Progresivne web aplikacije

Težak, Filip

Undergraduate thesis / Završni rad

2019

Degree Grantor / Ustanova koja je dodijelila akademski / stručni stupanj: **University of Zagreb, Faculty of Organization and Informatics / Sveučilište u Zagrebu, Fakultet organizacije i informatike**

Permanent link / Trajna poveznica: https://urn.nsk.hr/urn:nbn:hr:211:429237

Rights / Prava: Attribution-NonCommercial 3.0 Unported/Imenovanje-Nekomercijalno 3.0

Download date / Datum preuzimanja: **2024-03-29**

SVEUČILIŠTE U ZAGREBU
FAKULTET ORGANIZACIJE I INFORMATIKE
VARAŽDIN

Repository / Repozitorij:

Faculty of Organization and Informatics - Digital Repository

zir.nsk.hr

DIGITALNI AKADEMSKI ARHIVI I REPOZITORIJI

SVEUČILIŠTE U ZAGREBU

FAKULTET ORGANIZACIJE I INFORMATIKE

V A R A Ž D I N

Filip Težak

Progresivne web aplikacije

ZAVRŠNI RAD

Varaždin, 2019.

SVEUČILIŠTE U ZAGREBU

FAKULTET ORGANIZACIJE I INFORMATIKE

V A R A Ž D I N

Filip Težak

Matični broj: 44079/15–R

Studij: Informacijski sustavi

Progresivne web aplikacije

ZAVRŠNI RAD

Mentor:

Doc. dr. sc. Mario Konecki

Varaždin, rujan 2019.

Filip Težak

Izjava o izvornosti

Izjavljujem da je moj završni rad izvorni rezultat mojeg rada te da se u izradi istoga nisam koristio drugim izvorima osim onima koji su u njemu navedeni. Za izradu rada su korištene etički prikladne i prihvatljive metode i tehnike rada.

Autor potvrdio prihvaćanjem odredbi u sustavu FOI-radovi

_____

Sažetak

Tema rada su progresivne web aplikacije te će se u radu obraditi definicija progresivnih web aplikacija, njihovih karakteristika i područja primjene. U radu će se također opisati proces razvoja progresivnih web aplikacija te će biti dan praktičan primjer s detaljnim opisom elemenata potrebnih za funkcionalnost progresivnih web aplikacija. Osim toga u radu će se provesti detaljna usporedna analiza klasične web aplikacije, progresivne web aplikacije te izvornih (nativnih) aplikacija. Na temelju dobivenih znanja i iskustva o razvoju progresivnih web aplikacija i njihovoj primjeni izveden je zaključak.

Ključne riječi: pwa; progresivna web aplikacija; service workers; manifest; web;

Sadržaj

1 Uvod ……..1
2 Progresivne web aplikacije 2
2.1. Povijest ……4
2.2. Temeljni dijelovi PWA 4
2.2.1. Web App Manifest 5
2.2.2. Service worker 5
2.3. Lighthouse ……6
3 Zašto koristiti progresivne web aplikacije? 6
3.1. Usporedba web aplikacije i nativne aplikacije 6
3.2. Usporedba progresivnih web aplikacija sa nativnim i web aplikacijama 13
4 Izrada progresivne web aplikacije 16
4.1. ASP.NET MVC 16
4.1.1. Model ……16
4.1.2. Pogled ……17
4.1.3. Kontroler 18
4.2. Implementacija elemenata progresivnih web aplikacija 19
4.2.1. HTTPS ……20
4.2.2. Web app manifest 20
4.2.3. Service worker 21
4.3. Primjer progresivne web aplikacije 25
5 Zaključak 28
Popis literature 29
Popis slika 31

1 Uvod

U današnje vrijeme interneta, pametnih telefona, društvenih mreža i neograničenih količina informacija, usluga i alata koji su nam dostupni trenutno na dlanu, nije lako probiti se na tržište i privući, ali još i važnije zadržati korisnike. Veliki dio korištenja Interneta više se ne odvija putem računala već preko mobilnih uređaja. Donedavno su postojala dva principa razvoja aplikacija za mobilne uređaje, a to su izvorne, odnosno nativne aplikacije i web aplikacije. Svaka od tih opcija donosi neke prednosti, ali i nedostatke od kojih niti jedna nije zanemariva. U pokušaju eliminacije nedostataka pojavio se novi trend razvoja aplikacija za mobilne uređaje, a to su progresivne web aplikacije (PWA). Detaljnije će se o ova tri principa i njihovoj usporedbi govoriti u nastavku.

Tvorac ovog novog trenda razvoja aplikacija za mobilne uređaje, odnosno PWA je sam Google koji je jedno od najvećih imena u svijetu novih tehnologija, a samim time gotovo svaka nova tehnologija iza koje stane jedan takav tehnološki div doživi uspjeh ili barem veliki interes zajednice. U posljednje vrijeme vidi se povećanje interesa prema radu u oblaku, odnosno u cloud-u, a prema riječima analitičara i velikih vodećih svjetskih informatičkih kompanija poput Google-a, Amazona, Microsoft-a i drugih, cloud je budućnost te će se u budućnosti sve odvijati u cloud-u. Ako je to predviđanje točno, web aplikacije će postati jedina opcija, ali one imaju nedostatke koje su do sada mogle pokriti jedino nativne aplikacije, a sada uz PWA postoji mogućnost razvoja web aplikacija sa svim prednostima koje donose nativne i web aplikacije. Toliki potencijal koji PWA donosi u budućnosti je ujedno i inspiracija za pisanje ovoga rada te stjecanje novih znanja i iskustva u korak s vremenom.

2 Progresivne web aplikacije

Pojam progresivnih web aplikacija odnosno PWA je samo naziv za set funkcionalnosti i tehnologija koje se dodaju standardnim web aplikacijama kako bi ih unaprijedili te kako bi one izgledale i funkcionirale više kao nativne aplikacije [1].

Često se na prvi spomen progresivnih web aplikacija u ljudima pojavljuje povezanost s responzivnim web dizajnom što i nije krivo, responzivni dizajn je veoma bitan dio progresivnih web aplikacija, ali ovdje se ne misli na responzivni dizajn u smislu prilagodljivosti aplikacije kako bi izgledala dobro na svim vrstama uređaja. Ovdje je riječ o funkcionalnostima koje su nam poznate s nativnih aplikacija kao što su van mrežni način rada, ikona na početnom zaslonu, pristup aplikacije kameri mobilnog uređaja, gps lokaciji uređaja, sinkronizacije podataka u pozadini, itd.

PWA aplikacije ne funkcioniraju po sistemu sve ili ništa, implementacija PWA ne znači da aplikacija neće moći biti korištena na standardan web način ili da neće raditi na starijim pretraživačima te je to zapravo smisao naziva “progresivna web aplikacija” gdje se funkcionalnosti progresivno dodaju te nude najbolje iskustvo za korisnika [1].

Glavne značajke progresivnih web aplikacija su:

- **Pouzdanost** što znači da se aplikacija učitava brzo i ne prestaje s radom ukoliko nedostaje veza na Internet, odnosno podržava van mrežni način rada
- **Brzina,** odnosno osjetljivost na sve akcije korisnika
- **Privlačnost**, odnosno angažiranost što znači da aplikacija nudi osjećaj korištenja kao kod nativnih aplikacija

Ove značajke omogućuju progresivnim web aplikacijama da zasluže svoje mjesto na početnom zaslonu korisnikovog mobilnog uređaja [2].

Da bi se neka aplikacija smatrala progresivnom web aplikacijom ona mora steći niz uvjeta i zadovoljiti određene kriterije, a to su:

- **Progresivnost**
  - funkcionalna za sve korisnike bez obzira na odabir pretraživača
- **Responzivnost**
  - uklapa se u bilo koji format, desktop, mobilni, tablet ili bilo koji drugi
- **Neovisnost o konekciji**
  - unaprijeđena pomoću service workera (o kojima će više riječi biti u nastavku) u svrhu podržavanja van mrežnog načina rada
- **Poput nativne aplikacije**
  - koristi model dizajna nativnih aplikacija u svrhu pružanja korisničkog iskustva navigacije i interakcije kao na nativnim aplikacijama
- **Osvježena**
  - uvijek u korak s vremenom zahvaljujući service worker procesu ažuriranja
- **Sigurna**
  - koristi HTTPS konekciju
- **Vidljiva**
  - prepoznatljiva kao aplikacija
- **Ponovo angažirajuća**
  - omogućava ponovnu angažiranost koristeći se raznim funkcionalnostima od kojih je najkorištenija opcija push notifikacija
- **Instalirajuća**
  - omogućuje korisniku spremanje aplikacije na početni zaslon uređaja bez korištenja mrežne trgovine aplikacija
- **Djeljiva**
  - jednostavno dijeljenje putem linka bez potrebe za kompliciranom instalacijom

Kao što je već prije navedeno, aplikacija ne mora steći sve ovdje navedene uvijete da bi bila progresivna web aplikacija, dovoljno je implementirati samo one dijelove koji imaju svrhu u domeni aplikacije [3].

