasutades GSM (*Global System for Mobile Communications*) standardeid. SMS sõnumi pikkus on piiratud 160 tähemärgiga, kuid neid on võimalik kombineerida ühendatud sõnumiks kuni 918 tähemärgini. Tänapäeval on SMS saanud osaks igapäevaelust nii suhtluskanalina kui ka muudel eesmärkidel, näiteks mitmetasandilise autentimise realiseerimise osana [15]. Umbes 90% SMS sõnumitest loetakse esimese kolme minuti jooksul, mistõttu on see hea viis kiireloomuliste teavituste saatmiseks [14].

SMS sõnumite saatmiseks on vaja kasutada operaatorit, kes toimetaks sõnumi mobiilsideoperaatorile. Selliseid operaatoreid on turul mitmeid, näiteks Twilio ja Messente. Mõlemad teenusepakkujad võimaldavad saata SMS sõnumeid läbi rakendusliidese. Messente pakub omalt poolt kasutuselevõtmise lihtsustamiseks ka teeke erinevate programmeerimiskeelte jaoks. Teenusepakkujate kodulehtede andmetel pakuvad mõlemad ettemaksu põhist arveldamist, kuid Messentel on ühe sõnumi saatmise hind soodsam. Arvestades soodsamat hinda ning valmis teekide olemasolu valib autor SMS saatmise teenusepakkujaks Messente. [16], [17]

3.2.3 Tõuketeavitused

Tõuketeavituste on lühikesed teavitused, mida on võimalik kuvada seadme olekuribal või teavituskeskuses. Tõuketeavitustel on mitmeid eeliseid, näiteks nende kättetoimetamine on kiire ja saatmine enamasti ei tähenda täiendavaid kulusid. Samuti peetakse tõuketeavitusi ka vähem pealetükkivaks kui SMS sõnumeid, kuna neid on võimalik vastava rakenduse seadistustest vaigistada või nende saatmisest üldse keelduda. [14]

Olenevalt operatsioonisüsteemist ning rakenduse iseloomust võib toimuda teavituste saatmine erinevatel viisidel [18]. Kuna PWA töötab läbi veebilehitseja või iseseisvalt mähituna veebilehitseja sisse, siis on progressiivse veebirakenduste puhul võimalik kasutada teavituste saatmiseks Push API protokolli. Push API on standardiseeritud rakendusliides sõnumite saatmiseks serverist veebibrauserisse. Enamus kaasaegsed veebibrauserid toetavad seda. Üks suur erand on siiani Safari IOS operatsioonisüsteemis, mis tänaseni ei võimalda Push API teenuse kasutamist [19].

Lisaks on Push API standardile on veel erinevaid teenusepakkujaid, kes võimaldavad tõuketeavituste saatmist. Näiteks Google Firebase Messaging, Pusher Beams, WonderPush, OneSignal jne. Enamus neist on tasulised teenused. Mõned neist (näiteks Pusher Beams) toetavad IOS platvormile teavituste saatmist kasutades Apple Push Notification Service võimalust, kuid selleks on veel lisaks vaja tasulist Apple arendussertifikaati. [20]

Planeeritavas rakenduses ei soovi autor kasutada tõuketeavituste saatmiseks lisateenuseid ning arendab saatmise süsteemi kasutades Push API standardil põhinevat meetodit.

3.2.4 Teavitusviiside kokkuvõte

Autor soovib vabanenud praamipiletite informatsiooni kohta teavituste saatmiseks kasutada võimalikult operatiivseid teavitusmeetodeid, seetõttu võetakse kasutusele SMS ja tõuketeavitused. SMS saatmise teenusepakkujaks valib autor Messente, kuna see on Twilioga võrreldes soodsam ning tänu olemasolevatele teekidele lihtsamini kasutusele võetav. Tõuketeavituste saatmise kavatseb autor kasutusele võtta kasutades Push API standardil põhinevat meetodit. E-maili kasutamist ei plaanita.

4 Analüüs

Analüüsi käigus selgitatakse välja nõuded rakendusele ning valitakse arendamiseks sobilikud tehnoloogiad. Eraldi vaadeldakse tagarakenduse, eesrakenduse ning versioonihalduse tehnoloogiaid.

4.1 Nõuded rakendusele

Sarnaste lahenduste vaatlemisel ning planeeritava rakenduse vajadusi silmas pidades kirjutab autor välja nõuded kasutajalugudena. Kasutajalood on kirjutatud kujul „*Kellena* saan *mida teha*, et *saavutada midagi“*.

Nõuete nimekirja järjestab autor MoSCoW meetodil. MoSCoW meetodi puhul jagatakse kasutajalood nelja erinevasse gruppi. *Must have*, rakenduse töötamiseks hädavajalikud funktsionaalsused. *Should have*, olulised, aga mitte kriitilise tähtsusega nõuded. *Could have*, soovitud funktsionaalsus, mille puudumine otseselt ei sega rakenduse kasutamist ning võib välja jätta kui näiteks valmimise tähtaeg on nendest nõuetest tingituna ohus. *Won’t have this time*, nõuded mis ei mahu käesoleva projekti skoopi [21]. Prioriteetide järgi gruppidesse jagatud nõuded on toodud tabelis 1.

Tabel 1. Nõuete prioritiseerimine MoSCoW meetodi abil.

Must have	**Should have**
• Kasutajana saan valida praamiaja millele soovin teavitusi, et saaksin teavituse saabumisel osta pileti soovitud praamile. • Kasutajana saan SMS teavitusi praamiaja kohta, et saaksin osta sobiva pileti. • Kasutajana saan rakendusse sisse logida, et teha tellimusi teavitusteks.	• Kasutajana saan lisada, muuta ja kustutada telefoninumbreid sõnumite saatmiseks, et saaksin kasutada numbrit tellimuse kinnitamisel. • Kasutajana saan tellimuse loomisel kasutada eelsalvestatud telefoninumbrit, et saaks hõlpsamini tellimust kinnitada. • Kasutajana näen oma tellimuste ajalugu, et omada ülevaadet tehtud tellimustest.

Could have	**Won't have this time**
• Kasutajana saan kiiresti tellimust korrata, et ei peaks iga kord praamiaegasid uuesti valima. • Kasutajana näen saadetud SMS staatust, et näha kui sõnum on kohale toimetatud. • Kasutajana saan lisada ja kustutada seadmeid tõuketeavituste saatmiseks, et tõuketeavitusi saada soovitud veebilehitsejatega. • Kasutajana saan praamiaja kohta tõuketeavitusi, et soovitud praamipilet osta. • Administraatorina näen aegade kontrollimise logi, et tuvastada süsteemitõrkeid. • Administraatorina näen kasutajate nimekirja, et omada ülevaadet kasutajatest. • Administraatorina näen Messente konto jääki, et saaks vajadusel ettemakse teha. • Kasutajana saan paigaldada rakenduse oma seadmesse, et saaks selle käivitada eraldiseisvana. • Kasutajana saan rakenduse käivitada ilma võrguühenduseta ning rakendus kuvab selle kohta vastava info.	• Kasutajana saan sisse logida sotsiaalmeedia kontoga (Facebook, Google jms), et ei peaks iga kord eraldi sisse logima.

Vastavalt järjestamise tulemusele on võimalik planeeritav töö jagada erinevatesse faasidesse. Esimeste iteratsioonidega proovitakse täita *Must have* kategooria nõuded,

seejärel *Should have* ning lõpuks, kui ajagraafik võimaldab, *Could have* nimekirjas olevad kasutajalood.

4.2 Tehnoloogiate valik

Rakenduse loomiseks on palju võimalikke programmeerimiskeeli, raamistikke ning teeke. Võttes arvesse püstitatud eesmärki, sellele esitatud nõudeid ning lähtetingimusi analüüsitakse- ning valitakse käesolevas peatükis neist sobivaimad rakenduse loomiseks.

4.2.1 Tagarakenduse programmeerimiskeel

Lähtetingimusena seatud veebimajutuse piirang limiteerib tagarakenduse programmeerimiskeelteks PHP, Node.JS ja Go.

PHP on laialt levinud skriptimiskeel mis loodi juba 1995 aastal. Tänaseks kasutavad ligikaudu 80% kõikidest veebilehtedest PHP-d. Seda suuresti tänu sellele, et mitmed sisuhaldussüsteemid on kirjutatud PHP keeles. Enim levinud selliseks on Wordpress. Teine põhjus PHP suureks populaarsuseks on suur arv raamistikke, mis lihtsustavad arendaja tööd [22]. Autor on PHP keelt kasutanud varasemalt tööalaselt ning täiendanud teadmisi ka õpingute käigus.

Node.JS on Javascript käituskeskkond. Javascript algselt oli ainult veebirakenduste brauseris käivitatav osa, kuid Node.JS abiga on võimalik koodi käivitada ka serveris ning selle abil luua serveri poolseid rakendusi [23]. Node.JS kasutamiseks Zone.ee veebimajutuses on vaja teha täiendavaid seadistusi veebiserveril, ilma milleta ei ole võimalik rakendust avaliku veebi jaoks kättesaadavaks teha [24]. Autoril on varasem Javascripti kogemus ainult eesrakenduste loomisel, kuid Node.JS tagarakenduse loomiseks ei ole kasutanud.

Go on Google poolt 2009 aastal avaldatud avatud lähtekoodiga programmeerimiskeel [25]. Keel on palju populaarsust kogunud kuna see võimaldab mitmes lõimes samaaegset käitamist ning sisaldab kaasaegseid funktsioone nagu näiteks automaatne mälu puhastamine (*garbage collection*) [26]. Kuna autoril varasem kogemus Go programmeerimiskeelega puudub, siis jääb see valikus kõrvale.

