TALLINNA TEHNIKAÜLIKOOL
Infotehnoloogia teaduskond
Tarkvarateaduse instituut

Ilmar Lõhmus 142285

PROGRESSIIVNE VEEBIRAKENDUS KUI ALTERNATIIV MOBIILIRAKENDUSTE ARENDAMISEKS

Bakalaureusetöö

Juhendaja: Inna Švartsman
Magistrikraad

Tallinn 2017

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: Ilmar Lõhmus

22.05.2017

Annotatsioon

Käesoleva bakalaureusetöö eesmärkideks on uurida, kas progressiivsed veebirakendused on oma hetkelises arengujärgus võimelised asendama mobiilirakenduste arendamisel standardiks olevaid põlisrakendusi ning luua näide progressiivsest veebirakendusest. Töö käigus uuritakse progressiivseid veebirakendusi realiseerivaid tehnoloogiaid ning võrreldakse lahenduse omadusi põlis-, hübriid- ning traditsiooniliste veebirakendustega.

Töö esimeseks tulemuseks leiti, et progressiivne veebirakendus on Androidi platvormil põlisrakenduste suhtes sobilikuks alternatiiviks ning omab positiivseid tagajärgi ka vajalikke tehnoloogiaid mitte teostavatel platvormidel. Töö teiseks tulemiks valmis progressiivse veebirakenduse omadusi ning autori poolt esitatud nõudeid realiseeriv eesmärkide seadmist ning täitmist abistav rakendus.

Lõputöö on kirjutatud eesti keeles ning sisaldab teksti 29 leheküljel, 7 peatükki, 12 joonist, 4 tabelit.

Abstract

Progressive web application as an alternative for the development of mobile applications

The aim of this bachelor's thesis is to identify whether progressive web applications in their current status are capable of replacing native applications as the standard of developing mobile applications and develop an example of a progressive web application. The author explores the technologies behind progressive web apps and compares its characteristics to native, hybrid, and traditional web applications.

The first result of this thesis is that progressive web applications are a suitable alternative to native applications on Android devices and have a positive impact on platforms that do not fully implement the necessary technologies. The second result is a goal planning progressive web application that puts into practice previously introduced characteristics and requirements established by the author.

The application is built with Angular and written in TypeScript. The application’s data is stored in IndexedDB and its design follows material design’s guidelines. The final app is tested with Karma, Protractor, and Jasmine. Its code is hosted on GitHub and deployed to Firebase servers using Travis CI.

The thesis is in Estonian and contains 29 pages of text, 7 chapters, 12 figures, 4 tables.

Lühendite ja mõistete sõnastik

AJAX	Asynchronous Javascript and XML, asünkroonne JavaScript ja XML
API	Application Programming Interface, rakendusliides
CSS	Cascading Style Sheets, kaskaadlaadistik
DOM	Document Object Model, dokumendiobjektide mudel
HTTPS	HyperText Transport Protocol over SSL, hüperteksti
JSON	Javascript Object Notation, avatud standardiga võti-väärtus struktuuril andmekirjeldusformaat
NoSQL	Mitte-relatsiooniline andmebaas
npm	Node.js Package Manager, Node.js pakettide haldusvahend
PWA	Progressive Web App, Progressiivne veebirakendus
SCSS	Sassy Cascading Style Sheets, Sassy kaskaadlaadistik
SEO	Search Engine Optimization, otsingumootori optimeerimine
SPA	Single Page Application, üheleherakendus
SQL	Structured Query Language, struktuurpäringukeel
URL	Uniform Resource Locator, internetiaadress

Sisukord

Jooniste loetelu

Tabelite loetelu

1 Sissejuhatus

2015. aastal Google'i töötajate poolt sõnastatud termin „progressiivne veebirakendus" on järgnevatel aastatel levinud veebiarendajate seas kulutulena. Kuigi suurimaks propageerijaks on olnud Google ise, siis on tekkinud suur hulk arendajaid, kes ootavad mõistest AJAX-iga (*Asynchronous Javascript and XML*) sarnast revolutsiooni mobiilirakenduste arendamisel.

Leidub ka skeptikuid, kes ei näe tehnoloogias midagi muud kui lähiaja trendide jätkumist JavaScripti igale poole surumise näol ning kõhklevad Google õilsates kavatsustes. On ju läbi otsingumootorite leitavate rakenduste ning massiivsete hulkade reklaamitulude võita kõige rohkem ainult neil.

Käesoleva bakalaureusetöö eesmärgiks on leida vastus küsimusele, kas progressiivne veebirakendus on lihtsalt hetkel moekas termin või peaksid kõik infotehnoloogia valdkonna esindajad selle mõistega kursis olema.

Bakalaureusetöö teoreetilises osas annab autor ülevaate varasematele põlisrakenduste alternatiividele ja progressiivsetele veebirakendustele. Sellele järgnevalt analüüsib autor tugevaid ja nõrku külgi võrreldes progressiivsete veebirakendustega ning annab hinnangu uue lahenduse praegusele kasulikkusele.

Töö praktilises osas kirjeldatakse esialgu valikut tehnoloogiatest, millega on võimalik progressiivseid veebirakendusi arendada ning sellele järgneb teoorias käsitletud tehnoloogiaid ja omadusi realiseeriv näidisrakenduse kirjeldus. Valminud rakendusele esitatakse nõuded, analüüsitakse progressiivse veebirakenduse omadustele vastamist ning tuuakse välja võimalikud edasiarenduse võimalused.

1.1 Taust ja probleem

Maailmas on üle 2,6 miljardi nutitelefoni kasutaja, kes on loonud suure turu kaasaskantavatele seadetele mõeldavate programmide jaoks [1]. Levinuim viis

mobiilirakenduste loomiseks on arendada Android ja iOS rakendused ning neid toetav veebileht.

Põlisrakenduse loomisel tuleb arvestada platvormist sõltuvate tehniliste aspektidega. Teistsuguste teadmiste ja kogemuste vajadus toob tihti endaga kaasa spetsialiseerunud meeskonnad, kes tegelevad ühe operatsioonisüsteemiga. Seega tähendab platvormist sõltuvate rakenduste loomine suuremat aja- ning rahakulu.

Ka turustamine on põlisrakenduste puhul keeruline. Google Play ja App Store poodides on tohutul hulgal rakendusi, mille seas on raske välja paista ja kasutajaid leida. Enamuse tähelepanust saavad juba populaarsed edetabelites olevad programmid, samas nõuavad mõlemad poed vahendustasu. Lisaks sellele, ei ole head moodust, kuidas kasutajad suusõnaliselt rakendust levitada saaks. Andes nime tuleb kasutada otsingut ning suurte hulkade seast võib üles leidmiseks ka sellest väheseks jääda.

On tõestatud, et iga samm, mis sa kaustajalt enne rakenduse põhifunktsiooni kasutamist nõuad, vähendab tarbijaskonda 20% võrra [2]. Sammude hulka kuuluvad siis näiteks rakenduse otsimine, sellele lubade andmine ja installeerimine, avamine, registreerimine ja sisse logimine. Kõik need etapid on ajakulu, mis peletavad kasutajaid.

2 Varasemad lahendused

Käesolevas peatükis antakse ülevaade põlisrakenduste alternatiividele, mis on eksisteerinud juba enne progressiivse veebirakenduse mõiste kasutuselevõttu.

2.1 Mobiilsed veebirakendused

Mobiilsed veebirakendused põhinevad standardsetel Interneti tehnoloogiatel nagu HTML, CSS ja JavaScript. Rakendused, mis on kohendatud mobiilile, laetakse alla serverist ning töötavad veebilehitsejates [3].

Tänu standardiseeritud keeltele võimaldab veebirakenduste arendus pakkuda ühist kasutajakogemust kõigile platvormidele. Arendajate jaoks tähendab see aga kiiremat arendus- ja kergemat hooldusprotsessi. Lisaks sellele puudub vajadus kõrgetasemelistele kogemustele ja teadmistele erinevate platvormide kohta.

Veebirakenduste suurimaks probleemiks on teatud funktsionaalsuste puudumine - seadme kogu riistvarale ning andmetele pole võimalik ligi pääseda [3]. Selline olukord on loodud peamiselt turvakaalutlustel, kuid see tähendab, et seega ei saa rakenduses kasutada ära näiteks mobiilis paiknevaid kontakte ja telefoninumbreid.

Veebirakendusi ei ole võimalik levitada läbi rakenduspoodide, kasutajad pääsevad sellele ligi läbi veebilehe. Rakendused ei nõua installeerimist ja uuendamist ning seega tagavad kohese kasutusvõimaluse. Installeerimisvõimaluse puudus tähendab aga seda, et enamasti rakendused vajavad kasutamiseks internetiühendust [3].

