Progresivne mrežne aplikacije

Andrišić, Mirna

Sveučilište J.J. Strossmayera u Osijeku

Filozofski fakultet

Dvopredmetni diplomski studij Informacijske tehnologije i Nakladništva

Mirna Andrišić

Progresivne mrežne aplikacije

Diplomski rad

Mentor: doc. dr. sc. Tomislav Jakopec

Osijek, 2023.

Sveučilište J.J. Strossmayera u Osijeku

Filozofski fakultet

Odsjek za informacijske znanosti

Dvopredmetni diplomski studij Informacijske tehnologije i Nakladništva

Mirna Andrišić

Progresivne mrežne aplikacije

Društvene znanosti, Informacijske i komunikacijske znanosti,
Informacijski sustavi i informatologija

Mentor: doc. dr. sc. Tomislav Jakopec

Osijek, 2023.

Prilog: Izjava o akademskoj čestitosti i o suglasnosti za javno objavljivanje

Obveza je studenta da donju Izjavu vlastoručno potpiše i umetne kao treću stranicu završnog odnosno diplomskog rada.

IZJAVA

Izjavljujem s punom materijalnom i moralnom odgovornošću da sam ovaj rad samostalno napravio te da u njemu nema kopiranih ili prepisanih dijelova teksta tuđih radova, a da nisu označeni kao citati s napisanim izvorom odakle su preneseni. Svojim vlastoručnim potpisom potvrđujem da sam suglasan da Filozofski fakultet Osijek trajno pohrani i javno objavi ovaj moj rad u internetskoj bazi završnih i diplomskih radova knjižnice Filozofskog fakulteta Osijek, knjižnice Sveučilišta Josipa Jurja Strossmayera u Osijeku i Nacionalne i sveučilišne knjižnice u Zagrebu.

U Osijeku, datum 17.08.2023.

Mirna Andrišić, 0122228740

ime i prezime studenta, JMBAG

Sažetak

Progresivne mrežne aplikacije (PWA) nazivaju se progresivnima zbog svoje postupne transformacije iz konvencionalne mrežne stranice u mobilnu aplikaciju koja se može bez problema izvoditi na mobilnim uređajima. Osim sposobnosti preuzimanja, ove aplikacije imaju moć funkcioniranja u offline načinu, zahvaljujući pozadinskim servisnim radnicima (eng. *service workers*) koji omogućavaju nesmetan rad. PWA aplikacije su fleksibilne i prilagodljive različitim dimenzijama zaslona i operativnim sustavima jer su smještene unutar preglednika. Dodatno, njihova sigurnost osigurana je putem HTTPS protokola, koji je integrirani dio preglednika. No, važno je napomenuti da zastarjele verzije preglednika možda neće podržavati sve PWA značajke. Razvojem PWA, organizacijama i tvrtkama se otvaraju nove perspektive za brzu i ekonomičnu izradu aplikacija. Progresivne aplikacije donose novu dimenziju mobilnih aplikacija te postaju sve popularniji izbor programera zbog njihovih značajki koje se približavaju *native* aplikacijama, pritom omogućavajući znatno brži i jednostavniji proces razvoja. U ovom radu proći će se svi koraci koji su potrebni za izradu ispravne PWA aplikacije i njezino objavljivanje na Internet.

Ključne riječi: progresivne mrežne aplikacije, razvoj mobilnih aplikacija, PWA, PWA značajke, implementacija PWA

Sadržaj

1. Uvod

Razvoj tehnologije omogućio je nesmetano korištenje interneta na mnoštvo različitih uređaja i u raznim dijelovima svijeta. Ulaskom u 21. stoljeće osobna računala bila su primarno sredstvo za pristup internetu, pa su tako i mrežne (eng. *web*) stranice bile prilagođene uređajima s većim dimenzijama ekrana i mišem kao glavnim ulaznim uređajem. U to vrijeme mobilni uređaji nisu bili namijenjeni za surfanje internetom, a mobilne stranice imale su jednostavan izgled i ograničene sposobnosti interakcije. U nekoliko godina mobilni uređaji su postali pametni, odnosno imali su sposobnosti instaliranja mobilnih aplikacija, povezivanje i surfanje internetom, razvijene operativne sustave itd. Pametni mobiteli popularizirali su korištenje mobilnih aplikacija, koje su bile prilagođene malim dimenzijama zaslona. S vremenom, mobilni uređaji postali su vodeći način pristupa internetu za mnoge korisnike, što je dovelo do daljnjeg napretka i razvoja mobilnih stranica i aplikacija.

Očekivanja korisnika kontinuirano su rasla, a brzina, jednostavnost i mogućnost korištenja aplikacija izvan mreže (eng. *offline*) postali su ključni čimbenici za korisničko zadovoljstvo. U takvom kontekstu, mobilne aplikacije postale su preferirani izbor korisnika zbog njihove praktičnosti i brzine pokretanja. Omogućile su korisnicima da brzo obavljaju različite zadatke u samo nekoliko sekundi, dok tradicionalne mrežne stranice nisu mogle osigurati takvo efikasno korisničko iskustvo. Analizirajući ponašanje korisnika, organizacije i tvrtke odlučile su se fokusirati na mobilno iskustvo kao prioritet (eng. *mobile-first*) prilikom stvaranja svoje aplikacije, no suočile su se s izazovom.

Danas postoji mnoštvo različitih mobilnih uređaja, koji imaju razne dimenzije zaslona, što znači da se aplikacija mora moći prilagoditi za svaki od tih zaslona. Vodeći izbor programerima je stvaranje izvorne (eng. *native*) aplikacije, čiji proces nastanka zahtjeva veliko ulaganje resursa u samu izradu, za svaki pojedini operativni sustav. Kao odgovor na te izazove, nastala je ideja o progresivnim mrežnim aplikacijama (eng. *progressive web applications*, PWA), koje se ističu svojom sposobnošću pružanja vrhunskog korisničkog iskustva bez obzira na uređaj ili preglednik koji korisnik upotrebljava.

U ovom radu detaljno će se pojasniti i definirati progresivne mrežne aplikacije, navesti njihove značajke i ključne dijelove, te po čemu se razlikuju od ostalih aplikacija. Proći će se i kroz povijest progresivnih aplikacija, njihove prednosti i nedostatke u odnosu na tri vrste aplikacije. U

drugom dijelu rada predstaviti će se aplikacija Trenutno vrijeme, koja će poslužiti kao vizualni pokazatelj PWA značajki. Na samom kraju rada govoriti će se o izazovima s kojima se susrelo za vrijeme izrade aplikacije i o budućnosti PWA. Cilj ovog rada je detaljno istražiti koncept progresivnih mrežnih aplikacija, istaknuti njihove značajke, mogućnosti i potencijalne izazove u usporedbi s drugim vrstama mobilnih aplikacija. Svrha ovog rada je pružiti dublji uvid u progresivne mrežne aplikacije te istražiti njihov utjecaj na digitalno okruženje i korisničko iskustvo.

2. Progresivne mrežne aplikacije

2.1. Povijest i razvoj PWA

Svakih nekoliko godina dogodi se ključan trenutak koji predstavi promjenu u digitalnom okruženju. Razvoj novih ili spajanje već postojećih tehnologija često označava značajan napredak za Internet.[1] Jedan takav presudan trenutak dogodio se 2000. godine s razvojem XMLHttpRequest-a, API-ja koji je omogućio dohvaćanje podataka putem poveznica (eng. *Uniform Resource Locator*, URL) stranice bez potrebe za osvježavanjem preglednika. Na temelju te tehnologije, razvio se AJAX, koji je omogućio stvaranje boljih, bržih i interaktivnijih aplikacija koristeći tehnologije poput XML-a, HTML-a, CSS-a i Javascript-a.[2] Upravo te tehnologije poslužile su kao temelj za razvijanje ideje o premještanju klasičnog mrežnog okruženja u prostor mobilnih uređaja.

Razvojem pametnih mobilnih uređaja, pojavila se potreba za prilagođavanjem mrežnog okruženja mobilnom načinu korištenja, no taj koncept predstavljen je tek 2007. godine kada je Steve Jobs predstavio prvi iPhone uređaj.[3] Njegova je inovacija bila omogućiti korisnicima pregledavanje mrežnih stranica na mobilnom uređaju i njihovo prilagođavanje za manje dimenzije. Ideja Steve Jobs-a podsjeća na koncept današnjih PWA aplikacija, zbog njihove lakoće preuzimanja i dostupnosti.[4] Godinu dana kasnije, s pojavom mobilnih aplikacija, programeri su morali prilagoditi mrežne aplikacije mobilnom okruženju, uvodeći nove značajke koje su dotad

[1] Ater, Tal. Building Progressive Web Apps, Kalifornija: O'Reilly Media, 2017.
[2] Cruz, A. Progressive Web Apps: An in-depth look at the past, the present, and the future, 2021.
[3] Nav. dj. Cruz, A.
[4] Khan, A.;Al-Badi, A.; Al-Kindi, M. Progressive Web Application Assessment Using AHP. // Procedia Computer Science 155 (2019). str 289-294.

bile nezamislive. *Nativne* aplikacije preplavile su tržište i promijenile su način na koji se koristi Internet. Njihove napredne značajke, poput grafike, lokacije (eng. *geolocation*), slanje obavijesti (eng. *push notifications*), *offline* prikazivanja i aplikacijskih ikona, učinile su ih očiglednim izborom.[5] U usporedbi s mobilnim, mrežne aplikacije nisu imale šanse u očima korisnika. Mobilne aplikacije nastavile su se godinama razvijati, no programeri su shvatili da je za potrebu izgradnje i održavanja *nativnih* aplikacija nužno uložiti mnogo financijskih, ljudskih i vremenskih resursa. Također, korisnici nisu bili zadovoljni jer aplikacije nisu bile dostupne *offline*, te je preuzeta aplikacija zauzimala previše prostora na mobilnim uređajima.

Ključni trenutak dogodio se 2015. godine kada su dizajner Frances Berriman i inženjer Google Chrome-a Alex Russel po prvi put upotrijebili izraz progresivne mrežne stranice. Tim nazivom odlučili su opisati aplikaciju koja progresivno, uz sve češću interakciju sa korisnikom, postaje prava aplikacija. Ono što ih čini posebnima je to što se ne moraju na tradicionalan način instalirati na uređaj, koriste nove značajke podržane u najnovijim preglednicima, te se mogu prikazivati u pregledniku i *offline*.[6] Iako su Berriman i Russel osmislili naziv, PWA tehnologija postala je popularna tek 2018. godine, kada su Google i Microsoft izdali najnovije verzije Chrome-a 70 i Windows-a 10, koje su podržavale tu tehnologiju. Također, Apple je u iOS verziji 11.3 odlučio omogućiti svojim korisnicima instaliranje PWA aplikacije, u obliku prečaca, što znači da korisnik nije morao tražiti odobrenje od Apple-ove trgovine.[7]

Istraživanja pokazuju da je svake godine sve manja vjerojatnost da će korisnik instalirati aplikaciju, dok se održavanje tradicionalnih *nativnih* aplikacija pokazalo kao sve skuplje rješenje. Pojavom PWA otvorile su se nove mogućnosti za organizacije i tvrtke koje žele što brže i jeftinije napraviti aplikaciju.[8] Progresivne aplikacije predstavljaju novu razinu mobilnih aplikacija, te su sve češći izbor programera zbog svojih značajki koje sliče *nativnim* aplikacijama, uz mnogo brži i jednostavniji proces izrade. Stvaranje PWA postalo je jedno od ključnih vještina koje bi programeri trebali usvojiti i implementirati u svom radu.[9]

[5] Nav. dj. Ater, Tal.
[6] Tandel, S.; Jamadar, A. Impact of Progressive Web Apps on Web App Development, 2018.
[7] Brown, R. Progressive Web Apps (PWAs): Definition, How They Work, and Why You Need One?
[8] Nav. dj. Ater, Tal.
[9] Magomadov, V. Exploring the role of progressive web applications in modern web development. // Journal of Physics: Conference Series 1679, 2(2020).

2.2. Definicija i opis PWA

Prije svega, važno je razumjeti da PWA nisu definirane kao jedinstvena tehnologija, nego su zapravo koncept koji se razvija već nekoliko godina. Taj koncept može biti prilično jednostavan za razumjeti kada se razdvoje i pojedinačno definiraju svi dijelovi tog termina. Riječ „progresivno“ naglašava njihovu mogućnost napredovanja ili razvijanja iz jednostavnih aplikacija u aplikacije koje su vrlo slične *nativnim* aplikacijama. Ovaj pristup omogućuje PWA da se postepeno prilagođava i razvija s vremenom kako bi se ispunili zahtjevi korisnika i novih tehnologija.[10] Riječ „mrežno“ odnosi se na računalnu mrežu (eng. *World Wide Web*), koji se temelji na strukturi interneta HTTP-u (eng. *Hypertext Transfer Protocol*) i jednoznačnim adresama, odnosno URL poveznicama.[11] PWA koriste tehnologije HTML, CSS i Javascript kako bi svojim korisnicima omogućile pristup sadržaju putem interneta. Naposljetku, riječ „aplikacija“ označava program ili programski jezik koji se može izvoditi u mobilnom ili računalnom mrežnom okruženju.[12] PWA može funkcionirati kao samostalna aplikacija koja ne zahtjeva preuzimanje, a ipak nudi slične značajke i usluge kao *nativne* aplikacije. Rastavljajući i analizirajući naziv PWA, dobila se jasnija slika o tome što čini ovu tehnologiju drugačijom od svih ostalih.

