PROGRESĪVO TĪMEKĻA LIETOTŅU PERSPEKTĪVAS

PERSPECTIVES OF PROGRESSIVE WEB APPS

Autori: **Arnis RITIŅŠ**, e-pasts: arnis_ritins@inbox.lv;
Aleksejs SERGEJEVS, e-pasts: alexserg@inbox.lv
Zinātniskā darba vadītājs: **Sergejs Kodors**, Dr.sc.ing., e-pasts: Sergejs.Kodors@rta.lv

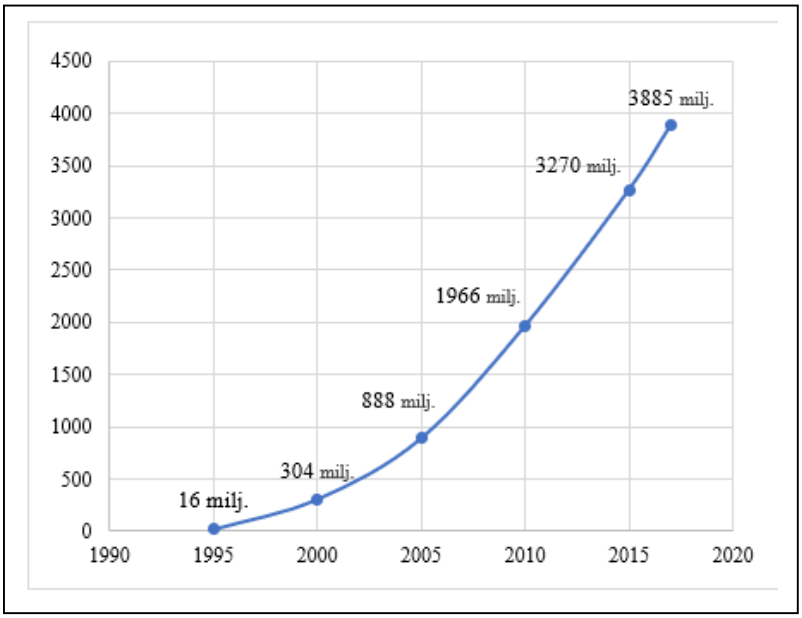
Abstract. *Given the monopoly over native mobile market for certain kind of apps, a cheaper and an easier to make alternative is necessary. One such alternative are the progressive web apps which solve some of the issues related to the native mobile apps. The goal of the work is to analyse the pros of progressive web apps and native mobile apps and estimate why and at which scenarios the progressive web apps can outcompete native mobile apps in terms of the pros.*

Keywords: *PWA, web, mobile, native, HTML, JavaScript.*

Ievads

Programmatūras izplatīšana un nogādāšana līdz gala lietotājam ir viens no svarīgākajiem jautājumiem programmatūras dzīves ciklā. Mūsdienās vidējais mobilo ierīču lietotājs mēnesī neinstalē pat vienu vienīgu lietotni. Šī iemesla dēļ tās nedaudzās lietotnes, kas tiek izmantotas, iegūst 80% no visiem lietotājiem. [1] Tā rezultātā jaunas un inovatīvas idejas paliek atstātas novārtā, jo tās nespēj iegūt kapitālu, kas nepieciešams, lai tās veiksmīgi turpinātu attīstīt. Viens no veidiem kā risināt šo problēmu ir progresīvo tīmekļa lietotņu izstrāde.

Dotā temata aktualitāti pamato interneta lietotāju skaita straujais pieaugums (sk. 1.att.), kā arī tīmekļa tehnoloģiju un interneta pārlūkprogrammu straujā attīstība. Mobilo iekārtu lietotāji nav īpaši ieinteresēti jaunu lietotņu lejupielādē, taču mājas lapu atvēršana un jaunu tīmekļa lietotņu izmēģināšana parasti lietotājiem nesagādā lielas grūtības.



1. attēls**. Interneta lietotāju skaita pieaugums periodā no 1995. līdz 2017. gadam** [10]

Izmantojot tīmekļa tehnoloģijas, mūsdienās ir iespējams izveidot lietotni, kas ir spējīga emulēt lielu daļu natīvo lietotņu funkcionalitātes. Šī darba mērķis ir noteikt progresīvo tīmekļa lietotņu izmantošanas perspektīvas.

http://dx.doi.org/10.17770/het2018.22.3637

Materiāli un metodes

Progresīvo tīmekļa lietotņu būtība

Pēc būtības progresīvā tīmekļa lietotne ir tā pati klasikā tīmekļa lietotne, kas ir izstrādāta ar tīmekļa tehnoloģijām, kā *HTML, CSS* un *JavaScript*, taču lietotājs to var darbināt un izmantot kā natīvo mobilo lietotni. Šī programmatūras tipa idejas pamatā ir apvienot mūsdienu pārlūku piedāvātās iespējas ar mobilo ierīču priekšrocībām.