2.1.1 Application Cache

Application *cache* ehk AppCache on HTML5-ga välja tulnud tööriist, millega on võimalik veebirakendustel realiseerida võrguühenduseta töötamine. Tänu sellele mehhanismile vahemälu ära kasutades on võimalik tagada ka rakenduse kiiremat laadimist ja serveri väiksem koormamine [4].

Töötamise tagamiseks, tuleb rakendusele lisada manifesti atribuut, mis omakorda viitab vahemälu manifesti failile. Sellesse faili lisavad arendajad nimekirja vahemällu salvestamist vajavatest ressurssidest. Kui rakendust külastades AppCache eksisteerib, laetakse vajalikud failid mälust ilma võrku kasutamata. Seejärel kontrollib veebilehitseja, kas serveris on manifest uuendatud. Juhul kui on, laeb sirvija alla uue versiooni ning selles nimetatud ressursid [4].

Praeguseks hetkeks, on AppCache arendus peatatud ning see ei kuulu enam veebistandarditesse. Selle tööriisata asendamiseks soovitatakse kasutada uuemat ja paremat vahendit nimega *service worker* [4].

2.2 Hübriidrakendused

Hübriidrakendused kombineerivad veebi- ja põlisrakenduste parimad osad. Nende põhiline arendus tehakse kasutades veebitehnoloogiaid, mis siis ümbritsetakse riistvara realiseeriva koodiga. Sarnaselt veebirakendustega, töötavad ka hübriidrakendused läbi veebilehitseja, kuid neid peab installeerima kasutades rakenduste poodi [3].

Hübriidrakendustel on võimalus kasutada seadme võimalusi nagu kaamera ja kontaktid. Lisaks sellele on võimalik saata ka teateid, et kutsuda kasutajat rakendust taaskordselt kasutama [5].

Kuna erinevatele platvormidele arendades on võimalik kasutada suures osas sama koodi, siis võrreldes põlisrakendustega on tööhulk väiksem. Negatiivseks küljeks on hübriidide väiksem kiirus ning riistvara täieliku funktsionaalsuse mitte ära kasutamine. Näiteks on Androidi telefonidel eraldi tagasimineku nupp, iOS seadmetel realiseeritakse seda funktsiooni ekraanil.

3 Progressiivsed veebirakendused

Antud peatükis antakse ülevaade progressiivsest veebirakendusest kui mõistest ning seda realiseerivatest tehnoloogiatest.

3.1 Progressiivse veebirakenduse definitsioon

Progressiivsed veebirakendused on veebi- ja põlisrakenduste parimaid omadusi kombineeriv kasutajakogemus, mis on töökindel, kiire ja kütkestav. Need on veebitehnoloogiatel põhinevad rakendused, mis on avalehele paigaldatavad, ühendusest sõltumata töötavad ning mobiiliteavituste saatmist võimaldavad [6].

Google poolt kirja pandud progressiivsete veebirakenduste omadused on järgmised [7]:

- **Progressiivne.** Olenemata veebilehitseja valikust, töötavad rakendused koheselt kõikide kasutajate jaoks. Installerimine ei ole kohustuslik
- **Kohanduv.** Rakendus on seadmetundlik ehk disain skaleerub vastavalt töövahendile.
- **Sõltumatu ühendusest.** Rakendust on võimalik kasutada ka halva internetiühendusega või selle täielikul puudumisel.
- **Põlisrakendusega sarnane.** Rakendus tundub kasutusel nagu tavaline mobiilirakendus pakkudes sarnast navigatsiooni ja funktsionaalsust.
- **Pidevalt uuendatud.** Rakendus on alati viidud tema viimasele versioonile, puudub vajadus anda rakenduste poodides uuenduste jaoks eraldi heakskiit.
- **Turvaline.** Rakendust hostitakse läbi HTTPS-i (*HyperText Transport Protocol over SSL*), et ründajal ei oleks võimalik muuta rakenduselt tulenevat sisu.
- **Avastatav.** Rakendus on otsingumootorite abil leitav.
- **Kütkestav.** Kasutajat on võimalik läbi mobiiliteavituste kutsuma rakendust taaskordselt kasutama.
- **Installeeritav.** Rakendust on võimalik paigutada avaekraanile läbi veebilehitseja.
- **Lingitav.** Rakendust võimalik jagada kasutades saates URL-i (*Uniform Resource Locator*) ning võimalus lisada see järjehoidjatesse.

PWA-d (*Progressive Web App*) töötavad progressiivse parenduse põhimõttel, mis tähendab, et rakendust on võimalik kasutada kõigi erinevate brauserite kui ka seadetega. Veebilehtede arendamine selle põhimõtte järgi toimub hoides eraldi sisu-, esitlus- ja käitumiskihti. HTML dokumendis on eraldi lingitud CSS ja JavaScript failid, mis laetakse alla vastavalt kasutaja seadmele, pakkudes nii võimalikult head kogemust [8]. Analoogselt peavad töötama ka PWA-d, mis tähendab, et rakendus avaneb ka veebilehitsejates, mis kõiki tema tehnoloogiaid ei võimalda.

Lihtsustatult võib PWA mõiste defineerida kui rakendus, mis kasutab ära tänapäevaseid veebitehnoloogiaid ja mõisteid nagu *web app manifest*, *service worker* ja *application shell*. Järgnevates peatükkides annab autor ülevaate neist PWA alustaladeks olevatest mõistest.

3.2 Web App Manifest

Web App Manifest on JSON-i (*Javascript Object Notation*) põhine fail, milles hoitakse meta-andmeid veebirakenduse kohta. Salvestatud andmete järgi on võimalik luua kasutajakogemusi, mis on sarnased põlisrakendustega, sealhulgas installeerimine avaekraanile [9].

Meta-andmete hulgas on rakenduse nimi, kirjeldus ja URL, mida veebirakendust käivitades avatakse. Lisaks hoitakse ka linke ikoonidele ja värvidele, mida kasutatakse rakenduse avalehel kuvamisel ning laadimisel. Näide ühest progressiivse veebirakenduse manifestist on joonisel (vt Joonis 1).

```
{
    "name": "Web App Name",
    "short_name": "App Name",
    "description": "Description of Web App",
    "start_url": "/index.html",
    "display": "standalone",
    "orientation": "portrait",
    "theme_color": "red",
    "icons": [{
         "src": "./images/icons/icon-192x192.png",
         "type": "image/png",
         "sizes": "192x192"
    },],
}
```

Joonis 1. Näidis Web App Manifest

3.3 Service Worker

Service worker on veebilehitseja poolt täidetav käsujada, mis loob rakenduse, brauseri ja võrgu vahele puhverserveri. See on mõeldud vastavalt võrgu olemasolule navigatsiooni- ja ressursipäringute töötlemiseks ning ressursside vahemällu salvestamiseks [10].

Service worker töötab asünkroonselt rakendust peamiselt juhtivast koodist eraldi ning ei oma ligipääsu DOM-ile (*Document Object Model*). Seega tema töö ei tõkesta rakenduse funktsioneerimist [10].

Tehnoloogia elutsükkel koosneb kolmest etapist: allalaadimine, installeerimine ja aktiveerimine. Kui veebileht, mida esmakordselt külastatakse, omab *service workerit*, siis laetakse see alla ning pärast installeerimist aktiveeritakse koheselt. Edaspidi peab alla laadimist tegema iga 24 tunni jooksul, et takistada segavate käsujadade tööd. Uue versiooni olemasolul toimub installeerimine taustprotsessina ning aktiveerimine toimub alles siis, kui vana varianti vajavate lehekülgede laadimine on lõppenud [10].

Service workerid on hetkel täielikult rakendatud Chrome, Firefox ja Opera veebisirvijates ning on arenduses Microsoft Edge-s. Lootust on ka, et Safari hakkab tulevikus seda tehnoloogiat toetama [11].

Läbi *service workerite* on võimalik kaaperdada ühendusi ning mõjutada päringute vastuseid. Turvalisuse huvides saab *service workereid* seetõttu kasutada ainult läbi HTTPS-i [11].

