Progresivne web aplikacije

Ivanušec, Sandi

Sveučilište Jurja Dobrile u Puli
Fakultet informatike u Puli

SANDI IVANUŠEC

PROGRESIVNE WEB APLIKACIJE

Završni rad

Pula, rujan, 2019.

Sveučilište Jurja Dobrile u Puli
Fakultet informatike u Puli

SANDI IVANUŠEC

PROGRESIVNE WEB APLIKACIJE

Završni rad

JMBAG: 0303069339, redoviti student
Studijski smjer: Informatika

Predmet: Poslovni informacijski sustavi
Mentor: doc. dr. sc. Darko Etinger
Komentor: dr. sc. Nikola Tanković

Pula, rujan, 2019.

IZJAVA O AKADEMSKOJ ČESTITOSTI

Ja, dolje potpisani _____, kandidat za prvostupnika _____ovime izjavljujem da je ovaj Završni rad rezultat isključivo mojega vlastitog rada, da se temelji na mojim istraživanjima te da se oslanja na objavljenu literaturu kao što to pokazuju korištene bilješke i bibliografija. Izjavljujem da niti jedan dio Završnog rada nije napisan na nedozvoljen način, odnosno da je prepisan iz kojega necitiranog rada, te da ikoji dio rada krši bilo čija autorska prava. Izjavljujem, također, da nijedan dio rada nije iskorišten za koji drugi rad pri bilo kojoj drugoj visokoškolskoj, znanstvenoj ili radnoj ustanovi.

Student

_____

U Puli, _____, _____ godine

IZJAVA
o korištenju autorskog djela

Ja, _____ dajem odobrenje Sveučilištu Jurja Dobrile u Puli, kao nositelju prava iskorištavanja, da moj završni rad pod nazivom

_____

_____

_____ koristi na način da gore navedeno autorsko djelo, kao cjeloviti tekst trajno objavi u javnoj internetskoj bazi Sveučilišne knjižnice Sveučilišta Jurja Dobrile u Puli te kopira u javnu internetsku bazu završnih radova Nacionalne i sveučilišne knjižnice (stavljanje na raspolaganje javnosti), sve u skladu s Zakonom o autorskom pravu i drugim srodnim pravima i dobrom akademskom praksom, a radi promicanja otvorenoga, slobodnoga pristupa znanstvenim informacijama.
Za korištenje autorskog djela na gore navedeni način ne potražujem naknadu.

U Puli, _____ (datum)

Potpis

_____

Sadržaj

1. Uvod

U današnje vrijeme, pri izradi običnih aplikacija još uvijek moramo paziti na to kako ćemo i za koju platformu napraviti aplikaciju. Jedan od najvećih problema današnjice je ako korisnik nema pristup internetu a potrebne su mu informacije koje se nalaze na web aplikaciji. Postoji više načina rješavanja tog problema: može se preuzeti i instalirati aplikacija putem trgovine aplikacija ili se izraditi progresivna web aplikacija.[1] Tema ovog rada je opis progresivnih web aplikacija sa više različitih pogleda. Cilj rada je teorijski analazitirati i upoznati se sa progresivnim web aplikacijama koje se posljednjih godina sve više koriste, te uvidjeti koje su pozitivne a koje negativne strane ovakvih aplikacija.

Podaci za ovaj rad prikupljeni su s internetskih izvora poput: web stranica i youtube videa koji se odnose na IT konferencije. Završni rad se sastoji od 14 poglavlja. U prvom poglavlju su iznesena uvodna razmatranja. Drugo poglavlje spominje početke progresivnih web aplikacija te njihov suštinski opis. U trećem se poglavlju uspoređuju web i aplikacije te njihove sličnosti. U četvrtom dijelu govori se o različitostima između običnih i progresivnih web aplikacija, o njihovom ponašanju te o alatima koji se koriste za njihovu izradu. U petom se poglavlju govori o ciljevima koje bi progresivne web aplikacije trebale zadovoljiti. U šestom dijelu opširno se govori o prednostima i nedostacima progresivnih web aplikacija te o statističkim podacima s kojima su razne tvrtke povećale brzinu, posjećenost i dobit. U sedmome se poglavlju opisuje tkz. „`App manifest`“ koji omogućuje konfiguriranje postavki progresivnih web aplikacija. U osmome se dijelu govori o tkz. „`Developer tools-ima`“ koji nam pomažu oko otklanjanja grešaka i poboljšavanju same aplikacije. U devetom dijelu govori se o tkz. „`Service worker-ima`“ koji se koriste za izvanmrežni način rada aplikacija. U desetome se dijelu govori o 3 različita načina kojima je moguće spremiti podatke. U jedanaestom poglavlju govori se o pretvorbi postojeće web stranice u progresivnu web aplikaciju. Dvanaesto poglavlje opisuje pozadinsku sinkronizaciju kao i njezin rad. U trinaestom dijelu opisane su tkz. „`Push notifikacije`“ kao i način na koji one rade. U četrnaestom poglavlju govori se o tome što bi se postiglo ukoliko se dogodi da se progresivne web aplikacije počnu češće koristiti i kako bi upravo one trebale poboljšati izvođenje i sam osjećaj web stranica koji je korisnicima vrlo bitan.

[1] https://www.youtube.com/watch?v=C6S-SOqAX-k&t=1s

2. Što su progresivne web aplikacije ?

Sve je počelo 2015. godine člankom koji je napisao Google-ov inženjer Alex Russel. Od tada je čitava industrija počela integrirati progresivne web aplikacije u web preglednike. Tvrtka Google potiče razvoj progresivnih web aplikacija iz razloga što žele da se njihovi servisi poput Google-ovih reklama, `Google analytics-a` itd. koriste što više. Google na ovaj način želi vratiti korisnike na web.

„Progresivna web aplikacija je razvojni pristup koji koristi skup tehnologija koje omogućuju da web sadržaj bude sličan iskustvu korištenja obične aplikacije, uključujući izvanmrežni način rada, notifikacije i pristup uređaju“

-Christian Heilmann

Riječ „**progresivna**“ označava da na platformi na kojoj se aplikacija izvodi koristimo što god nam je dostupno. Ako nam je još nešto dostupno tada dodajemo i te nove funkcionalnosti.

Riječ „**web**“ označava nešto što se oslanja na web tehnologiju koristeći HTML, JavaScript i CSS.

Riješ „**aplikacija**“ nam govori da nešto možemo pokrenuti kao običnu aplikaciju na mobilnim uređajima ili kao aplikaciju na osobnim računalima.[2]

Jedna od najboljih stvari kod progresivnih web aplikacija je ta da nemaju ograničenja kao tradicionalne aplikacije, koje su ograničile mogućnost rada samo s jednom platformom. To znači da progresivne web aplikacije moraju biti progresivne što je više moguće i raditi na što je više platformi moguće. Progresivne web aplikacije trebale bi raditi na bilo kojem web pregledniku koji korisnici koriste. One se ne mogu zvati progresivnima ukoliko ne mogu funkcionirati s nekim od web preglednika ili operacijskih sustava. Faktor koji izdvaja progresivne web aplikacije od tradicionalnih aplikacija je progresivno poboljšanje.[3]

[2] https://www.youtube.com/watch?v=j807OBKHKOk
[3] https://www.youtube.com/watch?v=0OJ24kCV-J8&t=627s

3. Usporedba Web-a i aplikacija

Ako pogledamo tablicu dolje, web radi dosta sličnih stvari kao i aplikacije, međutim osjećaj na web-u još uvijek nije jednak običnoj aplikaciji. Ako pogledamo aplikacije, one se izdvajaju po mnogočemu. Obične aplikacije imaju prednost u tome što mogu koristiti najnoviju tehnologiju poput: skeniranja otiska prsta, skeniranja lica, akcelometar itd.[4]

WEB	APLIKACIJE
Može se pokrenuti na bilo kojem uređaju	Obično ekosustavno orijentirano
Brzo se otvara i koristi	Preuzimanje i instalacija
Otvara se putem web preglednika	Otvara se putem „launchera“/datoteka
Uvijek se pokreće u web pregledniku	Radi samostalno
Uvijek se pokreće u kartici	Ima svoj prozor
Vjerojatno neće raditi u izvanmrežnom načinu rada	Obično radi dobro u izvanmrežnom načinu rada
Nije optimizirano za uređaj	Snažne mogućnosti / Pristup sustavu
Spojivo	Nije spojivo
„Koristim ovo“	„Posjedujem ovo“

Tablica 1 – prikaz razlika između Web-a i aplikacija

[4] https://www.youtube.com/watch?v=bMzUPHgm8vA

4. Usporedba običnih i progresivnih web aplikacija

Najveći problem kod progresivnih web aplikacija je taj što bi se sama progresivna web aplikacija trebala ponašati kao obična aplikacija. No u praksi nije nužno da se sve progresivne web aplikacije moraju tako ponašati. Primjeri za to bili bi: blog, e-knjiga, fotografije i sl.