Service Worker skripti

Viena no galvenajām sastāvdaļām, kas klasisko tīmekļa lietotni padara par progresīvo, ir *Service Worker* skripts.

Service Worker ir konkrēta veida tīmekļa strādnis (*web worker*). Pēc būtības tas ir *JavaScript* fails, kas tiek darbināts atsevišķi no galvenā pārlūka pavediena. [4] *Service Worker* galvenokārt ir paredzēts, lai nodrošinātu korektu lietotnes darbību bezsaistes režīmā, pārtverot tīkla pieprasījumus un ļaujot saglabāt kešatmiņā dažādus resursus (*HTML* dokumentus, *JavaScript* skriptus, *CSS* stilu lapas, attēlu datnes u.c.). Tāpat arī *Service Worker* skriptam ir iespēja izmantot pašpiegādes paziņojumus un datu sinhronizēšanas API. [6]

Tā kā *Service Worker* darbojas atsevišķi no galvenā pavediena, tad nav tieši atkarīgs no lietotnes. Tam ir daudzas dažādas sekas: [4]

- *Service Worker* nav paredzēts būt pilnībā asinhrons, līdz ar to sinhronie *XML HTTP* pieprasījumi, kā arī *localStorage* nevar tikt lietoti *Service Worker* skripta ietvaros; [4]
- neatkarīgi no tā, vai lietotne ir aktīva, vai ne, *Service Worker* ir spējīgs saņemt *push* ziņojumus no servera. Tas dod iespēju sūtīt pašpiegādes paziņojumus lietotājam pat tad, ja lietotne nav atvērta pārlūkprogrammā; [4]
- *Service Worker* skriptam nav tiešas piekļuves dokumentu objektu modelim *DOM*. Lai sazinātos ar tīmekļa lapu, *Service Worker* izmanto *postMessage()* metodi datus sūtīšanai un *message* notikumu uzturētāju (*event listener*) datu saņemšanai.

Shell arhitektūra

Shell arhitektūra ir viens no veidiem kā izstrādāt progresīvo tīmekļa lietotni, kas nodrošinātu ātru un drošu tās ielādi līdzīgi natīvajām lietotnēm. [7] Lietotnes *Shell* ir minimālais *HTML, CSS, JavaScript* saturs, kas ir nepieciešams, lai darbinātu lietotāja saskarni. Kad saglabāts kešatmiņā, bezsaistē tas var nodrošināt ātru un stabilu veiktspēju lietotājiem atkārtotos apmeklējumos. Tas nozīmē, ka lietotnes *Shell* netiek ielādēts no tīkla katru reizi. No tīkla tiek ņemts tikai nepieciešamais saturs, kas bieži mainās un atjaunināts. [7]

Lietotnes manifesta datne

Manifests ir vienkārša *JSON* datne, kas nodrošina izstrādātājam iespēju kontrolēt to, kā lietotne tiek parādīta lietotājam, kā arī definēt tās izskatu palaišanas brīdī. [5]

Manifesta datne tiek iekļauta *HTML* dokumentā ar *<link/>* taga palīdzību. Manifestā var konfigurēt šāda lietas:

- lietotnes nosaukums – īss nosaukums, kas tiek parādīts uz lietotāja sākuma ekrāna;
- ikona – tiek konfigurēts ikonu kopums, kas attēlojas, saglabājot lietotni uz sākuma ekrāna;
- sākuma lapa – lietotnes tīmekļa vietrādis (*URL*), kas tiek ielādēts, atverot lietotni;
- uzplaiksnījuma ekrāna stils – var definēt ekrāna krāsu un stilu, lietotnes palaišanas brīdī;
- ekrāna orientācija – var noteikt lietotnes orientāciju tās palaišanas brīdī [5].

Pārlūkprogrammu atbalsts

Progresīvās tīmekļa lietotnes ir relatīvi jauna tehnoloģija, kas balstās uz iespējām, ko nodrošina pārlūkprogramma, līdz ar to, šī lietotņu veida attīstība ir tieši atkarīga no tīmekļa pārlūkprogrammu attīstības.

Galvenā prasība progresīvo tīmekļa lietiņu darbībai ir *Service Worker* atbalsts. Uz doto brīdi *Service Worker* tiek atbalstīts šādā pārlūkprogrammās:

- *Google Chrome* (pilnīgs atbalsts);
- *Safari* (pilnīgs atbalsts);
- *Firefox* (pilnīgs atbalsts);
- *Microsoft Edge* (izstrādes stadijā) [8].