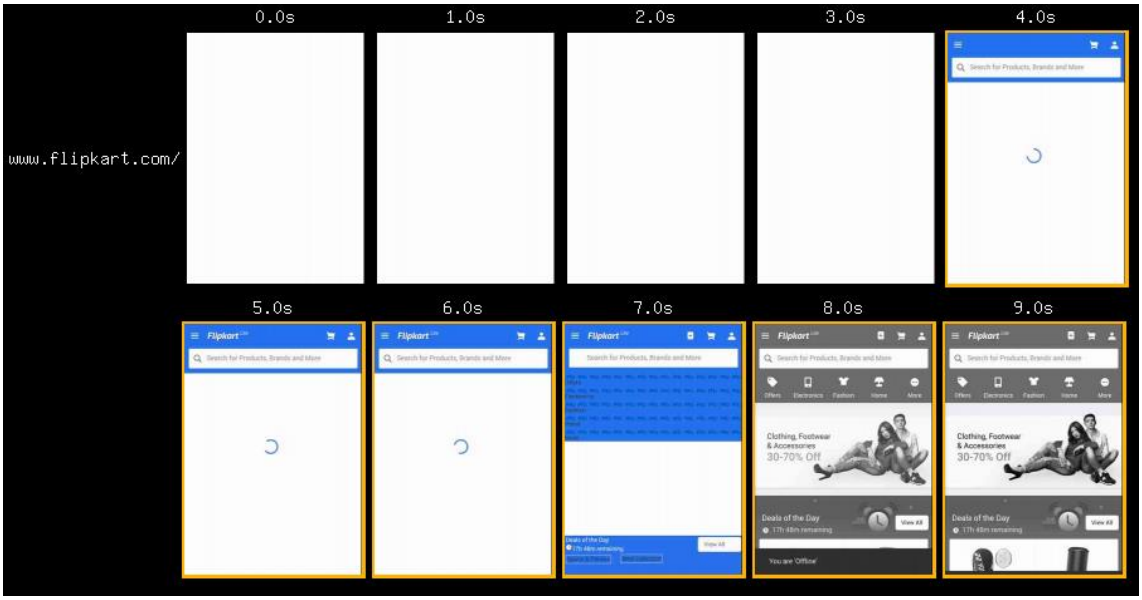
3.4 Application Shell

Application shell arhitektuuri kasutatakse põlisrakendustega sarnase väga kiire laadimise tagamiseks PWA-dele. See koosneb minimaalsest HTML, CSS ja JavaScript hulgast, mis on vajalik kasutajaliidese kuvamiseks. Arhitektuur sobib kõige paremini veebilehtedele ja -rakendustele, millel on muutuv sisu, kuid püsiv navigatsioon nagu näiteks SPA-del (*Single Page Application*) [12].

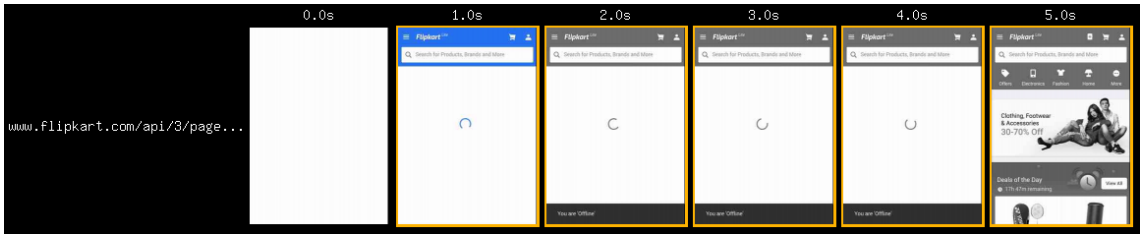
Salvestades *service workeriga* vajaliku info vahemällu, ei ole *application shelli* vaja igal külastusel uuesti laadida ning saab tagada rakenduse silmapilkse töö. Sealjuures on võimalik luua põlisrakendustele sarnane funktsionaalsus võrguga ühenduseta olukorras [12].

Flipkart Lite on India veebipoe Flipkart mobiilirakendus, mis on üks esimesi PWA-de edulugusid. Ettevõtte IT meeskond leidis, et kui brauseril on vajalikud tehnoloogiad võimaldatud, siis võib rakendus omada põlisrakendusega sarnast jõudlust [13].

Kasutades veebilehte WebPageTest.org on võimalik teha teste päris seadete peal. 3G võrgus Nexus 5 peal rakenduse laadimist 3G võrgus esmakordse (vt Joonis 2) ja korduva külastuse (vt Joonis 3) võrdlemisel on võimalik näha *service workeri* ja *application shelli* poolt tagatud rakenduse laadimise kiiruse paranemist.



Joonis 2. Flipkarti külastus 3G võrgus esmakordselt



Joonis 3. Flipkarti külastus 3G võrgus korduval külastusel

Esmasel külastusel tunneb kasutaja testi andmetel, et peamine sisu on laetud 6,6 sekundiga ning rakendus muutub visuaalselt täielikuks 8,9 sekundiga. Korduvatel külastusel on peamine sisu laetud 3,9 sekundiga ja visuaalselt täielik 4,7 sekundiga.

4 Lahenduste võrdlus

Järgnevates alampeatükkides võrdleb autor PWA-sid varasemate lahendustega ning analüüsib erinevate võimaluste tugevaid ja nõrku külgi.

4.1 Progressiivse veebirakenduse võrdlus traditsiooniliste veebirakendustega

Antud peatükis võrdleb autor progressiivseid veebirakendusi traditsiooniliste veebirakendustega või veebilehtedega, ehk rakendustega, mis PWA-de uudseid tehnoloogiaid ei kasuta. Kriteeriumid, mille järgi võrdlus toimub on järgmised:

- **Kasutatavad tehnoloogiad.** Arendamiseks vajalikud programmeerimiskeeled.
- **Võrguühenduseta töötamine.** Rakendus on võimeline töötama kas võrguühenduseta või piiratud võrguühendusega.
- **Turvalisus.** Võrreldakse kas turvalisus on tagatud läbi HTTPS protokolli kasutuse.
- **Avaekraanile lisatav.** Rakendust on võimalik lisada seadme avaekraanile, kust on võimalik rakendusele ligi pääseda ühe kliki läbi.
- **Teated.** Rakendus on võimeline kasutajale saatma autonoomselt teateid.
- **Kiired uuendused.** Kasutaja saab kiiresti kätte uusima versiooni ilma et peaks ise midagi tegema.
- **Ligipääs failisüsteemile.** Kasutajal on võimalik pääseda ligi seadmes leiduvatele andmetele ning neid kasutada.
- **Toetavad brauserid.** Rakenduse täielikku funktsionaalsust realiseerivad veebilehitsejad.

Vastavalt kriteeriumitele tehtud võrdlus on esitatud järgnevas tabelis (vt Tabel 1).

Tabel 1. PWA võrdlus traditsioonilise veebirakendusega

	Traditsiooniline veebirakendus	**Progressiivne veebirakendus**
Kasutatavad tehnoloogiad	Standardsed veebitehnoloogiad: HTML, CSS, JavaScript	Standardsed veebitehnoloogiad: HTML, CSS, JavaScript
Võrguühenduseta töötamine	Puudub	Võimalik tagada
Turvalisus	HTTPS kasutus võimalik	HTTPS kasutus kohustuslik
Avaekraanile lisatav	Ei	Jah
Teated	Ei	Jah
Ligipääs failisüsteemile	Ei	Jah
Kiired uuendused	Jah	Jah
Toetavad brauserid	Kõik	Firefox, Chrome, Samsung Internet, Opera

Ressursside vahemällu salvestamise ning kliendipoolsete andmebaaside kasutuselevõtt võimaldab PWA-del tagada võrguvaba töö. Tänu sellele ei nõua rakendus kasutajatel nii palju andemahtu ning rakenduse jõudlus suureneb. E-pood Konga suutis tänu PWA kasutuselevõtuga vähendada andmemahu hulka esimese tehingu korral võrreldes eelneva veebilahendusega 84% võrra [14]. Selline jõudluse ja kiiruse võit võib eriti oluline olla arengumaades, kus ühenduste kiirused on madalad ning sideteenused on kallid.

Kasutajatele on alati olnud turvalisus oluline ja nüüd kui Google agressiivselt HTTPS mitterealiseerivaid veebilehti ebaturvaliseks märgib ning otsingutulemustes tahaplaanile lükkab, siis PWA nõue HTTPS-i kasutada eraldi stiimulit ei vaja. Boonuseks on veel siiski uuema protokolli suurem kiirus.

Progressiivset veebirakendust ei pea installeerima, aga seadme avaekraanile lisamine võimaldab kasutajal rakendusele kiiremini ligi pääseda ning tagab nende suurema taaskasutuse. Viimasele nähtusele omab mõju ka kasutajale teadete saatmine. Ettevõte eXtra Electronics suutis tänu sõnumite saatmisele suurendada oma müükide arvu 100% võrra [15].

Hetkel veel ei ole võimalik progressiivset veebirakendust iOS-ile luua, sest Apple nõuab brauseritelt WebKit mootori kasutamist, mis aga ei toeta *service workereid*, mobiiliteateid

ja avaekraanile installeerimist. Siiski omab hästi loodud rakendus ka sellel platvormil positiivset mõju. Veebipood AliExpress avastas, et pärast PWA kasutuselevõttu suurenes nende tehingute arv üle kõigi veebilehitsejate 104% ja iOS peal 82%. [16] Saavutatud tulemus peaks rahuldama ka nõudlikemaid ärisid.