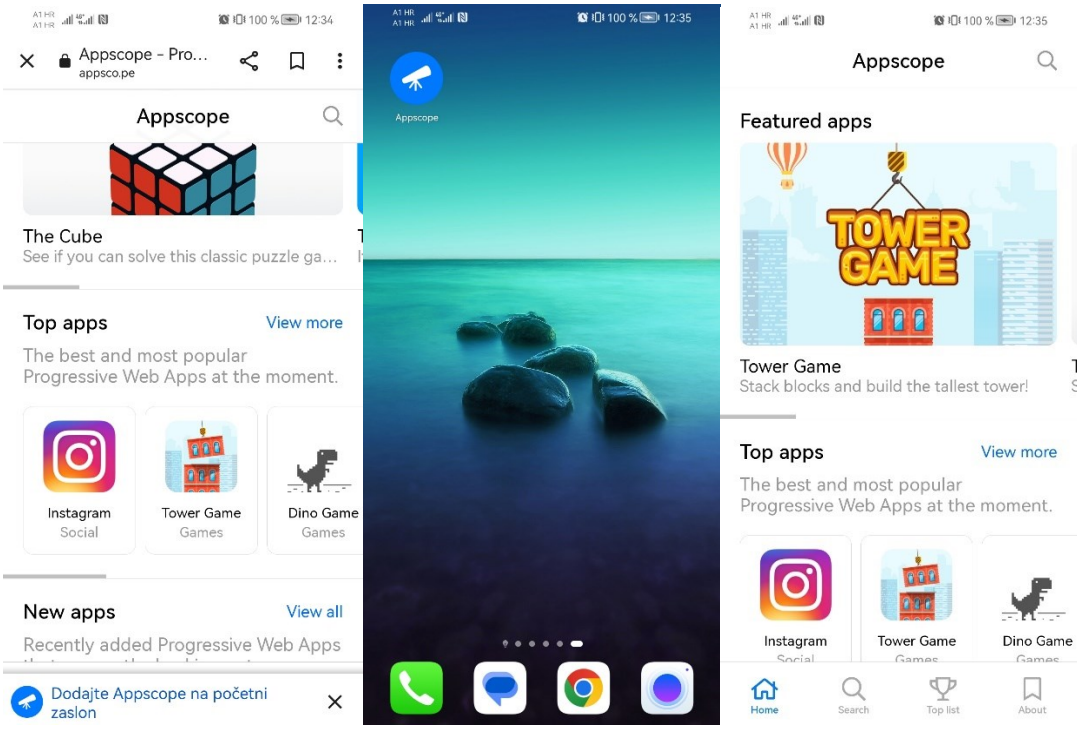
PWA aplikacije započinju kao obične mrežne stranice, ali progresivno stječu aplikacijske sposobnosti koje ih čine posebnima. Nakon čestog korištenja PWA aplikacije, preglednik predlaže dodavanje aplikacije na početni zaslon mobilnog uređaja. Tako PWA postaje dostupna kao prečac što omogućuje efikasan pristup i poboljšano korisničko iskustvo. Stvoreni prečac izgleda kao tradicionalna aplikacija koja ima svoju ikonu, a kada se pokrene aplikacija može raditi *offline* i poprimiti napredne značajke poput slanja obavijesti, pristup lokaciji i korištenja kamere. To je upravo ono što aplikaciju čini progresivnom, što je korisnici više posjećuju, pregledavaju i pretražuju informacije na aplikaciji, tako se aplikacija sve više razvija i dobiva više funkcionalnosti.[13]

2.3. Značajke i dijelovi PWA

[10] Nav. dj. Tandel, S.
[11] WWW. URL: https://www.enciklopedija.hr/natuknica.aspx?ID=66413
[12] Aplikacija. URL: https://www.enciklopedija.hr/natuknica.aspx?ID=3306
[13] Nav. dj. Tandel, S.

Raspravljajući za vrijeme večere, Frances Berriman i Alex Russel osim što su osmislili naziv koncepta progresivnih mrežnih aplikacija, napravili su i popis što one trebaju biti.[14] Na pitanje, Što čini PWA aplikaciju progresivnom, može se dobiti više odgovora, no u nastavku će se opisati neke od njezinih glavnih karakteristika.

Glavna značajka PWA je to što ih nije potrebno instalirati, već se mogu spremiti kao prečac na mobitel.[15] Aplikaciji se i dalje može pristupiti putem preglednika, no ova dodatna značajka omogućuje korisnicima jednostavniji i brži pristup samoj aplikaciji. Na slici 1. kao primjer te značajke pokrenuta je Appscope, trgovina PWA aplikacija. Prilikom prvog posjeta aplikacije na pregledniku, izbacio se skočni prozor o dodavanju aplikacije na početni zaslon. Nakon uspješnog preuzimanja prikazana je ikona aplikacije putem koje se otvara Appscope, koja izgleda kao klasična aplikacija.



Slika 1. Prikaz PWA aplikacije u pregledniku i instalirane kao prečac na mobilni uređaj

[14] Nav. dj. Russel, Alex.
[15] Korolov, S. Mobile App Development Approaches Explained, 2021.

Korisnici se često suočavaju s problemom nedovoljnog prostora za pohranu na svojim mobitelima, što može ograničiti njihov izbor aplikacija.[16] Na primjer, Starbucks, poznati lanac hrane i pića, razvio je PWA koji zauzima čak 99% manje prostora od svoje *nativne* verzije koja se mora instalirati. Ovakva ušteda prostora značajno povećava vjerojatnost da će korisnici odabrati PWA i koristiti ih redovito.[17] Veličina aplikacije je ponekad prevelika da bi se mogla instalirati, što znači da ju korisnik najvjerojatnije neće odabrati i koristiti. Radije će ju potražiti na internetu i koristiti ju putem preglednika. PWA djeluju kao most između preglednika i instaliranih aplikacija, navodeći korisnika na postavljanje prečaca, te potom i na postupno preuzimanje *nativne* verzije aplikacije putem Trgovine Play ili AppStore-a.[18] Ovo omogućuje glatku tranziciju, pružajući korisnicima potpunu kontrolu nad preuzimanjem i korištenjem aplikacije. Izgled prečaca PWA aplikacije dobiva se zapisivanjem ključnih metapodataka unutar JSON datoteke manifest.json.[19] Uobičajeni manifest sadrži naziv aplikacije, ikone koje će se koristiti te URL koji će se otvoriti kad se aplikacija pokrene. Ovi metapodaci ključni su za dodavanje aplikacije na početni zaslon uređaja i omogućuju joj ponašanje slično klasičnim aplikacijama.[20]

PWA aplikacije funkcioniraju *offline*, što korisnicima omogućuje pristup i korištenje čak i kada nisu povezani s internetom.[21] Korisnici očekuju da aplikacije uvijek prikazuju najnovije informacije i da usluge aplikacije budu dostupne i bez internetske veze. Međutim, mnoge tradicionalne aplikacije ne ispunjavaju ta očekivanja, često se ruše, prikazuju greške ili zahtijevaju internetsku povezanost.[22] PWA su riješile ovaj problem uz pomoć mehanizma servisnog radnika (eng. *service worker*)[23], koda napisanog u Javascript jeziku koji funkcionira odvojeno od ostatka aplikacije.[24] On djeluje kao sloj između aplikacije i mreže, omogućujući da se ključni resursi aplikacije pohrane u brzu priručnu memoriju preglednika (eng. *cache*). Time se postiže i brže učitavanje jer se statični dijelovi aplikacije mogu učitati iz *cache*-a preglednika, što se još zove i ljuska aplikacije (eng. *application shell*).[25]

[16] Darya, N. PWA VS native app: what to choose in 2023?, 2022.
[17] 12 Best Examples of Progressive Web Apps (PWAs) [Updated 2023].
[18] Wieckowska, S.; Bracisiewicz, M. PWA vs. MPA vs. SPA – What's the Best Choice for Your App?, 2022.
[19] Nyakundi, H. What is a PWA? Progressive Web Apps for Beginners, 2023.
[20] Jhala, D. A Study on Progressive Web Apps As A Unifier for Native Apps and the Web. // International Journal of Engineering Research & Technology (IJERT) 10, 5(2021). str. 207- 210.
[21] Nav. dj. Wieckowska, S.;
[22] Progressive Web Apps. URL: https://web.dev/progressive-web-apps/
[23] Detaljnije opisano u poglavlju 4.1. Service worker.
[24] Biørn-Hansen, A.; Majchrzak, T. A.; Grønli, T. Progressive Web Apps for the Unified Development of Mobile Applications, 2018.
[25] Nav. dj. Tandel, S.

Application shell je struktura aplikacije koju korisnik vidi kada po prvi put posjeti aplikaciju.[26] To je osnovni okvir korisničkog sučelja i temeljnih dijelova potrebnih za pokretanje aplikacije, iako obično ne sadrži stvarne podatke.[27] Nakon što se dodaju na početni zaslon, sve osnovne statičke datoteke kao što su HTML, CSS, JavaScript, slike i fontovi aplikacije, pohranjuju se na uređaju korisnika i postaju dostupne *offline*. Također, dinamički podaci mogu biti unaprijed pohranjeni za korištenje u *offline* ili slaboj internetskoj povezanosti. Ti će podaci biti osvježeni kad se pojave nove ažurirane informacije i kada se uređaj ponovno poveže na stabilnu mrežu.[28] Svaka sljedeća posjeta aplikaciji izvodit će se brže jer se većina temeljne strukture već nalazi u *cache*-u. Na taj način, kada korisnik nije spojen na internetsku mrežu, PWA i dalje funkcionira pouzdano i prikazuje posljednje spremljene informacije.[29] Kada se mobilni uređaj ponovno poveže na mrežu, podaci će se automatski uskladiti sa serverom.[30]

Osim što omogućuju *offline* funkciju, *service worker*-i također primaju obavijesti, ažuriraju aplikaciju i upravljaju mrežnim zahtjevima.[31] Od 2017. godine Safari više nije podržavao *service worker*-e, pa korisnici tog preglednika nisu mogli iskoristiti sve mogućnosti PWA, no od Safari 13 verzije to više nije problem.[32] Svi ostali moderni preglednici podržavaju PWA značajke, ukoliko se koristi najnovija inačica preglednika. Na stranici, Can I Use?, može se provjeriti koje sve verzije preglednika podržavaju, a koje ne podržavaju PWA (Slika 2).

[26] Nav. dj. Tandel, S.
[27] Nav. dj. Jhala, D.
[28] Majchrzak, T.; Biørn-Hansen, A.; Grønli, T. Progressive Web Apps: the Definite Approach to Cross-Platform Development? // 51st Hawaii International Conference on System Sciences (2018). str 5735-5744.
[29] Nav. dj. Progressive Web Apps. URL: https://web.dev/progressive-web-apps/
[30] Nav. dj. Khan, A.
[31] Staničić, O. Kraljević, S. ANALIZA ZNAČAJKI I PERFORMANSI PREOGRESIVNIH WEB APLIKACIJA. // Politehnika i dizajn 7, 4(2019). str. 240-247.
[32] Nav. dj. Biørn-Hansen, A.

Slika 2. Verzije preglednika koje podržavaju (zeleno), djelomično podržavaju (žuto) i ne podržavaju (crveno) PWA značajke [33]

Prilagođavanje sadržaja različitim dimenzijama uređaja (eng. *responsive*), predstavlja ključnu karakteristiku modernih aplikacija. Kako bi osigurali dosljedno i ugodno korisničko iskustvo, programeri često prilagođavaju aplikacije za tri glavne dimenzije: mobilnu (do 640px), tablet (između 641 i 1007px) i laptop ili računalnu verziju (više od 1008px).[34] Cilj je postići preglednost i estetski privlačan izgled na svakom uređaju i operativnom sustavu, što značajno poboljšava kvalitetu i pouzdanost aplikacije. PWA se prikazuju putem preglednika, što znači da se sadržaj automatski prilagođava dimenzijama uređaja. S druge strane, kod *nativnih* aplikacija, sadržaj se mora ručno prilagoditi za svaki specifičan uređaj, što u konačnici zahtjeva ulaganje dodatnog napora za programera.[35]

PWA aplikacije pružaju brže vrijeme učitavanja i prikazivanja sadržaja u usporedbi s drugim aplikacijama. Zahvaljujući *service worker*-ima, stranice se učitavaju u milisekundama, bez obzira na brzinu internetske veze. Prezentacija Google tima otkriva da PWA aplikacije ostvaruju nevjerojatnu brzinu koja je čak četiri puta veća od drugih vrsta aplikacija, bez obzira na snagu internetske veze. Stoga znatno veći broj korisnika i uređaja mogu istovremeno koristiti aplikaciju, ne ugrožavajući pritom kvalitetu usluge koja se pruža.[36] Brzina, odnosno sporost aplikacije uzrokuje slabo korištenje aplikacije i samim time smanjuje vjerojatnost da će ju posjetitelji htjeti instalirati i koristiti.

[33] Preuzeto sa izvora: URL: https://caniuse.com/serviceworkers
[34] Microsoft. Screen sizes and breakpoints, 2023.
[35] Nav. dj. Magomadov, V.
[36] Nav. dj. Brown, R.

Dodatna značajka PWA je njihova vidljivost i lakoća dijeljenja putem URL poveznice, kao što bi se učinilo s običnom mrežnom stranicom. Izravno su vidljive u pregledniku te ih se može pronaći bez potrebe posjećivanja trgovine aplikacija. Naime, pridržavaju se globalnih standarda i formata koji olakšavaju njihovo katalogiziranje, rangiranje i prikazivanje sadržaja na tražilicama.[37] Vidljivost i dostupnost PWA na webu čine ih snažnim alatom za poboljšanje korisničkog iskustva i postizanje većeg dosega publike. Međutim, ostvarile bi još veću vidljivost kada bi bile dostupne i za preuzimanje unutar trgovina. Korisnici imaju naviku pretraživanja i instaliranja aplikacija putem trgovina, pa će teže otkriti mogućnost instaliranja verziju aplikacije, koja zauzima manje prostora, putem preglednika.

Sigurnost koju PWA pružaju temelji se na implementaciji HTTPS protokola, koji osigurava šifriranje podataka koji se dijele između aplikacije i servera. Takve šifrirane podatke mogu čitati samo ovlaštene osobe s pravim ključem, dok neovlaštene osobe ne mogu razumjeti njihov sadržaj.[38] Da bi se ostvarila sigurnost, za registraciju *service worker*-a i njegovo reagiranje na događaje, nužno je koristiti HTTPS. HTTPS sprječava potencijalne zloupotrebe kao što su presretanje veza, izmišljanje ili filtriranje odgovora putem *service worker*-a.[39] Dodatna zaštita PWA osigurana je u pregledniku u kojem se aplikacija učitava, jer se kontinuirano unaprjeđuju načini zaštite i sigurnosti implementacijom standarda za razvoj sigurnih aplikacija i mrežnog okruženja.[40] Naravno, za samu sigurnost zaslužni su i *service worker*-i, koji automatski ažuriraju aplikaciju, što znači da programeri mogu na brz i jednostavan način otkriti i riješiti potencijalne probleme.[41]

Spoj svih ovih navedenih značajki čine progresivnu mrežnu aplikaciju, koja kombinira najbolje dijelove *nativnih* mobilnih aplikacija i tradicionalnih mrežnih stranica. Ovaj koncept je sve prisutniji u današnjem digitalnom svijetu, te je postao jedan od vodećih opcija u razvoju aplikacija.