2.1. Povijest

Daleko prije nego što je Google definirao progresivne web aplikacije, Apple je bio taj koji je 2007. godine na predstavljanju iPhone mobilnog uređaja prvi govorio o novom i inovativnom načinu kreiranja web aplikacija koje izgledaju i ponašaju se isto kao i nativne aplikacije. Samo 4 mjeseca od tog predstavljana Apple je promijenio smjer razmišljanja i predstavio SDK za razvoj nativnih aplikacija boljih performansi za iOS te u desetljeću kasnije zaradio više od 40 milijardi američkih dolara od svog službenog App Store-a.

Godine 2015. Google Chrome developer Alex Russel i dizajner Frances Berriman kreiraju progresivne web aplikacije što je zapravo kopija Apple-ove ideje, ali poboljšana modernim tehnologijama i podrškom Google-a za danji razvoj ideje. Od tada najveće svjetske kompanije poput Google-a i Microsoft-a promoviraju koncept progresivnih web aplikacija kao način za premostiti razlike između nativnih aplikacija i web aplikacija.

Jedni od prvih korisnika progresivnih web aplikacija su Twitter i Aliexpress te ih je ta odluka kasnije višestruko nagradila. Twitter je zahvaljujući uvođenju elemenata progresivnih web aplikacija smanjio promet podataka za 70%, broj poslanih tweet-ova za 75% i broj posjećenih stranica u jednoj sesiji za 65%, a Aliexpress je povećao vrijeme provedeno u aplikaciji za 74% [4], [5].

2.2. Temeljni dijelovi PWA

Kao što je već prije navedeno progresivne web aplikacije definiraju set funkcionalnosti i tehnologija koje se dodaju standardnim web aplikacijama kako bi ih unaprijedili, odnosno kako bi one poprimile poželjne karakteristike nativnih aplikacija.

Tri su osnovne komponente da bi progresivna web aplikacija bila progresivna web aplikacija, a to su:

- **HTTPS protokol –** s obzirom na to da progresivne web aplikacije mogu pristupati raznim funkcionalnostima operacijskog sustava, neophodno je da veza na server bude sigurna.
- **Web App Manifest**
- **Service Worker**

2.2.1. Web App Manifest

Web App Manifest je JSON deskriptivna datoteka koja sadrži informacije o aplikaciji poput naziva aplikacije, putanje do ikone aplikacije, URL aplikacije, opciju orijentacije ekrana, opciju prikaza preko cijelog zaslona, itd.

Kako bi progresivna web aplikacija ostavljala što veći dojam nativne aplikacije potrebno je koristiti što više mogućnosti koje nudi Web App Manifest, a to je mogućnost pokretanja aplikacije u prikazu preko cijelog ekrana bez prikazivanja pretraživačke trake, definiranje takozvanog splash ekrana prilikom pokretanja aplikacije, definiranja boje teme aplikacije, definiranje seta ikona aplikacije i još mnogo toga.

Web App Manifest je nužan za preuzimanje aplikacije, odnosno instalaciju aplikacije na početni zaslon uređaja pružajući korisniku brži pristup i bogatije korisničko iskustvo [3], [5], [6], [7].

2.2.2. Service worker

Service worker je tip web worker-a, odnosno JavaScript datoteka koja cijelo vrijeme radi u posebnom browser threadu odvojenom od glavnog browser thread-a, web stranice i korisničkih interakcija te može presretati mrežne zahtjeve, spremati i povratiti resurse iz predmemorije odnosno cache-a te slati push notifikacije.

Service worker predstavlja programabilni mrežni proxy koji dopušta kontrolu nad mrežnim zahtjevima. Korištenje service worker-a koji mogu presretati mrežne zahtjeve i modificirati odgovor je ujedno i razlog zašto je potrebno korititi HTTPS protokol. Još jedna od ključnih zadaća service workera je i ta da prilikom prvog posjeta jednoj progresivnoj web aplikaciji pokreće događaj instalacije gdje se na korisnikov uređaj instaliraju svi potrebni resursi.

Kada se ne koristi service worker postaje neaktivan te se ponovo pokreće kada je potreban. Kao što je već navedeno service worker omogućuje aplikaciji kontrolu nad mrežnim zahtjevima i spremanje tih zahtjeva u predmemoriju radi poboljšanja performansi te omogućavanja van mrežnog načina rada [3], [5], [6], [8].

2.3. Lighthouse

Lighthouse je automatski open-source alat koji služi za poboljšanje kvalitete web stranice te izvršava testove za performanse, dostupnost, progresivne web aplikacije i drugo. Lighthouse sadrži niz metrika koje pomažu voditi razvoj progresivne web aplikacije te je jedan od ključnih alata kod razvoja iste [9].

3 Zašto koristiti progresivne web aplikacije?

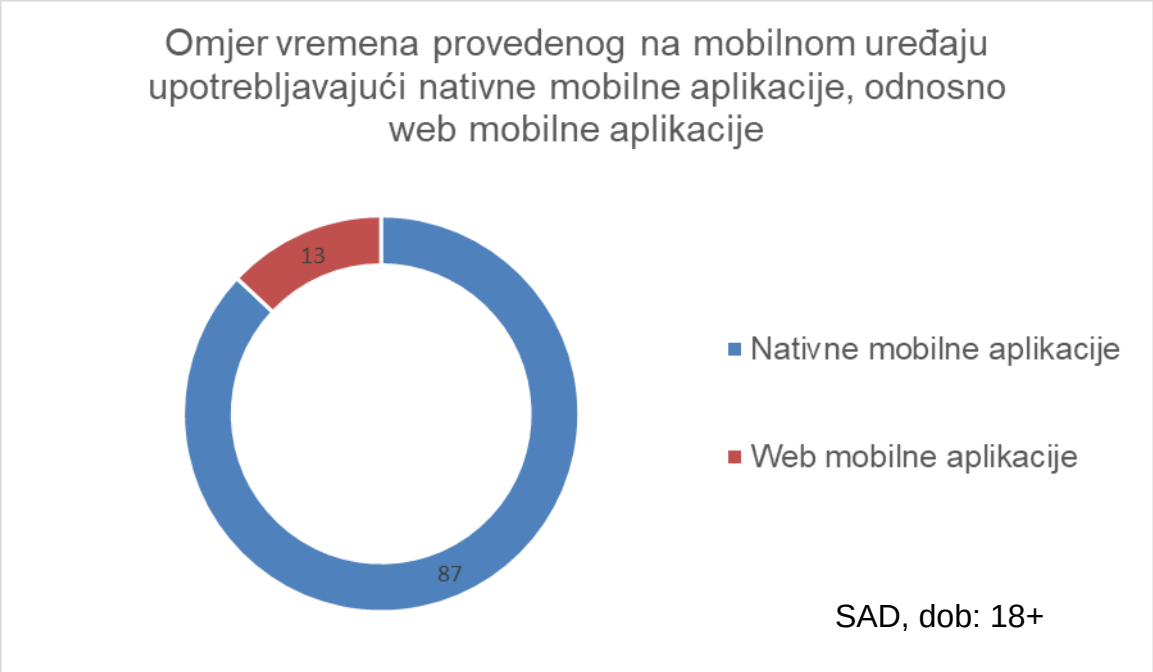
Kao što je već prethodno spomenuto donedavno su postojale dvije opcije kod razvoja aplikacija za mobilne uređaje, a to su nativne aplikacije i web aplikacije. Svaka od ovih opcija nudi neke prednosti, ali i nedostatke. Ovisno o funkcionalnosti same aplikacije odabirala se opcija koja je najbolje odgovarala potrebama, odnosno ona opcija koje je nudila najmanje nedostataka.

Dolaskom progresivnih web aplikacija pojavljuje se i treća opcija kao hibridna aplikacija između nativne aplikacije te web aplikacije koja nudi rješenje u potpunosti bez nedostataka prva dva modela ili barem njihovo minimiziranje.

U nastavku će biti napravljena usporedna analiza sve tri opcije na način da ćemo prvo usporediti koje prednosti, a koje nedostatke donose prve dvije opcije kao nativna aplikacije te web aplikacije. Zatim ćemo rezultate dobivene usporedbom analizirati u odnosu na ono što donose progresivne web aplikacije.