Kui praegu *service workereid* Microsoft Edge ja Safari ei toeta, siis on lootust, et tulevikus see võimalus tuleb. *Service workerid* on hetkel Edge peal arenduses ning WebKit-il on hetkel kaalumisel [17], [18].

Kuna progressiivne parendus on PWA-de üks olulisemaid omadusi, siis on PWA põhimõtete järgimine alati kasulik. Ka juba olemasolevatele veebilehtedele ning -rakendustele on võimalik kerge vaevaga tehnoloogiad realiseerida ning näha koheselt paranevaid tulemusi nii jõudluse kui ka kasutatavuse poolelt. Nagu edulugudest näha, siis sobivad progressiivsed veebirakendused väga hästi dünaamilise sisuga rakendustele – näiteks veebipoodidele.

4.2 Progressiivse veebirakenduse võrdlus hübriid- ja põlisrakendustega

Antud peatükis võrdleb autor progressiivseid veebirakendusi hübriid- ja põlisrakendustega. Kriteeriumid, mille järgi võrdlus toimub on järgmised:

- **Kasutatavad tehnoloogiad.** Arendamiseks vajalikud programmeerimiskeeled.
- **Installeerimis vajadus.** Enne rakenduse kasutamist tuleb see enda seadmesse installeerida.
- **Uuendamise vajadus.** Uusima versiooni omamiseks tuleb rakendusel lubada allalaadida ning installeerida uuendused.
- **Rakendustepoed.** Rakenduste hankimine toimub läbi neid vahendavate e-poodide.
- **Lingitav.** Rakenduse iga leht omab unikaalset URL-i.
- **Riistavarale ligipääs.** Rakendusel on võimalik kasutada seadme osasid nagu näiteks kaamera, Bluetooth.
- **iOS toetus.** Rakendus omab täielikku funktsionaalsust Apple mobiilioperatsioonisüsteemil.

Vastavalt kriteeriumitele tehtud võrdlus on esitatud järgnevas tabelis (vt Tabel 2).

Tabel 2. PWA võrdlus hübriid- ja põlisrakendusega

Kriteerium	**PWA**	**Hybrid**	**Native**
Tehnoloogiad	Standardsed veebitehnoloogiad	Veebitehnoloogiad + Operatsiooni süsteemist sõltuv keel	Operatsiooni süsteemist sõltuv keel
Installeerimis vajadus	Ei	Jah	Jah
Uuendamise vajadus	Ei	Jah	Jah
Rakendustepoed	Ei	Jah	Jah
Lingitav	Jah	Ei	Ei
Riistvarale ligipääs	Puudustega	Jah	Jah
iOS toetus	Ei	Jah	Jah

PWA-de arendamiseks piisab vaid standardsetest veebitehnoloogiatest ning seega on võimalik vähendada arendusele kuluvat aja- ning rahakulu. Nagu ikka, võidab aga spetsiaalselt kindlale seadmele arendatud programm jõudluses. See tähendab, et isegi kui vahemällu salvestavate ressursside kaudu on võimalik rakenduse kiirust suurendada ja andmemahtu vähendada, siis hästi arendatud hübriid- või põlisrakendusest on PWA aeglasem.

Progressiivse veebirakenduse installeerimise mittevajamine tähendab, et rakendus on kohe kasutaja jaoks kasutuskõlbulik ning kasutaja ei pea läbima motivatsiooni eemaldavaid etappe. Kui pärast esimest kasutust nähakse rakenduses hüvesid, siis on palju suurem võimalus panna kasutajat rakendust installeerima, mis PWA puhul tähendab lihtsalt avaekraanile lisamist.

PWA kasutamine eemaldab rakenduste vanade versioonide hooldamise probleemi. Kuna kasutaja ei pea ise uuenduseks luba andma ning see tehakse ära automaatselt, siis on kogu tarbijaskonnal üks ja sama variant. Seega on kasutuses alati uusim ja parim versioon ning arendajad ei pea tagama pärandrakenduste tööd.

Rakenduste leidmine ning installeerimine läbi Google Play ja App Store-i on nii kasuks kui ka kahjuks. Kuigi on tõestatud, et installeerimisprotsess on demotiveeriv ning enamus rakendustest kaob tohutute konkurentide hulka ära, siis on need poed siiski esimeseks

kohaks, kus kasutajad rakendusi otsivad. Seega tuleb arendamise käigus mõelda oma tarbijaskonna peale ning arvestada sellega, et PWA-sid neisse poodidesse lisada ei ole võimalik.

Progressiivsed veebirakendused on aga otsingumootori sõbralikud. Arendajatel on võimalik panna rõhku SEO-le (*Search Engine Optimization*) ning muuta ennast võrreldes rakenduspoodidega leitavamaks. Lisaks ei pea PWA-de puhul järgima rangeid, eriti App Store rakendustele pandud , reegleid ning ei pea maksma protsenti oma tulust poodide omanikele.

Kuna hästi arendatud PWA iga vaade peaks omama unikaalset URL-i, siis on kasutajal võimalik rakendust lisada järjehoidjatesse ning lihtsalt jagada. Eriti kasulik on see tunnusjoon sotsiaalrakendustel nagu Twitter, mille sisu jagamine läbi põlisrakenduse on võrreldes PWA-ga raskendatud.

Hetkel on progressiivsete veebirakenduste suurimaks negatiivseks omaduseks puudulik riistvara ligipääs. Funktsionaalsus on pidevalt arenduses ning juba on võimalik näiteks kasutada kaamerat ja mikrofoni, puudub aga näiteks Bluetoothi kasutusvõimalus. See tähendab, et PWA-sid ei ole võimalik arendada näiteks nutikellade jaoks. Ligipääs on olemas küll failisüsteemile, kuid pole võimalik kasutada näiteks kontakte.

Nagu traditsiooniliste veebirakendustega võrdluses välja tuli, siis hetkel veel pole võimalik PWA-sid luua iOS seadmetele. See tähendab, et mitmeplatvormiliseks arenduseks tuleb iOS seadmetele luua eraldi lahendus.

Rakenduste loomisel tuleb arvestada nii funktsionaalsete kui ka mittefunktsionaalsete nõuetega. Vähemalt hetkel veel, kus PWA on alles uudne lahendus ja sellel ei ole põlisrakendustega võrreldes kogu funktsionaalsust, ei saa öelda, et PWA oleks alati parem lahendus. Küll aga on nüüd vähenenud hübriidrakenduste kasutusvajalikkus. Hetkel, kus iOS võimalus on PWA-del piiratud, on nõuete täitmise korral sobilikuks lahenduseks PWA ja iOS rakenduse arendus.

5 Raamistikud ja tehnoloogiad

Antud peatükis toob autor välja peamised raamistikud ja tehnoloogiad, mida kasutatakse antud töös progressiivse veebirakenduse näidisprojekti arendusel.

5.1 TypeScript

TypeScript on Microsofti poolt arendatud avatud lähtekoodiga JavaScripti ülemhulk, mis pakub arendustööriistu suurte rakenduste produktiivseks arenduseks. TypeScripti kompileeritakse tavalisse JavaScripti, mis tagab rakenduse töötamise kõigis uuemates veebilehitsejates. Sama keele süntaksi kasutamine annab ka võimaluse kasutada olemasolevat JavaScripti koodi ning teeke. TypeScript hõlmab endas kõiki uusi JavaScripti võimalusi [19].

TypeScripti lisavõimaluste hulgas on tugev tüüpimine, klassid, liidesed, konstruktorid, kapseldamine, pärimine ja polümorfism. Kompileerimise käigus on arendajal võimalik kergemini ja kiiremini avastada tehtud vigu ning tagada koodi kõrgem kvaliteet.

Järgneval näitel (vt Joonis 4 ja Joonis 5) on kujutatud esialgset TypeScriptiga kirjutatud koodi ja sellest kompileeritud JavaScripti koodi.

```
class Task {
    title: string;
    complete: boolean;

    constructor(title: string, complete: boolean) {
        this.title = title;
        this.complete = complete;
    }

    public getTitle(): string {
        return this.title;
    }
}
```

Joonis 4. TypeScripti kood

```
var Task = (function () {
function Task(title, complete) {
        this.title = title;
        this.complete = complete;
    }
    Task.prototype.getTitle = function () {
        return this.title;
    };
    return Task;
}());
```

Joonis 5. TypeScripti kood kompileeritud JavaScripti

5.2 Angular

Angular on arendusraamistik, millega on võimalik luua mitmeplatvormilisi rakendusi, nii arvuti-, mobiili- kui ka progressiivseid veebirakendusi [20]. Angular pakub kliendipoolsesse arendusse varem ainult serveripoolse arenduse käsitluses olevaid võimalusi.