[37] Kremer, M. What is a PWA?, 2023.
[38] Nav. dj. Kremer, M.
[39] Sharma, V.; Verma, R.; Pathak, V.; Paliwal, M.; Jain, P. Progressive Web App (PWA) - One Stop Solution for All Application Development Across All Platforms. // International Journal of Scientific Research in Computer Science, Engineering and Information Technology (2019).
[40] Tran, J. PWA vs Native App and how to choose between them, 2022.
[41] Nav. dj. Staničić, O.

2.4. Prednosti i nedostaci PWA

Prednosti PWA mogu se gledati iz tri različite perspektive: svakodnevnih korisnika, programera koji stvaraju aplikaciju i organizacije ili tvrtke za koju je aplikacija namijenjena. Korisnici očekuju da se PWA aplikacije brzo učitavaju i ispunjavaju poslane zahtjeve, uz mogućnost prikaza sadržaja izvan mreže. Također, traže ažurnost sadržaja i estetski privlačan dizajn, kako bi im pružili ugodno i zanimljivo korisničko iskustvo.[42] S druge strane, programeri žele efikasno razvijati aplikaciju uz minimalan napor i vrijeme uloženo u izradu, a da ujedno završna aplikacija bude kvalitetna i sigurna za uporabu.[43] Konačno, organizacije imaju visoka očekivanja od PWA-a, zahtijevajući privlačnu aplikaciju koja ne zahtijeva ulaganje obilnih financijskih resursa za svoju izradu i promociju.[44] PWA bi trebale kvalitetno odražavati misiju i usluge organizacije, pružajući dosljedno iskustvo korisnicima. U svrhu informiranog odlučivanja, potrebno je pažljivo proučiti sve prednosti i nedostatke PWA aplikacija. Analizom ovih ključnih čimbenika može se utvrditi je li PWA pravi izbor za zadovoljenje potreba korisnika, olakšavanje razvojnog procesa za programere i postizanje ciljeva organizacija.

Prednosti i mogućnosti PWA su mnoge, te su neke već prethodno spomenute, pa će se u nastavku samo ukratko ponoviti neki od najvažnijih razloga za odabir PWA, radi usporedbe s njihovim nedostacima i izazovima. PWA su prilagodljive s različitim uređajima i operativnim sustavima, omogućujući široku dostupnost korisnicima na mobilnim uređajima, tabletima i osobnim računalima. Izrada PWA aplikacija je brža i jeftinija od drugih vrsta aplikacija, što privlači programere koji žele brzo pokrenuti svoje projekte, ali i organizacije koje žele izbjeći velika ulaganja u razvoj. Organizacije također nemaju potrebu za promocijom aplikacije jer je ona vidljiva na tražilicama i dostupna putem svakog preglednika.[45] Osim izbjegavanja potrebe za trgovinama aplikacija, pretraživanjem i instalacijama, korisnici PWA-a se oslobađaju potrebe za stalnim ažuriranjem ili prihvaćanjem ponuda za ažuriranja svaki put kada programer lansira novu verziju.[46] Također, PWA pružaju visoku razinu sigurnosti i mogućnosti *offline* rada, podržanu HTTPS protokolom.[47]

[42] PWA. URL: https://web.dev/what-are-pw
[43] Nav. dj. Tran, J.
[44] Nav. dj. Cruz, A.
[45] Nav. dj. Tran, J.
[46] Sharma, S.; Bhardwaj, A. Progressive Web Apps (PWA). // Journal of Emerging Technologies and Innovative Research 7(2021). str 696-706.
[47] Nav. dj. Nyakundi, H.

Service worker-i zaslužni su za veliku većinu ovih prednosti. Oni omogućuju brzo, interaktivno i jednostavno korištenje aplikacije, dohvaćaju podatke iz *cache*-a, šalju obavijesti i ažuriraju aplikaciju. PWA se automatski ažuriraju kad korisnici pristupe aplikaciji, čime se izbjegava potreba za preuzimanjem ažuriranja kao što je to slučaj kod tradicionalnih mobilnih aplikacija.[48] PWA također štede prostor za pohranu na uređajima korisnika, jer zauzimaju znatno manje prostora, ostavljajući pritom više mjesta za instaliranje drugih aplikacija.[49] Sve ove prednosti čine PWA popularnim izborom za različite vrste aplikacija, nudeći poboljšano korisničko iskustvo, jednostavnost rada i široku dostupnost.

Svaka nova tehnologija donosi sa sobom određene izazove koji mogu biti ključni prilikom odabira izrade aplikacije. PWA aplikacije bolje podržavaju Android uređaji u odnosu na iOS. Naime, Apple ne dopušta pristup podacima koji su potrebni za funkcioniranje PWA značajki kao što su Face ID, Touch ID, Bluetooth, baterija i svjetiljka. Osim toga, ove značajke nisu dostupne ni u starijim verzijama preglednika, što može predstavljati izazov za korisnike i njihovo korisničko iskustvo.[50] Stoga, korisnici bi uvijek trebali imati instaliranu najnoviju verziju preglednika kako bi iskoristili sve značajke PWA i osigurali kvalitetno korisničko iskustvo.

Također, važno je napomenuti da se PWA pokreću putem internetskog preglednika, što znači da može doći do kašnjenja i veće potrošnje baterije u usporedbi s *nativnom* aplikacijom.[51] Također, PWA nisu zastupljene u klasičnim trgovinama aplikacija, već se mogu pronaći samo putem pretraživanja u pregledniku, te postavljanjem prečaca na mobilni uređaj. Unatoč svojoj dostupnosti, zastupljenost PWA u trgovinama je minimalna, što smanjuje njihovu vidljivost i mogućnost korištenja. Trgovine aplikacija Android i Google upravljaju tržištem, oni određuju popularnost i rangiranje pojedinih aplikacija. Naravno, ako aplikacije nisu dostupne putem trgovine, korisnici ih teže mogu otkriti i koristiti. PWA aplikacije nemaju mogućnost prilagodbe specifičnim korisnicima, kao što su *loyalty* programi i personalizirani sadržaji, što može ograničiti njihovu privlačnost za određene organizacije.[52]

Izrada PWA zahtijeva više vremena i truda u odnosu na klasične mrežne stranice, što je dodatni posao za programere. Prvo je potrebno napraviti običnu aplikaciju na koju se naknadno trebaju dodati PWA značajke. Ujedno, te značajke nisu standardizirane, što predstavlja izazov

[48] Nav. dj. Staničić, O.
[49] Nav. dj. Darya, N.
[50] Simplelearn. Demystifying Progressive Web Apps: What They Are, How They Work, and Why They Matter, 2023.
[51] Nav. dj. Tran, J.
[52] IQUII. Progressive Web App (PWA): what they are, pros and cons and the main examples on the market, 2019.

prilikom implementacije minimalno održivog proizvoda (eng. *minimum viable product*, MVP) aplikacije u kratkom vremenskom roku.[53]

2.5. Primjeri PWA

Većina organizacija koje su stvorile PWA verziju svojih aplikacija primijetile su značajan porast posjećenosti i interakcije s korisnicima. Primjerice, Twitter je pokrenuo PWA Twitter Lite i zabilježio porast od 75% više *tweet*-ova i 20% manje napuštanja sa stranice, uz dodatnu uštedu podataka i brže učitavanje. Forbes, popularna novinarska stranica odlučila je poboljšati korisničko iskustvo te je pritom izabrala PWA. Prema njihovoj statistici, primijetili su 100% porast u angažmanu korisnika, 43% više sesija po korisniku i 20% veću vidljivost oglasa.[54] Kreativna društvena mreža Pinterest primijetila je da se samo 1% njihovih korisnika prijavilo i imaju instaliranu mobilnu aplikaciju, najvjerojatnije zbog lošeg korisničkog iskustva. Odlučili su stvorili PWA verziju koja je rezultirala povećanjem vremena provedenog na aplikaciji za 40%, 44% većom vidljivošću specifičnih oglasa i 60% većom interakcijom korisnika. Glazbena aplikacija Spotify odlučila je razviti PWA aplikaciju nakon što je Apple zahtijevao udio od 30% naknade za držanje aplikacije u svojoj trgovini. PWA verzija Spotify-ja znatno je brža i ima razvijeno sučelje koje mijenja pozadinu na temelju postupnog kretanja aplikacijom. Uber PWA još je jedan primjer uspješne implementacije, koji je omogućio rezerviranje i pozivanje taxi službe čak i na mreži brzine 2G. Uber PWA postala je idealno rješenje za vozače koji koriste starije mobitele ili nemaju pristup snažnoj internetskoj vezi. Aplikacija se može učitati na 2G mreži za samo 3 sekunde.[55] Svi ovi primjeri potvrđuju da PWA može značajno poboljšati korisničko iskustvo aplikacija, čineći ih izvrsnim izborom za organizacije koje žele ostvariti pozitivne rezultate i zadovoljiti potrebe svojih korisnika.

Prema istraživanju Stone Temple, uočen je porast u mobilnom pretraživanju za 63% u odnosu na prošlu godinu. Kissmetrics također navodi da 47% korisnika očekuje da se stranica učita za dvije ili manje sekunde i da će čak 40% korisnika napustiti stranicu ako joj je potrebno više od 3 sekunde za učitavanje sadržaja. TechCrunch je prije dvije godine izjavio da su 51%, odnosno većina američkih korisnika, prosječno instalirali nula aplikacija mjesečno.[56] Korisnici su zahtjevni,

[53] Nav. dj. Magomadov, V.
[54] Nav. dj. Brown, R.
[55] 12 Best Examples of Progressive Web Apps (PWAs) [Updated 2023].
[56] Nav. dj. Brown, R.

te od aplikacija očekuju najbolje moguće korisničko iskustvo. Razumljivo je da će napustiti aplikaciju ukoliko joj je potrebno dugo vremena za učitavanje ili provođenje zahtjeva. Aplikacije koje se sruše nestankom internetske veze, korisnicima pružaju dojam loše kvalitete i izdržljivosti. Za razliku od takvih aplikacija PWA nastavljaju svoj rad, prikazujući korisnicima da se trebaju spojiti na Internet ukoliko žele iskusiti sve čari aplikacije. Time ostvaruju bolji odnos sa svojim korisnicima, čime se povećava vjerojatnost ponovnog posjeta korisnika. Statistički podaci o popularnim PWA ukazuju na jasnu potražnju za takvim aplikacijama te sugeriraju da će njihova popularnost i korištenje u budućnosti samo rasti.

Osim navođenja primjera uspješnih PWA aplikacija potrebno je i usporediti PWA s drugim vrstama aplikacija, kako bi se u konačnici utvrdilo koje aplikacije su najbolje, najbrže i najkvalitetnije na tržištu.

3. Usporedba PWA s ostalim aplikacijama

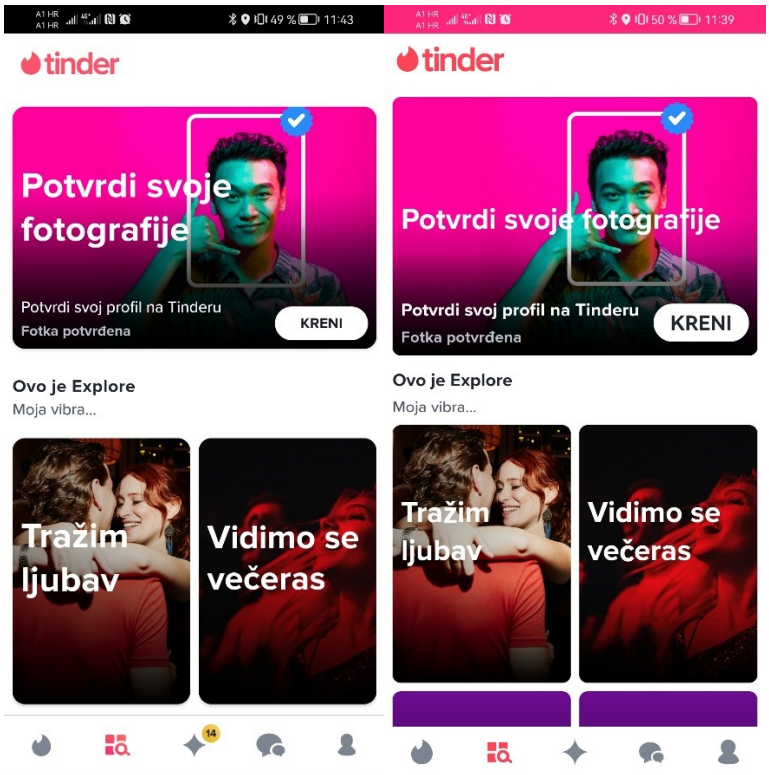
3.1. Usporedba PWA i *nativnih* aplikacija

Nativne mobilne aplikacije najpopularniji su izbor za razvoj mobilnih aplikacija među programerima i organizacijama. One su specifično izrađene za određeni operativni sustav, kao Android ili iOS, koristeći programski jezik i razvojno okruženje koje podržava taj operativni sustav.[57] Naziv *native* potječe od predinstaliranih aplikacija na platformama Mac i PC, poput Fotografija (Photos), Pošte (Mail) ili Kontakata (Contacts). U kontekstu mobilnih aplikacija, termin *native* označava aplikacije koje su dizajnirane da optimalno funkcioniraju na specifičnoj platformi. *Nativne* Android aplikacije razvijaju se korištenjem programskih jezika poput Java ili Kotlin, dok su *nativne* iOS aplikacije izrađene pomoću jezika Swift ili Objective-C.