Programmeerimiskeelte võrdlemisel arvestatakse autori varasemat kogemust keelega, rakenduse üles seadmise keerukust serveris ning kasutajaskonna suurust. Viimane on oluline praktilise töö ajal veebist abimaterjalide leidmisel. Tulemused on toodud tabelis 2.

Tabel 2. Tagarakenduse programmeerimiskeelte võrdlustabel.

	PHP	**Node.JS/Javascript**	**GO**
Varasem kogemus	Suur	Väike	Puudub
Üles seadmine serveris	Ei vaja täiendavaid seadistusi [8]	Vajab lisaseadistusi [24]	Vajab lisaseadistusi [27]
Kasutajaskonna suurus	Keskmine [28]	Suur [28]	Väike [28]

Arvestades eelnevat kogemust, kogukonna suurust ja rakenduse üles seadmise lihtsust serveris, valis autor tagarakenduse keeleks PHP.

4.2.2 Tagarakenduse raamistik

Raamistik on kogum funktsioone ja teeke, mis aitavad programmeerijal lihtsamini, kiiremini ning turvalisemat koodi kirjutada. PHP programmeerimiskeele jaoks on loodud mitmeid erinevaid raamistikke – Laravel, Symphony, Zend, CakePHP jpt. [29]

Käesoleva töö jaoks soovib autor kasutada lihtsat ja kompaktset raamistikku, mis oleks mõeldud ainult tagarakenduse jaoks. RapidApi ajaveebi soovitusel on parimateks mikroteenuste ja API liidese raamistikkudeks Lumen, Guzzle ja Slim. [30]

Lähemal uurimisel selgub, et Guzzle on pigem HTTP (*HyperText Transfer Protocol*) klient või teek veebiteenuste kasutamiseks [31], mis ei ole autori arvates piisav piletikontrollimise süsteemi tagarakenduse loomiseks.

Lumen on loodud Laraveli arendajate poolt ja põhineb osaliselt ka Laraveli komponentidel. Lumen sisaldab endas kõike vajalikku mikroteenuste ja API-de loomiseks – suunamised (*routing*); Eloquent ORM (*Object-relation management*) ja

migratsioonid mugavaks andmebaasi kasutamiseks; sõltuvuste süstimine (*depency injection*) ja palju muud. [32]

Slim dokumentatsioonist selgub, et tegemist on sarnase funktsionaalsusega nagu Lumenil, kuid puudu on näiteks andmebaasi tugi. Selleks on vaja kasutada lisa teeki. [33]

Võttes arvesse just Laraveli suurt kasutajate kogukonda [29] ning Lumeni sarnasust Laraveliga ja sisseehitatud andmebaasi tuge, valib autor tagarakenduse raamistikuks Lumeni.

4.2.3 Eesrakenduse programmeerimiskeel

Progressiivsed veebirakendused töötavad veebibrauseris nagu tavalised ühe lehe veebirakendused, seetõttu on nende arendamiseks ka kasutusel samad tehnoloogiad: HTML, CSS ja Javascript. Kaasaegses veebiarenduses on aga võimalik kasutada erinevaid raamistikke ja teeke, mis muudavad koodi veebibrauseri jaoks kasutatavaks. Sellisteks ühelehe rakenduse loomise raamistikeks on näiteks React, Vue.js, AngularJS ja paljud teised. Raamistike kasutamisel on ka võimalus kasutada Microsofti poolt arendatud Javascripti täiendkomplekti TypeScript, mis lisab keelele näiteks tüübikindluse ja see omakorda vähendab koodi kirjutades vigade tekkimise ohtu. [34]

4.2.4 Eesrakenduse raamistik

Eesrakenduse loomiseks soovib autor kasutada mõnda laialt levinud raamistikku ning selle leidmiseks võetakse appi RubyGarage veebilehel koostatud võrdlus, kus on toodud välja viis populaarseimat Javascript/Typescripti raamistikku: React, Angular, Vue, Ember ja Backbone [35]. Kuna kahe viimasega autor ei ole varasemalt kokku puutunud, siis antud töö käigus neid ei edasi ei uurita. React ja Vue on autorile tuttavad õpingute käigust ning Angulariga on kokku puutunud tööalaselt.

Kõik kolm raamistikku on omavahel väga erinevad. Angular on suur täieõiguslik raamistik ning seetõttu on tema õppimiskõver kõige järsem. React ja Vue on pigem teegid kui raamistikud, kuid oma lihtsuse tõttu on nendega ka töö alustamine kiirem. [35], [36]

Populaarsuselt on Vue kolmest valikust kõige viimane. Angular muutus peale teise versiooni välja tulemist väga palju ning suur osa kasutajaskonda ei ole veel oma süsteeme uuemale versioonile üle viinud. Seetõttu ei ole ka uusima Angulari kasutajaskond veel

väga suur. React on laialt levinud ning sellele on võimalik leida nii tuge kui valmis komponente. [35], [36]

Tabelis 3 toob autor välja kolme populaarseima Javascripti raamistiku põhilised võrdluspunktid, milleks on varasem kogemus, õpitavuse keerukus ning kasutajaskonna suurus.

Tabel 3. Eesrakenduse raamistike valik.

	React	**Angular**	**Vue**
Varasem kogemus	Väike	Väike	Väike
Õpitavus	Lihtne [36]	Keeruline [36]	Lihtne [36]
Kasutajaskonna suurus	Suur [36]	Keskmine [36]	Keskmine [36]

Kuna varasem kogemus kõigi kolme raamistikuga on autoril väike, siis põhilisteks argumentideks jäid õpitavuse lihtsus ning abi leidmiseks kasutajaskonna suurus. Vue ja React mõlemad on lihtsalt omandatavad, kuid kasutajaskond on Reactil suurem. Seeläbi on autoril võimalik ka avalikust veebist lihtsamini abi leida. Autori lõplik valik eesrakenduse loomiseks on React.

4.2.5 Versioonihaldus ja pideva paigalduse vahendid

Versioonihaldustarkvara on kasulik meeskondades töötades ühise lähtekoodi haldamiseks. Üksinda töötades on see abiks koodi muudatuste ajaloo säilitamiseks või mitmest erinevast seadmest sama koodiga töötamisel. Autor soovib kasutada versioonihaldustarkvara põhiliselt ajaloo säilitamiseks, kuid samuti ka CI/CD (*Continuous Integration/Continuous Delivery or Continuous Deployment)* juurutamiseks.

CI/CD hõlmab endas praktikaid, kus koodi kirjutatakse väikeste, kuid terviklike tükkidena nii, et seda on võimalik pidevalt tootmisserverisse toimetada. CI/CD rakendamiseks kasutatakse tihti konveiereid (*pipeline*), millega automatiseeritakse lähtekoodi transport, testimine, paigaldus serverisse jms. [37]

Käesoleva töö käigus soovib autor kasutada lähtekoodihoidlat ning CI/CD konveierit. Selleks vaatleme erinevaid teenuseid nagu Azure DevOps, Github ja Bitbucket.

Azure DevOps on Microsofti poolt pakutud pilveteenus. Seal on kokku toodud palju erinevaid arendajale vajalikke tööriistu – Azure Repos on lähtekoodihoidla, mis võimaldab luua ja hallata Git repositooriumeid; Azure Boards võimaldab luua agiilse töölaua ülesannete haldamiseks ning planeerimiseks; Azure Pipelines abil saab luua tarkvara kiireks paigaldamiseks mõeldud konveiereid. [38]

Bitbucket on Atlassiani poolt pakutud teenus, mis sisaldab endas Git repositooriumi ja kõike vajalikku sellega töötamiseks. Samuti on Bitbucketis olemas CI/CD konveierite tugi. Tööülesannete haldamiseks on Bitbucket integreeritud Atlassiani Trello ning Jira arendusprojektide haldamise tarkvaraga. [39]

Github pakub samuti Git koodihoidla teenust ning CI/CD konveiereid. Kodulehe andmetel pakub Github ka vigade haldamise süsteemi, kuid see ei ole sama nagu on Azure Boards või Trello, ehk tööülesannete planeerimise tööriist Githubil vähemalt tasuta versioonis puudub. [40]

Tabelis 4 toob autor välja kolme vaadeldud versioonihaldustarkvara põhilised omadused, mida kasutab valiku langetamiseks. Nendeks on varasem kogemus, hind, CI/CD konveierite tugi ning agiilne tööülesannete haldamise tööriist.

Tabel 4. CI/CD keskkondade võrdlus.

	Azure DevOps	**Bitbucket**	**Github**
Varasem kogemus	Keskmine	Puudub	Väike
Hind	Tasuta [38]	Tasuta [41]	Tasuta [40]
CI/CD konveierite tugi	Azure Pipelines [38]	Olemas [39]	Olemas [40]
Tööülesannete haldamise tööriist	Azure Boards [38]	Jira ja Trello [39]	Puudub [40]

Kõik kolm tööriista on antud projekti jaoks tasuta kasutatavad, kuid Githubil puudub tööülesannete haldamise ja planeerimise tööriist. Valikusse jäävad Bitbucket ning Azure DevOps, kuid võttes arvesse autori suuremat kogemust Azure DevOps osas, osutub valituks just see.

5 Lahenduse loomine

Rakenduse loomisel kasutatakse agiilset töömeetodit ning lahendus valmib etappide kaupa alustades kontseptsiooni tõestusest ja lõpetades valmis rakendusega.

5.1 Arendusmeetodite ja töövahendite kirjeldus

Praktilise osa läbiviimisel kasutab autor agiilset lähenemist. Rakenduse loomine jagatakse väiksemateks etappideks ehk iteratsioonideks. Iga etapiga luuakse lahendusele juurde väike osa funktsionaalsust ning etapi lõpus toimub manuaalne testimine ning lahenduse publitseerimine produktsioonikeskkonda.