3.1. Usporedba web aplikacije i nativne aplikacije

Na slici 1 prikazan je omjer vremena provedenog na mobilnom uređaju upotrebljavajući nativne aplikacije, odnosno web aplikacije. Prosječan korisnik pametnog telefona 87% vremena provedenog na mobilnom uređaju provodi koristeći nativne aplikacije, dok samo 13% vremena provodi koristeći web aplikacije preko pretraživača na svom mobilnom uređaju.



Slika 1. Omjer vremena provedenog na mobilnom uređaju upotrebljavajući nativne aplikacije, odnosno web aplikacije [10]

Na temelju informacija dobivenih sa slike 1 može se postaviti pitanje zašto bi netko izradio web aplikaciju ako je nativna toliko bolja. Na to i na još neka pitanja je odgovoreno u nastavku.

Na slici 2 možemo vidjeti prednosti i nedostatke nativnih aplikacija.

	Nativna mobilna aplikacija
Van mrežni način rada	+
Push notifikacije	+
Instalacija na početni zaslon uređaja	+
Rad preko cijelog ekrana	+
Pristup funkcionalnostima mobilnog uređaja	+
Podrška za sve vrste uređaja	-
Mogućnost indeksiranja od strane pretraživača	-
Ne zahtjeva preuzimanje	-
Ne zahtjeva ažuriranje	-

Slika 2. Prednosti i nedostaci nativnih aplikacija [11]

Na slici 2 možemo uočiti neke od najbitnijih prednosti, ali i nedostataka nativnih aplikacija. Najvažnije prednosti nativnih aplikacija su veoma bitne jer do prije pojave progresivnih web aplikacija jedina opcija ako želimo napraviti aplikaciju sa ovim funkcionalnostima je bila da se koristi princip nativnih aplikacija.

Funkcionalnosti koje donose nativne aplikacije su:

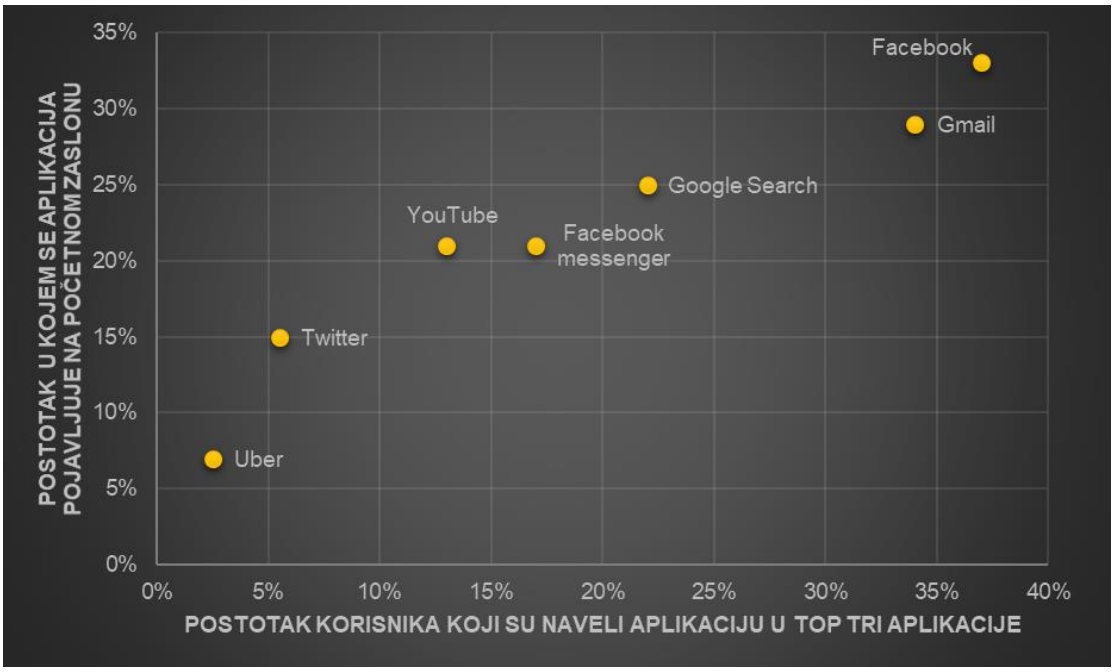
- **Van mrežni način rada** – aplikacija ne prestaje s radom ukoliko mobilni uređaj nema vezu na Internet, odnosno aplikacija ne javlja grešku bez sadržaja već nastavlja korisniku prikazivati sadržaj uz poruku o nedostupnosti mreže što pruža znatno bolje iskustvo za korisnika

- **Push notifikacije** – obavijesti koje korisnik prima preko svojeg mobilnog uređaja, a šalje mu ih aplikacija. Nativna aplikacija može slati push notifikacije kada je sama aplikacija zatvorena i neaktivna te na taj način privući pažnju korisnika na ponovo upotrebljavanje aplikacije

- **Instalacija na početni zaslon uređaja** – mogućnost aplikacije da na početnom zaslonu kreira ikonu preko koje je korisniku lagano omogućen pristup aplikaciji. Na slici 3 se može jasno uočiti povezanost prisustva aplikacije na početnom ekranu uređaja i važnosti aplikacije za korisnika. Što je aplikacija korisniku važnija to je veća šansa da će se aplikacija pojaviti na početnom zaslonu njegovog uređaja. Taj podatak govori dvije stvari, a to je da svakoj aplikaciji treba biti cilj osvojiti svoje mjesto na početnom zaslonu uređaja jer samim time dobiva i na važnosti, a drugo je to da mali postotak aplikacija ikada završi na početnom zaslonu jer nemaju dovoljno veliku važnost za korisnika.

- **Rad preko cijelog ekrana** – za aplikacije je veoma važan aspekt prikaza preko cijelog ekrana jer se time stvara osjećaj veće povezanosti korisnika i aplikacije te se stvaraju bolji uvjeti za interakciju korisnika s aplikacijom. Za razliku od web aplikacija koje se otvaraju unutar pretraživača i prikazuju traku za pretraživanje i ostale dodatne stvari koje umanjuju osjećaj povezanosti.

- **Pristup funkcionalnostima mobilnog uređaja** – funkcionalnost koja nativnim aplikacijama omogućuje da pristupa funkcijama mobilnog uređaja kao što su kamera, GPS lokacija, pohrana podataka i drugo. U današnjem svijetu stalne povezanosti i društvenih mreža pristup aplikacije kameri može biti od krucijalne važnosti. Uzmimo na primjer jednu od trenutno najpopularnijih platformi za dijeljenje fotografija, a to je Instagram. Instagramu je veoma važan elemenat aplikacije pristup kameri uređaja te dijeljenje fotografija. Ovakva aplikacija nije mogla biti u potpunosti realizirana web aplikacijama, već nativnim aplikacijama.



Slika 3. Odnos važnosti aplikacije za korisnika i sadržanosti aplikacije na početnom zaslonu [10]

Nativne aplikacije donose i neke nedostatke koji imaju veliki utjecaj u odluci da li razvijati nativnu aplikaciju ili pak web aplikaciju. Nedostaci nativne aplikacije su:

- **Nedostatak podrške za sve vrste uređaja** – mogućnost podrške dva glavana operacijska sustava mobilnih uređaja (Android i iOS) zahtjeva razvoj dvije različite aplikacije, odnosno programiranje u dva različita programska jezika. Takav pristup često zahtijeva dva razvojna tima gdje je svaki tim specijaliziran za jednu inačicu aplikacije. Prilikom ažuriranja promjene se moraju obaviti na oba dvije inačice aplikacije što veliki trošak razvoja na početku još dodatno povećava.

- **Nemogućnost indeksiranja od strane pretraživača –** nativne aplikacije pretraživač nije u mogućnosti indeksirati te ponuditi korisniku kao rezultat pretrage.
- **Zahtijeva preuzimanje** – prilikom instalacije nativne aplikacije na uređaj korisnika je potrebno spremiti sve datoteke potrebne za rad aplikacije što u nekim slučajevima može dosegnuti i do nekoliko desetaka megabajta.

- **Zahtijeva ažuriranja** – kada nativna aplikacija dobije novu verziju i ukoliko korisnik želi preuzeti noviju verziju aplikacije to se mora obaviti preko mrežne trgovine aplikacija što je po današnjim standardima dug i iritantan postupak.

Nativne aplikacije imaju svoje prednosti i nedostatke. Prednosti nativnih aplikacija su nešto čemu svaka aplikacija treba težiti, ali nativne aplikacije imaju i nedostatke koje bi trebalo riješiti.