Progresivne web aplikacije se mogu ažurirati u pozadini dok se obične aplikacije moraju ručno ažurirati. Za potrebe korištenja obične aplikacije potrebno je otići na neku od trgovina aplikacija, skinuti aplikaciju, pričekati da se ona instalira, tek tada ju otvoriti i onda ju možemo početi koristiti. Za potrebe korištenja progresivne web aplikacije dovoljno je tek otići na neki od `URL-ova` i aplikacija se može početi koristiti jer će se ona potiho instalirati u pozadini. Ako pogledamo statistiku trgovina aplikacija poput Google Play Store-a, AppStore-a možemo primjetiti da ljudi nisu zadovoljni, ljudi ih koriste jer moraju. Većinu aplikacija koju ljudi preuzmu, deinstaliraju ih u roku prvih 3 mjeseca. 66% aplikacija na Google Play Store-u nisu nikada niti skinute. Za razliku od obične web aplikacije, progresivne web aplikacije ne moraju imati `URL bar`.

Razvoj mobilnih aplikacija jako je težak. Za razvoj mobilnih aplikacija za Android i za iOS koriste se različiti programski jezici. Za Android se najčešće koristi Java, Kotlin, React Native dok se za iOS koriste: Objective C i Swift. Razvoj mobilnih aplikacija jako je skup, osim toga postoje i različite cijene za postavljanje aplikacije na App Store ili na Google Play.

Jedna od boljih strana progresivnih web aplikacija je ta što koriste Javascript i `JSON`. Imaju većinu prednosti kao i obične aplikacije uključujući performanse. Velika prednost progresivnih web aplikacija je ta što jednom izrađena aplikacija radi na svim platformama dok se kod običnih aplikacija zasebno mora raditi aplikacija za Android i zasebno za iOS. Još jedna od velikih prednosti progresivnih web aplikacija je i samo `scroll-anje` koje je puno bolje za razliku od obične aplikacije.[5]

[5] https://www.youtube.com/watch?v=dlvevNPxHd8&t=2s

5. Ciljevi koje bi progresivne web aplikacije trebale zadovoljiti

Progresivne web aplikacije trebale bi raditi i kad konekcija na internet nije dostupna, moraju biti brze i moraju raditi na različitim uređajima jednako dobro. Poželjno je da progresivne web aplikacije koristite `HTTPS`.

Progresivne web aplikacije trebale bi biti:
Napredne
Osjetljive
Neovisne o mreži
Poput aplikacije
Ponovno osvježene
Sigurne
Lako pronađene
Ponovno zahvaćene
Instalirane
Povezane

Tablica 2 – prikaz svojstava koje bi progresivne web aplikacije trebale imati

Progresivne web aplikacije trebale bi biti napredne, responzivne, neovisne o mreži, sličiti normalnoj aplikaciji, čiste, sigurne, lako pronađene na internetu i imati mogućnost instaliranja. Sve ove navedene stvari dobro je imati u aplikaciji.[6]

[6] https://www.youtube.com/watch?v=saGLwEbcfU0&t=830s

6. Prednosti i nedostaci progresivnih web aplikacija

Dobar primjer velikih prednosti progresivnih web aplikacija je `Tinder`: smanjeno je vrijeme čekanja sa 11.91 na 4.69 sekundi te je zauzeće same aplikacije 90% manje od obične aplikacije. Još jedan dobar primjer je `Uber` progresivna web aplikacija kojoj treba manje od 3 sekunde da se otvori. Popularna aplikacija `Trivago` nakon izrađene progresivne web aplikacije povećala je korisnički angažman za čak 150% te su se klikovi na hotelske ponude povećale za 97%. Isto tako povećao se broj korisnika za 67% koji ostaju koristiti aplikaciju nakon što se prekine internetska veza. `Pinterest` je povećao korisnički angažman za 60% te se povećao prihod za 44% od oglasa koji generiraju korisnici. Vrijeme korištenja web stranice povećalo se za 40%.

Više korisnih statistika može se pronaći na web stranici: https://www.pwastats.com.
Web stranica prikazuje statistiku tvrtki koje su uz pomoć progresivnih web aplikacija poboljšali svoje poslovanje.[7]

6.1 Prednosti:

- Obične s gledišta korisnika
- Mogu raditi skoro sve sa perspektive multimedijalnih stvari poput: pokretanja videozapisa, izvanmrežnog gledanja itd.
- Koriste podatke koje je aplikacija `cache-irala` od posljednjeg puta kada je imala interakciju s internetom
- Dostupne su i kada nema interneta
- Vrlo korisne za poslovanja u kojima se koriste katalozi, gdje korisnik ne mora svaki puta osvježavati stranicu, što omogućuje i dulje zadržavanje korisnika na stranici
- Brze su zbog tehnologija s kojima su napravljene poput: `app shell-a`, `web manifest-a` i `service workers-a`
- Rade na više platforma što smanjuje i sam trošak izrade aplikacija
- Pronalazak web developera koji rade sa HTML-om, CSS-om i JavaScript-om je lakše
- Posjetitelji nisu usmjereni na trgovinu aplikacija

[7] https://www.pwastats.com/

- Zauzimaju zanemarljivo malo prostora na mobilnim uređajima
- Pošto imaju URL kao i web stranice, mogu se indeksirati i linkati, SEO tehnike su također moguće
- Imaju mogućnost prikaza push notifikacija
- Svi korisnici automatski imaju najnoviju verziju aplikacije
- Lako se mogu podijeliti
- Većina ih je sigurno jer koriste HTTPS protokol
- Brze su na sporoj internet vezi
- Na Android-u je omogućen pristup Google Assistent-u[8]

6.2 Nedostaci:

- Nedostajanje velike količine korisnika koji primarno pretražuju aplikacije u trgovinama aplikacija
- Veća potrošnja baterije
- Limitiran pristup hardware-u. Npr. ne može se pristupiti NFC-u, Bluetooth-u, Touch ID-u, Face ID-u, informacijama o bateriji, informacijama o korisniku, Proximity sensor-u, Ambient light-u, Advanced camera controls-ima itd.
- Nisu podržani u svim web preglenicima
- Prijava unutar njih koristeći drugu aplikaciju poput Facebook-a, Instagrama ili slično nije podržana
- Nije podržana integracija sa Siri asistenticom na iOS-u[9] [10]

[8] https://vaadin.com/pwa/learn/pros-and-cons
[9] https://clutch.co/app-developers/resources/pros-cons-progressive-web-apps
[10] https://jaba.com.au/blog/pros-and-cons-of-progressive-web-apps

Odlika	Progresivna web aplikacija	Obična aplikacija
Radi izvanmrežno	✓	✓
Mobilno specifična navigacija	✓	✓
Push notifikacije	✓	✓
Pristup početnom zaslonu	✓	✓
Nije potrebno preuzimanje	✓	✗
Zaobilaženje trgovine aplikacija	✓	✗
Povezljiv i dijeljiv	✓	✗
Indeksiran od strane Google-a	✓	✗
Niski zahtjevi za podacima	✓	✗
Ne zahtijeva ažuriranja	✓	✗

Tablica 3 – usporedba progresivne web aplikacije i obične aplikacije

7. App manifest

Web app manifest je JSON datoteka koja nam omogućuje konfiguriranje postavki naše progresivne web aplikacije, spojena je u head područje HTML datoteke.