Tagarakenduse loomiseks kasutab autor JetBrains PHPStorm integreeritud arenduskeskkonda (*Integrated Development Environment,* IDE). Eesrakenduse loomisel on kasutusel Microsoft Visual Studio Code koodiredaktor. Tööülesannete ning leitud vigade haldamiseks kasutatakse Azure DevOps Boards töölauda.

5.2 Esimene iteratsioon – kontseptsiooni tõestus

Selleks, et aru saada kas valitud lahendus võiks sobida praamipiletite saamiseks loob autor kontseptsiooni tõestuse. Autor loob PHP käsujada, mis pärib andmebaasist kõikide aktiivsete tellimuste kuupäevad. Iga kuupäeva kohta teeb praamid.ee rakendusliidesele ühe päringu. Tulemusi võrreldakse ning kui vabade sõiduki piletite arv on suurem kui null, saadetakse SMS sõnum autori mobiiltelefonile ning andmebaasipäringuga salvestatakse tellimus lõpetatuks.

Lähtekood paigaldatakse Zone serverisse ning seadistatakse Cron töö, mis automaatselt käivitab PHP käsujada iga viie minuti tagant.

Tellimuste haldamine on võimalik ainult otse andmebaasi kirjutades. Zone keskkond võimaldab seda teha läbi phpMyAdmin graafilise kasutajaliidese.

Esimeste katsetustega leiab autor, et vaba pileti leidmisel teavitamine SMS teel töötab ja peale teavituse saamist on võimalik soovitud kellaajaks sõidupilet osta.

5.3 Teine iteratsioon – rakendusliides

Teise arendusetapi eesmärk on kasutusele võtta Lumen raamistik ning Azure DevOps versioonihaldus ning CI/CD.

Tagarakenduse loomiseks paigaldab autor Lumen raamistiku kasutades Composer paketihaldussüsteemi. Selleks käivitab käsurealt käsu, mis on kujutatud joonisel 2.

```
composer create-project --prefer-dist laravel/lumen piletid_api
```

Joonis 2. Lumen projekti loomine käsureal.

Järgmise sammuna seadistab autor raamistiku andmebaasi ühenduse. Selleks on vaja lisada konfiguratsioonifaili andmebaasi ühenduseks vajalikud parameetrid: serveri aadress, kasutajanimi, parool ning andmebaasi nimi. Lumen raamistikul on erinevad võimalused andmebaasipäringute koostamiseks – näiteks kasutades andmebaasi fassaadi või Eloquent mudeli põhist päringute loomist. Lumeni andmebaasi fassaad on fassaadi tarkvarakujundusmustril põhinev käskude kogum mis lihtsustab päringute kasutamist. Eloquent omakorda lihtsustab koodis kirjeldatud andmemudeli ja andmebaasipäringute koos kasutamist. Eloquent negatiivseks küljeks on suuremahuliste ja keerukamate päringute puhul väiksem kiirus ning suurem süsteemiressursi kasutus [42]. Erinevaid andmepäringute iseloome silmas pidades võtab autor kasutusele mõlemad võimalused. Selleks on vaja Lumeni eellaaduri failis Eloquent ja andmebaasi fassaadimustri kasutamine sisse lülitada, mis on kujutatud joonisel 3.

```
$app->withFacades();
$app->withEloquent();
```

Joonis 3. Lumen eellaaduri seadistamine.

Lumenil raamistikul on sisse ehitatud käsurea tööriist Artisan, mis on sobilik vahend ajastatud praamipiletite kontrollimiseks. Autor loob uue faili app/Console/Commands kausta. Faili nimeks on CheckTrips.php, mis sisaldab „CheckTrips“ klassi. Uus klass laiendab Lumeni „Command“ klassi. Koodis on vaja lisada uuele klassile „signature“ parameeter ning „handle“ meetod, mis on mõlemad kujutatud joonisel 4. Samuti lisatakse viide uuele klassile app/Console/Kernel.php faili, mis võimaldab käsurealt käivitamise nagu on näha joonisel 5.

```
class CheckTrips extends Command
{
    protected $signature = 'check';
    protected $description = 'Check for trips in praamid.ee';

    public function handle()
    {
        echo "hello world!";
    }
}
```

Joonis 4. Artisan käsurea klass.

```
php artisan check
```

Joonis 5. Artisan käsu käivitamine käsureal.

Järgmise sammuna kohandab autor esimeses iteratsioonis loodud koodi CheckTrips klassi kasutades Lumeni andmebaasi- ning HTTP fassaade. Täiendusena esialgsele koodile võeti kasutusele Messente teek, mida on võimalik Composer paketihalduri kaudu rakendusele paigaldada.

Edasise arenduse ja serverisse paigaldamise hõlbustamiseks juurutab autor pideva integratsiooni konveieri. Azure DevOps keskkonnas luuakse piletite kontrollimise rakenduse jaoks uus projekt. Projekti sees luuakse tagarakenduse jaoks uus Git lähtekoodihoidla ning rakendusele tehakse vajalikud seadistused lähtekoodihoidlaga ühendamiseks. Seejärel lisatakse ka Zones asuvasse veebiserverisse Git lähtekoodihoidla ühendus ning luuakse Azure DevOps keskkonnas konveier lähtekoodi uuendamiseks. Konveieri tööpõhimõte on selline – koodi tõukamisel Azures asuva koodihoidla peaharusse luuakse ühendus üle SSH kanali Zones asuvasse veebiserverisse ning tõmmatakse Git uuendused ning seejärel käivitatakse käsureal „composer install“, mis paigaldab lisatud sõltuvused.

Viimase sammuna käesolevas iteratsioonis tehakse vajalikud muudatused Cron perioodiliste tööde käivitamises. Eelnevalt lihtsalt PHP faili käivitamise asemel käivitatakse nüüd loodud Artisan käsk praamipiletite kontrollimiseks.

5.4 Kolmas iteratsioon – lihtne veebirakendus React/MUI eesrakendus

Kolmanda arendustsükliga loob autor ühe kasutaja jaoks loodud veebirakenduse. Veebirakenduse funktsionaalsus on sõidu kuupäeva ning kellaja aja valik ja käsitsi telefoninumbri salvestamine.

Esmalt loob autor React rakenduse NPM (*Node Package Manager*) paketihalduri ja Create React App teegi abil kasutades Typescripti malli. Selleks käivitatakse käsurealt vastav käsk, mis on kujutatud joonisel 6.

```
npx create-react-app piletid-react --template typescript
```

Joonis 6. React rakenduse loomine käsureal.

Seejärel lisasb autor samuti NPM paketihalduri abil abistavad teegid, näiteks Material UI, Axios, React Router. Material UI on kogum kasutajaliidese komponente, mis kohanduvad vastavalt kasutatava seadme ekraani suurusele ning hõlbustavad rakenduse mobiilivaate loomist. Axios on HTTP klient, mis lihtsustab päringute tegemist tagarakendusele ja teistele vaja minevatele API-dele. React Router võimaldab React üheleherakendusel navigeerida erinevate vaadete vahel.

Klientrakenduses loodi praami sõidusuundade ning -aegade kuvamise vorm, kus kasutajal on võimalik valida millisele praamile on soov piletit osta. Praamide aegu, millele on vabu pileteid saada, rakendus valida ei luba. Samuti on vormil tekstilahter mobiiltelefoni numbri sisestamiseks ning nupp tellimuse salvestamiseks.

Eesrakenduse paigaldamiseks juurutab autor Azure DevOps keskkonnas eraldi Git lähtekoodihoidla ning konveieri. Lähtekoodi tõukamisel hoidlas olevasse peamisesse harusse käivitatakse rakenduse loomine lähtekoodist "npm install“ ja „npm build“ käskudega. Seejärel tõstetakse loodud käivitusfailid üle SSH kanali Zone veebiserverisse.

Tagarakenduses lisab autor vajaliku rakendusliidese kontrolleri tellimuse salvestamiseks. POST meetodiga kasutatav otspunkt saab JSON kujul tellimuse andmed ning salvestab tellimuse andmebaasi. Veel lisatakse tagarakenduses SMS sõnumite kohaletoimetamise raportite vastuvõtmise otspunkt. Selleks on vaja täiendada ka piletite kontrollimise meetodit lisades SMS saatmisel täiendav URL (*Uniform Resource Locator)* ning Messentest tagastatav identifikaatori salvestamine andmebaasi. Sõnumi edastamise staatuse uuendamisel postitab Messente uue staatuse teenusest kaasa antud aadressile

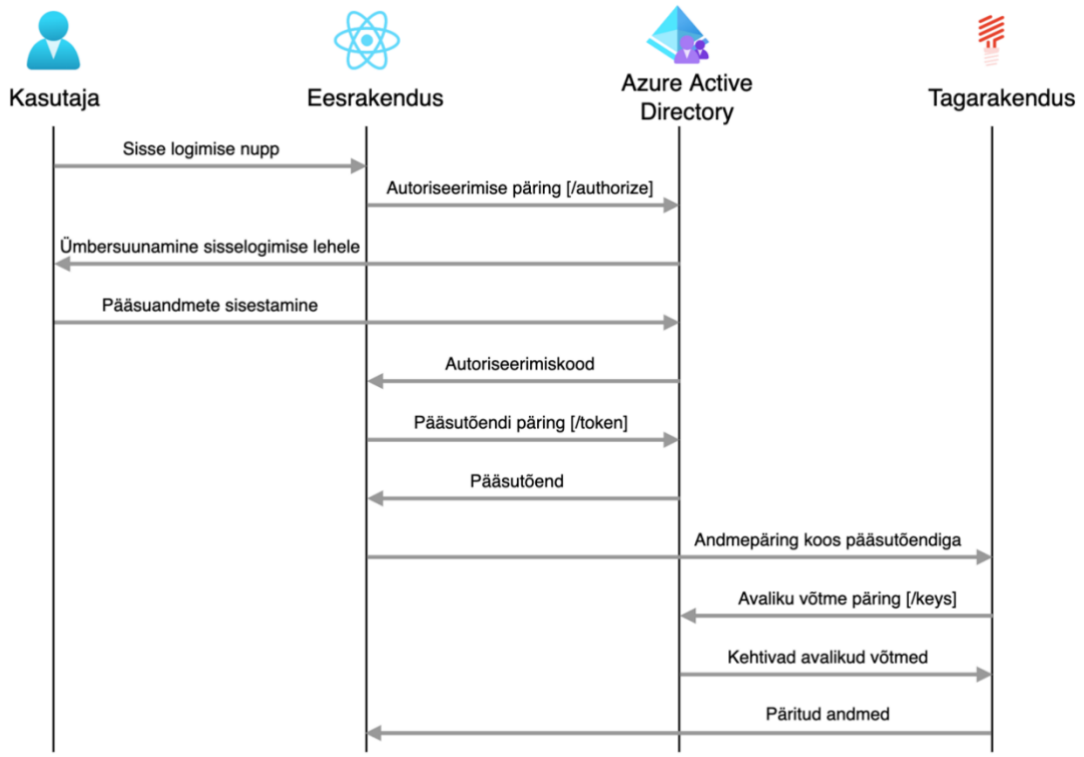
ning rakendus salvestab selle andmebaasi. Selle abil on võimalik järgmises iteratsioonis tellimuste ajaloo kuvamise vaates näidata, kas sõnumi saatmine oli edukas või mitte.