Nakon nativnih aplikacija dolaze web aplikacije koje zajedno sa nativnim aplikacijma zajedno čine Jin i jang, odnosno prednosti nativnih aplikacija postaju nedostaci web aplikacija, a nedostaci postaju prednosti. Ovakav odnos je prikazan na slici 4.

	Nativna mobilna aplikacija	Web mobilna aplikacija
Van mrežni način rada	+	-
Push notifikacije	+	-
Instalacija na početni zaslon uređaja	+	-
Rad preko cijelog ekrana	+	-
Pristup funkcionalnostima mobilnog uređaja	+	-
Podrška za sve vrste uređaja	-	+
Mogućnost indeksiranja od strane pretraživača	-	+
Ne zahtjeva preuzimanje	-	+
Ne zahtjeva ažuriranje	-	+

Slika 4. Prednosti i nedostaci nativnih i web aplikacija [11]

Kao što je već prije navedeno nedostaci web aplikacija su ujedno i prednosti nativnih aplikacija. Stoga web aplikacija ne može raditi bez pristupa na mrežu, korisniku se pritom ne prikazuje nikakav sadržaj već samo greška koju je poslužio preglednik.

Push notifikacije su nedostupne za web aplikacije jer aplikacije nema neposrednu komunikaciju sa uređajem i mrežom kada je ugašena.

Iako se u većini mobilnih OS-ova može kreirati prečac web aplikacije na mobilni uređaj to nije jednako kao i instalacija aplikacije na početni uređaj. Prečac aplikacije nudi korisniku da jednim klikom u pregledniku otvori taj zadanu URL, dok aplikacije instalirane na početnom zaslonu rade mnogo više od toga.

Web aplikacije ne može pokrenuti rad preko cijelog ekrana, a razlog tome je direktno to što sam preglednik nema tu opciju pa ga stoga nema ni web aplikacija koja se u njemu pokreće

Nakon nedostataka web aplikacija tu su i prednosti koje su ujedno i nedostaci nativnih aplikacija, a to su:

- **Podrška za sve vrste uređaja** – za razliku od razvoja nativnih aplikacija gdje se aplikacije trebaju pokretati na različitim OS-ovima i gdje je često potrebno razvijati nekoliko različitih inačica iste aplikacije, kod web aplikacija to nije slučaj. Web aplikacije su aplikacije koje se pokreću u pregledniku bez obzira o kojem se OS-u radilo te su osnovni jezici za programiranje istih uvijek HTML, CSS i Javascript.

- **Mogućnost indeksiranja od strane pretraživača** – Kao što je navedeno za razliku od nativnih aplikacija, web aplikacije se pokreću unutar pretraživača, odnosno do njih se pristupa posjetom određenog URL-a. samim time pretraživač takvu aplikaciju može indeksirati i ponuditi korisniku kao odgovor na pretraživanje.

- **Ne zahtijeva preuzimanje** – web aplikacije ne zahtijevaju preuzimanje nikakvih datoteka jer im one nisu potrebne na uređaju kako bi mogle raditi. Svi resursi potrebni web aplikaciji za rad nalaze se na serveru te im aplikacija pristupa preko interneta što ujedno i odgovara na tvrdnju zašto web aplikacije ne rade u van mrežnom načinu rada.

- **Ne zahtijeva ažuriranja** – prilikom dolaska nove verzije aplikacije nativna aplikacija mora obaviti ažuriranje preko mrežne trgovine aplikacija, dok s druge strane web aplikacija prima ažuriranja gotovo trenutno. Aplikacija se u pozadini na serveru ažurira i prilikom dolaska korisnika na aplikaciju i osvježavanja iste on odmah počinje koristiti noviju verziju aplikacije bez preuzimanja novije verzije na uređaj.

3.2. Usporedba progresivnih web aplikacija sa nativnim i web aplikacijama

Nakon definiranja prednosti i nedostataka nativnih i web aplikacija vrijeme je da se u priču uključe i progresivne web aplikacije. Kao što smo prethodno zaključili kako nativne aplikacije i web aplikacije zajedno čine skup funkcionalnosti kojima treba težiti svaka aplikacija, tako sada možemo promatrajući sliku 5 zaključiti kako su progresivne web aplikacije upravo ono čemu svaka aplikacija treba težiti.

Na slici 5 možemo primijetiti kako progresivne web aplikacije imaju sve dobre karakteristike nativnih aplikacija, ali i web aplikacija.

	Nativna mobilna aplikacija	Web mobilna aplikacija	Progresivna web aplikacija
Van mrežni način rada	+	-	+
Push notifikacije	+	-	+
Instalacija na početni zaslon uređaja	+	-	+
Rad preko cijelog ekrana	+	-	+
Pristup funkcionalnostima mobilnog uređaja	+	-	+
Podrška za sve vrste uređaja	-	+	+
Mogućnost indeksiranja od strane pretraživača	-	+	+
Ne zahtjeva preuzimanje	-	+	+
Ne zahtjeva ažuriranje	-	+	+

Slika 5. Usporedba nativne i web aplikacija sa progresivnim web aplikacijama

Jednako kao i nativne aplikacije tako i progresivne web aplikacije podržavaju van mrežni način rada. Kod progresivnih web aplikacija za to su zaslužni service workeri koji mogu u predmemoriju pohraniti podatke potrebne za van mrežni rad te u trenutku kada aplikacije ostane bez veze na Internet aplikacija može sadržaj spremljen u predmemoriju prikazati korisniku što čini korištenje aplikacije korisniku veoma ugodnije.

Push notifikacije su još jedna od stvari koje su progresivne web aplikacije preuzele iz svijeta nativnih aplikacija što sada omogućuje jednoj web aplikaciji odnosno progresivnoj web aplikaciji serviranje push notifikacija korisniku te tako privući pažnju korisnika na aplikaciju.

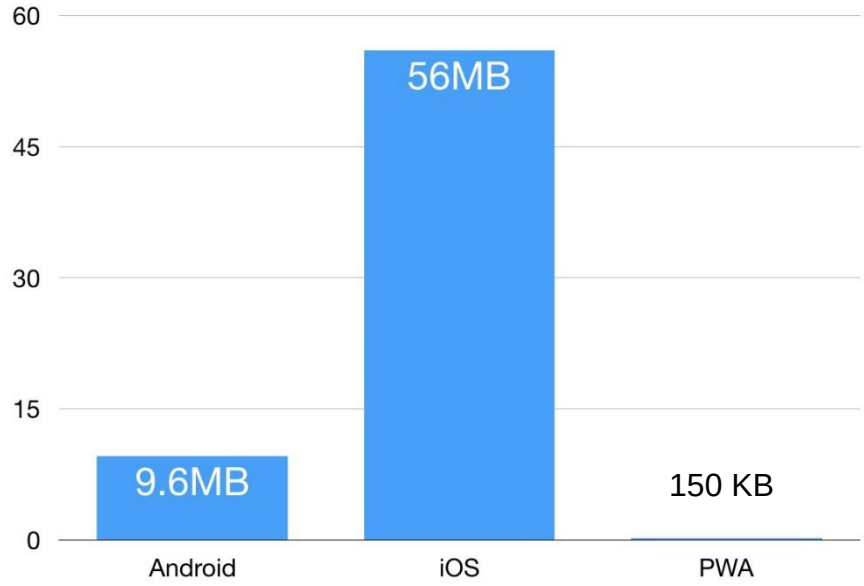
Progresivne web aplikacije mogu na početni zaslon instalirati svoju ikonu čime su korisniku lako dostupne i potiču korisnika na interakciju.

Za razliku od obične web aplikacije, progresivne web aplikacije mogu pristupati funkcionalnostima mobilnog uređaja te tako mogu koristiti primjerice kameru te uslikati fotografiju ili pristupati GPS lokaciji samog mobilnog uređaja i još mnogo više.

Osim što progresivne aplikacije sadrže sve prednosti nativnih aplikacija one mogu sve što mogu i web aplikacije, a to je da podržavaju sve vrste uređaja, omogućuju indeksiranje od strane pretraživača te ne zahtijevaju preuzimanje niti ažuriranje.

Zanimljiv je podatak na primjeru migracije platforme Pinterest sa nativnih aplikacija za Android i iOS na progresivnu web aplikaciju o količini sadržaja koji je potrebno preuzeti kako bi se aplikacija instalirala na uređaj korisnika. Nativna aplikacija platforme Pinterest za Android zahtijevala je preuzimanje oko 9.6MB podataka, a za iOS čak 56MB podataka.