Angulari rakendus koosneb komponentidest, mis koosnevad komponendi klassist ja HTML mallist. Mallide tööd juhivad komponendid ning rakendusloogikat sisestatakse läbi teenuste. Rakendused on modulaarsed, moodulites ühendatakse komponendid ja teenused [21].

Raamistik pakub võimalusi nagu andmete sidumine, marsruutimine ja animatsioonid [21]. Angulari populaarseimaks funktsiooniks on kahesuunaline andmete sidumine, millega on võimalik automaatselt kuvada rakenduse muutusi kasutajale ning vastupidi võtta arvesse kasutaja tehtud muutusi komponentides.

Angulari rakenduste arendamise lihtsamaks tegemiseks on loodud tööriist Angular CLI. Vahendiga on võimalik kiiresti luua koheselt töötavaid uusi projekte, genereerida Angulari komponente ning käivitada teste [22].

Töö näidisprojektiga alustati versiooniga Angular 2. 2017. aasta märtsis tehti kättesaadavaks Angular 4.0 lõplik versioon. Antud väljalase on Angular 2 suhtes tahapoole ühilduv ning seega jätkati näidisprojekti arendamist versiooniga 4.0.

5.3 Material Design

Material design on Google poolt kirja pandud kujundusjuhtnööride kogum, mille eesmärgiks on tagada ühine kasutajakogemus kõigil seadmetel. Samal ajal järgitakse hea disaini põhimõtteid ning pidevalt uuenevaid tehnoloogilisi ja teaduslikke võimalusi [23]. Juhtnööride hulka kuuluvad erinevate komponentide nagu näiteks nuppude, dialoogiakende ja menüüde kujundamine, elementide kompositsioon ja nende liigutamine ning värvipalettide ja kirjatüübi soovitused.

Käesolevas projektis kasutatakse Angulari meeskonna poolt loodud *material designi* komponente. Projekt on beeta staadiumis ning mitmed komponendid on alles arenduses või tuleviku plaanides.

5.4 Sass

Sass on kujunduskeel, mis laiendab CSS funktsionaalsust. Täienduse juures on muutujate kasutamine, CSS reeglite reasisene importimine ja nende sügavuti kirjutamine. Viimane neist võimaldab kirjutada sisemisi reegleid, mis kehtivad vaid ülemise klassi selektoritele [24].

Sass-i kirjutatakse üldjuhul kasutades SCSS (*Sassy Cascading Style Sheets*) süntaksit, mis on CSS-i laiendus ja mille failid on .scss laiendiga. Sass on täiesti ühilduv CSS-iga ning iga CSS korrektselt kirjutatud CSS fail on ka SCSS fail. Et HTML oskaks kujunduskeelt kasutada, kompileeritakse Sass CSS-iks [24]. Järgneval joonistel (vt Joonis 6 ja Joonis 7) on kujutatud sügavuti kirjutamise, muutujate kasutamise ning ühe Sass-i funktsiooni kompileerimist.

```
$task-color: #90CAF9;
$task-edit-color: #3F51B5;

.task {
  color: $task-color;

  .edit-task {
    color: $task-edit-color;
  }

  .edit-task:hover {
    color: lighten( $task-edit-color, 20% );
  }
}
```

Joonis 6. Sassiga kirjutatud SCSS kood

```
.task {
  color: #90CAF9;
}
.task .edit-task {
  color: #3F51B5;
}
.task .edit-task:hover {
  color: #8591d5;
}
```

Joonis 7. Sassiga kirjutatud SCSS koodist kompileeritud CSS

Töös kasutatav Angular CLI tagab SCSS-i automaatse kompileerimise CSS-iks. Kujunduskeel osutub vajalikuks *material designi* teema loomisel, komponentide kujundamisel kasutatakse traditsioonilist CSS-i.

5.5 IndexedDB

IndexedDB on populaarseim kliendipoolseks struktureeritud andmete salvestamiseks loodud API (*Application Programming Interface*). See on JavaScripti põhine andmebaas, mis kasutab võtmega indekseerimist, mis tagab suure jõudluse andmete otsimisel. IndexedDB operatsioonid on asünkroonsed, see tähendab, et andmebaasiga tehtavad toimingud ei takista rakenduse poolt tehtavaid ülesandeid [25].

IndexedDB ei kasuta SQL-i (*Structured Query Language*) ja on objektorienteeritud. See tähendab, et andmebaas ei ole relatsiooniline ning tabelitest koosnev. Andmete salvestamiseks luuakse andmehoidlast objekt, millesse on võimalik lisada JavaScripti objekte võti-väärtus paarides [26].

Näidisrakenduse arendamiseks kasutatakse ühe progressiivsete veebirakenduste põhilise propageerija Jake Archibaldi poolt loodud IndexedDB teeki. Teek lihtsustab originaalse API kasutamist asendades andmebaasi päringu objekti uue lihtsama asünkroonse päringuga.

5.6 Firebase

Firebase on Google mobiilplatvorm, mis võimaldab arendada kõrgekvaliteedilisi rakendusi kiiremini. Selle eesmärgiks on läbi lihtsa API asuda kliendi probleeme lahendama ilma, et peaks looma keerulise infrastruktuuri. Tööriistu on võimalik kasutada nii iOS, Android kui ka veebirakenduste ning Unity mängumootoril arendatud mängude jaoks. Platvormi kasutamist on võimalus alustada tasuta, suurema rakenduste jaoks oma hinnapoliitika [27].

Firebase-i kasutamine rakenduste arendamisel tagab kiiruse, kuna puudub vajadus serveripoolsele programmeerimisele. Firebase võimaluste seas on nii andmete talletamine ja sünkroniseerimine, kasutajate autentimine, erinevad pilveteenused kui ka hosting. Kuna tegemist on NoSQL andmebaasiga, siis nagu enamustel teistel süsteemidel, hoitakse ka Firebase-iga info JSON formaadis.

Näidisprojektis kasutatakse Firebase vaid veebihostingu jaoks. Kuna progressiivsed veebirakendused tuleb hoida turvatud veebiserverites, siis on selle nõude täitmiseks Firebase väga sobilik. Rakenduse edasiarenduses oleks võimalik kasutada ka platvormi andmebaasi võimalusi ning läbi selle tagada andmete sünkronisatsioon kliendi- ja serveripoolses andmehoidlas.

5.7 Jasmine, Karma ja Protractor

Angulari rakenduste testimiseks on mitmeid erinevaid tehnoloogiaid. Antud töös kasutatakse Angulari ametlikus dokumentatsioon välja toodud kolme hästi töötavat tööriista. Jasmine testimisraamistik võimaldab kirjutada puhta ja arusaadava süntaksiga teste. Karma on tööriist, mis loob Angulari rakenduse jaoks testkeskkonna ilma raske seadistuseta ning võimaldab ühiktestide kirjutamist ja käivitamist. Protractori abil on võimalik testida, kas rakendus käitub algusest lõpuni nii nagu eeldatud [28].

Näidisprojektis kontrollitakse rakenduse käitumist vaadete vahetumisel, sealhulgas tööriistaribas oleva pealkirja muutumist ning vaadete põhielementide olemasolu. Lisaks kontrollitakse ühiktestide näol funktsioone, mis pole otseselt seotud IndexedDB-ga ning mis pole seotud ainult kasutajaliidesega.

5.8 Travis CI

Travis CI on pidevat integratsiooni ja juurutamist pakkuv teenus GitHubi projektidele. Avalikele hoidlatele on mõeldud tasuta variant, privaatsete hoidlate kasutamiseks pakutakse tasulist versiooni [29]. Tarvis CI võimaldab tarkvaraarendust suures hulgas programmeerimiskeeltes, sealhulgas JavaScript-is.

Teenuse seadistamiseks lisatakse projekti juurkataloogi fail nimega .travis.yml. Failis lisatava info järgi on võimalik Travis CI-le öelda näiteks millist programmeerimiskeelt kasutatakse ja milliseid käske peab täitma ning millised sõltuvused realiseerima enne juurutamist. Lisaks on võimalik teavitusi protsessi toimumisest või ebaõnnestumisest [30].

Näidisprojekti jaoks loodud seadistusfail on esitatud järgneval joonisel (vt Joonis 8).

```
language: node_js
node_js:
  - "6.9"
branches:
  only:
    - master
before_script:
  - npm install -g @angular/cli@latest
  - npm install -g firebase-tools
  - npm install --save @angular/material
  - npm install --save @angular/animations
  - npm install --save hammerjs
script:
  - ng build --prod --aot
  - npm run precache
after_success:
  - firebase deploy --token $FIREBASE_TOKEN
notifications:
  email:
    on_failure: change
    on_success: change
```

Joonis 8. Näidisprojekti jaoks seadistatud travis.yml

Kuna Angular kasutab sõltuvuste installimiseks npm-i (*Node.js Package Manager*), siis on keeleks seadistatud Node.js. Arendusel harusid ei kasutata, kogu kood liidetakse vaikimisi *master* haruga. Enne juurutamist paigaldatakse vajalikud sõltuvused, siis aga kompileeritakse rakendus ning genereeritakse vahemällu salvestav *service worker*. Järgnevalt laetakse rakendus Firebase veebihostingusse ning teavitatakse protsessi tulemist läbi e-posti.