U JSON datoteci možemo specificirati sve postavke koje korisnici vide kada pokrenu našu aplikaciju. Neke od stvari koje možemo specificirati u JSON datoteci mogu biti: zadana boja pozadine pri učitavanju aplikacije, ikona koja se instalira na početni zaslon, ime aplikacije, URL koji će se otvoriti kada korisnik klikne na ikonu koja se nalazi na pozadini, kako će se aplikacija otvoriti kada se prvi puta pokrene (da li će se prikazati adresna traka ili ne).

```
{
        "name": "Squoosh",
        " start_url": "/",
        "scope": "/",
        "display": "standalone",
        "background_color": "#fff",
        "theme_color": "#f78f21",
        "icons": [ ... ]
}
```

Slika 1 – primjer App manifest-a koji je Google-ov tim izradio za squoosh.app (aplikaciju za kompresiranje slika)

Name, *start_url* i *icons* koriste se za izradu prečaca koji omogućuje pokretanje aplikacije. Display govori web pregledniku na koji način se aplikacija treba otvoriti kada se pokrene (u ovom slučaju) standalone, pri čemu se otvara prozor aplikacije nezavisan o web pregledniku. Postoje i ostali načini prikaza, međutim standalone je jedan od najpoželjnijih. Theme_color omogućuje uporabu različitih boja za naslovnu traku aplikacije. Scope određuje opseg aplikacije, on je važan za točno ponašanje povezivanja. Preglednik može presresti navigaciju koja spada u opseg naše aplikacije te tada usmjeriti veze da se otvore u prozoru naše aplikacije.
Kada se izradi manifest, moramo dodati vezu (*eng. link*) na sve stranice web aplikacije.

```
<link rel="manifest" href="/manifest.json">
```

Kada se aplikacija prvi put pokrene može proći neko vrijeme dok web preglednik učita web aplikaciju. Umjesto da se prikazuje bijeli ekran (koji korisniku može izgledati kao da je aplikacija stala), web preglednik Chrome će pokazati tkz. „splash screen“ prije prve pozadinske boje. Chrome će automatski napraviti splash screen uzimajući u obzir postavke manifest-a u kojemu su navedeni: ime, pozadinska boja i ikona web aplikacije.[11]

8. Developer tools kod progresivnih web aplikacija

Browser developer tools-e koristimo kako bi nam pomogli debug-irati progresivne web aplikacije. Pošto je Google Chrome jedan od najpopularnijih web preglednika danas, u nastavku ćemo govoriti najviše o njemu.

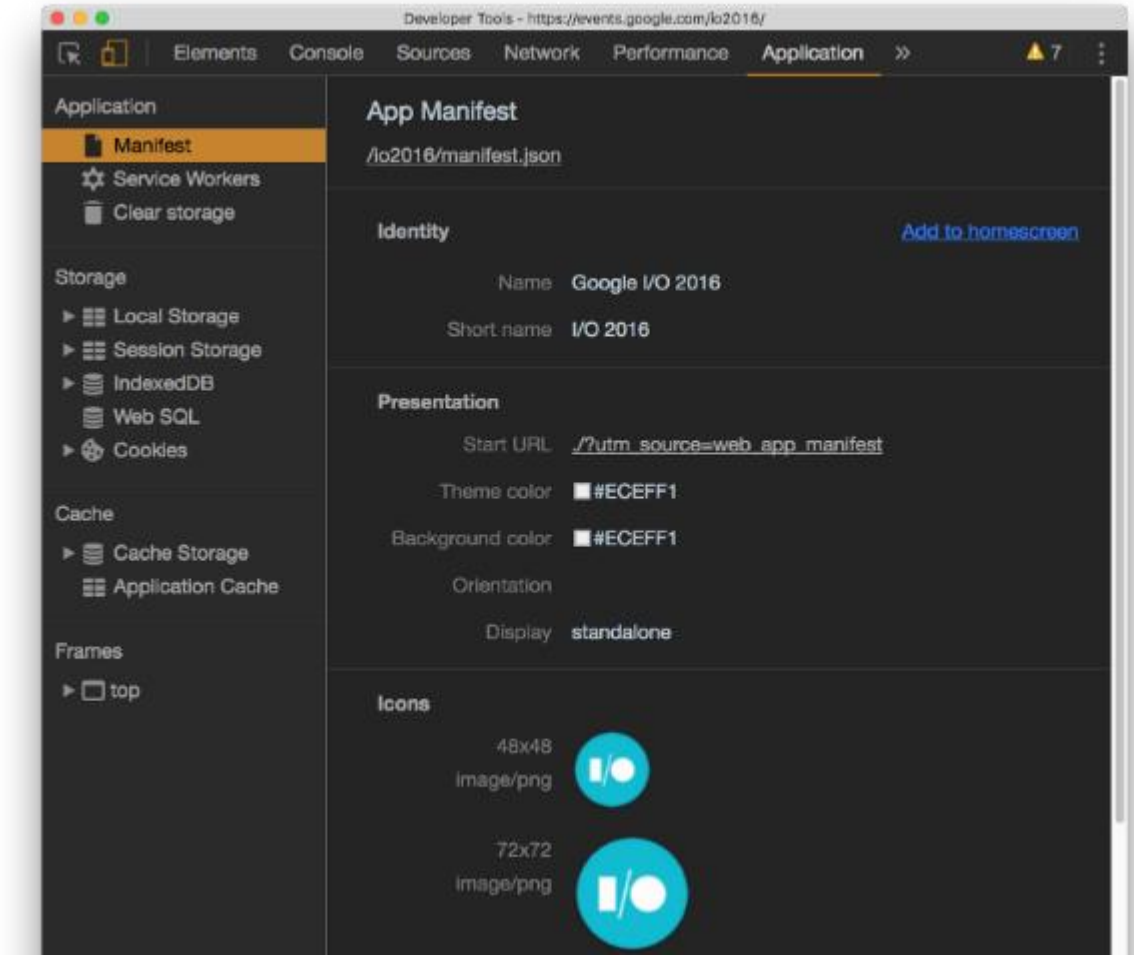
Kada otvorimo developer tools-e, možemo primjetiti nekoliko panela poput: Elements, Console, Sources, Network, Performance, Application. Pokazalo se da ljudi koji izrađuju web stranice ili web aplikacije najviše koriste: Console, Elements i Network panele. Application panel grupira više elemenata koji su ključni za progresivnu web aplikaciju.[12]

8.1 Manifest

[11] https://developers.google.com/web/fundamentals/web-app-manifest/
[12] https://www.raymondcamden.com/2017/10/17/devtools-tips-for-pwas

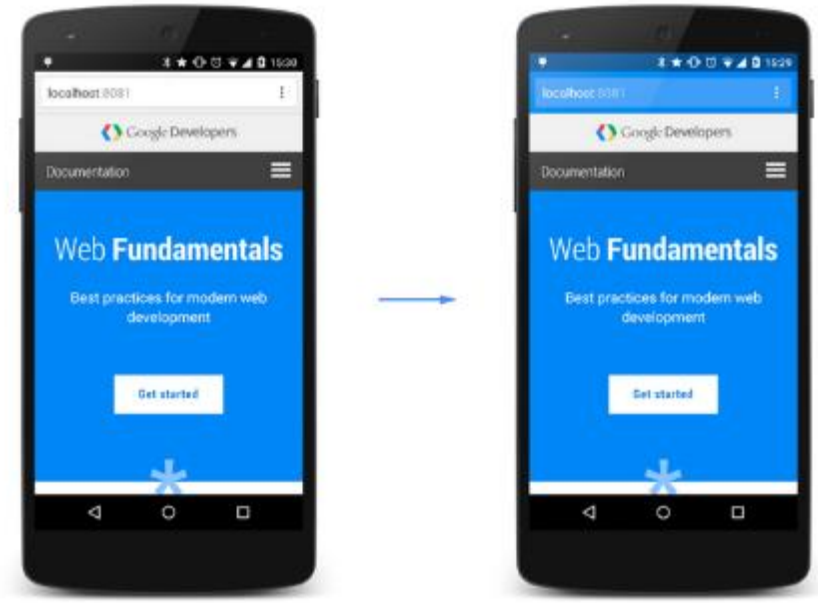
`Manifest` omogućuje tkz. „`Add to home screen`“ opciju. On pruža detalje o tome kako bi se aplikacija trebala ponašati jednom kada se instalira na mobitel. Ako nešto pođe po zlu oko definiranja `manifest-a`, javit će se pogreška.

U `manifest` panelu možemo vidjeti ime aplikacije, kraće ime za pozadinu i pregled ikona u različitim veličinama. Kod prezentacijskog djela imamo stvari poput: `StartURL-a`, `Theme color-a`, `Background color-a`, `Orientation-a`, `Display-a`.