Kolmanda iteratsiooni käigus registreerib autor rakenduse jaoks ka veebiaadressi pilet.info ning teeb vajalikud DNS (*Domain Name System*) suunamised veebiserveri peale. Tagarakendus töötab aadressil https://api.pilet.info ning eesrakendus on kättesaadav aadressil https://pilet.info. Kuna rakendusel ei ole veel sisse ehitatud autentimise süsteemi lisati veebiserveri poolt klientrakendusele liht-pääsuautentimine (*Basic Access Authentication*).

5.5 Neljas iteratsioon – veebirakendus

Neljandas iteratsioonis lisab autor rakendusele autentimise, autoriseerimise, tellimuste ajaloo kuvamine ning administraatori rolliga kasutajatele Messente konto jäägi vaatamise võimalused.

Kasutajate haldamiseks võeti kasutusele Microsofti identiteeditenuse platvorm Azure AD (*Active Directory*). Azure identiteeditenus võimaldab rakendustel autentida kasutajaid Oauth2 turvaprotokolli kasutades, mille tööpõhimõtet on kujutatud joonisel 7.



Joonis 7. Azure AD autentimise skeem [43].

Eesrakenduses kasutatakse MSAL (*Microsoft Authentication Library*) teeki, mis on Microsofti enda poolt soovitatud vahend turvalise autentimise lahenduse loomiseks[44]. MSAL abil on võimalik kasutada AuthenticatedTemplate komponenti, mis peidab autentimata kasutajatele komponendi sisu. Samuti on võimalik MSAL funktsioone kasutada näiteks sisse logitud kasutaja pääsuvõtmest loetud info nägemiseks - näiteks nime, kasutajagruppide jmt. Eesrakendusest päringute loomisel tagarakendusele lisatakse HTTP päisesse JWT (*JSON Web Token*) ligipääsuvõti.

Tagarakenduses luuakse API päringute autentimiseks vahetarkvara kiht (*Middleware*), mis kontrollib päringute päises oleva pääsuvõtme kehtivust enne kontrollerisse saatmist. JWT pääsukoodi dekodeerimisel võetakse appi Firebase/php-jwt teek. Võtme dekodeerimisel kasutatakse Azure AD poolt pakutavat avalikku võtit, millega tagatakse ka võtme verifitseerimine.

Kasutajate autoriseerimine toimub läbi Azure AD kasutajagruppide. Azure portaalis on võimalik kasutajatele lisada gruppe. Autentimisel lisatakse grupid pääsuvõtme sisse. Nii klientrakenduses kui tagarakenduses kontrollitakse vajadusel gruppide olemasolu.

Tagarakenduses luuakse ka kontroller ning vastavad otspunktid kasutajate sisselogimise registreerimiseks ning -nimekirja kuvamiseks, kasutaja telefoninumbrite lisamiseks, muutmiseks, kustutamiseks ja lugemiseks. Tellimuse kinnitamise funktsionaalsust täiendatakse kasutaja andmete salvestamisega tellimuse andmete juurde.

Eesrakenduses luuakse vaated tellimuste ajaloo vaatamiseks, telefoninumbri haldamiseks ning tellimuse kinnitamise vormil muudetakse varasem käsitsi telefoninumbri sisestamise lahter rippmenüüst valitavaks.

Neljanda arendustsükli käigus lisatakse tagarakendusele Messentest ettemaksukonto jäägi ning SMS hindade pärimise funktsionaalsus ning otspunktid klientrakendusest selle info saamiseks. Samuti tehakse vajalikud täienduses klientrakenduses Messentest saadava informatsiooni näitamiseks.

5.6 Viies iteratsioon – progressiivne veebirakendus

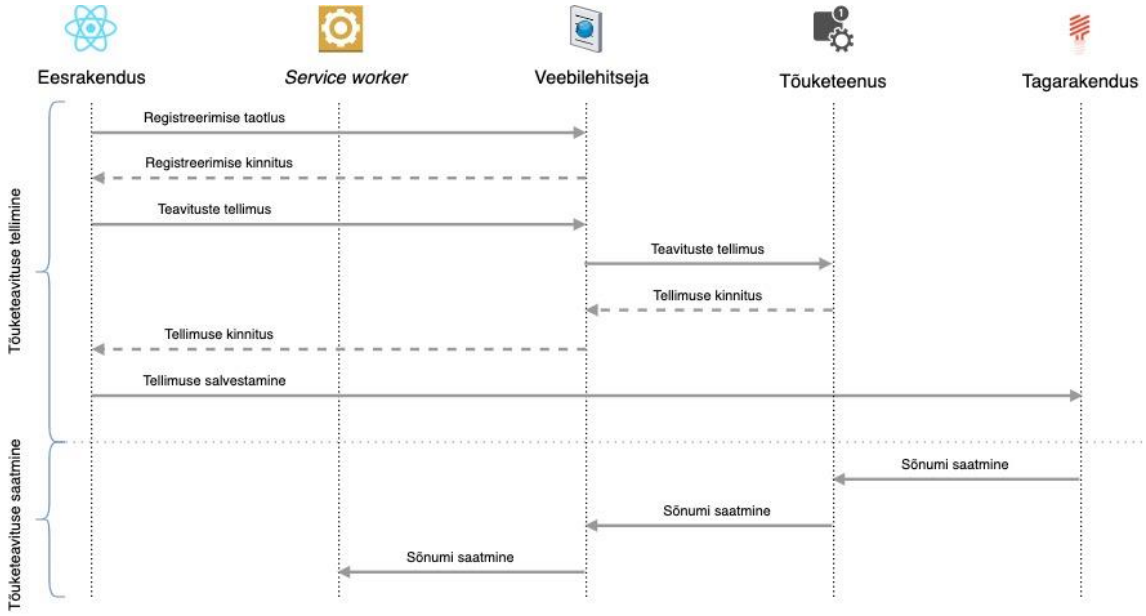
Viienda iteratsiooniga lisatakse rakendusele manifest.json fail ja *service-worker* koodijada. Funktsionaalsusest lisab autor rakendusele tõuketeavituste haldamise võimaluse ning tõuketeavituste saatmise.

Manifest on JSON formaadis fail, milles kirjeldatakse kuidas veebirakenduse sisu nähtavaks teha operatsioonisüsteemile. See on üks oluline komponent, et rakendust oleks võimalik paigaldada ning iseseisvalt käivitada. Manifestis kirjeldatakse ära rakenduse nimi, ikoon, teema värv, avamise viis jms. [45]

Service-worker on Javascript käsujada mis töötab veebirakendusest eraldi tööruumis ning puudub otsene ligipääs veebilehe objektidele. *Service-worker* on vaja rakenduse poolt esmakordsel kasutamisel alla laadida, paigaldada ning aktiveerida. Peale seda saab rakendus kasutada *service-worker* koodijada andmete ja failide vahemällu lugemiseks ning veebipäringute kinni püüdmiseks. Mõlemad on vajalikud mobiiliseadmes võimalikult sujuvaks kasutajakogemuseks ning näiteks võrguühenduse puudumisest tekkinud tõrgete silumiseks. [46]

Tõuketeavituste saatmiseks kasutab autor Push API teenust. Tõuketeavituste saatmiseks on vaja esmalt luua *service-worker* registreerimine tõuketeavituste tellimus (*push subscription*). Tellimuse käigus saadab veebilehitseja registreerimise andmed tõuketeenuse serverisse. Tellimuse kinnitamisel tagastab tõuketeenuse server pääsukoodid ja teavituste saatmiseks kasutatava veebiaadressi. Tagastatud andmed salvestab praamipiletite rakendus läbi tagarakenduse andmebaasi. Vaba praamipileti leidmisel loetakse salvestatud pääsukoodid ja veebiaadress andmebaasist ning kasutatakse tõuketeavituse saatmiseks tõuketeenusesse. Tõuketeenuse server edastab teate veebilehitsejale ning see omakorda *service-worker*`ile [47]. Teavituste tellimise ja saatmise protseduur on toodud joonisel 8.

Tõuketeenuse server on veebilehitseja poolt kasutatav rakendusliides. Iga veebilehitseja võib kasutada erinevat tõuketeenuse serverit, kuid nende liides on standardiseeritud ning seetõttu on nende kasutamine võimalik kõikides lehitsejates ühte moodi. [48]



Joonis 8. Push API teenuse kasutamise skeem [49].