6 Progressiivse veebirakenduse näidisrakendus

Valminud näidisrakendus on ligipääsetav veebiaadressil https://success-planner.firebaseapp.com/ ning antud töö hoidla asub aadressil https://github.com/lohmuss/success-planner. Rakenduse loomisel on kasutatud eelnevas peatükis mainitud raamistikke ning tehnoloogiaid ning realiseeritud PWA-de põhiomadused. Antud peatükis toob autor välja rakenduse nõuded, ülesehituse, hinnangu nõuetele vastavuse kohta ning edasiarenduse võimalused.

6.1 Rakenduse tutvustus

Käesoleva töö raames valminud näidisrakenduse eesmärk on võimaldada kasutajal jälgida oma ülesannete täitmist ning harjumuste kujundamist. Loodud rakenduse nimi on „Success Planner“ ning selle funktsionaalsuse tagamiseks on vaja täita järgmised funktsionaalsed nõuded:

- Kaustaja saab lisada päevaülesandeid
- Kasutaja saab lisada kuuülesandeid
- Kasutaja saab luua uue harjumuse
- Kasutaja saab loodud ülesannet/harjumust muuta
- Kasutaja saab loodud ülesannet/harjumust kustutada
- Kasutaja saab loodud ülesannet märkida tehtuks
- Kasutaja saab loodud harjumust märkida tehtuks iga nädalapäev
- Kasutaja saab jälgida kuu jooksul tehtud päevaülesannete ja harjumuste kohta statistikat
- Süsteem loob uue kuu algul ülesannete salvestamiseks uue kuu
- Süsteem loob uue nädala algul harjumuste salvestamiseks uue nädala
- Kasutaja saab jälgida eelnevatel nädalatel ja kuudel tehtud tööd
- Kasutaja saab vaadata päevaülesandeid tähtaja järgi kategoriseerituna
- Kasutaja saab vaadata tähtaja ületanud ning tänase päeva ülesandeid ja tänase päeva harjumuste täitmist esilehel
- Kasutaja saab meeldetuletuse uue kuu algusest

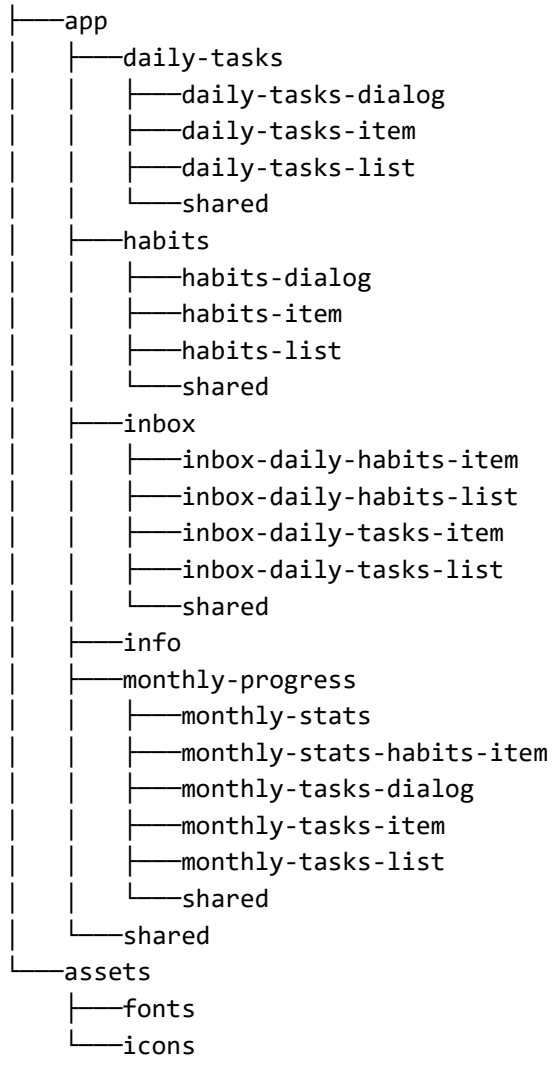
Rakenduse kvaliteediomadusi silmas pidades, peab rakendus täitma järgnevaid mittefunktsionaalseid nõudeid:

- Rakendus laadimine pärast esimest külastust peab toimima kiiremini
- Rakendus peab pärast esimest külastust töötama sõltumatult ühenduse kvaliteedist
- Rakendus peab põlisrakenduse sarnase kasutajakogemuse saavutamiseks täitma *material designi* juhtnööre
- Rakenduse hostinguga tegelev võrguühendus peab olema turvaline
- Rakendus peab läbima Lighthouse testid vähemalt tulemusele 90 punkti

6.2 Rakenduse struktuur

Rakendus koosneb viiest moodulist: juurmoodul, postkasti moodul, päevaste ülesannete moodul, harjumuste moodul ja kuu saavutuste moodul. Juurmoodul ühendab ülejäänud moodulid üheks tervikuks, teised moodulid ühendavad vaateid moodustavaid komponente, malle ja teenuseid.

Projekti struktuuri loomisel jälgiti Angulari ametlikku stiilijuhist. Allikaskataloogi kaustade struktuuri on näidatud järgneval joonisel (vt Joonis 9).



Joonis 9. Projekti allikaskataloogi kaustade struktuur

Igal komponendil on oma kaust, ühe mooduli komponendid on ühes kaustas. Enamus moodulite infot hoidvates kataloogides on ka ühiskaust, kus hoitakse teenuseid ning moodulit ennast. Eraldi kaust on ka varade jaoks, mis hoiab endas kirjatüüpe ja ikoone.

6.3 Kasutajaliides

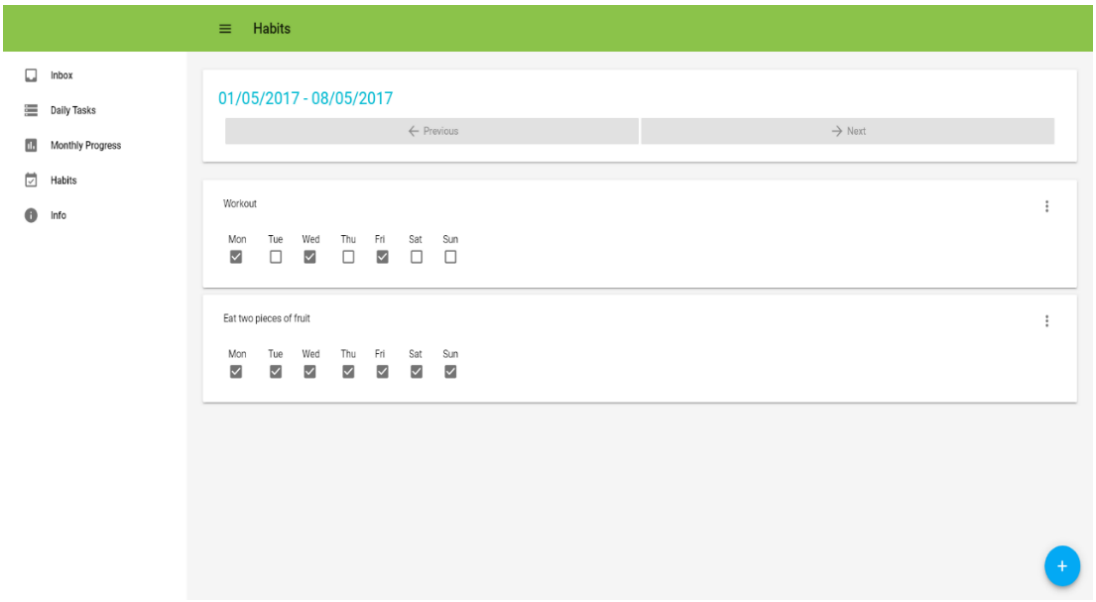
Rakenduses on loodud 5 erinevat vaadet, mis on mõeldud funktsionaalsuse tagamiseks ning info andmiseks. Vaated on kirjeldatud järgnevas tabelis (vt Tabel 3).