Slika 2 – prikaz App manifest sučelja

`StartURL` je `URL` koji će uređaj učitati kada korisnik pokrene web aplikaciju sa pozadine. `Theme color` ukazuje na temu stranice. Google Chrome koristi tu mogućnost kako bi obojio neke djelove `UI-a` web preglednika kao što je to npr. adresna traka. `Background color` omogućuje specificiranje pozadinske boje web aplikacije. Takva pozadina prikazuje se kao ekran učitavanja prije nego što CSS bude dostupan. Ovakav način pospješuje bolje iskustvo za korisnika. Nakon što je CSS dostupan, pozadinska boja učitavanja je prebrisana sa stvarnim web dizajnom.



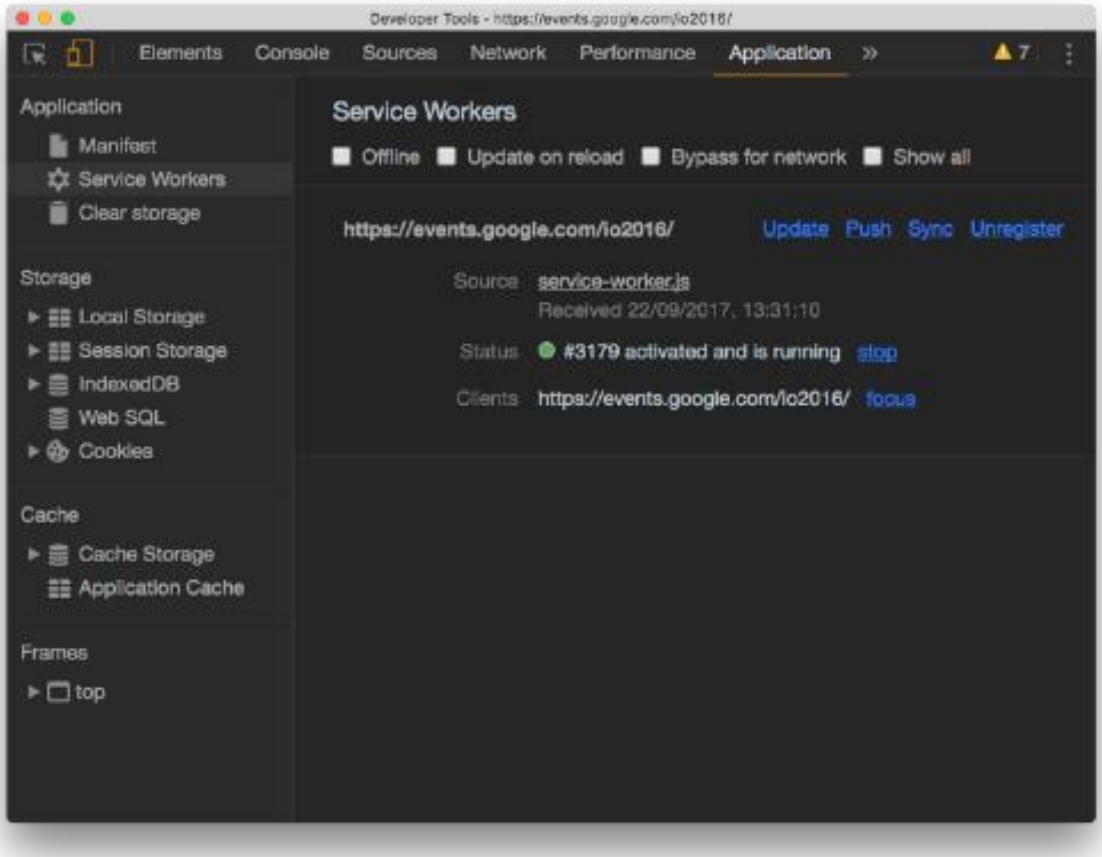
Slika 3 – prikaz korisničkog sučelja bez i sa korištenom temom

`Orientation` specificira zadanu orijentaciju koja može biti: `any`, `natural`, `landscape` i `portrait`. `Display` definira na koji način će aplikacija biti predstavljena. Postoji mogućnost `fullscreen` koja otvara aplikaciju kroz čitavu veličinu zaslona uređaja. `Standalone` prikazuje standardnu statusnu traku i sistemski `back` gumb. `Minimal-ui` pruža korisniku `back`, `forward` i `reload button-e`. `Browser` prikazuje normalni `UI` web-a koji uključuje i adresnu traku.

Zadnja mogućnost je `Add to home screen link`. Na računalima se pokreće `Add to home screen` opcija odmah nakon učitavanja web aplikacije koja nam omogućuje dodavanje aplikacije na policu aplikacija.
Na mobilnim uređajima web preglednik traži da se aplikacija instalira tj. da se aplikacija doda na početni zaslon.[13]

8.2 Service workers

`Service workers` je tehnologija koja omogućuje progresivnoj web aplikaciji da radi izvanmrežno. Omogućuju presretanje mrežnih zahtjeva i koristi `Cache API` za lokalno spremanje resursa.



Slika 4 – Prikaz Service workers sučelja

[13] https://www.raymondcamden.com/2017/10/13/some-pwa-tips/

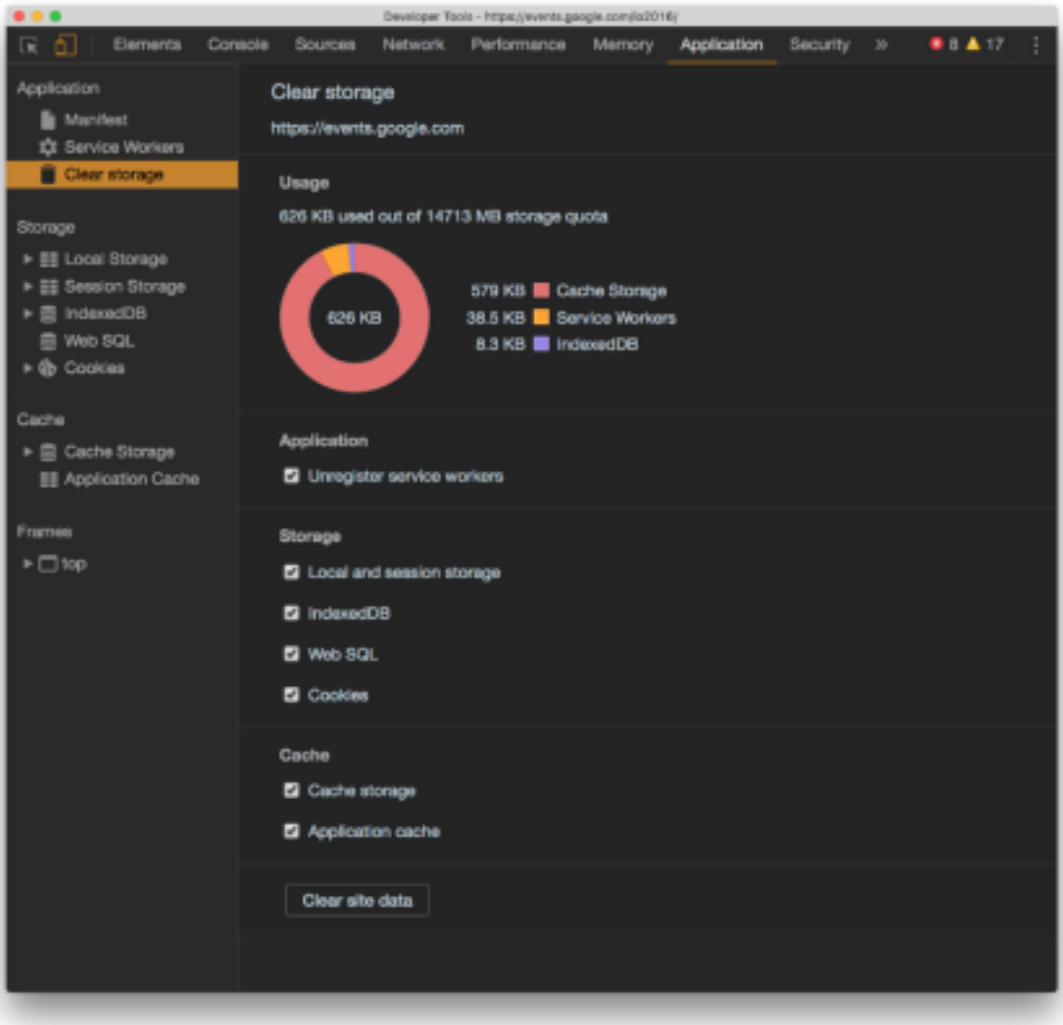
U service workers postavkama imamo mogućnost forsiranja izvanmrežnog načina rada. Update on reload je vrlo koristan pri otklanjanju grešaka.
Service workers-i su instalirani na uređaju pri prvom početku samog učitavanja te se ne ažuriraju sve dok se ne promijeni kod. No i da ažuriramo service worker, on se neće koristiti na web aplikaciji sve dok se stari service worker ne obriše tj. dok korisnik ne zatvori sve kartice koji se odnose na web aplikaciju. Bypass for network jako je koristan pri otklanjanju grešaka. On omogućuje da se u potpunosti isključi caching koji je omogućen sa Service Worker-om. To onemogućuje aplikaciji da koristi cache-irane resurse kada želimo direktan pristup prema mreži.
Show all je opcija koja omogućuje brzi pristup svim instaliranim Service Workers-ima na uređaju. Service workers-i su na listi i imaju statusni indikator. Svaki od service worker-a možemo zaustaviti ili ponovno pokrenuti.