Tõuketeavituste tellimise jaoks teeb autor muudatused klientrakenduses, kus „Minu konto“ lehel telefoninumbrite juurde lisatakse ka kõik teavituste saamiseks registreeritud veebilehitsejate nimekiri. Samuti võimaldatakse kasutajal tellimusi kustutada ning lisatakse nupp lehitseja lisamiseks. Vastavate tegevuste jaoks loob autor ka tagarakenduse otspunktid.

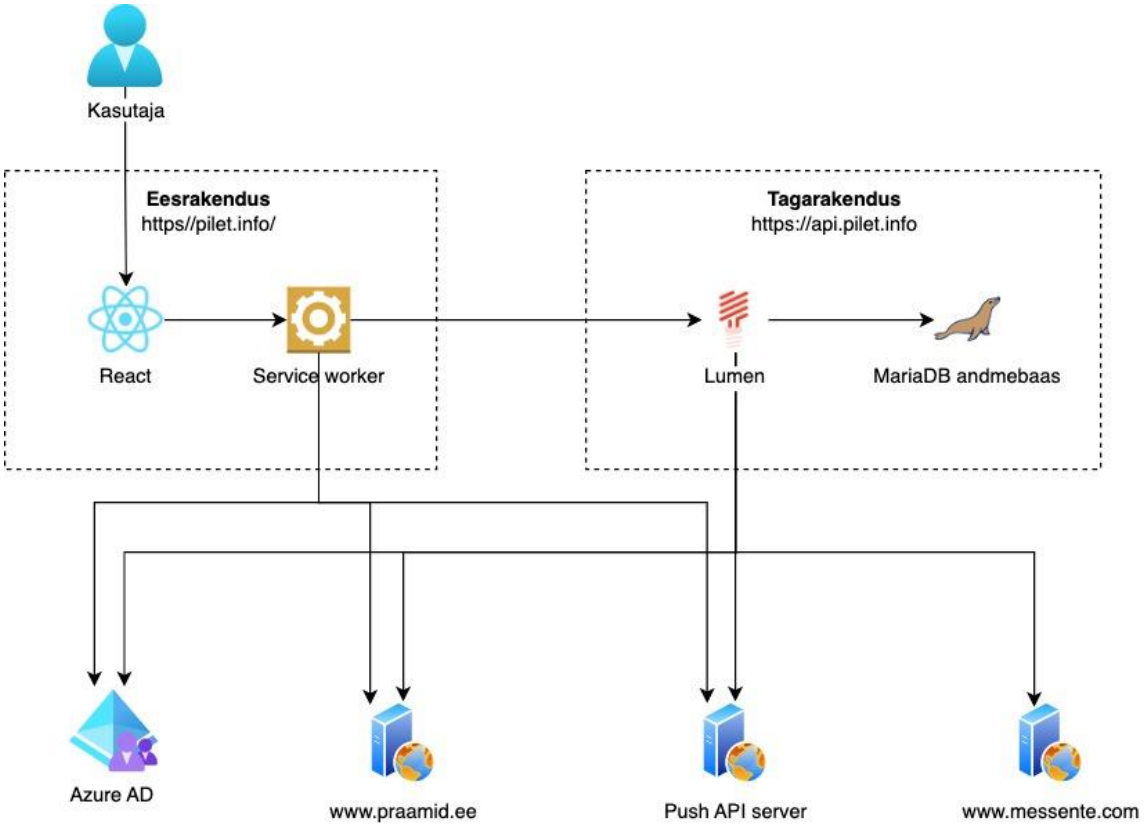
Autor testib tõuketeavituste registreerimist ning saatmist erinevate seadmete ning veebilehitsejatega. Näiteks Windows ning MacOS operatsioonisüsteemidega Safari, Mozilla Firefox ning Chrome veebilehitsejatega. Samuti Android operatsioonisüsteemiga Chrome veebilehitsejaga. Rakendust paigaldades töötab see mähituna seadme vaikimisi veebilehitsejas ning katsetamisel töötab vastavalt ootustele. Varasemalt on teada, et Apple mobiilsetel seadmetel Push API ei tööta, seepärast proovib autor IOS operatsioonisüsteemiga Safari ja Chrome veebilehitsejas, kas kõik eelmistes arendustsüklites loodud funktsionaalsused töötavad nii nagu peab.

6 Valmis rakenduse kirjeldus

Kuuendas peatükis kirjeldatakse valminud rakenduse arhitektuuri ning vaadeldakse rakenduse ees- ja tagarakendust. Samuti tuuakse välja testkasutajate tagasiside ning võimalused edasisteks arendusteks.

6.1 Rakenduse arhitektuur

Valminud rakendus on arhitektuurilt hajusa ülesehitusega. Hajussüsteemi puhul rakenduse komponendid suhtlevad omavahel üle võrgu ning kasutaja puutub kokku ainult eesrakendusega [50]. Eesrakendus loeb andmeid nii tagarakendusest kui ka väliste teenusepakkujate rakendusliidestest. Klientrakendus kasutab andmete vahemällu salvestamiseks *service-worker* koodijada, mistõttu on võimalik kasutajale silutud veateateid näidata ka võrguühenduseta olekus. Rakenduse komponendid ning nende vahelised seosed on toodud joonisel 9.

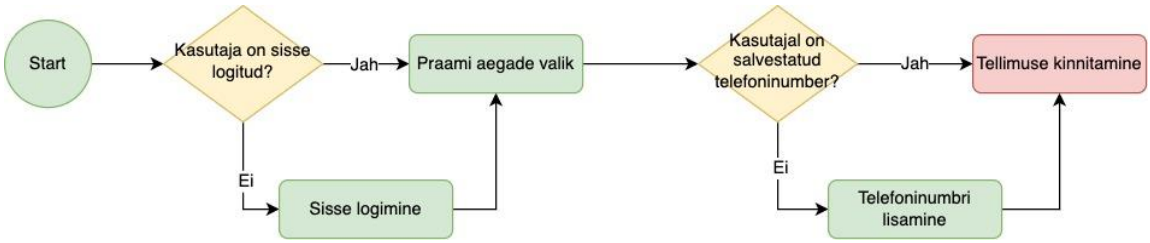


Joonis 9. Rakenduse arhitektuur.

6.2 Eesrakendus

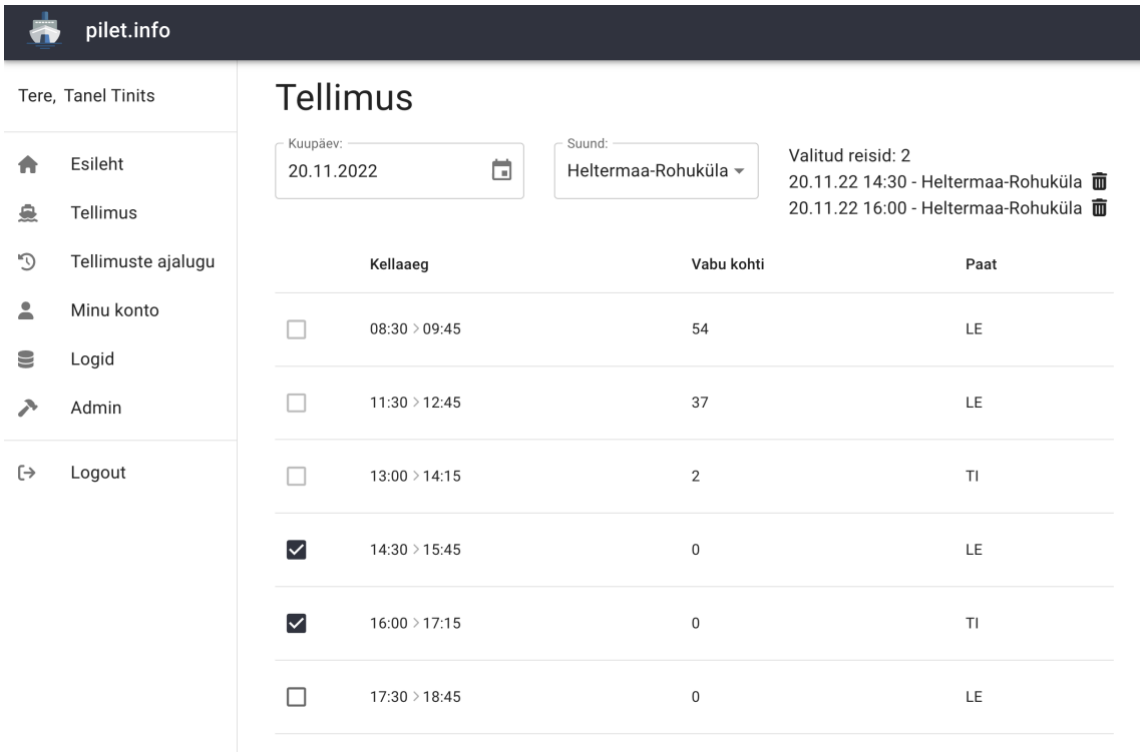
Valminud rakenduses on sisse logitud tavakasutajal võimalik liikuda nelja erineva lehe vahel – esileht, tellimuse vormistamise-, tellimuste ajaloo- ja kasutaja konto leht. Administraatori õigustega kasutajatel on lisaks veel leht logide vaatamiseks ning kasutajate- ja Messente andmete vaade. Sisse logimata kasutajatele rakenduse sisu ei kuvata.

Tellimuse vormistamiseks on vajalik sisse logitud kasutajal esmalt valida soovitud praamide ajad. Seejärel, kui kasutajal on juba eelnevalt salvestatud telefoninumber, on võimalik tellimis kohe kinnitada. Kui telefoninumbrit ei ole eelnevalt sisestatud, on see vaja enne tellimuse kinnitamist lisada „Minu konto“ lehel. Tellimuse vormistamise voog on kujutatud joonisel 10.