Nativne aplikacije obično pružaju vrhunsku kvalitetu korisničkog iskustva, ali zahtijevaju angažiranje različitih razvojnih timova i dodatne napore kako bi se stvorila verzija za svaku platformu. To može dovesti do povećanja troškova i produženja vremena razvoja aplikacije. Također, održavanje *nativnih* aplikacija može biti izazovno, jer zahtijeva brigu o svakoj pojedinačnoj aplikaciji stvorenoj za specifične platforme.[58] Na slici 3. prikazana je *nativna* i PWA

[57] BEGINNER'S GUIDE TO APPS: NATIVE VS HYBRID VS PWA, 2021.
[58] Gillis, A. S. Native app.

verzija poznate aplikacije Tinder. Na prvi pogled aplikacije izgledaju vrlo slično, no postoje male razlike poput veličine proreda između sadržaja, statusne trake, pa i veličine tipografije.



Slika 3. Usporedba *nativne* (lijevo) i PWA Tinder aplikacije (desno)

Uspoređujući *nativne* i PWA aplikacije, može se primijetiti da dijele većinu značajki, kao što su pristup kameri, lokaciji, primanje obavijesti, *offline* način rada. Međutim, PWA aplikacije imaju određena ograničenja u usporedi s *nativnim*.[59] Pojedine zastarjele verzije preglednika ne podržavaju PWA značajke, dok *nativne* aplikacije imaju iste sposobnosti na svim verzijama. Također, mogućnosti uporabe pojedinih značajki su naprednije, u odnosu na PWA. Međutim, PWA su privlačne jer se mogu implementirati s manjim financijskim, ljudskim i vremenskim ulaganjima u odnosu na izradu *nativnih* aplikacija. Prednost PWA aplikacija leži u mogućnosti razvoja samo jedne verziju koja se automatski prilagođava različitim preglednicima. Prednost *nativnih* aplikacija

[59] Nav. dj. Tran. J.

je njihova zastupljenost u trgovinama aplikacija, čime se poveća njihova vidljivost i dostupnost putem procesa optimizacija trgovine aplikacija (eng. *App Store Optimization,* ASO).[60]

Iako PWA aplikacije pružaju veću sigurnost od klasičnih mrežnih stranica, zbog HTTPS protokola, ipak se ne mogu usporediti s *nativnim* aplikacijama po pitanju sigurnosti. *Nativne* aplikacije omogućuju nadogradnju dodatnih sigurnosnih mjera, čime postižu višu razinu pouzdanosti. Korisnici imaju više povjerenja u mobilnu aplikaciju od URL poveznice, koja se može nalaziti na nepouzdanim stranicama. Također, postavljanje prečaca takvih aplikacija može biti riskantno zbog opasnosti virusa i zlouporabe osobnih podataka korisnika. *Nativne* aplikacije ostvaruju bolje korisničko iskustvo u odnosu na PWA aplikacije zahvaljujući bržem i snažnijem kodu.[61] Ovo znači da će *nativne* aplikacije raditi brže, a također će trošiti manje baterije na mobilnom uređaju u usporedbi s PWA aplikacijama.[62] S jedne strane, PWA aplikacije su praktičnije i ekonomičnije za izradu, s mogućnošću rada na raznim platformama. S druge strane, *nativne* aplikacije nude naprednije značajke i sigurnost, ali zahtijevaju više ulaganja u izradu i održavanje. PWA su dobar izbor kada se želi brzo razviti aplikaciju, no treba uzeti u obzir da je kvaliteta *nativne* aplikacije znatno bolja i samim time predstavlja najbolji odabir za izradu aplikacije.

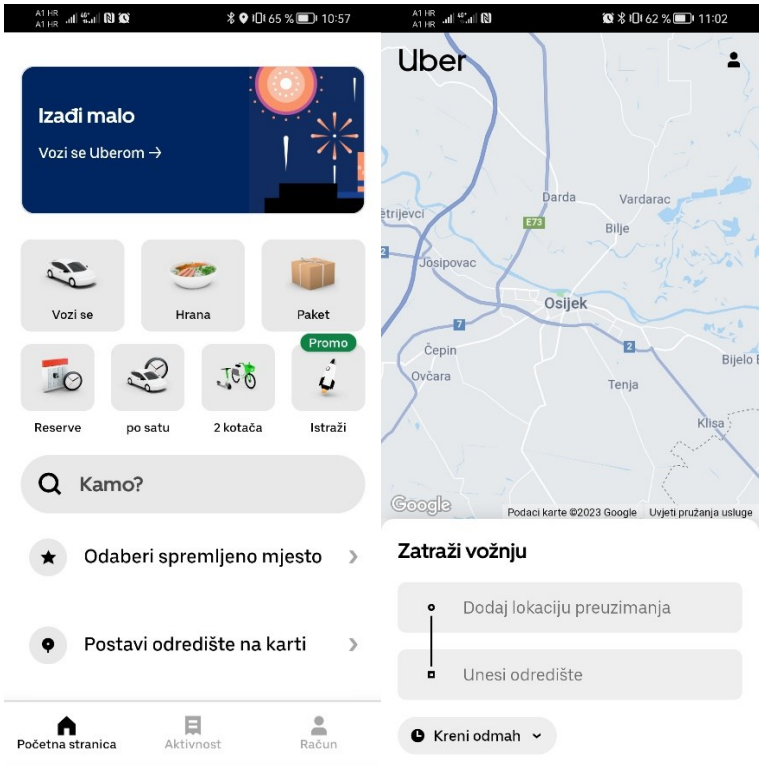
Tablica 1. Usporedba značajki PWA i *nativnih* aplikacija

Kriterij	**PWA aplikacije**	***Nativne* aplikacije**
Razvoj i novčana ulaganja	Manja ulaganja, brži razvoj	Veća ulaganja, sporiji razvoj
***Offline* način prikaza**	Dostupno	Dostupno
Brzina	Brzo učitavanje	Brže učitavanje
Prilagođenost	*Responzivni* dizajn za različite uređaje	Prilagođene svakom operativnom sustavu
Značajke	Samo jednostavne značajke	Napredne značajke
Sigurnost	Visoki standard sigurnosti, HTTPS protokol	Napredne sigurnosne mjere, veća pouzdanost
Dostupnost i vidljivost	Lako dostupne putem preglednika, postavljanje prečaca na mobilni uređaj	Preuzimanje putem trgovine aplikacija

[60] Nav. dj. Nyakundi, H.
[61] Nav. dj. Tran, J.
[62] Nav. dj. Darya, N.

3.2. Usporedba PWA i hibridnih aplikacija

Hibridne (eng. *hybrid*) aplikacije su vrsta mobilnih ili mrežnih aplikacija koje nastaju korištenjem HTML, CSS i Javascript tehnologija, zajedno s određenim *nativnim* elementima koji su specifični za svaku platformu. Naziv „hibridne“ odnosi se na njihovu sposobnost spajanja mrežnih elemenata s mobilnim, stvarajući aplikaciju koja je dostupna putem trgovina aplikacija ili preko preglednika. Ključna karakteristika hibridnih aplikacija je njihova integracija u *nativni* spremnik (eng. *container*) WebView, koji omogućava bogato korisničko iskustvo (eng. *user experience*, UX) te pristup značajkama poput kamere i lokacije.[63] Primjeri hibridnih aplikacija su: Evernote, Amazon App Store, Instagram, Uber i Gmail.[64] Na slici 4. prikazana je usporedba hibridne i PWA verzije Uber aplikacije. Hibridna verzija ima veći izbor opcija, dok je PWA aplikacija namijenjena za samo najjednostavnije, rezerviranje taxi vožnje.



Slika 4. Usporedba hibridne (lijevo) i PWA Uber aplikacije (desno)

[63] Nav. dj. Korolov, S.
[64] Dharmwan, S. 10 Examples of Hybrid Apps that have taken Businesses to the next Level, 2023.

Hibridne aplikacije dijele sličnosti s PWA jer su prilagođene operativnim sustavima Android i iOS, koristeći isti kod. Međutim, izrada hibridnih aplikacija zahtijeva veće ulaganje financijskih sredstava, budući da se PWA značajke mogu jednostavno integrirati u već postojeće aplikacije, što značajno skraćuje vrijeme i smanjuje troškove razvoja. Prednost PWA je mogućnost pružanja sadržaja *offline*, dok se hibridne aplikacije ne mogu koristiti bez pristupa internetu. Također, brzina hibridnih aplikacija može biti izazovna, često zahtijevajući optimizaciju kako bi se osiguralo glatko korisničko iskustvo.[65] PWA aplikacije podložne su izmjenama, svaka promjena u kodu automatski se prikazuje i prilagođava u aplikaciji. S druge strane, hibridne aplikacije imaju ograničene postavke prilagodbe, jer se može dogoditi da pojedini operativni sustav ne podržava te promjene.[66] Dostupnost i vidljivost PWA može se postići putem tražilice i preglednika, uz mogućnost postavljanja prečaca na mobilni uređaj, dok se hibridne aplikacije moraju instalirati kao obične aplikacije, iako se mogu koristiti i putem preglednika.[67]

Hibridne aplikacije prilagođene su s različitim mobilnim uređajima korištenjem istog koda, što rezultira uštedom vremena i financijskih sredstava za organizacije prilikom razvoja. Unatoč tome, hibridne aplikacije nisu u mogućnosti pružiti optimalno korisničko iskustvo te možda neće biti sposobni iskoristiti sve jedinstvene značajke operativnog sustava. Iako su prikladne za izradu MVP-a, mogu postojati bolji izbori za potpuno razvijene digitalne proizvode.[68]

[65] PWA vs. Native App vs. Hybrid vs. SPA vs. MPA vs. Responsive Website, 2023.
[66] Nav. dj. Dharmwan, S.
[67] Nav. dj. PWA vs. Native App vs. Hybrid vs. SPA vs. MPA vs. Responsive Website, 2023.
[68] Isto PWA vs. Native App vs. Hybrid vs. SPA vs. MPA vs. Responsive Website, 2023.

Tablica 2. Usporedba značajki PWA i hibridnih aplikacija

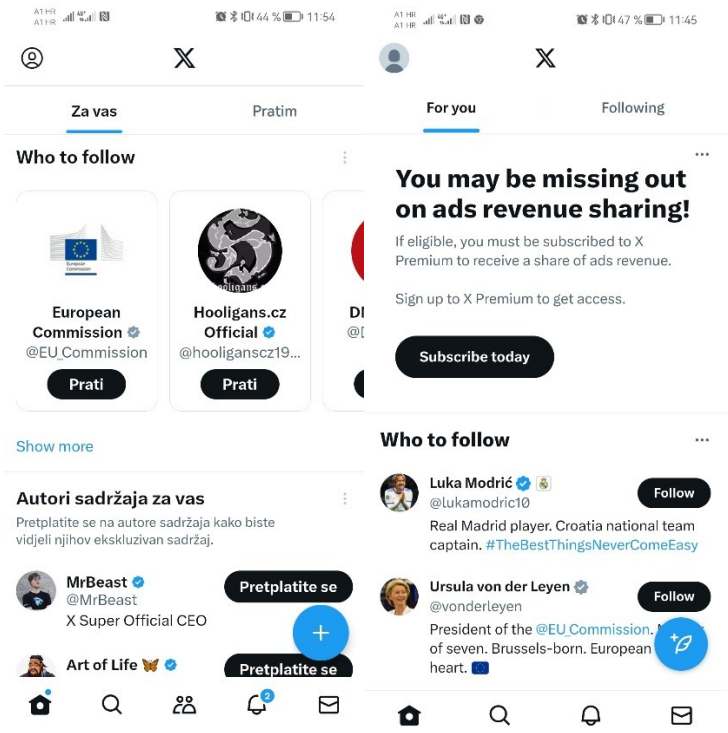
Kriterij	PWA aplikacije	Hibridne aplikacije
Prilagođenost	Isti kod za sve platforme	Isti kod za Android i iOS
Razvoj i ulaganja	Manja ulaganja i brži razvoj	Veća ulaganja i sporiji razvoj
Integracija u postojeće aplikacije	Jednostavna	Zahtjeva dodatne napore i resurse
***Offline* način prikaza**	Dostupno	Nedostupno
Korisničko iskustvo	Ograničeno iskustvo, ali bolje od hibridnih	Ograničeno iskustvo, neke značajke se ne mogu koristiti
Dostupnost i vidljivost	Dostupno putem preglednika, prečaci	Klasično preuzimanje aplikacije

3.3. Usporedba PWA i SPA aplikacija

Tradicionalna aplikacija sastoji se od više stranica koje su međusobno povezane. Svaku od tih pojedinačnih stranica, preglednik otvara kao novi entitet, uključujući HTML dokument i sve ostale izvore poput CSS i Javascript datoteke. SPA (eng. *Single-page application*) je, kao i PWA, jedna od novijih vrsta aplikacija, a razlikuje se po tome što sadrži samo jedan HTML dokument koji postupno mijenja svoj izgled kao rezultat korisnikove interakcije sa sadržajem, bez da se učitava cijela stranica. Kada korisnik klikne unutarnje linkove, Javascript dohvaća novi sadržaj sa servera i ažurira ključne dijelove stranice.[69] Primjeri SPA aplikacija su: Gmail, Twitter, PayPal, LinkedIn i Airbnb.[70] Na slici 5. prikazana je usporedba Twitter (X) SPA i PWA verzije aplikacije. SPA verzija prikazuje sadržaj na hrvatskom jeziku i ima jednu dodatnu opciju u glavnom izborniku, dok PWA aplikacija ima samo četiri opcije. Međutim, uočeno je da je PWA aplikacija mnogo brža od SPA verzije.

[69] What is a progressive web app?. URL: https://developer.mozilla.org/en-US/docs/Web/Progressive_web_apps/Guides/What_is_a_progressive_web_app#pwas_and_single-page_apps

[70] Nav. dj. Wieckowska, S.