Od svih Service worker mogućnosti najviše se koristi Service worker lifecycle events simulation. Kod njih se mogu koristiti sljedeći eventi:

- Update – forsirat će ažuriranje Service worker-a
- Push – prati push event
- Sync – prati pozadinski sync event, omogućuje korisniku izvanmrežne akcije i komunikaciju prema server-u kada se konekcija na internet ponovno ostvari.
- Unregister – poništava registraciju Service worker-a, tako da se može početi sa čistim stanjem.

8.3 Clear storage

`Clear storage` pokazuje ukupnu veličinu mjesta za pohranu koja se koristi kod web aplikacije. Isto tako pokazuje koliko slobodnog prostora je još ostalo te postoji mogućnost odabira koju pohranu obrisati.



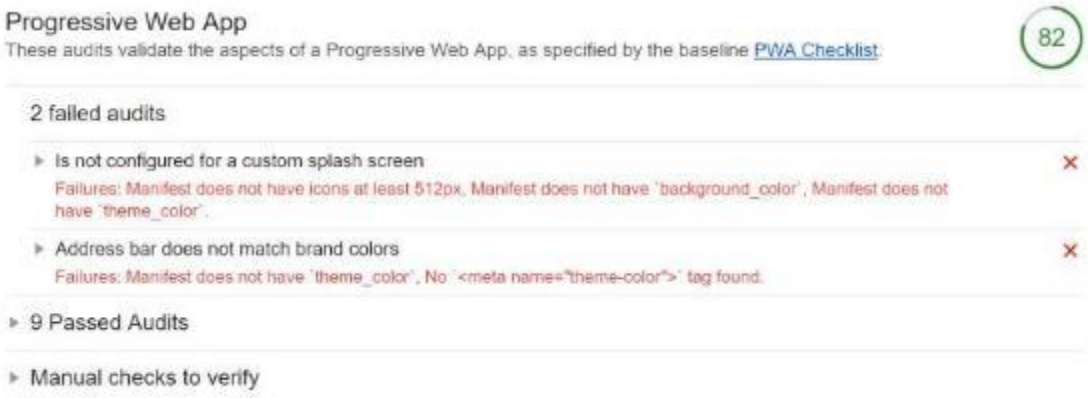
Slika 5 – Prikaz Clear storage sučelja

8.4 Storage

`Storage` sadži alate koji nam omogućuju interakciju s uobičajnim opcijama za pohranu poput `Local/Session Storage`, `IndexedDB` i `Cookies`.

8.5 Lighthouse

`Lighthouse` je alat koji provjerava razne aspekte web stranice. Lighthouse omogućuje bodovanje performansi i ostalih stvari koje su bitne kako bi progresivna web aplikacija bila što bolja. Lighthouse je uključen u same `developer tools-e` tako da se može pokrenuti direktno u Google Chrome-u.

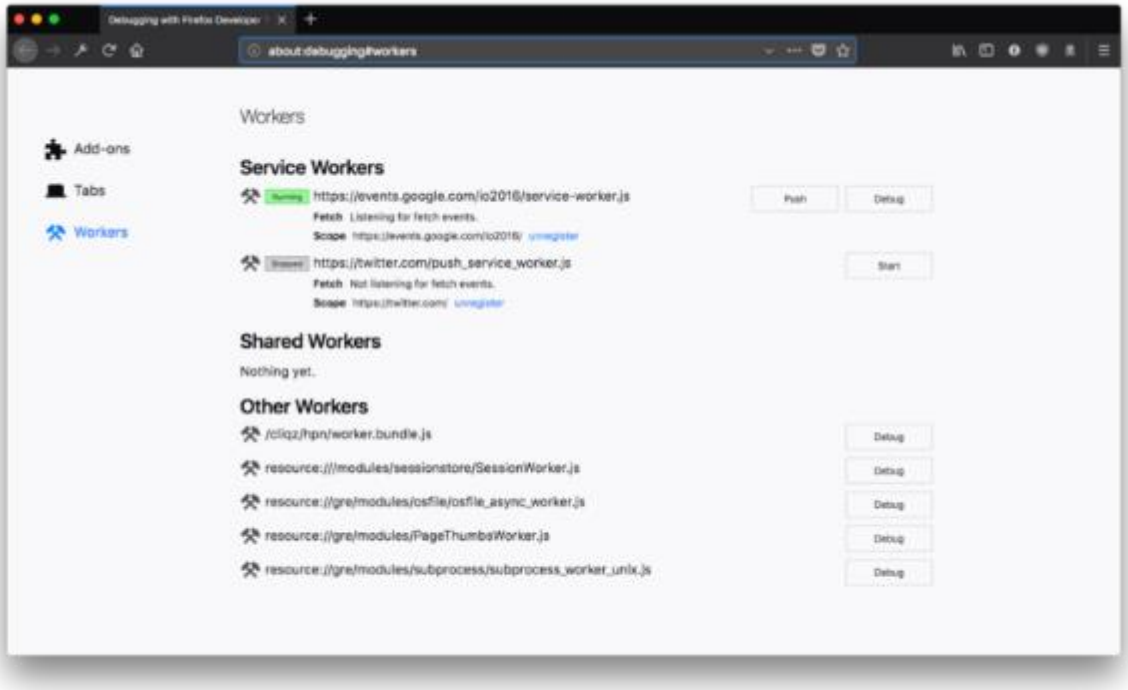


Slika 6 – Prikaz bodova i nekih od svojstava koji nisu zadovoljeni u Lighthouse-u

8.6 Firefox web preglednik

Firefox podržava progresivne web aplikacije kao i `Service Worker-e`, no za razliku od Google Chrome-a ne prikazuje `developer tools-e` jednako dobro. Kod Firefox `developer tools-a` imamo mogućnost poništiti registraciju svakog `Service`

worker-a i otvoriti kod u debugger-u za bilo koji tip worker-a. Isto tako možemo pokrenuti i Push API push event kako bi debugg-irali push evente.
U vrijeme pisanja ovog teksta nije postojala mogućnost simuliranja evenata, forsiranog ažuriranja ili zaobilaska Service worker-a kao što je to slučaj u Google Chrome-u.[14]



Slika 7 – Prikaz Developer tools-a u web pregledniku Firefox

9. Service worker API

Service worker je mrežni proxy koji koristi mrežne zahtjeve. Ovo je tehnologija koja omogućuje da se naše web aplikacije izvode izvanmrežno i tehnologija koja omogućuje da naše web aplikacije budu što više nalik običnim aplikacijama. U praksi se pokazalo da je pri razvoju aplikacije puno bolje raditi prije offline aplikacije nego online. Pri razvoju offline aplikacija moramo uzeti u obzir da ne radimo aplikacije poput aplikacija za: financijsko praćenje, cijene na burzama, vrijeme dolaska vlakova (ako se radi o praćenju dolazaka vlakova u živo). Ne želimo takve podatke cache-irati na 5 minuta.

[14] https://www.freecodecamp.org/news/how-to-debug-progressive-web-apps-using-browser-developer-tools-bad1cd3db784/

Korisnicima za koje radimo progresivnu web aplikaciju vrlo je bitno napomenuti da su u izvanmrežnom načinu rada jer oni to možda neće sami primjetiti. Većina današnjih modernih web preglednika koristi `Service worker API`.