Kao što je i prikazano na slici 6, prelaskom na progresivnu web aplikaciju ove brojke su se svele na svega 150KB podataka i za Android i za iOS jer kao što je prethodno navedeno, kod korištenja progresivnih web aplikacija podržani su svi uređaju. Na prvi pogled se može činiti kako se ovdje već i prije radilo o malim i zanemarivim brojkama, ali na milione korisnika ovdje se radi o poprilično velikim brojkama, a poboljšanje od nekoliko stotina puta je impresivan podatak. [12]



Slika 6. Prikaz količine podataka potrebnih za preuzimanje prilikom instalacije [12]

4 Izrada progresivne web aplikacije

Kao primjer progresivne web aplikacije napravljena je aplikacija Pamtim. Aplikacija služi studentima za praćenje broja izostanaka, odnosno koliko su puta izostali sa pojedinog predavanja, laboratorijskih vježbi ili slično. Kao dodatnu funkcionalnost aplikacija nudi i pregled rasporeda sati svih aktivnosti koje student ima na fakultetu.

Aplikacija Pamtim se sastoji od prijave i registracije korisnika, elementa za praćenje izostanaka te rasporeda. Registracija korisnika se obavlja na standardan način korisnikova unosa podataka u aplikaciju te potvrđivanje email adrese putem linka poslanog na istu email adresu. Element za praćenje izostanaka se sastoji od dodavanja novih semestara, zatim dodavanja kolegija u semestar, dodavanja aktivnosti u kolegije te se za svaku aktivnost mogu definirati termini u tjednu kad se ta aktivnost izvršava i izostanci. Za svaku od prethodno navedenih dijelova podržava takozvane CRUD operacije, odnosno operacije kreiranja, čitanja, uređivanja te brisanja.

Za izradu aplikacije korišten je ASP.NET MVC razvojni okvir za izradu web aplikacija unutar Visual Studio razvojnog okruženja. Navedeni alati su odabrani radi prethodnog znanja u korištenju istih te veoma dobre podrške i dokumentacije za navedene tehnologije. Slijed razvoja je bio takav da se prvo kreirala funkcionalna web aplikacija bez implementacije tehnologija progresivnih web aplikacija te su se te tehnologije naknadno ugrađivali u već funkcionalnu aplikaciju kako bi ju poboljšali i učinili progresivnom web aplikacijom.

4.1. ASP.NET MVC

ASP.NET MVC je razvojni okvir za izradu web aplikacija razvijen od strane Microsoft-a koji na okvir ASP.NET primjenjuje uzorak Model-View-Controller odnosno model, pogled i kontroler. ASP.NET je okvir za izradu web aplikacija na strani servera koji omogućuje razvoj dinamičkih web stranica [13],[14].

4.1.1. Model

Unutar ASP.NET MVC, modeli su implementirani u obliku klasa koje predstavljaju podatke koje koristimo, a ti su podaci najčešće spremljeni u bazu podataka. Na slici 7 možemo vidjeti dio koda iz aplikacije koji predstavlja kod modela podataka za semestre koji ujedno predstavlja i sve atribute semestra u bazi podataka.

U bazi podataka u tablici za semestre nalaze se tri atributa, a to su identifikacijski broj semestra, odnosno primarni ključ, zatim vanjski ključ na tablicu korisnika kao identifikacijski broj korisnika te naziv samog semestra.

```
8  namespace Pamtim.Models
9  {
10     public class SemesterViewModel
11     {
12         public string ID_Semester { get; set; }
13         public string Name { get; set; }
14     }
15
16     public class ListofSemesterViewModel
17     {
18         public List<SemesterViewModel> Items { get; set; }
19     }
20
21     public class EditSemesterViewModel
22     {
23         public string User_ID { get; set; }
24         public int ID_Semester { get; set; }
25
26         [Required]
27         [Display(Name = "Naziv")]
28         public string Name { get; set; }
29     }
30 }
```

Slika 7. Model za semestre

4.1.2. Pogled

Pogled odnosno View je predložak koji dinamički generira HTML kod na osnovu podataka koje mu prosljeđuje kontroler kako bi podatke prikazao korisniku [13],[14].

Na slici 8 možemo vidjeti kod iz aplikacije koji predstavlja kod pogleda za prikaz semestara. Kod na slici 8 za svaki semestar dohvaćen iz baze podataka u kontroleru (slika 9) kreira njegovu karticu zajedno sa podacima za taj semestar.

Pogled će automatski prikazati element prikladan vrsti podatka te će tako podatak tipa datum dobiti element za odabir datuma, podatak za vrijeme će dobiti element za odabir vremena, itd.

```
@foreach (SemesterViewModel item in Model.Items)
{

    <div class="col-10 col-md-6 col-lg-4 my-3">
        <div class="card">

            <a href="@("\\Courses\\Pregled\\?Semester_ID=" + item.ID_Semester)" style="text-decoration: none;">
                <div class="card-body">
                    <h5 class="card-title">@item.Name</h5>
                </div>
            </a>

            <div class="row card-footer text-center mx-0 my-0 p-1">

                <a href="@("\\Semestri\\Edit\\" + item.ID_Semester)" class="align-self-center col-6 p-0">
                    <img src="..\..\Content\edit.png" height="32" width="32" />
                </a>
                <a href="@("\\Semestri\\Delete\\" + item.ID_Semester)" onclick="javascript: return confirm('Obriši?');"
                    <img src="..\..\Content\delete.png" height="32" width="32" />
                </a>

            </div>
        </div>
    </div>
    <div class="w-100 d-block d-md-none"></div>
}
```

Slika 8. Pogled za pregled semestara

4.1.3. Kontroler

Kontroler upravlja interakcijom sa korisnikom, osvježuje model, odabire koji će pogled prikazati te prosljeđuje informacije pogledu za prikaz korisniku [13],[14].

Na slici 9 možemo vidjeti kontroler zaslužan za prikaz semestara. Kao što možemo vidjeti kontroler ima mogućnost spajanja na bazu, povlačenja potrebnih podataka iz baze, osvježavanje modela novim podacima te na kraju prosljeđivanje podataka odabranom pogledu za prikaz.

```
[Authorize]
public ActionResult Pregled()
{
    var User_ID = Request.Cookies["User_ID"].Values;

    DataTable dt = new DataTable();
    using (var conn = new SqlConnection(ConfigurationManager.ConnectionStrings["pamtim"].ConnectionString))
    {
        conn.Open();
        if (conn.State == ConnectionState.Open)
        {

            var upit = "select * from Semester where User_ID = '" + User_ID + "'";
            using (var cmd = new SqlCommand(upit, conn))
            {
                cmd.CommandType = CommandType.Text;
                var sqlda = new SqlDataAdapter(cmd);
                sqlda.Fill(dt);
            }
        }
    }
    ViewBag.podaci = dt;

    var semesters = new ListofSemesterViewModel();

    semesters.Items = new List<SemesterViewModel>();

    foreach (DataRow r in dt.Rows)
    {
        var semester = new SemesterViewModel();
        semester.ID_Semester = r["ID_Semester"].ToString();
        semester.Name = r["Name"].ToString();

        semesters.Items.Add(semester);
    }
    return View(semesters);
}
```

Slika 9. Kontroler za pregled semestara

4.2. Implementacija elemenata progresivnih web aplikacija

Kao što je prethodno u radu navedeno, osnovni uvjeti koje svaka web aplikacija mora ostvariti da bi bila progresivna web aplikacija su korištenje HTTPS protokola te implementacija web app manifest-a i service workera. U nastavku će biti prikazana implementacije prethodno navedenih uvjetu u izrađenom primjeru.

4.2.1. HTTPS

HTTPS (eng. Hypertext Transfer Protocol Secure) predstavlja nadogradnju na HTTP protokol koji donosi sigurnu komunikaciju preko mreže. Komunikacija preko ovog protokola je kriptirana koristeći SSL (eng. Secure Socket Layer). Za potrebe ostvarivanja mogućnosti korištenja HTTPS protokola mora se steći i valjani SSL certifikat, certifikat može biti potpisan od strane pouzdanih izvora što nosi određenu cijenu, a može biti i vlastito potpisan što smanjuje sigurnost i dovodi do nestabilnosti u redu. Za potrebe aplikacije napravljen je valjani certificirani Rapid SSL certifikat koji je zatim instaliran na server na kojem se nalazi domena aplikacije.