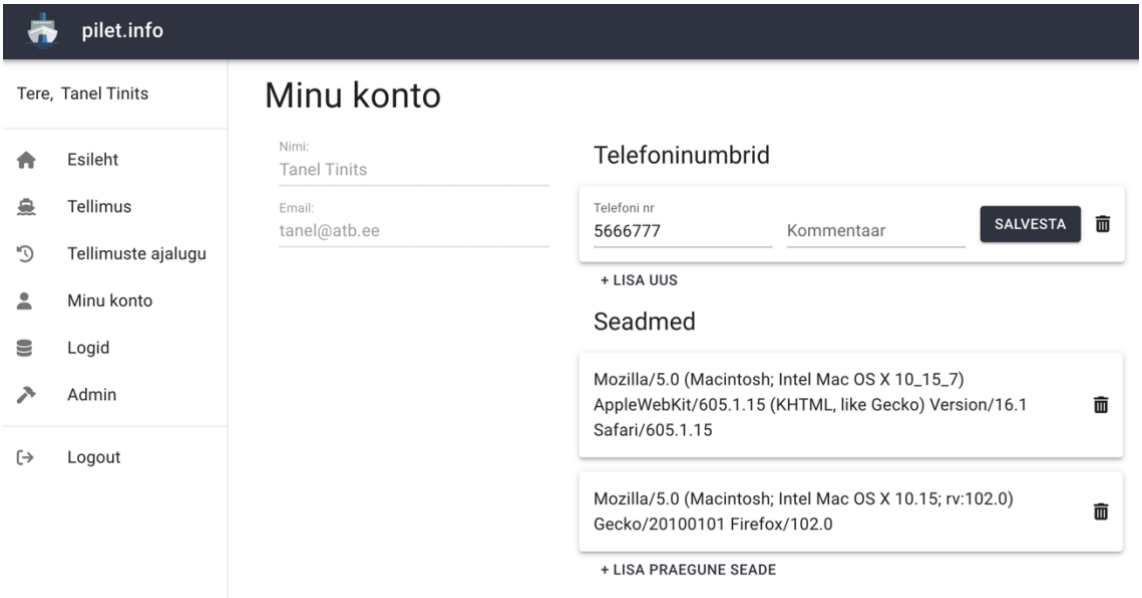
Joonis 10. Teavituse tellimise kasutajavoo diagramm.

Joonisel 11 kujutatud teavituste tellimise vormil näidatakse kõiki valitud praamide aegu. Ühele tellimusele saab valida ka erinevate päevade- ja suundade aegasid. Praamide ajad laetakse otse praamid.ee lehelt, nii on aegade nimekiri alati ajakohane.



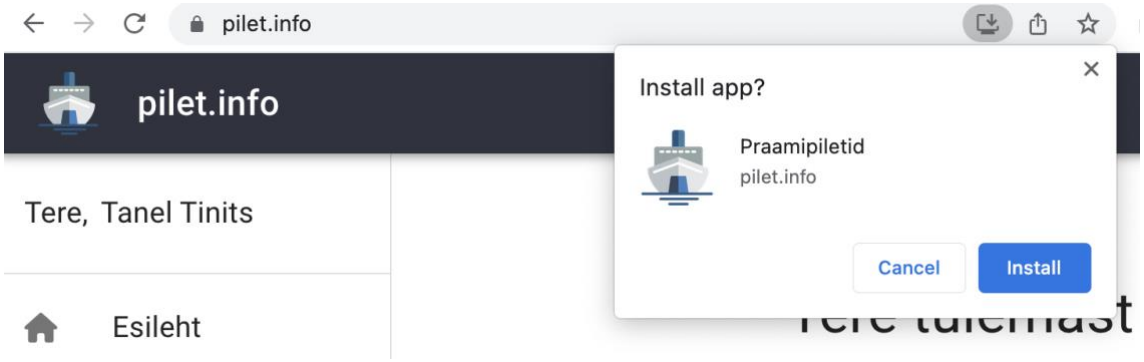
Joonis 11. Teavituse tellimisvorm.

„Minu konto“ lehel on võimalik hallata oma telefoninumbreid ning tõuketeavitusteks lisatud seadmeid. Kasutajale kuvatakse ka tema sisselogimiseks kasutatav e-posti aadress ning nimi. Muuta neid ei ole võimalik, kuna need laetakse Azure Active Directory identiteeditеenuse rakendusliidese kaudu. „Minu konto“ leht on kujutatud joonisel 12.



Joonis 12. Konto haldamise leht.

Veebilehte brauseris vaadates on võimalik see paigaldada nii mobiiliseadmes kui arvutis eraldi rakendusena, nagu on kujutatud joonisel 13. Peale paigaldamist on võimalik seda avada töölaualt vastava otsetee kaudu ning rakendus avaneb iseseisva rakendusena, mitte enam veebibrauseris.



Joonis 13. Rakenduse paigaldamine.

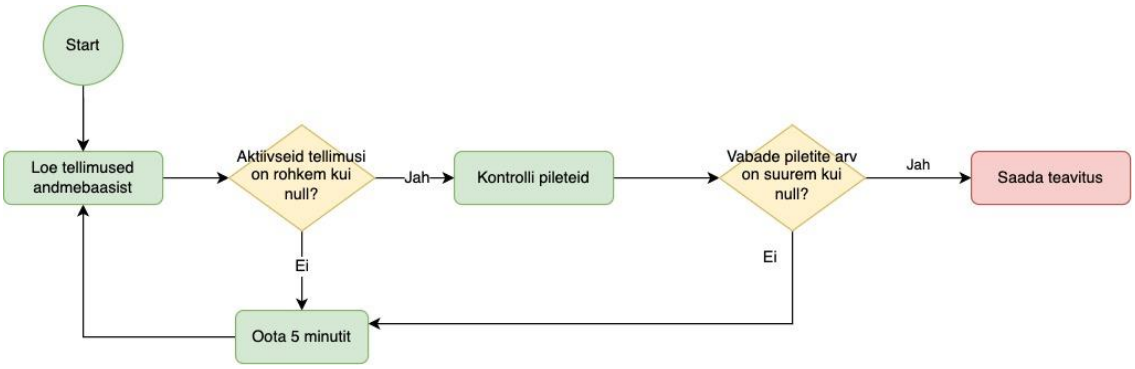
6.3 Tagarakendus

Valminud tagarakendus täidab kahte eesmärki. Esmalt on see rakendusliides eesrakenduse jaoks, mis võimaldab andmebaasist andmeid lugeda ning salvestada. Tabelis 5 on välja toodud erinevad võimalused, mille jaoks rakendusliidese otspunktid loodi.

Tagarakenduse teine eesmärk on perioodiline praamipiletite kontrollimine praamid.ee kodulehelt. Kontrollimiseks käivitatakse käsurealt koodijada, mis loeb andmebaasist kõik aktiivsed tellimused, seejärel teeb päringu praamid.ee kodulehele ning kontrollib, kas soovitud pileteid on vabanenud. Kui soovitud praamiaegadele leiti vabu pileteid saadetakse kasutajale teavitus. Kui aktiivseid tellimusi ei leitud või soovitud praamiaegadele ei leitud vabu pileteid, ootab koodijada viis minutit ning alustab seejärel tööd algusest. Perioodilise piletite kontrollimise voog on toodud joonisel 14.

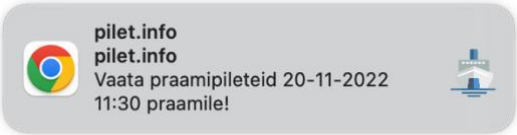
Tabel 5. Rakendusliidese otspunktid.

Ressurss	**HTTP meetod**	**Kirjeldus**
/accounts/login	POST	Salvestab sisselogimise info
/accounts/phones	GET	Tagastab kasutaja telefoninumbrid
/accounts/phones	POST	Lisab kasutajale uue telefoninumbri
/accounts/phones/{id}	PUT	Salvestab telefoninumbri
/accounts/phones/{id}	DELETE	Kustutab telefoninumbri
/accounts/subscriptions	GET	Tagastab kasutaja tõuketeavituste registreerimise informatsiooni
/accounts/subscriptions	POST	Lisab tõuketeavituste registreerimise
/accounts/subscriptions/{id}	DELETE	Kustutab tõuketeavituste registreerimise
/admin/balance	GET	Tagastab Messente kontojäägi
/admin/users	GET	Tagastab kasutajate nimekirja
/dlr	POST	Salvestab sõnumi saatmisraporti
/logs	GET	Tagastab logide nimekirja
/trips	GET	Tagastab tellitud reiside nimekirja
/trips	POST	Lisab uue tellimuse
/trips/{id}/redo	POST	Lisab olemas olevast tellimusest koopia
/trips/{id}	DELETE	Tühistab tellimuse