`Service Worker` radi u zasebnoj izvršnoj dretvi. Nema pristup `DOM-u` no postoji `API` koji se naziva `postMessage` i koji omogućuje propuštanje poruka između `service workers-a` i javascript frontend-a.

Progresivne web aplikacije imaju mogućnost interakcije sa svim mrežnim zahtjevima. `Service workers-i` se najčešće koriste za upravljanje memorijom tj. `cache-om` ali može se koristiti i za ostale primjene. Kako bi se `service worker` implementirao potrebno je dio javascript koda dodati u `head` područje html datoteke.

Na donjoj slici prikazan je početni zahtjev prema našem `API-ju`. Pri pokretanju dohvaćaju se podaci, pri povratku se podaci spremaju te se oni tada prikazuju korisnicima, korisnici tada imaju interakciju sa web stranicom (osvježe stranicu ili odu na neku od drugih stranica kojoj su potrebni isti podaci). Zahtjev se potom ugasi no ovaj puta koristi ga `service worker` i odlučuje kako će odgovoriti na njega. Ako on već ima podatke onda će odgovoriti sa podacima spremljenih u `cache-u`, ako ne onda će potražiti podatke sa server-a te će se tada ažurirati korisničko sučelje.

Slika 8 – prikaz koraka od početnog zahtjeva, service worker-a do ažuriranja korisničkog sučelja

Service worker se nalazi između web preglednika i mreže, ponašajući se poput proxy poslužitelja. Service workeri prestaju raditi kada se ne koriste i ponovno se pokreću kada je to potrebno. Ovakvo ponašanje omogućuje im smanjivanje opterećenja procesora i manju potrošnju baterije.

Service worker-i zahtijevaju da web aplikacija koristi HTTPS protokol kako bi se spriječili tkz. „man-in-the-middle“ napadi. HTTPS spriječava mnoge uobičajene napade.[15]

Service worker mora biti registriran od strane web stranice. Budući da neki web preglednici još uvijek ne podržavaju service worker-e trebala bi se izvršiti provjera značajki prije registracije service worker-a.
Service workeri prolaze kroz tri koraka tijekom svog životnog ciklusa:

- **Registracija** – govori web pregledniku gdje se nalazi service worker i započinje instalaciju u pozadini

- **Instalacija** – pokreće se „install“ događaj

- **Aktivacija** - aktivira se samo kad nema više učitanih stranica koje i dalje koristi stari service worker-i

Za vrijeme pisanja ovog teksta svi moderni preglednici su podržavali service worker-e, barem cache-iranje. Chrome, FireFox i Edge web preglednici podržavaju izvorne push notifikacije dok Chrome jedini ima podršku za sinkronizaciju u pozadini.

[15] https://techbeacon.com/app-dev-testing/how-use-service-workers-progressive-web-apps

Neke od stvari koje service workeri ne mogu zadovoljiti su sljedeće:

- Ne mogu pristupiti objektu prozora koji se nalazi u korisnickom sučelju
- Zahtijevaju HTTPS
- Samo su asinkroni [16]

Service workeri koriste API-ove kako bi web aplikacije učinili više nalik običnim aplikacijama. Neki od API-a su:

- **Notification API** - način prikazivanja i interakcije s obavijestima pomoću obavijesti operativnog sustava.

- **Push API** - API koji aplikaciji omogućuje pretplatu na push notifikacije i primanje push notifikacija

- **Background Sync API** - Omogućuje odgađanje radnji sve dok korisnik nema stabilnu povezanost

- **Channel Messaging API** – omogućuje komunikaciju između web workers-a i service worker-a

Service workeri su još relativno novi na web-u te pružaju novu dimenziju mogućnosti koju programeri tek počinju učiti. Model programiranja service worker-a znatno se razlikuje od tradicionalnog web programiranja.[17]

[16] https://developers.google.com/web/ilt/pwa/introduction-to-service-worker
[17] https://love2dev.com/blog/what-is-a-service-worker/

10. Spremanje podataka

Kod progresivnih web aplikacija postoje tri glavna načina spremanja podataka: Local storage, Cache i IndexedDB. U Cache se najčešće spremaju statički podatci dok se kod IndexedDB-a spremaju dinamički podatci. Npr. slike, datoteke poput: .html-a, .css-a, .js-a spremaju se u Cache dok se u IndexedDB spremaju dinamički JSON podatci. Local storage vrlo je popularan način spremanja podataka najviše zbog toga što je lak za korištenje. Jako je sličan Service worker cache-u ali radi na sinkronizirani način, tako da ne radi unutar service worker-a. U daljnjem tekstu opisani su samo kompleksniji načini spremanja podataka poput IndexedDB-a i Cache-a.[18]

10.1 IndexedDB

IndexedDB radi asinkrono, najčešće se koristi za spremanje velikih količina podataka. Jako je dobar u brzom traženju podataka radi indexa koje koristi. Podaci se spremaju najčešće u JSON formatu međutim tome ne mora biti tako. IndexedDB nije relacijska baza podataka i ne koristi SQL upitni jezik.

Nakon što otvorimo bazu podataka imamo mogućnost pohraniti podatke direktno ili napraviti objektna spremišta. Svaka baza podataka može imati više objektnih spremišta. Npr. možemo imati objektno spremište „korisnik“ koje sadržava sve podatke o korisniku. U IndexedDB podatci su spremljeni kao parovi ključa i vrijednosti. Svaki od objektnih spremišta može imati svoj primarni ključ koji unikatno indentificira svaki objekt u spremištu.

IndexedDB podržava uobičajne CRUD operacije poput: Create, Retrieve, Update i Delete. Koriste se operacije poput: add, get, put, delete, getAll, cursor. Sve ove navedene operacije su uobičajene osim cursor-a. Cursor omogućuje izdvajanje vrijednosti po indeksu ili ključu.
Indexed DB omogućuje dobivanje blok podataka na temelju ključa ili indeksa.

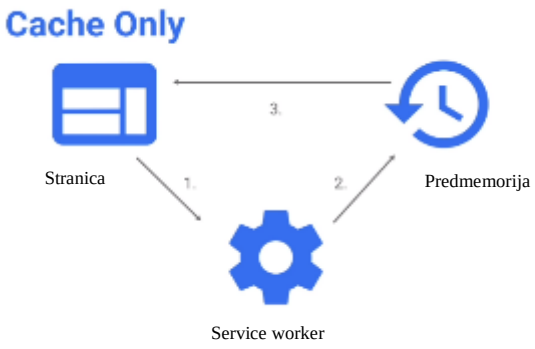
[18] https://www.youtube.com/watch?v=VNFDoawcmNc

`IndexedDB` omogućuje snimanje akcija koje korisnik odradi dok je u izvanmrežnom načinu rada na način da se akcije spreme u `IndexedDB`. Kada korisnik ponovno ostvari konekciju na internet, akcije koje je korisnik prethodno odradio se izvršavaju.[19]

10.2 Cache API

`CacheAPI` daje nam potpunu kontrolu nad `cache-iranjem` resursa. `Caching` radimo kako bi poboljšali performanse aplikacije. Postoji više strategija `cache-iranja` poput: `Cache only`, `Network only`, `Cache first`, `Network first`, `Cache then network`, `Stale while revalidate`.

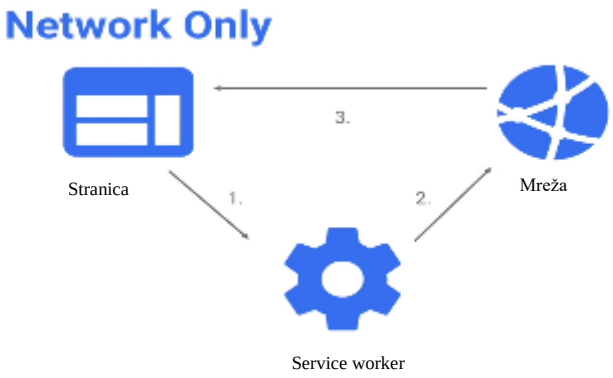
10.2.1 `Cache only` - koristi samo `cache`, idealno za stranice koje su statičke



Slika 9 - prikaz Cache Only strategije

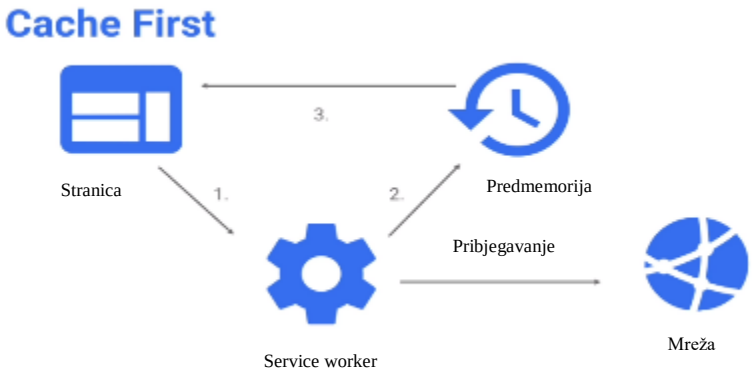
[19] https://www.youtube.com/watch?v=k1eoekN3nkA

10.2.2 `Network only` – suprotan pristup `Cache only-u`, koristi se u slučajevima kada nema `offline` ekvivalenta poput analitike, `pings-ova` ili `non-GET` zahtjeva koji traži dinamički odgovor od strane servera.