Nakon instalacije certifikata na server jedino je preostalo preusmjeriti sve zahtjeve sa HTTP protokola na HTTPS protokol što se definiralo u Web.config datoteci aplikacije te je taj dio koda prikazan na slici 10.

```
<rewrite>
  <rules>
    <rule name="Force HTTPS" enabled="true">
      <match url="(.*)" ignoreCase="false"/>
      <conditions>
        <add input="{HTTPS}" pattern="off"/>
      </conditions>
      <action type="Redirect" url="https://www.pamtim.com/" appendQueryString="true" redirectType="Permanent"/>
    </rule>
  </rules>
</rewrite>
```

Slika 10. Preusmjeravanje zahtjeva na HTTPS

4.2.2. Web app manifest

Web App Manifest je JSON deskriptivna datoteka koja sadrži informacije o aplikaciji te je jedan od uvjeta nužan za postizanja progresivnih web aplikacija. Slika 11 prikazuje kao je web app manifest implementiran u primjeru.

Informacije koje sadrži ovaj web app manifest su puno ime aplikacije, skraćeno ime aplikacije, opis, boja teme, dohvat web app manifesta koji je u ovom slučaju cijeli projekt, boja pozadine, način prikaza aplikacije, startni url kod pokretanja aplikacije te set ikona različitih veličina koje su potrebne za prikaz na početnom zaslonu uređaja te za prikaz na splash screen-u aplikacije.



Slika 11. Primjer web app manifest-a

4.2.3. Service worker

Da bi instalirali service worker potrebno je potaknuti taj proces instalacije registracijom service workera na web stranici. Proces registracije je zadužen da pruži informaciju pretraživaču gdje se nalazi service worker JavaScript datoteka. U primjeru proces registracije je pokrenut u datoteci _Layout.cshtml čija je svrha da sadrži standardne dijelove korisničkog sučelja kao što je navigacijska traka i podnožje aplikacije te je sastavni dio svake stranice aplikacije. Isti proces se mogao pozvati i u zasebnoj JavaScript datoteci, ali je u ovom slučaju odabran drugačiji pristup.

Na slici 12 je prikazan dio koda koji u primjeru predstavlja proces registracije service workera. Ovaj kod najprije provjerava da li je service worker uopće podržan od strane pretraživača te ukoliko jest tada započinje proces registracije istog koji će biti registriran jednom kada je stranica učitana.

```
if ('serviceWorker' in navigator) {
    window.addEventListener('load', function() {
        navigator.serviceWorker.register('../../serviceworker.js').then(function() {
            console.log('Service Worker Registered');
        });
    });
}
```

Slika 12. Registracija service workera

Jednom kada je service worker registriran radnja se prebacuje na JavaScript datoteku samog service workera gdje se pokreće proces instalacije service workera. Proces instalacije service workera ujedno služi kao prilika za spremanje statičkih datoteka, odnosno datoteka koje se rijetko mijenjaju kao što su .CSS datoteke ili pak slike i JavaScript datoteke u predmemoriju. U primjeru to je nekoliko slika korištenih u korisničkom sučelju te nekoliko .CSS datoteka. Na slici 13 je prikazan primjer implementiranog procesa instalacije service workera.

```
var CACHE_STATIC_NAME = 'static-v7';

self.addEventListener('install',
    function(event) {
        console.log('[Service Worker] Installing Service Worker... ', event);
        event.waitUntil(
            caches.open(CACHE_STATIC_NAME)
            .then(function(cache) {
                console.log('[Service Worker] Precaching...');
                cache.addAll([
                    '/Content/bootstrap.min.css',
                    '/Content/Site.css',
                    '/Content/calendar.png',
                    '/Content/delete.png',
                    '/Content/edit.png',
                    '/Content/enter.png',
                    '/Content/list.png'
                ]);
            })
        )
    });
```

Slika 13. Instalacija service workera

Kao i sve drugo i service worker je je moguće ažurirati, a to će se dogoditi kada se dogode i najmanje promjene u JavaScript datoteci service workera. Svaka pa čak i najmanja promjena dovoljna je da pretraživač taj service worker smatra novim. Svakim pristupom korisnika aplikaciji pristupa se ponovo datoteci, uoči li se razlika pretraživač će u pozadini preuzeti izmijenjenu datoteku i pokrenuti ponovo proces instalacije novog service workera. Nakon procesa instalacije u aplikaciji je i dalje aktivan stari service worker dok je novi u fazi čekanja. Jednom kada se sve otvorene stranice aplikacije zatvore stari service worker će biti izbrisan i novi će postati aktivan te se u tom trenutku pokreće proces aktivacije.

Kao što je prethodno navedeno proces instalacije sprema podatke u predmemoriju te ukoliko želimo te podatke osvježiti odnosno stare zamijeniti novima to je moguće učiniti upravo pomoću service workera i to u procesu aktivacije. Ovaj postupak brisanja starih podataka nije moguće napraviti u procesu instalacije iz razloga što je sve do procesa aktivacije i dalje aktivan stari service worker te ukoliko izbrišemo podatke koje je taj service worker pohranio neće više biti u mogućnosti ispravno raditi. Upravo iz tog razloga proces brisanja starih podataka potrebno je obaviti u procesu aktivacije ujedno kada se izbriše i stari service worker.

Na slici 14 je prikazana implementacija procesa aktivacije na primjeru gdje se prilikom procesa aktivacije stari podaci u memoriji brišu.

```
23  self.addEventListener('activate',
24      function (event) {
25          console.log('[Service Worker] Activating Service Worker... ', event);
26          event.waitUntil(
27              caches.keys()
28                  .then(function(keyList) {
29                      return Promise.all(keyList.map(function(key) {
30                          if (key !== CACHE_STATIC_NAME) {
31                              console.log('[Service Worker] Removing old cache.', key);
32                              return caches.delete(key);
33                          }
34                      }));
35                  })
36          );
37          return self.clients.claim();
38      });
```

Slika 14. Primjer procesa aktivacije

Nakon provedenih procesa registracije, instalacije i aktivacije service worker je spreman za rad. Jednom kada web aplikacija ispunjava sva tri uvjeta za postizanje progresivne web aplikacije, a to su HTTPS protokol, web app manifest i service worker stranica je ujedno ispunila i sve kriterije kako bi postala instalirajuća, odnosno u tom trenutku korisnik može instalirati aplikaciju na početni zaslon uređaja.

Jedna od glavnih funkcionalnosti service workera je i presretanje svih mrežnih zahtjeva i isporuka kreiranog odgovora. Service worker odgovor može dohvatiti sa interneta, može ga dohvatiti iz predmemorije te je još mnogo različitih načina na koji service worker može proslijediti promijenjeni odgovor na neki mrežni zahtjev aplikacije. Upravo za presretanje mrežnih zahtjeva zadužen je fetch event.

Prilikom presretanja mrežnih zahtjeva i isporuke promijenjenog odgovora postoje mnoge strategije na koje načine se to može izvesti. Neke od strategija se isporuka sadržaja samo putem predmemorije, isporuka sadržaja samo preko mrežnog zahtjeva, isporuka sadržaja prvo preko predmemorije zatim preko mrežnog zahtjeva, isporuka prvo preko mrežnog zahtjeva zatim preko predmemorije, isporuka unaprijed predviđenog sadržaja i još mnoge druge strategije. Sve ove navedene strategije mogu se kombinirati i koristiti kako to ima najviše smisla.

Neke od strategija su fokusirane na to da se što prije korisniku sadržaj prikaže na ekranu uređaja te su za tu namjenu najbolje metode prvog pokušaja dostave sadržaja iz predmemorije, ali ovakve strategije ne nude uvijek najnoviji sadržaj koji strategije gdje se prvo sadržaj dobavlja preko mrežnog zahtjeva nude.

U primjeru aplikacije odabrana je strategija dohvata sadržaja putem mrežnog zahtjeva, a ukoliko sadržaj preko mrežnog zahtjeva nije moguć korisniku se prikazuje unaprijed spremljen sadržaj u predmemoriji koji prikazuje generiranu stranicu s porukom upozorenja kako sadržaj nije dostupan što sprječava aplikaciju od prikaza standardne greške o nedostupnosti sadržaja.