Slika 5. Usporedba Twitter (X) SPA (lijevo) i PWA aplikacije (desno)

SPA ujedno može biti i PWA, što je često slučaj, ali PWA ne mora nužno biti SPA.[71] SPA su brze, ne zauzimaju puno prostora i jednostavne su za korištenje. Uspoređujući brzinu SPA s PWA aplikacijama, vrlo su slične, no PWA su malo brže, a svaka sekunda se broji kod korisnika koji je nestrpljiv i želi što brže dobiti tražene informacije. Ipak, vrijeme potrebno za razvoj SPA aplikacija je znatno kraće u odnosu na PWA aplikacije, te su ujedno i jeftinije za izradu. Što se tiče korisničkog iskustva, PWA pružaju kvalitetnu uslugu i imaju sposobnost učitavanja sadržaja i interaktivnih značajki u rekordnom vremenu i to u *offline* načinu. Upravo je to razlog zašto se sve više organizacija odlučuje za pretvaranje SPA u PWA aplikacije. PWA također pružaju bolju sigurnost jer djeluju pod HTTPS protokolom, koji osigurava visoki standard sigurnosti. Ako aplikacija zahtijeva unos osobnih podataka ili kartičnih informacija, PWA je bolji izbor. U odnosu na vidljivost i dostupnost aplikacije, PWA još jednom izlaze kao pobjednik. Mogu se pokrenuti mrežno i *offline*, te omogućavaju dodavanje prečaca na mobitele, što korisniku olakšava svakodnevno korištenje.[72]

[71] Nav. dj. What is a progressive web app?
[72] Nav. dj. Wieckowska, S.

Tablica 3. Usporedba značajki PWA i SPA aplikacija

Kriterij	PWA aplikacije	SPA aplikacije
Korisničko iskustvo	Bolje, interaktivnost, brzina	Dobro, ali ograničeno
Razvoj i novčana ulaganja	Veća ulaganja i sporiji razvoj	Manja ulaganja i brži razvoj
Brzina	Brža aplikacija	Brza aplikacija
Sigurnost	Visok standard sigurnosti, HTTPS protokol	Standardna sigurnost
***Offline* način prikaza**	Dostupno	Nedostupno

4. Izrada PWA pomoću React-a

Prije samog procesa stvaranja aplikacije, potrebno je definirati jasne odgovore na nekoliko ključnih pitanja: Koja je svrha aplikacije?, Za koga je namijenjena?, Koje usluge pruža? i Postoji li potražnja za takvom aplikacijom? Za svrhu ovog rada, aplikacija se ne planira lansirati širokoj javnosti, stoga se nije potrebno baviti analizom tržišne potražnje. Svrha ovog rada je razviti aplikaciju koja dosljedno zadovoljava glavne PWA standarde, te na samom kraju testirati i implementirati aplikaciju. Za potrebe ovog rada, izraditi će se jednostavna aplikacija nazvana Trenutno vrijeme[73], koja koristi OpenWeather API u svrhu prikazivanja trenutnog vremena bilo gdje u svijetu. Usluga je besplatna i može se koristiti nakon registracije na OpenWeather stranici, na kojoj se dobiva i pristup vlastitom API ključu. Aplikacija će pružati informacije o trenutnoj temperaturi, ilustraciji vremenskih uvjeta i opisu za određeni grad za koji se traži informacija.

Za izradu korisničkog sučelja progresivnih web aplikacija (PWA) nužne su tehnologije kao što su HTML5, CSS3 i JavaScript programski jezik. Također, preporučeno je odabrati odgovarajući okvir (eng. *framework*) za razvoj aplikacije, od kojih su Angular, React i Vue najpopularniji izbori.[74] Ova aplikacija izraditi će se u React *framework*-u, JavaScript biblioteci za

[73] Github repozitorij aplikacije: https://github.com/mandrisic/PWA_weather

[74] Huber, S.; Demetz, L.; Felderer, M. PWA vs the Others: A Comparative Study on the UI Energy-Efficiency of Progressive Web Apps, 2021.

izgradnju korisničkih sučelja. Kao razvojno okruženje, odabran je Visual Studio Code (VSC), alat koji je široko prihvaćen i omogućuje učinkovitu izradu i uređivanje koda.

S obzirom na to da je najvažniji dio ovog rada implementiranje PWA značajki, samo će se ukratko opisati dio razvoja koji nije vezan uz taj koncept. Da bi se izradila React aplikacija potrebno je instalirati Node.js, koji omogućuje preuzimanje raznih paketa s interneta. Zatim, u VSC-u potrebno je kreirati i otvoriti novi direktorij u kojem će se aplikacija razvijati. U Terminalu potrebno je unijeti osnovnu naredbu:

```
npx create-react-app ./
```

Naredbom će započeti preuzimanje svih React paketa iz Javascript biblioteke unutar stvorenog direktorija. Postupak traje nekoliko minuta, a nakon završetka preuzimanja može se započeti stvarati aplikacija. Naredba

```
npm start,
```

izvodi se kako bi pokrenuli aplikaciju na lokalnom serveru, pri čemu će se stranica automatski osvježavati svaki put kada napravimo izmjene u kodu. Unutar glavnog direktorija aplikacije nalazi se „public“ direktorij, u kojem se nalazi index.html datoteka, i „src“ direktorij u kojem će se pisati glavni kod aplikacije. Jedna od najvažnijih datoteka u „src“ direktoriju je App.js u kojoj je zapisan veliki dio React koda. Za potrebe pohranjivanja i prikazivanja datuma zadnjeg *online* posjeta aplikaciji za vrijeme *offline* povezanosti, potrebno je preuzeti dva paketa: serviceworker-storage i date-fns. Serviceworker-storage je paket koji omogućuje lokalno pohranjivanje (eng. local storage) podataka koji se dohvaćaju u aplikaciji pomoću *service worker*-a.[75] Date-fns je Javascript biblioteka koja se koristi za rad s datumima. Osim date-fns, može se koristiti i Moment.js, no biblioteka date-fns je manja i lakša za korištenje, jer se dohvaćaju samo one funkcije koje su potrebne aplikaciji.[76] Naredbe koje su se koristile za preuzimanje tih paketa su

```
npm install --save serviceworker-storage i

npm install date-fns –save.
```

Također, osim navedenih paketa bilo je potrebno dohvatiti i datoteku fetchWeather.js na kojoj je definirana adresa API-ja, s kojeg se dohvaćaju podaci, i jedinstven API ključ, koji se dobije

[75] Sagawara, R. Serviceworker-storage, 2017. URL: https://github.com/RyotaSugawara/serviceworker-storage

[76] How to format datetime in ReactJS, 2023. URL: https://www.altcademy.com/blog/how-to-format-datetime-in-reactjs/

prilikom prijave na OpenWeather stranici. Unutar fetchWeather datoteke potrebno je preuzeti još jednu Javascript biblioteku Axios, koja olakšava izvođenje HTTP zahtjeva. Za preuzimanje Axios paketa koristila se naredba:

```
npm install axios.
```

Ostale datoteke unutar „src“ direktorija su index.js koji dohvaća App.js datoteku i App.css koja stilizira elemente unutar App.js datoteke. Svaka PWA aplikacija započinje kao obična aplikacija na koju se naknadno primjenjuju značajke, pa je tako i ovdje bio slučaj. Izrada osnovne React aplikacije nije bila previše zahtjevna te je brzo izvedena u relativno kratkom vremenskom roku.

4.1. Service worker

Glavna značajka koja PWA aplikacije čini posebnima su *service worker*-i, koji su zapravo obična Javascript datoteka smještena u „public“ direktoriju aplikacije. *Service worker* radi konstantno, aktivira se čim se aplikacija pokrene, te čak nastavlja raditi nakon što se aplikacija zatvori. Stoga su izvanredan alat za implementiranje *offline* mogućnosti unutar PWA aplikacija. Osnovni cilj je omogućiti aplikaciji da funkcionira i kad korisnici nemaju pristup internetu.

Prvo je potrebno kreirati serviceworker.js datoteku u kojoj će se zapisivati sav kod koji *service worker* treba obavljati u pozadini aplikacije. U ovoj aplikaciji postoje dvije stranice index.html i offline.html, koje će se prikazati kada korisnik nije povezan na Internet. U sljedećem dijelu proći će se kroz cijeli kod unutar serviceworker.js datoteke, kako bi se vizualno moglo što jasnije opisati što znači specifičan dio koda. Na samom početku serviceworker,js datoteke bilo je potrebno definirati verzije *cache*-a i popisa URL-ova koji će biti pohranjeni u tom *cache*-u. Prvo je definirana varijabla „CACHE_NAME“, koja se odnosi na *cache* preglednika. Kada korisnik pokrene aplikaciju, pozadina i tražilica se ne moraju svaki put dohvaćati i učitavati, već su spremljeni u *cache*-u. Stvorila se inačica „CACHE_NAME“ pod nazivom „version-1“. Ako se kasnije naprave izmjene u *application shell*-u, potrebno je preimenovati „CACHE_NAME“ u „version-2“. Na primjer, ako želimo dodati još jednu stranicu unutar aplikacije koju želimo pohraniti u *cache*, tada ga moramo preimenovati. Naziv *self* u ovom slučaju odnosi se na *service worker*-a, te ga se mora povezati s *this*, što predstavlja *service worker*-a u SW dokumentaciji.

```
const CACHE_NAME = "version-1";
const urlsToCache = ['index.html', 'offline.html'];

const self = this;
```

Slika 6. Definiranje verzije i popisa URL-ova za *cache*

Budući da se može instalirati, PWA bi trebao pravilno funkcionirati čak i u situacijama kada nema internetske veze, koristeći skladišne mehanizme u preglednicima kao što je IndexedDB.[77] Na slici 7. može se uočiti kod u kojem se stvara IndexedDB baza podataka. Funkcija

`openDatabase()`

otvara i stvara bazu podataka nazvanu *myDatabase*, u koju će se pohranjivati svi podaci koje korisnik bude pretraživao za vrijeme jedne sesije. Podaci će se spremiti u tzv. *dataStore*, i dohvaćati prema jedinstvenom ključu tog podatka.

```
// Handle storing and retrieving data in IndexedDB
function openDatabase() {
    return new Promise((resolve, reject) => {
        const request = indexedDB.open('myDatabase', 1);

        request.onupgradeneeded = (event) => {
            const db = event.target.result;
            const store = db.createObjectStore('dataStore', { keyPath: 'key' });
        };

        request.onsuccess = (event) => {
            const db = event.target.result;
            resolve(db);
        };

        request.onerror = (event) => {
            reject(event.target.error);
        };
    });
}
```

Slika 7. Stvaranje IndexedDB baze

U nastavku koda, koji se može vidjeti na slici 8. i 9., stvaraju se funkcije *storeData* i *retrieveData* za spremanje i dohvaćanje podataka iz IndexedDB baze podataka. Funkcija

[77] Nav. dj. Biørn-Hansen, A.

storeData prima ključ i podatke koji se žele pohraniti, tako što otvara bazu podataka i stvara novi objekt za *dataStore*. Ti podaci će biti pohranjeni u bazu podataka pod specifičnim ključem. Funkcija *retrieveData* prima ključ pod kojim se žele dohvatiti traženi podaci. Ako je zahtjev već jednom proveden onda se naredbom

`request.onupgradeneeded`

ažuriraju podaci, ako je zahtjev uspješan, odnosno

`request.onsuccess`,

dohvaćaju se podaci, a ako se dogodi greška, odnosno

`request.onerror`,

odbacuje se zahtjev. Ovim se funkcijama ostvaruje interakcija s IndexedDB bazom podataka.

```
27  function storeData(key, data) {
28      openDatabase().then((db) => {
29          const transaction = db.transaction(['dataStore'], 'readwrite');
30          const objectStore = transaction.objectStore('dataStore');
31          objectStore.put({ key, data });
32      });
33  }
```

Slika 8. Funkcija za spremanje podataka u bazu

```
35  function retrieveData(key) {
36      return new Promise((resolve, reject) => {
37          openDatabase().then((db) => {
38              const transaction = db.transaction(['dataStore'], 'readonly');
39              const objectStore = transaction.objectStore('dataStore');
40              const request = objectStore.get(key);
41
42              request.onsuccess = (event) => {
43                  const data = event.target.result ? event.target.result.data : null;
44                  resolve(data);
45              };
46
47              request.onerror = (event) => {
48                  reject(event.target.error);
49              };
50          });
51      });
52  }
```

Slika 9. Funkcija za dohvaćanje podataka iz baze

Nakon stvaranja baze podataka potrebno je pokrenuti instalaciju *service worker*-a, kao što je vidljivo na slici 10. Instalacijom se pokreće događaj (eng. *event*) kojim se želi dodati *urlsToCache*, odnosno adrese stranica same aplikacije. Ako je sve uspješno instalirano u konzoli će se pojaviti odgovor „Otvoren cache“.

```
54  // Install SW
55  self.addEventListener('install', (event) => {
56      event.waitUntil(
57          caches.open(CACHE_NAME)
58              .then((cache) => {
59                  console.log('Otvoren cache');
60                  return cache.addAll(urlsToCache);
61              })
62      )
63  });
```

Slika 10. Instalacija *service worker*-a

Osim instalacije, potrebno je definirati što *service worker* treba raditi kada primi specifičan zahtjev, npr. prikaz stranice ili slike. Ako se zahtjev ne uspije izvršiti znači da korisnik nije spojen na Internet i tada *service worker* treba otvoriti offline.html stranicu aplikacije. Na slici 11. može se vidjeti kako to izgleda u kodu.

```
66  // Listen for requests
67  self.addEventListener('fetch', (event) => {
68      event.respondWith(
69          caches.open(CACHE_NAME)
70              .then((cache) => {
71                  return cache.match(event.request)
72                      .then((cacheResponse) => {
73                          if (cacheResponse) {
74                              return cacheResponse;
75                          }
76
77                          if (event.request.url.includes('api.openweathermap.org')) {
78                              return fetch(event.request)
79                                  .then((fetchResponse) => {
80                                      const responseToCache = fetchResponse.clone();
81                                      cache.put(event.request, responseToCache);
82                                      return fetchResponse;
83                                  })
84                                  .catch(() => caches.match('offline.html'));
85                          }
86
87                          return fetch(event.request)
88                              .catch(() => caches.match('offline.html'));
89                      });
90              })
91      );
92  });
```

Slika 11. Izvođenje zahtjeva i preusmjeravanje s index.html na offline.html datoteku