Tabel 3. Rakenduse vaated

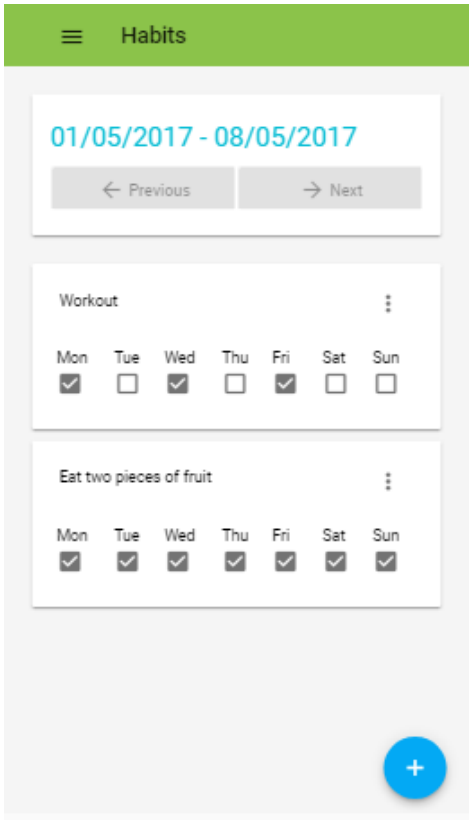
Nimi	**URL**	**Eesmärk**
Postkast (*Inbox*)	/	Rakenduse esileht, mis kuvab üle aja läinud, tänase päeva ülesandeid ning tänase päeva harjumusi.
Päevased ülesanded (*Daily tasks*)	/daily-tasks	Päevaste ülesannete vaade, kus on võimalik luua uusi ülesandeid, neid muuta, kustutada ning vaadata neid lähtuvalt kuupäevast jaotatuna.
Harjumused (*Habits*)	/habits	Harjumuste vaade, kus on võimalik luua uusi harjumusi ning vaadata nädalata kaupa nende täitmist.
Kuu saavutused (*Monthly Progress*)	/monthly-progress	Vaade, kus on võimalik luua tähtpäevatuid kuuülesandeid, neid muuta, kustutada ning näha statistikat kuu jooksul tehtud päevaste ülesannete ning harjumuste kohta.
Info	/info	Rakendust tutvustav vaade.

Lisaks antud vaadetele on eraldi vaade juurkomponendil, mis sisaldab rakenduse navigatsiooni ning loob koha teiste vaadete kuvamiseks. Nagu õigele PWA-le kohane, siis igat vaatele on võimalik ligi pääseda kasutades unikaalset URL-i. Liikudes tabelis paiknevatele URL-idele, paigutatakse juurkomponenti vastav vaade.

Rakendus on vastavalt PWA nõuetele seadmele kohanduv, mis tähendab, et lisaks mobiilis kasutusele, on võimalik rakendust kasutada ka personaalarvutis. Vastavalt seadme dimensioonidele on rakenduse elementidel erinev paigutus, seda erinevust on kujutatud järgnevatel joonistel (vt joonis 9 ja Joonis 10).



Joonis 10. Harjumuste vaade personaalarvutis



Joonis 11. Harjumuste vaade mobiiltelefonis

Suurim vahe väiksema ja suurema ekraaniga seadme kasutajaliideses on navigatsioon. Kui suuremal esitluspinnal on menüü automaatselt avatud, siis väiksemal pinnal ei ole selle jaoks ruumi ning kui menüü on avatud, ei ole võimalik kasutada rakenduse teisi funktsionaalsusi.

6.4 Progressiivse veebirakenduse nõuetele vastavus

Antud peatükis võrreldakse valminud näidisrakendust progressiivsete veebirakenduste nõuete ja omadustega. Google poolt kirja pandud omaduste täitmist selgitakse järgnevas tabelis (vt Tabel 4).

Tabel 4. PWA omaduste tagamine näidisrakenduses

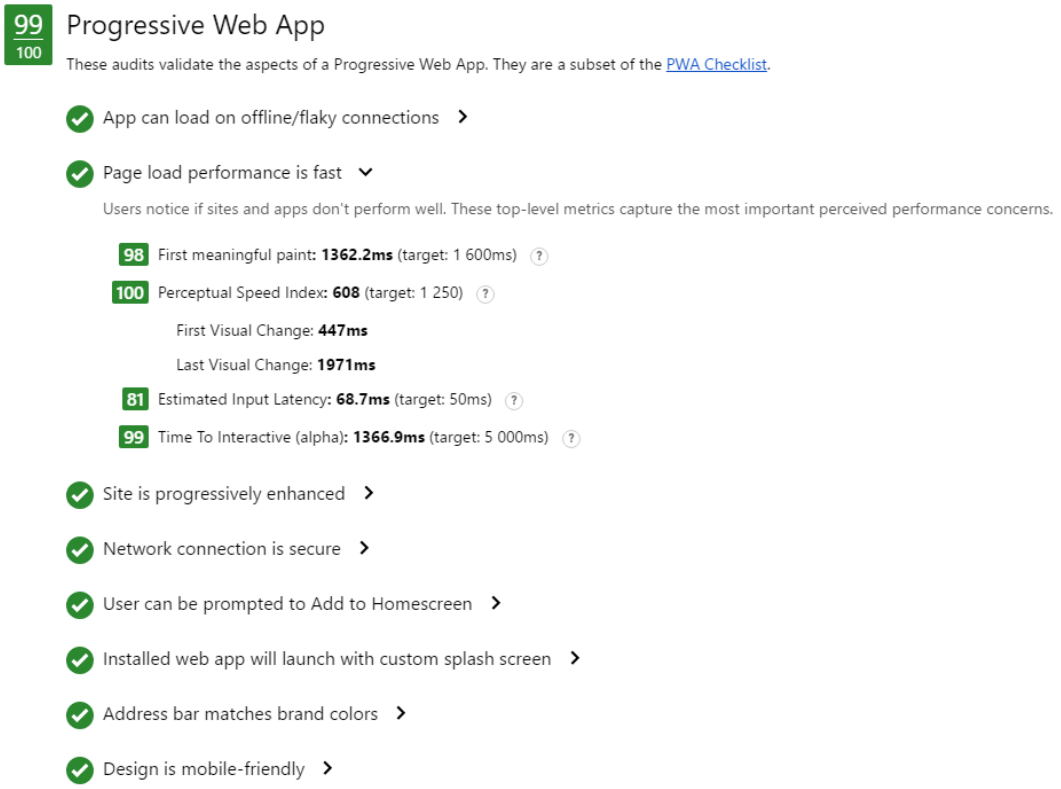
Omadus	**Lahenduse selgitus**
Progressiivne	Veebirakendust on võimalik avada iga veebilehitsejaga. Ka ilma JavaScripti toetuseta veebilehitsejates näidatakse kasutajale sisu.
Kohanduv	Rakenduse disain kohandub vastavalt kasutatava seadme dimensioonidele.
Sõltumatu ühendusest	Rakendust on võimalik korduval külastusel kasutada ilma internetiühenduseta.
Põlisrakendusega sarnane	Rakenduse navigatsioon ning funktsionaalsus on tänu *material designi* komponentide kasutamisele sarnane põlisrakendustega.
Pidevalt uuendatud	Uuenduste korral ei pea kasutaja uuendust kinnitama.
Turvaline	Rakendust hostimiseks kasutatakse Firebase-i, mis kasutab HTTPS-i.
Avastatav	Rakendus on otsingumootorite abil avastatav. Juhul kui esineks soov rakenduse avaldamiseks laiemale kasutajaskonnale, tuleks tegeleda SEO-ga.
Kütkestav	Rakendus tuletab kasutajale uue kuu alguses meelde lisada kuu ülesandeid. Tegemist on vanema tehnoloogiaga, mis ei ole mobiilseadmetel realiseeritud. Mobiilseadmete jaoks tuleks kasutada pilveteenuse sõnumeid.
Installeeritav	Rakendusel on realiseeritud manifest, mis võimaldab rakendust lisada avaekraanile.
Lingitav	Rakenduse igal vaatel on unikaalne URL.

6.4.1 Lighthouse

Lighthouse on avatud lähtekoodiga tööriist, millega on võimalik hinnata veebilehe jõudlust, juurdepääsetavust ning progressiivse veebirakenduse nõuetele vastavust. Tööriistale tuleb ette anda vaid URL, vahend teeb automaatselt testid ning genereerib

aruande tulemustest. Tulemuste hulgas mainitakse miks antud kontroll on oluline ning kuidas tulemust parandada [31].

Tööriista on võimalik käivitada kolmel viisil: Chrome veebilehitseja laiendina, käsurealt või Node moodulina. Antud töös kasutati neist esimest viisi ning PWA testide tulemused on toodud järgneval joonisel (vt Joonis 11).



Joonis 12. Lighthouse testi tulemused

Testide tulemuste järgi on tegemist täisfunktsionaalse progressiivse veebirakendusega. Tööriista legendi järgi kategoriseeritakse tulemused 75-100 heaks, kõik näidisrakenduse testid ka selle tulemuse saavutavad. Testide korduval jooksutamisel on rakenduse jõudluse tulemustes erinevused, mis tulenevad pigem veebihostingust kui rakendusest endast.