Na slici 15 možemo vidjeti primjer koda koji služi za spremanje predefiniranog odgovora koji se korisniku prezentira u slučaju nedostatka Internet veze u predmemoriju.

```
if ('serviceWorker' in navigator) {
    window.addEventListener('load', function() {
        console.log('[Semestri] Pregled semestara je učitan!');
        event.waitUntil(
            caches.open('Offline')
            .then(function(cache) {
                console.log('[Service Worker] Precaching...');
                cache.add('https://www.pamtim.com/Offline/Offline');
            })
        )
    });
}
```

Slika 15. Spremanje predefiniranog odgovora u predmemoriju

S obzirom na odabranu strategiju spremanjem predefiniranog odgovora u predmemoriju može se definirati i sama strategija uporabom fetch eventa. Na slici 17 moguće je vidjeti kako je fetch event implementiran u primjeru. Service worker pomoću fetch eventa presreće mrežni zahtjev te na njega odgovara jednakim mrežnim zahtjevom koji će biti proslijeđen kao odgovor ukoliko je on dostupan i ispravan. Ukoliko se prilikom dohvaćanja odgovora mrežnog zahtjeva ili pak nedostatkom Internet veze javi greška, service worker će automatski kao odgovor isporučiti predefinirani odgovor koji smo prethodno spremili u predmemoriju.

```
40  self.addEventListener('fetch',
41      function (event) {
42          event.respondWith(
43              fetch(event.request)
44              .catch(function (err) {
45                  return caches.open('Offline')
46                      .then(function (cache) {
47                          return cache.match('https://www.pamtim.com/Offline/Offline');
48                      });
49
50              })
51          );
52      });
```

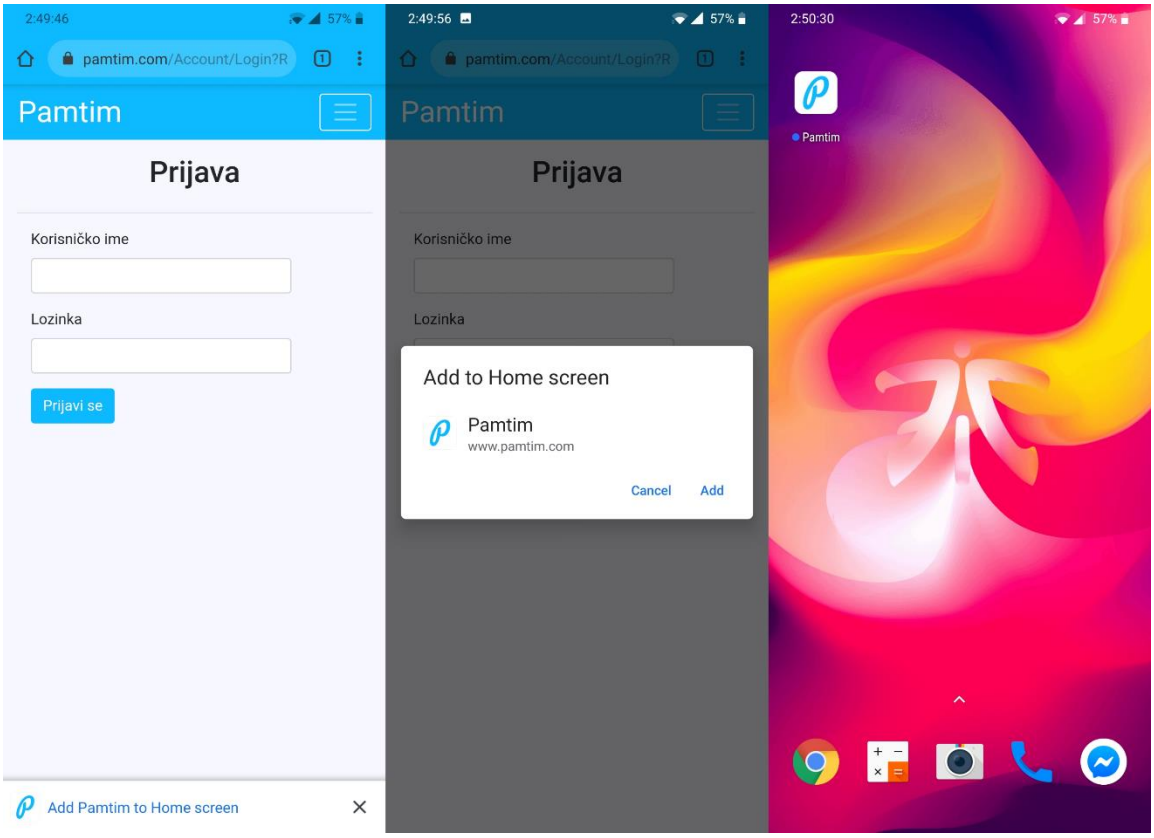
Slika 16. Primjer primjene fetch eventa i implementacija odabrane strategije

4.3. Primjer progresivne web aplikacije

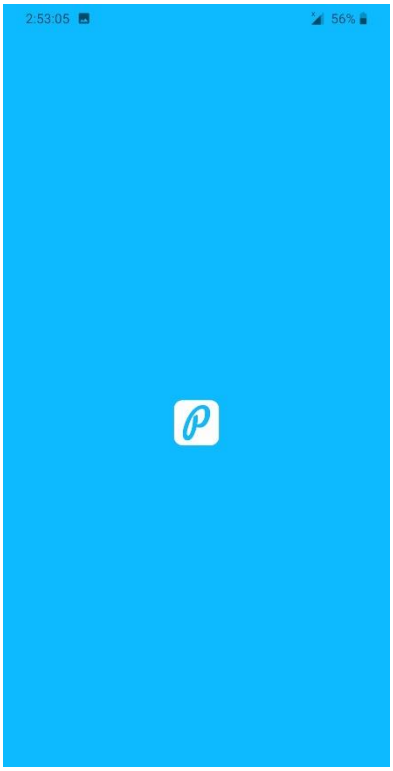
Implementacijom prethodno definiranih elemenata pretvorili smo standardnu ASP.NET MVC aplikaciju u progresivnu web aplikaciju koja uz pomoć novo definiranih elemenata nudi funkcionalnosti koje kao standardna web aplikacija nije omogućavala.

Prva primjetna razlika kod novog korisnika aplikacije vidljiva je već u prvom trenutku kada pretraživač korisniku prezentira mogućnost dodavanja aplikacije na početni zaslon uređaja, odnosno instalacije aplikacije na uređaj korisnika.

Slika 18 prikazuje trenutak u kojem se korisniku prezentira navedena mogućnost, što se događa kada korisnik prihvati instalaciju aplikacije na uređaj te samu pojavu ikone aplikacije na početnom zaslonu uređaja. Sama se ikona na početnom zaslonu uređaja ničime ne razlikuje od ikone bilo koje druge nativne aplikacije.



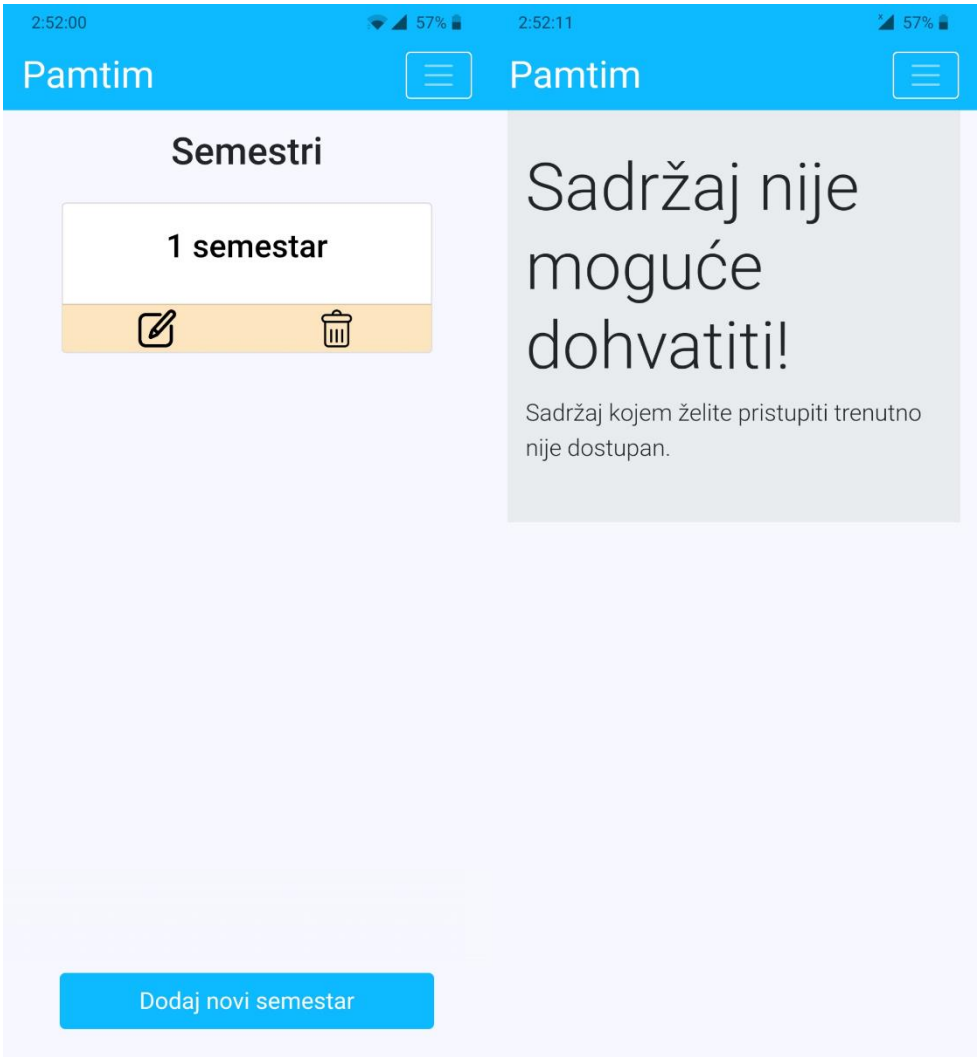
Slika 17. Postupak instalacije progresivne web aplikacije



Slika 18. Primjer splash screena progresivne web aplikacije

Kako bi korisniku ponudile najbolje iskustvo koje nude nativne aplikacije, progresivne web aplikacije nakon što je aplikacija instalirana na uređaj korisnika prilikom pokretanja aplikacije prikazuju splash screen koji je u primjeru prikazan na slici 19. Karakteristike splash screena definirane su u web app manifestu.