Nakon toga, dolazi se do dijela koda koji omogućava komunikaciju između *service worker*-a i aplikacije putem poruka. Aplikacija može zatražiti pohranu ili dohvaćanje podataka u *online* ili *offline* stanju, a *service worker* odgovara izvršavanjem odgovarajućih funkcija i slanjem podataka natrag stranici. Ovaj dio koda može se vidjeti na slici 12.

```
96  // Handle message from page
97  self.addEventListener('message', (event) => {
98      if (event.data.action === 'storeData') {
99          const { key, data } = event.data;
100         storeData(key, data);
101     } else if (event.data.action === 'retrieveData') {
102         const { key } = event.data;
103         retrieveData(key).then((data) => {
104             event.source.postMessage(data);
105         });
106     } else if (event.data.action === 'getOnlineStatus') {
107         event.source.postMessage({ type: 'onlineStatus', isOnline: navigator.onLine });
108     }
109 });
```

Slika 12. Komunikacija između *service worker*-a i aplikacije

Aktivacijom *service worker*-a provjeravaju se postoje li neke izmjene na aplikaciji, a ako postoje *service worker* treba izbrisati sve prethodne verzije i omogućiti prikaz najnovije inačice aplikacije. „CACHE_NAME“ se želi uvijek zadržati i to se čini pisanjem naredbe:

`cacheWhitelist.push(CACHE_NAME).`

Ako cacheWhitelist ne sadrži cacheName, *service worker* ga briše, u suprotnom će ga sačuvati. Ovo se često koristi kako bi se osiguralo ažuriranje podataka i obavijestilo korisnike o promjenama u *online*/*offline* statusu. Na slici 13. prikazan je cijeli kod za aktivaciju *service worker*-a.

```
111  // Activate the SW
112  self.addEventListener('activate', (event) => {
113      const cacheWhitelist = [];
114      cacheWhitelist.push(CACHE_NAME);
115
116      event.waitUntil(
117          caches.keys().then((cacheNames) => Promise.all(
118              cacheNames.map((cacheName) => {
119                  if (!cacheWhitelist.includes(cacheName)) {
120                      return caches.delete(cacheName);
121                  }
122              })
123          )).then(() => {
124              self.clients.matchAll().then(clients => {
125                  clients.forEach(client => {
126                      client.postMessage({ type: 'onlineStatus', isOnline: navigator.onLine });
127                  });
128              });
129          })
130      )
131  });
```

Slika 13. Ažuriranje izmjena na aplikaciji

Posljednji korak obuhvaća izradu offline.html stranice, koja će biti aktivna u *offline* načinu rada. Ova datoteka sadrži jednostavan tekst koji će se prikazati korisnicima kada nemaju pristup internetu. Aplikacija će i dalje raditi u pozadini, ali neće prikazivati sadržaj stranice kao inače, te neće moći izvoditi zahtjeve sve dok se korisnik ne spoji na internet. Na slici 14. vidljivo je da je aplikacija pokrenuta na mobilnom uređaju, dok korisnik nije spojen na Internet. U takvom slučaju, korisniku će se prikazati poruka: „Za provjeru trenutnog vremena, molimo vas da se spojite na internet“.

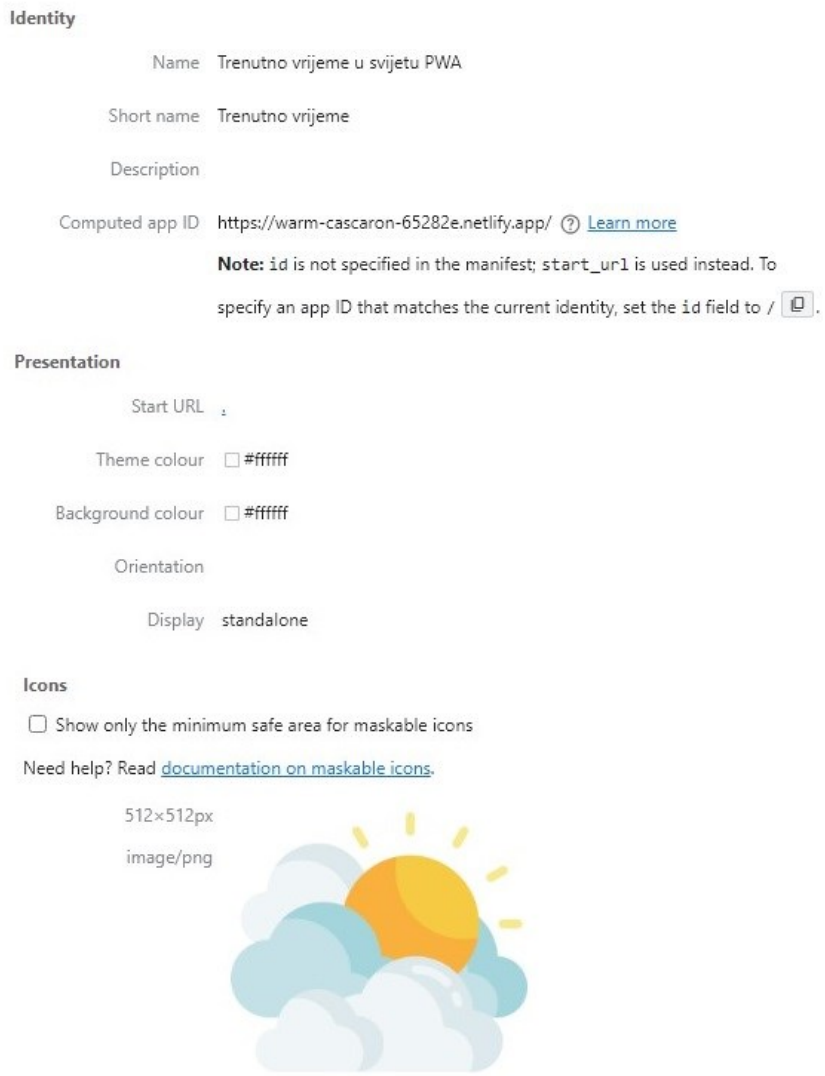
Slika 14. Aplikacija u *offline* prikazu

4.2. Manifest

Manifest je JSON datoteka koja sadrži važne metapodatke koji opisuju izgled i ponašanje aplikacije. Najčešće se definira naziv aplikacije, opis, paleta boja i ikone. Ikona aplikacije treba biti visoke rezolucije kako bi izgledala kvalitetno na svim dimenzijama zaslona. Preporuča se i korištenje ikone u više dimenzija, onda je potrebno svaku ikonu posebno definirati i naglasiti o kojoj dimenziji se radi. Različite dimenzije ikona će se bolje uklopiti na različitim Android i Apple uređajima, pa je poželjno pripremiti više ikona. Aplikacija Trenutno vrijeme koristi samo jednu ikonu, koja je preuzeta s platforme Icons8, s besplatnom dozvolom za korištenje. Bez manifest datoteke aplikacija se ne može računati kao PWA aplikacija i neće uspješno proći Lighthouse testiranje. Cijeli kod napisan u manifest.json datoteci može se vidjeti na slici 15. Chrome DevTools omogućava prikaz svih podataka zapisanih u manifest.json datoteci, što se može uočiti na slici 16.

```json
{
    "short_name": "Trenutno vrijeme",
    "name": "Trenutno vrijeme u svijetu PWA",
    "icons": [
        {
            "src": "/images/logo512.png",
            "type": "image/png",
            "sizes": "512x512",
            "purpose": "any maskable"
        }
    ],
    "start_url": ".",
    "display": "standalone",
    "theme_color": "#ffffff",
    "background_color": "#ffffff"
}
```

Slika 15. Sadržaj manifest.json datoteke



Slika 16. Podaci opisani u Manifest-u prikazani u Chrome pregledniku

5. Testiranje i implementiranje PWA

Lighthouse Chrome-ov je alat koji je dostupan svima na uporabu, a posebice je koristan programerima koji su se odlučili za razvoj PWA aplikacije. Lighthouse omogućuje sveobuhvatno testiranje aplikacije, te generira izvješće na temelju dvije stavke: preuzimanja i optimizacije aplikacije po PWA standardima. Nakon toga, rezultati neuspjelih stavki mogu poslužiti kao pokazatelji kako poboljšati aplikaciju s detaljnim objašnjenjem zašto je potrebno napraviti izmjenu i na koji način poboljšati i popraviti aplikaciju.[78]

Lighthouse alat može se otvoriti desnim klikom miša na našoj stranici, čime se otvara prozor i odabire provjeri (eng. *inspect*) opcija. S desne strane otvara se Chrome izbornik koji pruža uvid u kod i, što je nama najbitnije, omogućuje izvođenje testa aplikacije. U gornjem dijelu nalaze se opcije elementi (eng. *elements*), konzola (eng. *console*), uvidi u izvedbu (eng. *performance insights*), izvori (eng. *sources*) i mreža (eng. *network*). Nakon tih opcija dolaze dvije strelice koje otvaraju dodatne opcije izvedba (eng. *performance*), memorija (eng. *memory*), aplikacija (eng. *application*), sigurnost (eng. *security*) i Lighthouse. Klikom na Lighthouse opciju, otvara se alat koji može testirati izvedbu (eng. *performance*), pristupačnost (eng. *accessibility*), najbolje primjere iz prakse (eng. *best practices*), SEO i PWA aplikacije.

Sve navedene opcije testiranja pružaju dubok uvid u trenutno stanje aplikacije, ističući područja koja zahtijevaju izmjene kako bi se postigla veća brzina i pouzdanost. Odabirom opcije za generiranje PWA testiranja, pokreće se Lighthouse testiranje kojim se želi utvrditi može li se aplikacija instalirati i optimiziranost PWA značajki. Testiranje može trajati nekoliko sekundi, a sam proces je vidljiv u mobilnom prikazu s lijeve strane, koji simulira otvaranje i učitavanje sadržaja aplikacije. Nakon što se testiranje završi, generiran je popis svih značajki aplikacije koji čine aplikaciju upravo PWA, te rezultati Trenutno vrijeme aplikacije na toj skali. Aplikacija je uspješno ispunila sve zahtjeve za preuzimanje i optimizaciju, čime je službeno postala prava PWA aplikacija.

[78] Lighthouse. URL: https://developer.chrome.com/docs/lighthouse/overview/

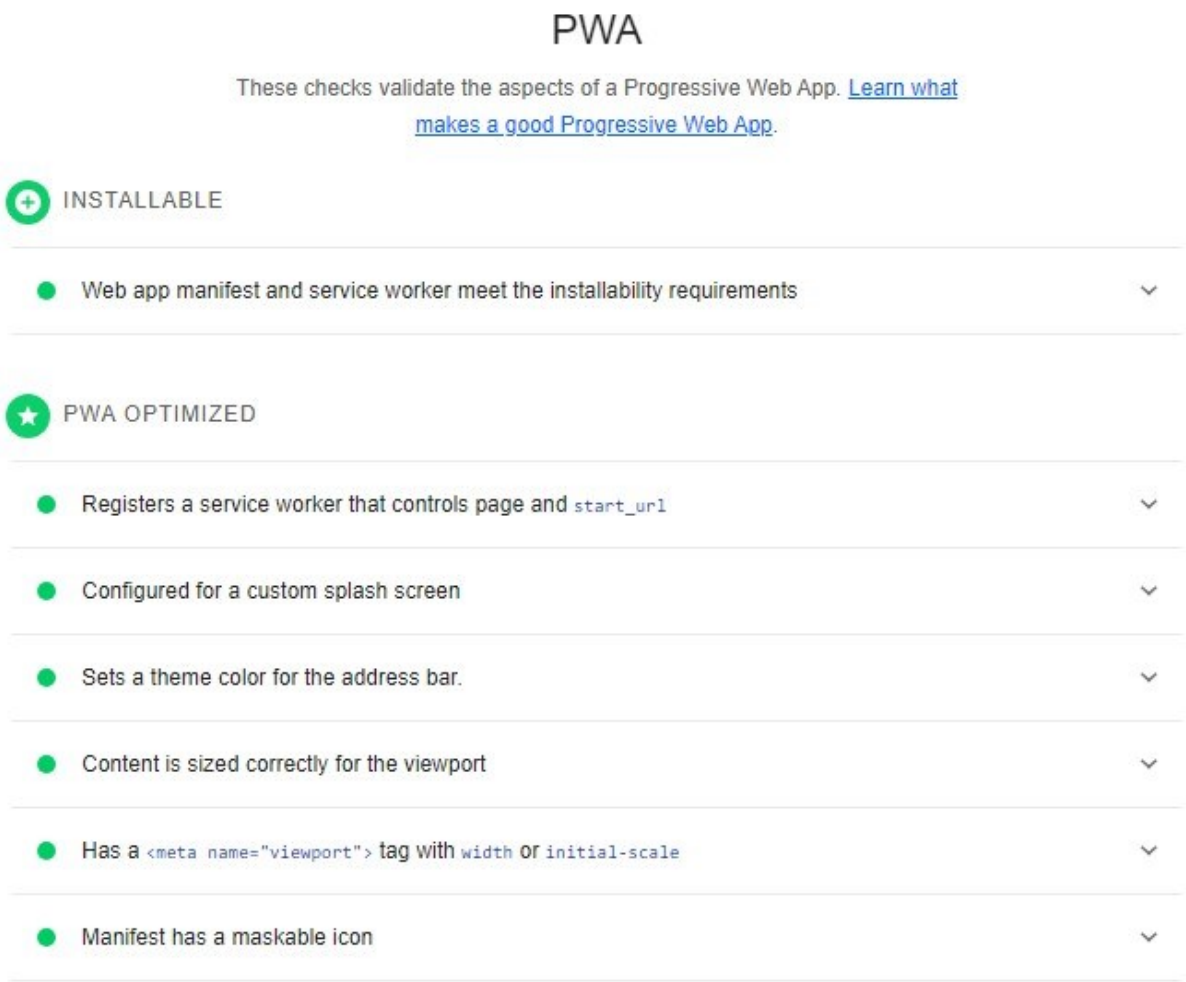
Slika 17. Aplikacija ispunjava sve zahtjeve za preuzimanje i optimizaciju

Zahtjev za preuzimanje provjeravao je rad manifest-a i *service worker*-a. Ako aplikacija nema manifest.json datoteke ili registriranog servisnog radnika, tada ne može biti instalirana. Manifest datoteka osim što treba imati definirane metapodatke aplikacije, mora se povezati sa index.html datotekom putem `<link>` oznake (eng. *tag*). Tek nakon uspješnog izvođenja ovih koraka, preglednik prikazuje skočni prozor za preuzimanje aplikacije, odnosno kao mogućnost postavljanja aplikacije kao prečac na mobitel.