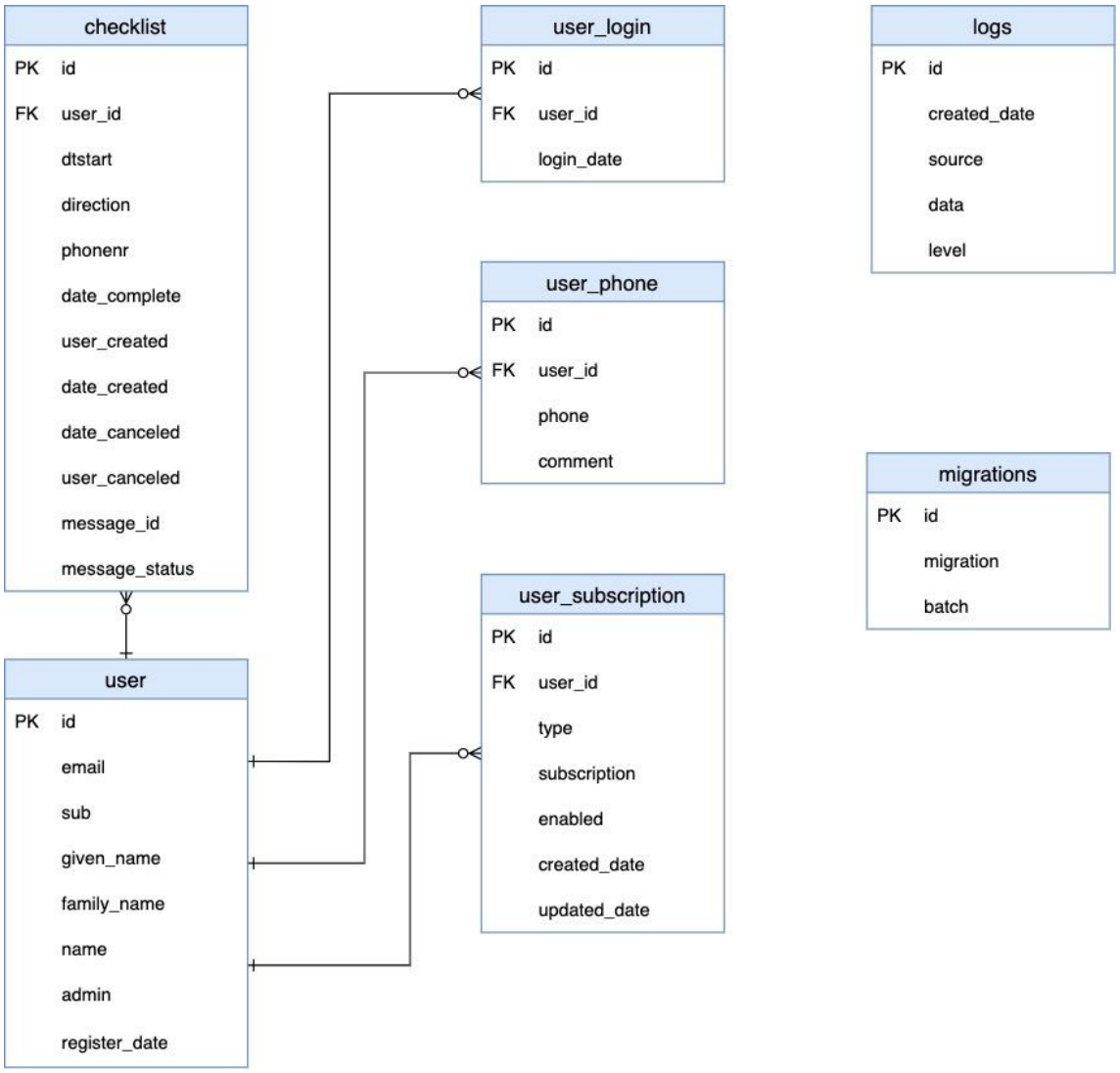
Joonis 14. Koodijada voodiagramm.

Tagarakenduse poolt kasutajale saadetud SMS ja tõuketeavitustes näidatakse ära praami kuupäev ja kellaaeg, millele saab piletit ostma minna. Tõuketeavituse eelvaade on joonisel 15.



Joonis 15. Tõuketeavitus.

Joonisel 16 kujutatud andmebaasiskeem koosneb kokku seitsmest tabelist. Migratsioonide tabel on loodud Lumen raamistiku poolt ning selles hoitakse andmebaasimigratsioonide ajalugu. Logide tabel „*logs*“ sisaldab vigade infot, kui näiteks mingil põhjusel ei ole õnnestunud praamipiletite aegu kontrollida. Rakenduse keskne tabel on „*user*“ ehk kasutaja tabel. Kasutajaga on seotud „*checklist*“, „*user_phone*“, „*user_subcsription*“ ning „*user_login*“ tabelid ehk vastavalt kasutaja tellimuste, telefoninumbrite, tõuketeavituste tellimuste ning sisselogimiste ajaloo tabelid.



Joonis 16. Lihtsustatud olemi-suhte diagramm.

6.4 Testimine ja kasutajate tagasiside

Rakenduse valmimise perioodil kasutasid rakendust viis testkasutajat. Iga valminud iteratsiooni lõpus jagas autor kasutajatele uuenduste ja muudatuste kohta informatsiooni. Kasutajad andsid töö autorile tagasisidet nii rakenduse kasutajakogemuse, leitud vigade kui ka õnnestunud või mitteõnnestunud praamipiletite soetamise kohta.

Üheks veaks, mille tuvastasid testkasutajad, oli peale autentimissessiooni lõppemist sisselogimise ebaõnnestumine Safari veebilehitsejaga. Probleem tekkis, kui autor oli uuendanud MSAL teeki, kuid manuaalse testimise käigus ei teadnud täpset situatsiooni läbi proovida. Probleem sai lahendatud teegi vanemale versioonile tagasi minnes.

Kasutajate tagasisidest tuli veel välja, et mõnedel kordadel kui kasutajale tuli teavitus vabanenud pileti kohta ning seejärel asus kasutaja piletit ostma oli see juba välja müüdud. Autor peab üheks põhjuseks piletite kontrollimise viieminutilist perioodi. Lühendades seda kontrollimiste vahelist perioodi näiteks kahele minutile oleks selliseid eksitavaid teavitusi vähem, kuid koormus praamid.ee kodulehele oleks suurem. Arutelu käigus kasutajate otsustati kontrollimise periood jätta viie minuti peale ning vajadusel saab eesrakenduses tellimuse uuesti luua.

Üldiselt on kasutajate tagasiside positiivne ning rakendus on hõlbustanud praamipiletite soetamist isegi tipptundideks ka lühikest aega reise ette plaanerides.

6.5 Võimalused edasiseks arenduseks

Edasise arendusena võiks lisada sotsiaalmeediakontoga sisselogimise – näiteks Facebook, Google vms. Selliselt oleks uutel kasutajatel konto loomine mugavam ning ei vajaks administraatori poolset tööd. Rakenduse seisukohast tähendab see tagarakendusele täiendavate veebivõtmete valideerimise lisamist. Eesrakenduses tuleb välja vahetada Microsoft Azure AD-ga autentimist võimaldav MSAL teek mõne universaalsema vastu. Samuti on vaja täiendada kasutajate haldamise vaadet ning võimaldada kasutajate ligipääsude lukustamine.

Laiemale üldsusele rakenduse kättesaadavaks tehes on võimalik lisada ka tasulise kasutamise võimalus. Näiteks võimaldada SMS teavitust ainult tasulise teenusena või lisada tasuta teenuse kasutajate teavitusele mõningane viivitus teavituse saatmisel.

Edasiste arendustena on võimalik laiendada ka rakenduste spetsiifikat – näiteks võib hakata kontrollima ka lennupiletite saadavust või hindasid ning teavitama vastavalt kasutaja ettemääratud tingimustele. Enne selliste laienduste tegemist tuleb kindlasti uurida turu nõudlust ning võimalusi vastavate kontrollide loomiseks.

7 Kokkuvõte

Käesoleva lõputöö eesmärk oli luua rakendus, mis võimaldab teavitada kasutajat praamid.ee lehel soovitud kellaaegadel välja müüdud Hiiumaa- ja Saaremaa praamipiletitest juhul, kui need vabanevad. Selle eesmärgi täitmiseks loodi progressiivne veebirakendus, kus on võimalik tellida teavitus valitud praamiajale SMS ja tõuketeavitusena.

Rakenduse veebimajutuseks valis autor zone.ee keskkonna ning seeõttu oli kasutatavate tehnoloogiate valik mõnevõrra piiratud. Analüüsi tulemusena valis autor lahenduse tagarakenduse loomiseks PHP keele ja Laravel Lumen raamistikku, eesrakenduse jaoks Typescript ning React raamistiku. Andmebaasina võeti kasutusele MariaDB. Autentimise ja autoriseerimise jaoks kasutati Azure Active Directory identiteediplatvormi.

Rakenduse loomine toimus agiilsel meetodil iteratiivsete tsüklitega. Esimeses arendustsüklis loodi lihtne käsujada kontseptsiooni tõestamiseks, mille abil prooviti, et kas sellise teavituse abil on üldse võimalik pileteid soetada. Teise iteratsiooniga võeti lihtsa käsujada asemel kasutusele Lumen raamistik. Kolmanda iteratsiooni käigus loodi lihtne ühe kasutaja jaoks mõeldud eesrakendus. Neljanda arendustsükliga tõsteti rakenduse turvalisust lisades autentimiseks ja autoriseerimiseks liidestuse Azure Active Directory identiteediplatvormiga. Viienda iteratsiooniga lisas autor tõuketeavituste saatmise ning kõik vajaliku, et rakendus vastaks progressiivse veebirakenduse tunnustele.

Töö käigus loodud rakendus sai valmis ning testkasutajad on saanud teavituste abiga pileteid osta. Valminud lahendus täitis seatud eesmärgi, kuid edasiste arendustega on võimalik rakenduse spetsiifikat laiendada. Näiteks lisades lennupiletite kontrollimise ning teavituste saatmise funktsionaalsuse.

Kasutatud kirjandus

[1] P. Saccomani, „Native, Web or Hybrid Apps? What's The Difference?“, *MobiLoud,* 1. oktoober 2018. https://www.mobiloud.com/blog/native-web-or-hybrid-apps (vaadatud 21. november 2022).

[2] „Kõigi aastate statistika | Praamid.ee“, 17. märts 2022. https://www.praamid.ee/online/statistics/xlsx?lang=et (vaadatud 2. jaanuar 2023).

[3] „Aruandlus | Praamid.ee“, 17. märts 2022. https://www.praamid.ee/aruandlus-2/ (vaadatud 29. oktoober 2022).