Još jedna od prethodno definiranih funkcionalnosti implementiranih u primjer progresivne web aplikacije je i predefiniran odgovo na mrežni zahtjev u slučaju greške ili nedostatka veze na Internet. Na slici 20 je prikazan ekran pregleda semestara na čijem su primjeru prethodno objašnjem koncept MVC te prikaz istoga ekrana, ali u slučaju da nedostaje veza na Internet kada se korisniku servira prikaz predefiniranog odgovora.



Slika 19. Prikaz implementacije predefiniranog odgovora u slučaju nedostatka veze na internet

5 Zaključak

Progresivne web aplikacije zaista nude veoma korisne i inovativne funkcionalnosti koje do sada nisu bile moguće. Veoma bitno za istaknuti je to da progresivne web aplikacije ne predstavljaju samostalno aplikaciju već je to set funkcionalnosti kojima se nadograđuju klasične web aplikacije kako bi one poprimile funkcionalnosti nativnih aplikacije i time pružile najbolje iz oba svijeta.

S obzirom na to da u progresivne web aplikacije relativno nov pojam u svijetu razvoja web aplikacije dostupno je mnogo dokumentacije i materijala kojih je svakim danom sve više. Iza ove tehnologije stoji jedna od najvećih svjetskih kompanija kao tvorac progresivnih web aplikacija, a to je Google uz podršku ostalih vodećih svjetskih kompanija kao što su Amazon i Microsoft. Google tvrdi kako prelaskom svijeta u cloud doba, progresivne web aplikacije predstavljaju budućnost razvoja i korištenja web aplikacija. Proučavanjem funkcionalnosti koje nude progresivne web aplikacije zaista možemo biti sigurni da je tome tako, jer progresivne web aplikacije predstavljaju mogućnost zamjene nativnih aplikacija progresivnim web aplikacijama.

Implementacija osnovnih funkcionalnosti progresivnih web aplikacija na klasične web aplikacije je vrlo jednostavna i u svega par dorada i promjena klasična web aplikacija može poprimiti funkcionalnosti za koje nikada ne bi rekli da jedna web aplikacija može imati.

Aplikacija Pamtim koja je izrađena kao primjer progresivnih web aplikacija mogla je biti implementirana i kao klasična web aplikacija, ali implementacijom kao progresivna web aplikacija ona je poprimila sasvim nov način korištenja same aplikacije. Jednom instalirana aplikacija Pamtim ne pokazuje nikakve znakove kako se radi o web aplikaciji te korisnik nema dojam da ne koristi nativnu aplikaciju, ali uz sve prednosti koje donose web aplikacije, a to su trenutne promjene na aplikaciji bez potrebe za ažuriranjem, instalacija aplikacije u nekoliko sekundi, količina podataka preuzetih tijekom instalacije svedena na svega stotinjak kilobajta i još mnogo toga.

Popis literature

[1] M. Schwarzmüller, (5.2019.) „Progressive Web Apps (PWA) – The ¸ Complete Guide," Udemy [Video datoteka]. Dostupno: https://www.udemy.com/course/progressive-web-app-pwa-the-complete-guide/learn/lecture/7888676#overview [pristupano 14.8.2019.]

[2] „Progressive Web Apps" (bez dat.) Google [Na internetu]. Dostupno: https://developers.google.com/web/progressive-web-apps/ [pristupano 14.8.2019.]

[3] A. Osmani, „Getting started with Progressive Web App", 23.12.2015. [Na internetu]. Dostupno: https://addyosmani.com/blog/getting-started-with-progressive-web-apps/ [pristupano 16.8.2019.]

[4] ScandiPWA, „History of Progressive Web Apps", 7.2019. [Na internetu]. Dostupno: https://medium.com/progressivewebapps/history-of-progressive-web-apps-4c912533a531 [Pristupano 16.8.2019.]

[5] „Progressive web applications," (bez dat.). u Wikipedia, the Free Encyclopedia. Dostupno: https://en.wikipedia.org/wiki/Progressive_web_applications [pristupano 17.8.2019.]

[6] A. Gustafson, „Yes, That Web Project Should Be a PWA", 30.8.2017. [Na internetu]. Dostupno: https://alistapart.com/article/yes-that-web-project-should-be-a-pwa/ [pristupio 20.8.2019.]

[7] „Web App Manifest" (bez dat.) Mozilla [Na internetu]. Dostupno: https://developer.mozilla.org/en-US/docs/Web/Manifest [pristupano 20.8.2019.]

[8] K. Končić, „PROGRESIVNE Web-Aplikacije," Mreža: za IT profesinalce, broj 02, 2.2018.

[9] „Lighthouse PWA Analysis Tool" (bez dat.) Google [Na internetu]. Dostupno: https://developers.google.com/web/ilt/pwa/lighthouse-pwa-analysis-tool [pristupano 20.8.2019.]

[10] B. Martin, „The Global Mobile Report" (2017.) comScore [Na internetu]. Dostupno: https://www.comscore.com [Pristupano 22.8.2019.]

[11] J. DeJung, „What's Next in Mobile: Progressive Web Apps“ (10.1.2018.) Onenorth [Na internetu]. Dostupno: https://www.onenorth.com/blog/post/whats-next-in-mobile-progressive-web-apps [Pristupano 22.8.2019.]

[12] A. Osmani „A Pinterest Progressive Web App Performance Case Study“ (29.11.2017.) Medium [Na internetu]. Dostupno: https://medium.com/dev-channel/a-pinterest-progressive-web-app-performance-case-study-3bd6ed2e6154 [Pristupano 25.8.2019.]

[13] A. M. Žinić, IZRADA WEB-APLIKACIJA NA PLATFORMI ASP.NET [Diplomski rad]. Sveučilište u Zagrebu, Zagreb, 2017, Dostupno: https://repozitorij.pmf.unizg.hr/islandora/object/pmf:4544/preview [Pristupano 29.8.2019.]

[14] „ASP.NET MVC,“ (bez dat.). u Wikipedia, the Free Encyclopedia. Dostupno: https://en.wikipedia.org/wiki/ASP.NET_MVC [pristupano 1.9.2019.]

Popis slika

Slika 1. Omjer vremena provedenog na mobilnom uređaju upotrebljavajući nativne aplikacije, odnosno web aplikacije [10]..... 7
Slika 2. Prednosti i nedostaci nativnih aplikacija [11] 8
Slika 3. Odnos važnosti aplikacije za korisnika i sadržanosti aplikacije na početnom zaslonu [10] 10
Slika 4. Prednosti i nedostaci nativnih i web aplikacija [11] 11
Slika 5. Usporedba nativne i web aplikacija sa progresivnim web aplikacijama 14
Slika 6. Prikaz količine podataka potrebnih za preuzimanje prilikom instalacije [12] 15
Slika 7. Model za semestre.....17
Slika 8. Pogled za pregled semestara 18
Slika 9. Kontroler za pregled semestara 19
Slika 10. Preusmjeravanje zahtjeva na HTTPS 20
Slika 11. Primjer web app manifest-a 21
Slika 12. Registracija service workera 22
Slika 13. Instalacija service workera 22
Slika 14. Primjer procesa aktivacije 23
Slika 15. Spremanje predefiniranog odgovora u predmemoriju.....24
Slika 16. Primjer primjene fetch eventa i implementacija odabrane strategije 25
Slika 17. Postupak instalacije progresivne web aplikacije 26
Slika 18. Primjer splash screena progresivne web aplikacije 26
Slika 19. Prikaz implementacije predefiniranog odgovora u slučaju nedostatka veze na internet 27