1. tabula

Rezultāti un to izvērtējums

Priekšrocība	Progresīvās tīmekļa lietotnes	Natīvās mobilās lietotnes
Var lietot lokāli bez interneta pieslēguma	✓	✓
Ir centralizēts veikals, kur ir apkopotas lietotnes lielā apmērā	✗	✓
Spēj veikt vairākas operācijas paralēli	✗	✓
Zems baterijas resursu patēriņš	✗	✓
Spēj izmantot visu iespējamo ierīces funkcionalitāti	✗	✓
Ātri un vienkārši izstrādājams	✓	✗
Vienkārša lietotnes atjaunināšana un uzturēšana	✓	✗
Daudz-platformu atbalsts	✓	✗
Viegli reklamējams	✓	✗
Iespēja lietot bez instalēšanas	✓	✗

Balstoties uz veikto analīzi un iegūtajiem rezultātiem (sk. 1. tabulu), var spriest, ka progresīvo tīmekļa lietotņu izstrāde ir lētāka, ātrāka un to ir vienkāršāk reklamēt, tāpēc nelieliem uzņēmumiem šīs lietotnes ir ideālais variants. Natīvās lietotnes ir pārākas funkcionalitātes un ātrdarbības ziņā, taču tās izstrādes izdevumi ir daudz lielāki, tāpēc šis variants ir ieteicams tikai lieliem uzņēmumiem, vai arī īpašos gadījumos, kad progresīvo tīmekļa lietotņu pieejamās iespējas neapmierina projekta prasības.

Secinājumi

1. Progresīvās tīmekļa lietotnes nav optimālākais risinājums situācijās, kad ātrdarbībai un zemam baterijas resursu patēriņam ir liela nozīme;
2. Galvenā progresīvo tīmekļa lietotņu priekšrocība ir daudzplatformu atbalsts, kas savukārt būtiski atvieglo izstrādi un uzturēšanas darbus;
3. Izstrādājot progresīvo tīmekļa lietotni, var samazināt kopējās izstrādes izmaksas, jo nav nepieciešams atsevišķi algot atsevišķus programmētājus katrai mobilo ierīču platformai;
4. Turpinoties pārlūkprogrammu attīstībai, progresīvo tīmekļa lietotņu un ierīču atbalsts tikai pieaugs;
5. Alternatīvi progresīvajām tīmekļa lietotnēm un natīvajām mobilajām lietotnēm eksistē hibrīdie varianti, kam piemīt daudzas no šo abu lietotņu priekšrocībām.

Summary

With the rising popularity of the internet, web technologies only get more sophisticated. Given that hardware seems to be less of an issue nowdays, technologies like HTML5 get to be used in several different industries outside it's original domain of expertise. As stated once by the Co-founder of Stack Overflow Jeff Atwood: „any application that can be written in JavaScript, will eventually be written in JavaScript". [9] Frankly, given that there are even

Desktop apps written in JavaScript, it might eventually come true. It was inevitable that a technology like progressive web apps emerged.

Literatūra

1. Google Chrome Developers. Why Build Progressive Web Apps?. Sk Internetā (04.09.2017) https://www.youtube.com/watch?v=1QILz1lAzWY
2. Binariks. Pros of progressive Web Apps. Sk. Internetā (29.01.2018) http://www.binariks.com/blog/tips/progressive-web-app-development-process-pros-cons-benefits-costs/
3. Harshith. Do PWAs (progressive web apps) have experience advantages over native apps? Sk. internetā (29.01.2018) https://ux.stackexchange.com/questions/108830/do-pwas-progressive-web-apps-have-experience-advantages-over-native-apps
4. Google. Introduction to Service Worker. Sk. Internetā (07.02.2018) https://developers.google.com/web/ilt/pwa/introduction-to-service-worker
5. Google. The Web App Manifest. Sk. Internetā (11.02.2018) https://developers.google.com/web/fundamentals/web-app-manifest/
6. MDN Web Docs, Service Worker API (11.02.2018) https://developer.mozilla.org/en-US/docs/Web/API/Service_Worker_API
7. Google. The App Shell Model. Sk. Internetā (13.02.2018) https://developers.google.com/web/fundamentals/architecture/app-shell
8. Vaadin, PWA Browser support. Sk. Internetā (21.04.2018) https://vaadin.com/progressive-web-applications/learn/browser-support
9. Atwood J. *The principle of Least Power* Sk. Internetā (22.04.2018) https://blog.codinghorror.com/the-principle-of-least-power/
10. Internet growth statistics. Sk. Internetā (22.04.2018) https://www.internetworldstats.com/emarketing.htm