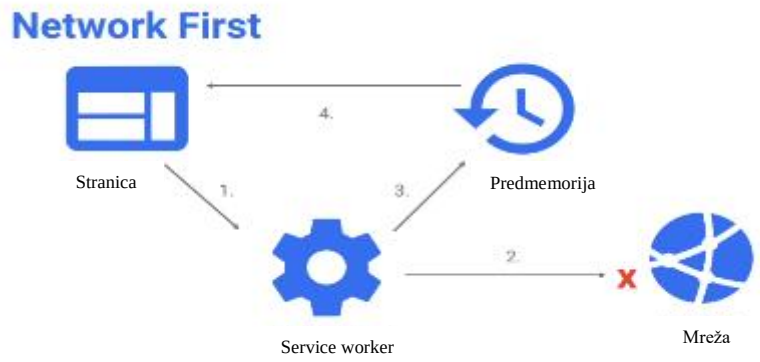
Slika 10 - prikaz Network Only strategije

10.2.3 `Cache first` – strategija koja je poznata i pod nazivom „`falling back to the network`“. `Service worker` prvo gleda da li je nešto spremljeno u `cache-u`, ako resurs nije pronađen tada se okreće prema mreži.



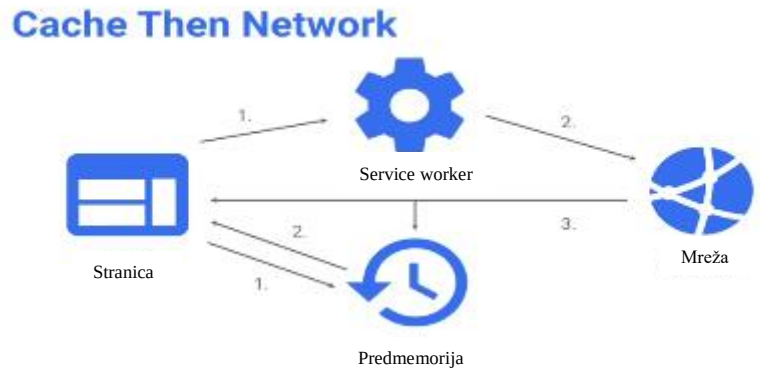
Slika 11 - prikaz Cache First strategije

10.2.4 `Network first` – strategija koja je poznata i pod nazivom „`network falling back to cache`“. Ova strategija idelna je za popravljanje resursa koji se često ažuriraju bez obzira na verziju web stranice.



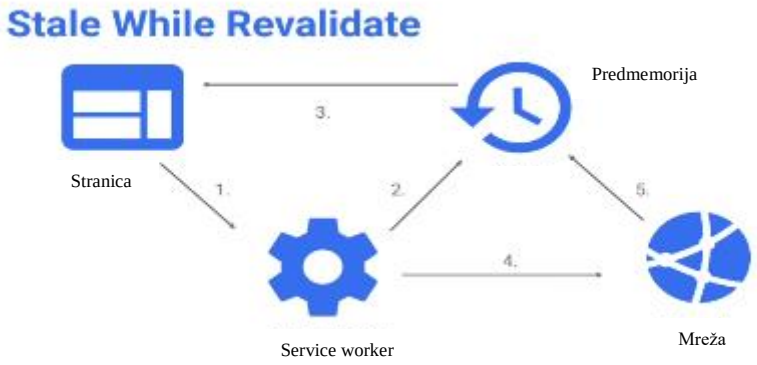
Slika 12 - prikaz Network First strategije

10.2.5 `Cache then netowrk` – ovaj pristup obilazi `cache` i mrežu. Prvo koristi odgovor `cache-a`, zatim ažurira stranicu i `cache` nakon što mreža odgovori. Ovakav pristup najviše se koristi za sadržaj koji se često ažurira. `Service worker` radi dva zahtjeva, jedan prema `cache-u` i jedan prema mreži. Ideja je prvo prikazati `cache-irane` podatke te tada ažurirati stranicu kada ili ako mrežni podaci stignu.



Slika 13 - prikaz Cache Then Network strategije

10.2.6 `Stale while revalidate` – strategija idealna za često ažuriranje resursa, gdje nije bitno da li je stranica ažurirana na zadnju verziju. Sa ovom strategijom stranica se ne mora ažurirati svaki puta kada se vrate podaci s mreže, već se ažurira samo `cache`.[20]



Slika 14 - prikaz Stale While Revalidate strategije

[20] https://www.youtube.com/watch?v=FY1r7Y5rf2E

11. Pretvorba postojeće web stranice u progresivnu web aplikaciju

1. korak

U `JSON`-u formatu određujemo ime aplikacije, ikonu koja će se pojaviti na početnom zaslonu, `theme colors` itd.

```
1.  +{
2.  + "name": "Mikemjharris Blog",
3.  + "short_name": "Blog",
4.  + "icons": [{
5.  +       "src": "images/favicon.png",
6.  +              "sizes": "20x20",
7.  +              "type": "image/png"
8.  + }],
9.  + "start_url": "/",
10. + "display": "standalone",
11. + "background_color": "#5cd65c",
12. + "theme_color": "#5cd65c"
```

Slika 15 – Prikaz JSON formata za određivanje osnovnih svojstava vezanih uz aplikaciju

```
13. +}
```

2. korak

U `head` dio HTML-a povezujemo `JSON` i dodajemo boju za navigacijsku traku.

```
1.  +       <link rel="manifest" href="/manifest.json">
```

Slika 16 – Prikaz povezivanja JSON-a i određivanje boje za navigacijsku traku u HTML-u

```
2.  +       <meta name="theme-color" content="#5cd65c">
```

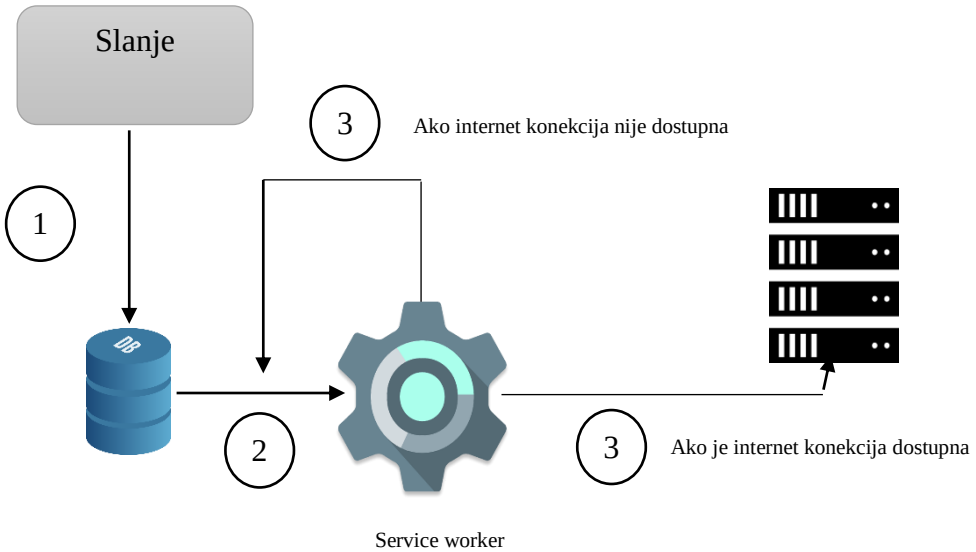
3. korak

Service worker koristi JavaScript kod i `Proxy` server koji komunicira između web preglednika i servera. Upravljanje `cache-om` je programsko. U osnovi, imamo pristup svemu što je na frontend-u.[21]