Trenutno postoji šest kriterija koji procjenjuju PWA optimizaciju aplikacije. Neki od ovih zahtjeva mogu se jednostavno izmijeniti u kodu, kao na primjer dodavanje `purpose` elementa sa `maskable` ili `any-maskable` vrijednostima unutar područja s ikonama manifest.json datoteci. `Maskable` je novi format koji omogućuje da ikona aplikacije izgleda visokokvalitetno na svim Android uređajima. Iako Lighthouse ne provjerava vizualni izgled ikone, to je zadaća programera, svejedno zahtijeva korištenje `maskable` vrijednosti kako bi se osigurala kompatibilnost. Zahtjev koji većina današnjih stranica koriste u svojoj index.html datoteci je `<meta name="viewport">` *tag* koji sadrži atribute `width` ili `initial-scale`. Prilikom stvaranja početnog koda svake HTML5 datoteke, generira se nekoliko linija koda među uključujući i spomenuti `<meta>` *tag*, tako da je to prvih izvedivih zahtjeva. Također, potrebno je dodati još jedan `<meta>` *tag*, koji definira tematsku boju aplikacije. Boja se određuje unutar `content` atributa, u HEX ili RGB obliku, i time se definira boja navigacije i učitavanja aplikacije nakon preuzimanja.

Sljedeći zahtjev odnosi se na prilagođavanje sadržaja aplikacije, definiranjem `viewport` veličine koja odgovara optimalnoj dimenziji prikazanog sadržaja na različitim uređajima. Ključno je prilagoditi aplikaciju barem za klasičan i mobilni prikaz. Korisnici s pravom očekuju da će

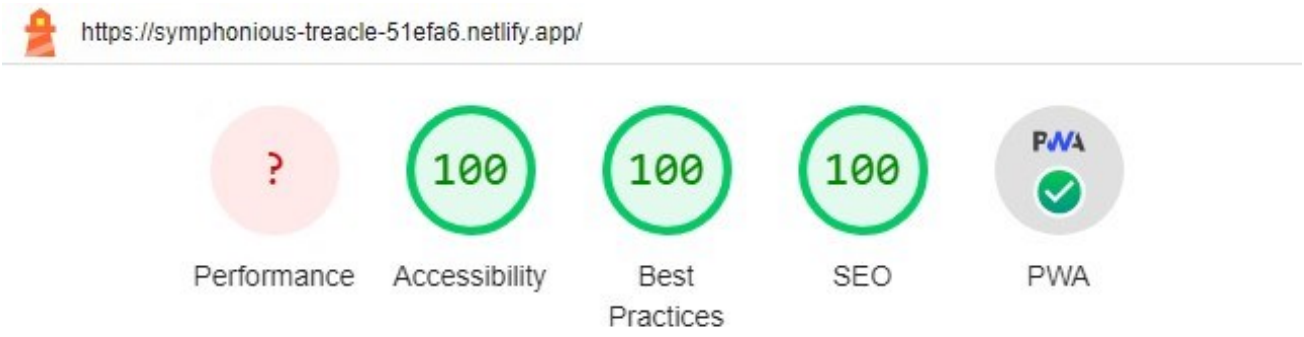
sadržaj aplikacije biti prilagođen dimenzijama uređaja na kojima koriste aplikaciju. Nedostatak prilagodbe sadržaja može dovesti do dojma zastarjelosti i niske kvalitete aplikacije, što u konačnici može odvratiti korisnike od njenog korištenja. Za posljednja dva zahtjeva optimizacije nužno je korištenje manifesta i *service worker*-a. Manifest je neophodan za definiranje početnog izgleda aplikacije prilikom njezinog učitavanja, uključujući prikaz tematske boje i ikone minimalne veličine od 512x512px. Tek kada su svi zahtjevi preuzimanja i PWA optimizacije ispunjeni dobije se ispravna PWA aplikacija, kao što je prikazano na slici 18. Aplikacija Trenutno vrijeme uspješno je ispunila sve zahtjeve i uspješno je prošla Lighthouse testiranje.



Slika 18. Stavke koje su ispunjene za PWA ispravnu aplikaciju

Nakon uspješnog PWA testiranja provelo se i testiranje svih ostalih ponuđenih opcija Lighthouse alata. Aplikacija je uspješno prošla sve testove, osim testa izvedbe (eng. *performance*),

jer se aplikacija nije uspjela učitati unutar očekivanog vremena. Nakon nekoliko pokušaja aplikacija i dalje nije uspjela postići rezultat na ovom testu, vjerojatno jer joj je trebalo previše vremena za učitavanje sadržaja i izvršavanje svih zahtjeva. Na sljedećem prikazu može se uočiti rezultat testiranja na sve testove Lighthouse alata.



Slika 19. Rezultati svih Lighthouse testova

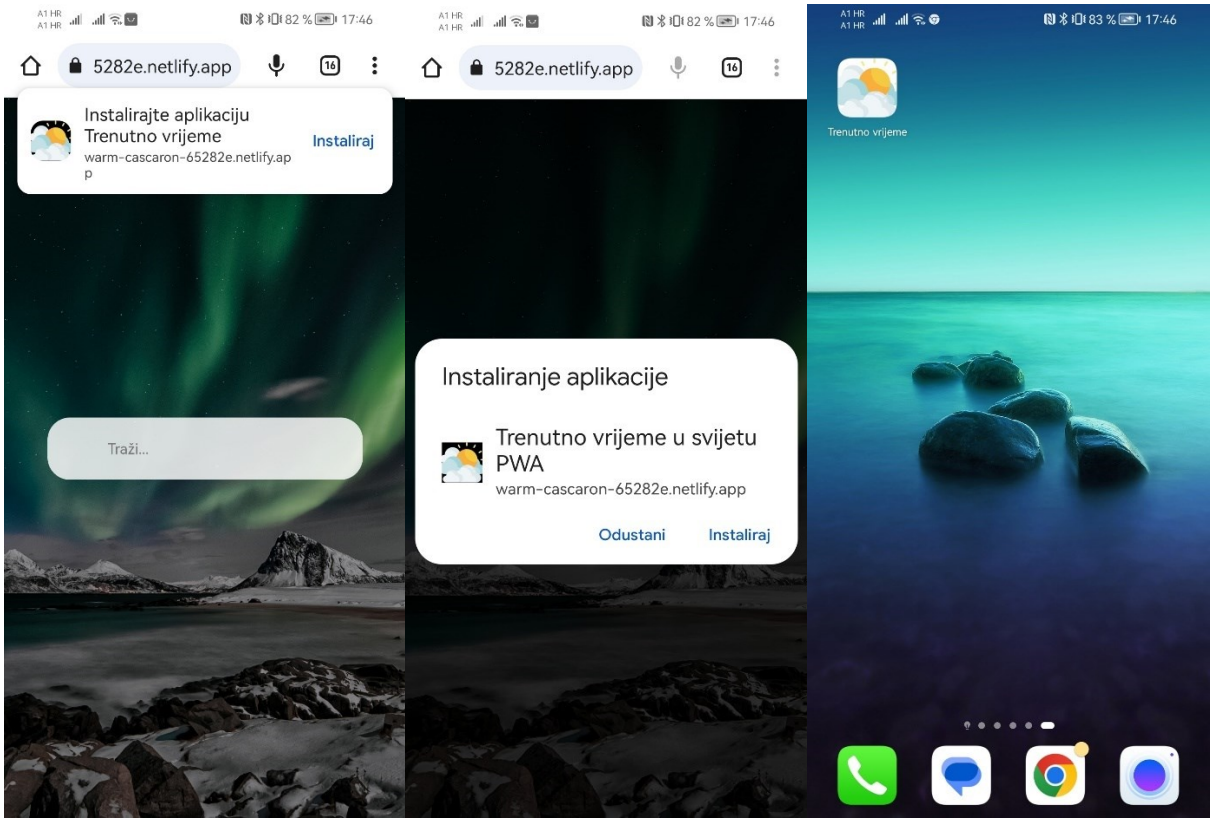
Osim samog procesa testiranja, trebalo je osigurati da aplikacija bude dostupna putem Netlify-ja. Netlify je platforma koja omogućuje razvoj i jednostavnu dostupnost dinamičnih aplikacija putem Interneta za nesmetano korištenje svima. Za objavljivanje aplikacije na Netlify potrebno je postaviti direktorij aplikacije na platformu. React PWA aplikacija u sebi ima direktorij „public“ i „src“. Unutar „public“ direktorija nalaze se samo HTML datoteke, dok se ostatak koda napisanog u Javascript-u nalazi u „src“ direktoriju. Međutim, postavljenjem cijelog direktorija na Netlify, aplikacija ne bi bila objavljena i spremna za korištenje. Prije samog postavljanja na Netlify potrebno je pokrenuti naredbu:

```
npm run build
```

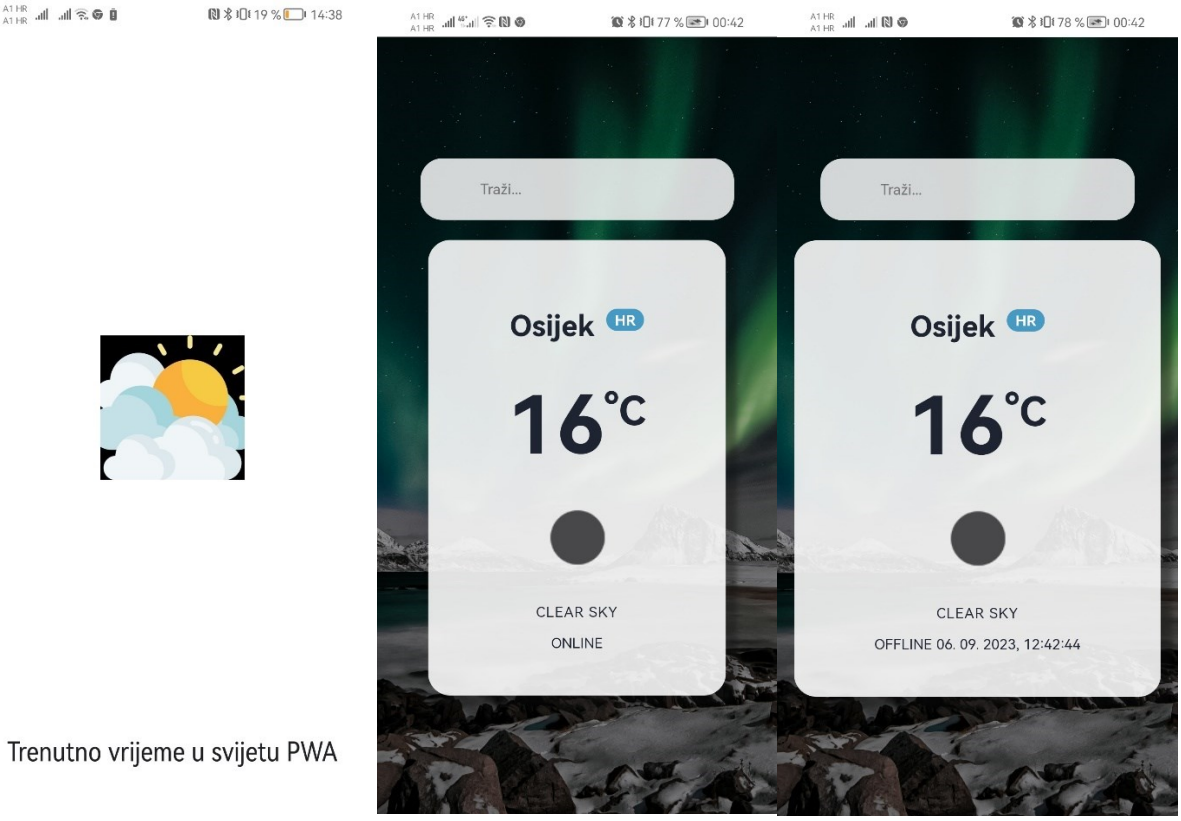
u VSC terminalu, kojom se generira „build“ direktorij koji predstavlja potpunu aplikaciju. Novonastali direktorij može se postaviti na platformu Netlify, koja automatski objavljuje aplikaciju na Internet i generira URL kojim se omogućuje dijeljenje stranice s drugima.

Otvaranjem aplikacije putem Netlify poveznice uspješno se izvršilo učitavanje Trenutno vrijeme aplikacije, te se pretraživanje i dohvaćanje rezultata s OpenWeather besplatnog API-ja izvodilo brzo i pregledno. Na svakom posjetu aplikaciji, izbacuje se skočni prozor koji potiče korisnika na instaliranje aplikacije, sve dok korisnik to i ne učini. Nakon instaliranja aplikacije ne vidi se razlika između PWA i ostalih aplikacija na mobilnom uređaju. Ikona ne odaje da se radi o

aplikaciji koja se zapravo pokreće s preglednika. Na slici 20. prikazan je hodogram instaliranja aplikacije na mobilni uređaj, a na slici 21. prikaz otvaranja i korištenja aplikacije.



Slika 20. Prikaz učitavanja aplikacije u pregledniku i njezino preuzimanje na početni zaslon



Slika 21. Otvaranje aplikacije i dobivanje rezultata za grad Osijek (*online* i *offline*)

Trenutno vrijeme aplikacija sada je dostupna svima putem specifične poveznice, omogućene putem Netlify platforme.[79] Nažalost, aplikacija trenutno nije vidljiva u tražilicama, kao ni u trgovinama aplikacija, jer za potrebe ovog rada to nije bilo potrebno učiniti. Aplikacija je vidljiva u pregledniku, sadržaj se prilagodljivo prikazuje na različitim uređajima, te se dobiva skočni prozor za preuzimanje aplikacije. Instalirana aplikacija na početnom zaslonu mobilnog uređaja izgleda kao i svaka druga aplikacija, ima svoju ikonu i učitava se u nekoliko sekundi.