6.5 Täiendus võimalused

Kuna progressiivsed veebirakendused ei saa oma täies funktsionaalsuses hetkel veel iOS peal töötada, siis ei ole näidisrakendus arendatud mõeldes sellele platvormile. Arenduse käigus selgus, et rakenduses kasutatav IndexedDB on küll iOS-i peal väidetavalt toetatud, siis esineb sellega liiga palju probleeme.

Hoides silmas tulevikku ning lootes, et Apple jõuab teistele veebilehitsejatele järgi progressiivsete veebirakenduste koha pealt, siis tuleks IndexedDB kindlasti asendada millegi universaalsemaga. Kuna rakenduse kõik funktsioonid on seotud andmete salvestuse ning nende lugemisega, siis aitab siin ka progressiivne parendus ning praegusel kujul on rakendust võimalik kasutada ainult Androidi seadmete peal ning personaalarvutites kõikides veebilehitsejates peale Safari.

Funktsionaalsuse koha pealt oleks võimalusi täiendada tohutult. Enamus sarnastes rakendustes on olemas ülesannete grupeerimine rühmadesse, nende kirjeldused ning ajalised kordused.

Kuna rakendus kasutab hostinguks Firebase-i siis oleks võimalik kergesti juurde lisada kaustajaks registreerimine, pilves oleva andmebaasi kasutamine ning pilve kaudu teadete saatmine. Sellise funktsionaalsuse lisamine võimaldaks rakendust kasutada erinevate seadmete peal üheaegselt ning samade andmetega. Lahenduse keeruliseks osaks oleks andmete sünkroniseerimine lokaalse ja pilves oleva andmebaasi vahel.

7 Kokkuvõte

Käesoleva bakalaureusetöö eesmärgiks oli leida vastus küsimusele, kas progressiivse veebirakenduse mõiste on lihtsalt moekas mitte midagi tähendav termin või on tegemist millegi kasulikuga ning revolutsioonilisega. Töö tulemid on mõeldud infotehnoloogia valdkonda esindavatele inimestele, kes tegelevad mobiilirakenduste arendusega.

Töö käigus uuris autor olemasolevaid alternatiive põlisrakendustele mobiilirakenduste arendamiseks, progressiivsete veebirakenduste tehnilist sisu ning võrdles nende võimaluste tugevaid ja nõrku külgi. Teoreetilisele osale lisaks arendas autor näidisrakenduse ühest Androidi seadmetel toimivast progressiivsest veebirakendusest.

Autori poolt seotud eesmärk sai täidetud ning käesoleva töö tulemusena leiti, millistel juhtudel on võimalik põlisrakendusi asendada progressiivse veebirakendustega ning millal hetkeoludes ainult veebitehnoloogiatel põhinevast veebirakendusest ei piisa. Töö tulemusena valmis ka progressiivse veebirakenduse näidisrakenduse, mis realiseerib kõiki temale esitatud nõudeid.

Progressiivne veebirakendus on võimas lahendus, millega on võimalik tagada seda realiseerivatel seadmetel põlisrakendustega sarnane kasutajakogemus. Praegusel hetkel on tegu aga uue arenguga, mis tähendab, et ei ole võimalik võrreldes põlisrakendustega tagada kogu funktsionaalsust ning iOS platvormi puudused nõuavad tihti vähemalt ühte spetsiaalselt ühele seadmele loodavat rakendust. Küll aga võib öelda, et progressiivsed veebirakendused on hübriidrakenduste rolli arenduses üle võtnud ning traditsioonilistele veebirakendustele ei ole kunagi kahjuks progressiivsete omaduste realiseerimine.

Käesoleva töö käigus valminud rakendus on kättesaadav aadressil https://success-planner.firebaseapp.com/.

Kasutatud kirjandus

[1] Piejko, P. 16 mobile market statistics you should know in 2016. – *DeviceAtlas, 2016.* [WWW] https://deviceatlas.com/blog/16-mobile-market-statistics-you-should-know-2016 (26.04.2017)

[2] Cselle, G. Every Step Costs You 20% of Users. – *Tables of Creation, 2012.* [WWW] http://blog.gaborcselle.com/2012/10/every-step-costs-you-20-of-users.html (26.04.2017)

[3] Xanthopoulos, S., Xinogalos, S. A comparative analysis of cross-platform development approaches for mobile applications. – *Proceedings of the 6th Balkan Conference in Informatics (BCI '13),* 2013. Thessaloniki, Greece, 213-220. [Online] http://dl.acm.org/citation.cfm?id=2490292 (28.04.2017)

[4] Using the application cache. [WWW] https://developer.mozilla.org/en-US/docs/Web/HTML/Using_the_application_cache (14.04.2017)

[5] Bristowe, J. What is a Hybrid Mobile App? – *Telerik Developer Network, 2015.* [WWW] http://developer.telerik.com/featured/what-is-a-hybrid-mobile-app/ (15.04.2017)

[6] LePage, P. Your First Progressive Web App. – *Google Developers Web Fundamentals.* [WWW] https://developers.google.com/web/fundamentals/getting-started/codelabs/your-first-pwapp/ (08.03.2017)

[7] Progressive Web Apps. [WWW] https://developers.google.com/web/progressive-web-apps/ (08.03.2017)

[8] Wells, J., Draganova, C. Progressive enhancement in the real world. – *Proceedings of the eighteenth conference on Hypertext and hypermedia (BCI '07),* 2007- Manchester, UK, 55-56. [Online] http://dl.acm.org/citation.cfm?id=1286259 (02.04.2017)

[9] Web App Manifest. [WWW] https://www.w3.org/TR/appmanifest/ (10.03.2017)

[10] Service Worker API. [WWW] https://developer.mozilla.org/en-US/docs/Web/API/Service_Worker_API (11.03.2017)

[11] Gaunt, M. Service Workers: an Introduction. – *Google Developers Web Fundamentals.* [WWW] https://developers.google.com/web/fundamentals/getting-started/primers/service-workers (11.03.2017)

[12] Osmani, A. The App Shell Model. – *Google Developers Web Fundamentals.* [WWW] https://developers.google.com/web/fundamentals/architecture/app-shell (13.03.2017)

[13] Nagaram, A. Progressive Web App: A New Way To Experience Mobile. – *Flipkart Tech Blog, 2015.* [WWW] http://tech-blog.flipkart.net/2015/11/progressive-web-app/ (15.03.2017)

[14] Konga Case Study. [WWW] https://developers.google.com/web/showcase/2016/konga (16.03.2017)

[15] eXtra Electronics Case Study. [WWW] https://developers.google.com/web/showcase/2016/extra (16.03.2017)

[16] Aliexpress Case Study. [WWW] https://developers.google.com/web/showcase/2016/aliexpress (16.03.2017)

[17] Microsoft Edge Platform Status. [WWW] https://developer.microsoft.com/en-us/microsoft-edge/platform/status/serviceworker/ (24.03.2017)

[18] WebKit Feature Status. [WWW] https://webkit.org/status/#specification-service-workers (24.03.2017)

[19] TypeScript. [WWW] https://www.typescriptlang.org/ (11.03.2017)

[20] Angular Features & Benefits. [WWW] https://angular.io/features.html (11.03.2017)

[21] Angular Architecture Overview. [WWW] https://angular.io/docs/ts/latest/guide/architecture.html (12.03.2017)

[22] Angular CLI. https://cli.angular.io/ (16.04.2017)

[23] Material Design Introduction. [WWW] https://material.io/guidelines/material-design/introduction.html (15.03.2017)

[24] Sass (Syntactically Awesome StyleSheets). [WWW] http://sass-lang.com/documentation/file.SASS_REFERENCE.html (20.04.2017)

[25] IndexedDB API. [WWW] https://developer.mozilla.org/en/docs/Web/API/IndexedDB_API (17.03.2017)

[26] IndexedDB API Basic Concepts. [WWW] https://developer.mozilla.org/en-US/docs/Web/API/IndexedDB_API/Basic_Concepts_Behind_IndexedDB (17.03.2017)

[27] Firebase. [WWW] https://firebase.google.com/ (18.03.2017)

[28] Angular Testing. [WWW] https://angular.io/docs/ts/latest/guide/testing.html (15.05.2017)

[29] Travis CI GitHub. [WWW] https://github.com/travis-ci/travis-ci (22.03.2017)

[30] Travis CI Customizing the Build. [WWW] https://docs.travis-ci.com/user/customizing-the-build/ (22.03.2017)

[31] Lighthouse. [WWW] https://developers.google.com/web/tools/lighthouse/ (05.05.2017)