12. Pozadinska sinkronizacija

Pozadinska sinkronizacija (*eng. Background sync*) je dio `service worker-a` koji omogućuje ponovno pokretanje mrežnih operacija ukoliko aplikacija uspostavi vezu prema mreži. Ovakva opcija omogućuje korisnicima da nesmetano čitaju ili pišu dok nemaju pristup internetu. Neki od primjera korištenja `background sync-a` su: slanje email-ova ili učitavanje slika kada korisnik dobije vezu na mrežu.[22]

`Background sync` radi na način da korisnik prvo napravi zahtjev. Prije nego zahtjev bude prihvaćen, on dolazi do `service worker-a`. `Service worker` provjerava da li korisnik ima internet konekciju, ako ima zahtjev će biti poslan. Ako se desi da korisnik još nije uspostavio konekciju na internet, `service worker` će čekati sve dok se to ne desi. Kada se uspostavi konekcija na internet tada će `service worker` proslijediti poslani zahtjev nakon što uzme podatke iz `IndexedDB-a`. Korisna mogućnost `background sync-a` je ta što će on poslati zahtjev i u slučaju da korisnik napusti stranicu.[23]



Slika 17 - Prikaz rada pozadinske sinkronizacije

[21] https://www.youtube.com/watch?v=vhg01MI-8pI&t=371s
[22] https://davidwalsh.name/background-sync
[23] https://vaadin.com/pwa/learn/background-sync

13. Push notifikacije

Push notifikacije su poruke koje se šalju od servera do pretplaćenog klijenta. Omogućuju obavještavanje korisnika i kad oni ne koriste aplikaciju. Rade u pozadini čak i kada je web preglednik zatvoren. Neki od primjera push notifikacija su: obavijest da je paket uspješno poslan, obavijest da je korisnik dobio novu poruku i sl. Push notifikacije su dobar marketinški alat za poslovanje jer pomažu pri zadržavanju kontakata s korisnicima.[24]

Push notifikacije rade na način da web aplikacija izbaci iskočni popup prozor koji pita korisnika da li se želi pretplatiti na notifikacije. Korisnik ima mogućnost prihvaćanja ili odbijanja pretplate na push notifikacije. Service worker push manager odgovoran je za pretplatu na koju se korisnik pretplatio. Kako bi se poruke sa servera uspješno poslale određenim korisnicima koristi se ID korisnika. Svaki korisnik može imati prilagođeno iskustvo na temelju pretplatničkog ID-a. Service worker uz pomoć push-a osluškuje i spreman je prihvatiti bilo koje nadolazeće poruke. Kada uređaj dobije push notifikaciju, poruka se prikazuje na zaslonu. Ako korisnik nije aktivan na svom mobilnom uređaju, uređaj će prikazati notifikaciju na zaslonu zaključavanja.[25]

[24] https://auth0.com/blog/introduction-to-progressive-web-apps-push-notifications-part-3/
[25] https://love2dev.com/pwa/push-notifications/

14. Zaključak

Ne smijemo odustati od običnih aplikacija još, međutim jednog dana bi smo se toga trebali riješiti skroz. Najveća stvar koja bi se trebala postići ako se sve aplikacije pretvore u progresivne web aplikacije je ta da bi se trebalo smanjiti `HTTP` zahtjeva tj. trebao bi se smanjiti promet između frontend-a i backend-a. Progresivnu web aplikaciju možemo jednom izraditi i radit će na svim platformama. Ova tehnologija web standarda poboljšati će izvođenje i osjećaj web stranica.

Literatura

1. https://www.youtube.com/watch?v=C6S-SOqAX-k&t=1s
2. https://www.youtube.com/watch?v=j807OBKHKOk
3. https://www.youtube.com/watch?v=0OJ24kCV-J8&t=627s
4. https://www.youtube.com/watch?v=bMzUPHgm8vA
5. https://www.youtube.com/watch?v=dlvevNPxHd8&t=2s
6. https://www.youtube.com/watch?v=saGLwEbcfU0&t=830s
7. https://www.pwastats.com/
8. https://vaadin.com/pwa/learn/pros-and-cons
9. https://clutch.co/app-developers/resources/pros-cons-progressive-web-apps
10. https://jaba.com.au/blog/pros-and-cons-of-progressive-web-apps
11. https://developers.google.com/web/fundamentals/web-app-manifest/
12. https://www.raymondcamden.com/2017/10/17/devtools-tips-for-pwas
13. https://www.raymondcamden.com/2017/10/13/some-pwa-tips/
14. https://www.freecodecamp.org/news/how-to-debug-progressive-web-apps-using-browser-developer-tools-bad1cd3db784/
15. https://techbeacon.com/app-dev-testing/how-use-service-workers-progressive-web-apps
16. https://developers.google.com/web/ilt/pwa/introduction-to-service-worker
17. https://love2dev.com/blog/what-is-a-service-worker/
18. https://www.youtube.com/watch?v=VNFDoawcmNc
19. https://www.youtube.com/watch?v=k1eoekN3nkA
20. https://www.youtube.com/watch?v=FY1r7Y5rf2E
21. https://www.youtube.com/watch?v=vhg01Ml-8pI&t=371s
22. https://davidwalsh.name/background-sync
23. https://vaadin.com/pwa/learn/background-sync
24. https://auth0.com/blog/introduction-to-progressive-web-apps-push-notifications-part-3/
25. https://love2dev.com/pwa/push-notifications/

Slike:

Tablice:

Sažetak

Trenutno se puno govori o progresivnim web aplikacijama. Česta je tema u JavaScript zajednici. Može se reći da su progresivne web aplikacije zlatna sredina između responzivne web-aplikacije i obične aplikacije. One su manje zahtjevne aplikacije dizajnirane za prilagodbu bilo kojem uređaju, rade izvan mreže kad je to potrebno te se ponašaju kao obične aplikacije.

U ovome radu istražiti ćemo što je progresivna web aplikacija te koje su njihove prednosti i ograničenja. Zatim ćemo malo dublje istražiti tehnologije koje stoje iza progresivnih web aplikacija poput service workera, posebno ćemo istražiti mehanizam koja stoji iza service workers-a i kako će utjecati na izradu backend-a. Proučit ćemo razvoj izvanmrežnog načina rada te alate i tehnologije potrebne za izradu izvanmrežnih aplikacija. Usporediti ćemo izvanmrežne mehanizame za pohranu i njihova ograničenja. Progresivne web aplikacije nisu samo za mobilne uređaje, omogućuju isporuku visokokvalitetnih i sposobnih aplikacija i na Windows, Mac, Linux i Chrome OS uređaje.

Razlika između onoga što web može učiniti u odnosu na ono što može učiniti obična aplikacija stvarno je mala. Izrada ovakve aplikacije mogle bi biti korisne polaznicima koji imaju ideju za softver koji bi željeli izraditi ali ne razumiju kako današnji kompleksni web funkcionira. Progresivne web aplikacije nisu samo nešto što je novo, već su to web-bazirane aplikacije koje imaju velikih potencijala.

Ključne riječi: Progresivne web aplikacije, izvanmrežni razvoj, web, obična aplikacija, web-bazirane aplikacije

Summary

There is a lot of talk about Progressive Web Apps at the moment. It is a common topic in the JavaScript community. Progressive web apps are a happy middle ground between a responsive web app and a native app. They are lightweight apps designed to conform to any device, work offline, and when appropriate, feel like a native app.

We will explore what a Progressive Web App is. We will look at the benefits of Progressive Web Apps and more importantly their limitations. We then look a bit deeper at the technologies behind Progressive Web Apps such as Service Workers, in particular we will look at the mechanics behind Service Workers, and how they will affect back-end builds. We will look at offline development, the offline first life-cycle and the tools and technologies needed to start building offline applications. Comparing offline storage mechanisms, and their limitations. Progressive Web Apps aren't just for mobile any more, they make it possible to deliver high quality, capable apps on Windows, Mac, Linux and Chrome OS.

The gap of what the web can do versus what must be done in a native application is really small these days. This could possibly connect some of the dots for attendees that have ideas for software they'd like to build, but not necessarily understand just how far the web has come. Progressive Web Apps aren't just a worthwhile endeavor because it's something new. These web-based apps hold a ton of potential.

Keywords: Progressive Web Apps, offline development, web, native application, web-based apps