[4] „Korduma kippuvad küsimused | Praamid“, 18. mai 2021. https://www.praamid.ee/korduma-kippuvad-kusimused/ (vaadatud 29. oktoober 2022).

[5] ERR, „Hiiumaal oli praamile jälle mitmetunnine järjekord“, *ERR,* 1. august 2021. https://www.err.ee/1608294831/hiiumaal-oli-praamile-jalle-mitmetunnine-jarjekord (vaadatud 29. oktoober 2022).

[6] „App Service Pricing | Microsoft Azure“. https://azure.microsoft.com/en-us/pricing/details/app-service/windows/ (vaadatud 3. jaanuar 2023).

[7] „Pricing Overview | DigitalOcean“. https://www.digitalocean.com/pricing (vaadatud 3. jaanuar 2023).

[8] „Virtuaalserverite detailne võrdlus“, *Zone.ee*. https://www.zone.ee/et/virtuaalserver/vordlus/ (vaadatud 29. oktoober 2022).

[9] „Meiega on MNT sõidueksami järjekord paar päeva. Ära oota 3 kuud. | Eksamile“, *Eksamile - Kiirelt transpordiameti sõidueksamile!* https://eksamile.ee/telli-teenus (vaadatud 15. detsember 2022).

[10] „Toote saadavus“. https://www.reserved.com/ee/et/help-product-availability (vaadatud 15. detsember 2022).

[11] V. Kaushal, „Movie Ticket Availability Notifier“. 26. jaanuar 2021. Vaadatud: 15. detsember 2022. [Online]. Available at: https://github.com/kaushalvivek/Ticket-Availability-Notifier

[12] admin, „What are the advantages and disadvantages of web applications?“, *iTrobes - Software Consulting | Digital Transformation Services*, 7. september 2021. https://www.itrobes.com/what-are-the-advantages-and-disadvantages-of-web-applications/ (vaadatud 4. jaanuar 2023).

[13] „What Is a Progressive Web Application?“, *Codecademy News*, 17. september 2021. https://www.codecademy.com/resources/blog/what-is-a-progressive-web-application/ (vaadatud 27. november 2022).

[14] J. Tolentino, „SMS vs. Push vs. Email: When Should You Use Which?“, *TNW | Future-Of-Communications,* 9. veebruar 2015. https://thenextweb.com/news/sms-vs-push-vs-email (vaadatud 28. november 2022).

[15] „What is SMS, and How is it Different from Text Messages?“, *TextMagic*. https://www.textmagic.com/blog/what-is-sms-and-how-is-it-different-from-text-messages/ (vaadatud 28. november 2022).

[16] M. C. Ltd, „SMS Pricing - Calculate Your Business Messaging Costs with Messente“. https://messente.com/pricing (vaadatud 3. jaanuar 2023).

[17] „SMS Pricing in Estonia for Text Messaging“, *Twilio*. https://www.twilio.com/sms/pricing (vaadatud 3. jaanuar 2023).
[18] AppMySite, „Mobile App Push Notifications - A complete guide“, *AppMySite*, 29. september 2020. https://www.appmysite.com/blog/mobile-app-push-notifications-guide/ (vaadatud 19. detsember 2022).
[19] „Push API | Can I use... Support tables for HTML5, CSS3, etc“. https://caniuse.com/push-api (vaadatud 19. detsember 2022).
[20] Pusher, „Pusher Beams Docs | Build web notifications for Safari users“. https://pusher.com/docs/beams/getting-started/web/configure-safari/ (vaadatud 19. detsember 2022).
[21] A. Business, „Chapter 10: MoSCoW Prioritisation“. https://www.agilebusiness.org/dsdm-project-framework/moscow-prioririsation.html (vaadatud 19. detsember 2022).
[22] „PHP Market Share in 2022“, *Kinsta*®. https://kinsta.com/php-market-share/ (vaadatud 6. november 2022).
[23] H. McMurdy, „A Brief History of JavaScript“, *Medium*, 11. september 2020. https://javascript.plainenglish.io/a-brief-history-of-javascript-9289a4d344d2 (vaadatud 29. oktoober 2022).
[24] P. Marvet, „Node.js veebirakendused nüüd Zone virtuaalserveris - kauba peale PM2, mod_proxy ja portide suunamine“, *Zone.ee Blogi*, 22. juuli 2016. https://www.zone.ee/blogi/2016/07/22/node-js-veebirakendus-websocket/ (vaadatud 29. oktoober 2022).
[25] „Frequently Asked Questions (FAQ) - The Go Programming Language“. https://go.dev/doc/faq (vaadatud 29. oktoober 2022).
[26] javinpaul, „Is Golang worth learning in 2022? Why should you learn Go Programming Language?“, *Javarevisited*, 21. aprill 2022. https://medium.com/javarevisited/what-is-go-or-golang-programming-language-why-learn-go-in-2020-1cbf0afc71db (vaadatud 29. oktoober 2022).
[27] „Golang paigaldamine“, *help.zone.eu*. https://help.zone.eu/kb/golang-paigaldamine/ (vaadatud 28. november 2022).
[28] „Stack Overflow Developer Survey 2020“, *Stack Overflow*. https://insights.stackoverflow.com/survey/2020/?utm_source=social-share&utm_medium=social&utm_campaign=dev-survey-2020 (vaadatud 28. november 2022).
[29] „The Most Popular PHP Frameworks to Use in 2022“, *Kinsta*®, 28. september 2020. https://kinsta.com/blog/php-frameworks/ (vaadatud 6. november 2022).
[30] R. Davis, „Top 7 Best PHP Frameworks for REST APIs (2021) | RapidAPI“, *The Last Call - RapidAPI Blog*, 6. mai 2020. https://rapidapi.com/blog/php-api-frameworks/ (vaadatud 6. november 2022).
[31] „Welcome to Guzzle — Guzzle documentation“. https://guzzle3.readthedocs.io/getting-started/overview.html (vaadatud 6. november 2022).
[32] Elena, „Lumen - Diving into the Stunningly Fast Micro-Framework by Laravel“, *FastComet Blog*, 20. juuli 2017. https://www.fastcomet.com/blog/lumen-micro-framework-by-laravel (vaadatud 6. november 2022).
[33] „Slim 4 Documentation“, *Slim Framework*. https://www.slimframework.com/docs/v4/ (vaadatud 6. november 2022).
[34] D. Spínola, „TypeScript is a superset of JavaScript, meaning it’s a layer around JS with more methods and that…“, *Medium*, 29. juuli 2017. https://medium.com/@daspinola/typescript-is-a-superset-of-javascript-meaning-its-

a-layer-around-js-with-more-methods-and-that-46bbc9368be1 (vaadatud 27. november 2022).
[35] „The Best JS Frameworks for Front End“. https://rubygarage.org/blog/best-javascript-frameworks-for-front-end (vaadatud 20. detsember 2022).
[36] S. Martin, „Angular vs React vs Vue.js: Which is the Best Choice for 2022“, *plainenglish.io/blog/angular-vs-react-vs-vue-js-which-is-the-best-choice-for-2022*, 3. veebruar 2022. https://plainenglish.io/blog/angular-vs-react-vs-vue-js-which-is-the-best-choice-for-2022 (vaadatud 30. november 2022).
[37] „What is CI/CD?“ https://www.redhat.com/en/topics/devops/what-is-ci-cd (vaadatud 12. november 2022).
[38] „Azure DevOps Services | Microsoft Azure“. https://azure.microsoft.com/en-us/products/devops/ (vaadatud 20. detsember 2022).
[39] Atlassian, „Bitbucket Overview“, *Bitbucket*. https://bitbucket.org/product/guides/getting-started/overview (vaadatud 20. detsember 2022).
[40] „Pricing · Plans for every developer“, *GitHub*. https://github.com/pricing (vaadatud 20. detsember 2022).
[41] Atlassian, „Bitbucket - Pricing“, *Atlassian*. https://bitbucket.org/product/pricing (vaadatud 20. detsember 2022).
[42] M. Akash, „Laravel Eloquent Vs DB Query Builder [Performance and other statistics]“. https://devsenv.com/tutorials/laravel-eloquent-vs-db-query-builder-performance-and-other-statistics (vaadatud 12. november 2022).
[43] davidmu1, „Microsoft identity platform and OAuth 2.0 authorization code flow - Microsoft Entra“. https://learn.microsoft.com/en-us/azure/active-directory/develop/v2-oauth2-auth-code-flow (vaadatud 4. jaanuar 2023).
[44] j-mantu, „Tutorial: Create a React single-page app that uses auth code flow - Microsoft Entra“. https://learn.microsoft.com/en-us/azure/active-directory/develop/tutorial-v2-react (vaadatud 19. november 2022).
[45] „Web app manifest“, *web.dev*. https://web.dev/learn/pwa/web-app-manifest/ (vaadatud 16. detsember 2022).
[46] „Service Worker API - Web APIs | MDN“. https://developer.mozilla.org/en-US/docs/Web/API/Service_Worker_API (vaadatud 16. detsember 2022).
[47] „Push API - Web APIs | MDN“. https://developer.mozilla.org/en-US/docs/Web/API/Push_API (vaadatud 16. detsember 2022).
[48] „How push works“, *web.dev*. https://web.dev/push-notifications-how-push-works/ (vaadatud 16. detsember 2022).
[49] „Push API“. https://www.w3.org/TR/push-api/ (vaadatud 4. jaanuar 2023).
[50] „What is a Distributed System?“, *StackPath*. https://www.stackpath.com/edge-academy/what-is-a-distributed-system/ (vaadatud 4. jaanuar 2023).

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

Mina, Tanel Tinits

1. Annan Tallinna Tehnikaülikoolile tasuta loa (lihtlitsentsi) enda loodud teose „Progressiivne veebirakendus vabade praamipiletite leidmiseks“, mille juhendaja on Kristiina Hakk
   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.

05.01.2023

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.