6. Izazovi i ograničenja u procesu izrade PWA aplikacije

PWA aplikacije na prvi pogled izgledaju kao savršeno rješenje svim programerima, zbog svoje jednostavnosti, prilagodljivosti i brzine razvoja. Čak se i implementiranje PWA značajki u već postojeće aplikacije čini jednostavno. Postoje razni tečajevi i YouTube *tutorial*-i o tome kako

[79] Poveznica aplikacije: https://sweet-belekoy-596315.netlify.app

se razvijaju PWA aplikacije.[80] U čemu je onda problem? Sva dokumentacija, *tutorial*-i i tečajevi o PWA često su zastarjeli, odnosno objavljeni su prije četiri ili pet godine, što znači da je i PWA tehnologija doživjela zastoj. Problem leži u činjenici da PWA tehnologija nije toliko tražena koliko se možda činilo, a većina običnih korisnika nije ni svjesna da postoje ovakve vrste aplikacija, kamoli da ih koriste. Pronalaženje ažuriranih i dovoljno detaljnih materijala i dokumentacije za razvoj PWA aplikacija često predstavlja izazov. Unatoč tome, uspješno su implementirane glavne PWA karakteristike: *offline* način rada, preuzimanje, prilagodljivost i mogućnost dijeljenja URL poveznica aplikacije. Ipak, valja napomenuti da PWA često neće imati pristup svim naprednim mogućnostima uređaja koje su dostupne npr. *nativnim* aplikacijama. To ograničava mogućnosti nekih aplikacija, osobito onih koje ovise o pristupu kameri, senzorima i drugim komponentama uređaja. S obzirom na postojeće aplikacije razvijene tijekom godina putem različitih arhitektura (*nativne*, *web* i hibridne), postoji skeptičnost prema potrebi, uspjehu i prihvatljivosti PWA.[81]

Iako je jedna od ključnih prednosti PWA sposobnost rada u *offline* načinu, implementacija pravilne *offline* funkcionalnosti može biti izazovna. Osiguravanje da aplikacija ispravno radi i pruža korisničko iskustvo čak i bez internetske veze zahtijeva napredno rukovanje podacima i sinkronizacijom. Uz to, testiranje *offline* funkcionalnosti ponekad predstavlja izazov. Simuliranje različitih scenarija s prekidima veze, sinkronizacijom i promjenama podataka može biti naporno, ali je nužno za osiguravanje pouzdane izvedbe aplikacije u različitim uvjetima. Chrome DevTools omogućuju ispitivanje *offline* funkcionalnosti aplikacije, no ispravnost te funkcionalnosti nije uvijek valjana, te se aplikacija ponaša kao i u *online* stanju prikaza.

Ono što PWA aplikacije čini posebnima je njihova mogućnost preuzimanja na uređaj, bez potrebe posjećivanja trgovine aplikacija. Ta značajka može biti izuzetno korisna za neke korisnike, zahvaljujući jednostavnosti preuzimanja i manjem zauzimanju prostora na uređaju, no drugima može predstavljati problem Skočni prozor koji se pojavljuje svakim posjetom stranice, koji potiče korisnika na preuzimanje aplikacije, može smetati i u konačnici negativno utjecati na korisničko iskustvo. Također, preuzimanje aplikacije iz preglednika u odnosu na klasično preuzimanje putem trgovine aplikacija nije pouzdana, te uvijek postoji sumnja o sigurnosti aplikacije.

[80] Za potrebe aplikacije Trenutno vrijeme poslužili su YouTube tutorial-i: The Net Ninja. PWA Tutorial for Begginers, 2019. URL: https://www.youtube.com/playlist?list=PL4cUxeGkcC9gTxqJBcDmoi5Q2pzDusSL7 , te JavaScript Mastery. Build and Deploy a React PWA – Why Progressive Web Apps are the Future of the Web, 2020. URL: https://www.youtube.com/watch?v=IaJqMcOMuDM

[81] Adetunji, O.; Ajaegbu, C.; Otuneme, N.; Omotosho, O. Dawning of Progressive Web Applications (PWA): Edging Out the Pitfalls of Traditional Mobile Development. // American Scientific Research Journal for Engineering, Technology, and Sciences (ASRJETS) 68, 1(2020). str. 85-99.

Lighthouse je iznimno koristan Chrome-ov alat, koji je sposoban analizirati aplikaciju i donijeti detaljne rezultate i preporuke kako poboljšati aplikaciju. Redovitim ponavljanjem testova uočeno je da se testiranje ponekad ne može izvršiti na vrijeme ili rezultati variraju, čak i kada nisu napravljene izmjene na aplikaciji. Razlog za ovu pojavu može ležati u različitim faktorima, uključujući promjene u okruženju testiranja ili ažuriranja Lighthouse alata. Stoga je važno uzeti u obzir da rezultati Lighthouse testiranja nisu uvijek apsolutno konzistentni i ispravni te da bi se trebali promatrati kao smjernice za unaprjeđenje, a ne kao konačni rezultati testa.

7. Zaključak

PWA aplikacije označavaju uvod u novo poglavlje u današnjem digitalnom okruženju, otvarajući vrata za inovativne načine aplikacijskog razvoja. Danas, stvaranje kvalitetne aplikacije više nije dovoljno. Ključno je istaknuti se i pružiti napredne značajke koje će osvojiti pažnju korisnika i zadovoljiti njihove sve zahtjevnije potrebe. PWA aplikacije dokazale su da su sposobne konkurirati s trenutno najpopularnijom vrstom aplikacija - *nativnim* aplikacijama. Naravno, još uvijek nisu dostigle njihovu razinu popularnosti, možda nikada ni neće, no svejedno su ostvarile odlične rezultate u kratkom roku. PWA aplikacije nude korisnicima iskustvo slično *nativnim* aplikacijama, uz istodobnu lakoću pristupa i preuzimanja, čime postavljaju novi standard za pristup razvoju mobilnih i mrežnih aplikacija. *Nativne* aplikacije najbolji su izbor za izradu aplikacije, no organizacije koje nemaju dovoljno financijskih, ljudskih i vremenskih resursa, mogu bolje proći odabirom PWA. Bitno je za zapamtiti da je PWA više koncept nego aplikacija. Implementiranjem PWA značajki mnoge aplikacije mogu podignuti svoje performanse na sljedeću razinu. Uspoređujući PWA s *nativnim*, hibridnim i SPA aplikacijama utvrdilo se da su jedne od najjeftinijih i najboljih aplikacija po pitanju brzine učitavanja sadržaja, implementaciji *offline* mogućnosti i sigurnosti zbog HTTPS protokola.

Prednosti PWA obuhvaćaju prilagodljivost različitim uređajima i operativnim sustavima, brži i jeftiniji razvoj te visoku razinu sigurnosti. Implementacija *service worker*-a ključna je za ostvarivanje tih prednosti, omogućujući brzu interakciju, dohvaćanje podataka iz *cache*-a, slanje obavijesti i automatsko ažuriranje aplikacije. Unatoč brojnim prednostima, PWA aplikacije suočavaju se s nekim izazovima. Nisu jednako podržane na svim uređajima i preglednicima, posebno u iOS okruženju. PWA također mogu trošiti više baterije mobilnih uređaja te se ne mogu pronaći u tradicionalnim trgovinama aplikacija, što utječe na njihovu vidljivost i dostupnost. Odluka o izradi PWA aplikacije zahtijeva temeljito razmatranje prednosti i nedostataka, potreba korisnika, razvojne izazove i ciljeve organizacija. Iako donose brojne prednosti, važno je prepoznati i ograničenja te procijeniti hoće li PWA odgovarati konkretnom projektu i okruženju.

Napravljena aplikacija Trenutno vrijeme mogla je biti klasična mrežna stranica, no implementiranjem PWA značajki dobila je novu dimenziju, te je postala prava aplikacija. Vrijeme koje je bilo potrebno za njezino stvaranje nije dugo trajalo, oko tjedan dana. Najteže je bilo stvoriti *service worker*-a i shvatiti na koji način funkcionira. No, jednom kada se to uspješno usvojio sve ostalo je išlo glatko. Manifest je bio vrlo jednostavan za shvatiti, bilo je potrebno samo pripremiti

ikone u odgovarajućim dimenzijama i zapisati osnovne podatke o aplikaciji. Također, postavljanje aplikacije na Netlify platformu nije bilo komplicirano, te je u kratkom roku bila vidljiva i postavljena na Internet.

Za vrijeme procesa stvaranja aplikacije, uočeno je da malo ljudi zna za PWA tehnologiju, iako je prisutna već neko godina. Istraživanjem u području PWA spominje se da bi u budućnosti mogle zamijeniti *nativne* aplikacije, no mogućnosti PWA su još uvijek siromašnije u odnosu na *nativne.* Primjeri aplikacija koje su uspješno implementirale PWA verziju svoje aplikacije doživjele su ogroman uspjeh, no te aplikacije su već popularne i već su ostvarile svoje mjesto na ljestvici u trgovinama aplikacija. Organizacije koje su tek u početnim fazama stvaranja aplikacije ne mogu doživjeti takvu vidljivost i angažman korisnika, kao aplikacije koje se mogu instalirati putem trgovine. Poznata činjenica je da Apple i Google kontroliraju tržište implementiranjem postupka rangiranja, time promovirajući vlastite aplikacije i aplikacije koje su najpopularnije. Svaka aplikacija mora platiti naknadu da bude prisutna u trgovini, što je mala cijena za mogućnost pristupa, preuzimanja i dostupnosti aplikacije, koje su od velike važnosti za organizacije. Korisnici će posjetiti trgovinu aplikacija, instalirati one aplikacije koje su najpopularnije i koje imaju pozitivne recenzije. Rijetko tko će htjeti instalirati aplikaciju putem preglednika, zbog navike klasičnog instaliranja aplikacija, ali i zbog manjka pouzdanosti preuzimanja aplikacije s preglednika.

Teško je predvidjeti kako će PWA aplikacije opstati u budućnosti i na koji način će se razvijati. Kako bi PWA aplikacije napredovale moraju poboljšati svoje značajke kao i jednaku kvalitetu korisničkog iskustva neovisno o verziji preglednika koju korisnik koristi. Uz izazove koje nosi digitalno okruženje, PWA aplikacije imaju potencijal postati trajan i značajan trend. Njihova fleksibilnost, sposobnost rada u *offline* načinu i brza interakcija s korisnicima samo su neki od čimbenika koji ih čine primamljivima za organizacije i programere. I dok budućnost PWA aplikacija ostaje otvorena, jedno je jasno - kontinuirani trud na unaprjeđenju i prilagodbi omogućit će im da ostanu relevantne i korisne u promjenjivom svijetu tehnologije.

8. Literatura

KNJIGE

1. Ater, Tal. Building Progressive Web Apps, Kalifornija: O'Reilly Media, 2017.

STRUČNI ČLANCI

2. Adetunji, O.; Ajaegbu, C.; Otuneme, N.; Omotosho, O. Dawning of Progressive Web Applications (PWA): Edging Out the Pitfalls of Traditional Mobile Development. // American Scientific Research Journal for Engineering, Technology, and Sciences (ASRJETS) 68, 1(2020). str. 85-99. URL: https://asrjetsjournal.org/index.php/American_Scientific_Journal/article/view/5812 (2023-08-27).

3. Biørn-Hansen, A.; Majchrzak, T. A.; Grønli, T. Progressive Web Apps for the Unified Development of Mobile Applications, 2018. URL: https://www.researchgate.net/publication/325823248_Progressive_Web_Apps_for_the_Unified_Development_of_Mobile_Applications (2023-08-13).

4. Huber, S.; Demetz, L.; Felderer, M. PWA vs the Others: A Comparative Study on the UI Energy-Efficiency of Progressive Web Apps, 2021. URL: https://www.researchgate.net/publication/351487772_PWA_vs_the_Others_A_Comparative_Study_on_the_UI_Energy-Efficiency_of_Progressive_Web_Apps (2023-08-13).

5. Jhala, D. A Study on Progressive Web Apps As A Unifier for Native Apps and the Web. // International Journal of Engineering Research & Technology (IJERT) 10, 5(2021). Str. 207- 210. URL: https://www.ijert.org/a-study-on-progressive-web-apps-as-a-unifier-for-native-apps-and-the-web (2023-08-27).

6. Khan, A.;Al-Badi, A.; Al-Kindi, M. Progressive Web Application Assessment Using AHP. // Procedia Computer Science 155 (2019). str 289-294. URL: https://www.sciencedirect.com/science/article/pii/S187705091930955X (2023-08-27).

7. Magomadov, V. Exploring the role of progressive web applications in modern web development. // Journal of Physics: Conference Series 1679, 2(2020). URL:

https://www.researchgate.net/publication/347180526_Exploring_the_role_of_progressive_web_applications_in_modern_web_development (2023-08-13).

8. Majchrzak, T.; Biørn-Hansen, A.; Grønli, T. Progressive Web Apps: the Definite Approach to Cross-Platform Development? // 51st Hawaii International Conference on System Sciences (2018). str 5735-5744. URL: https://www.semanticscholar.org/paper/Progressive-Web-Apps%3A-the-Definite-Approach-to-Majchrzak-Bi%C3%B8rn-Hansen/672324aaa5de7c41a57d951898d2a7dd21b15f04 (2023-08-27).

9. Sharma, S.; Bhardwaj, A. Progressive Web Apps (PWA). // Journal of Emerging Technologies and Innovative Research 7(2021). str 696-706. URL: https://www.jetir.org/view?paper=JETIR2107594 (2023-08-27).

10. Sharma, V.; Verma, R.; Pathak, V.; Paliwal, M.; Jain, P. Progressive Web App (PWA) - One Stop Solution for All Application Development Across All Platforms. // International Journal of Scientific Research in Computer Science, Engineering and Information Technology (2019). str. 1120–1122. URL: https://doi.org/10.32628/CSEIT1952290 (2023-08-27).

11. Tandel, S.; Jamadar, A. Impact of Progressive Web Apps on Web App Development, 2018. URL: https://www.researchgate.net/publication/330834334_Impact_of_Progressive_Web_Apps_on_Web_App_Development (2023-08-13).

MREŽNI IZVORI

12. 12 Best Examples of Progressive Web Apps (PWAs) [Updated 2023]. URL: https://www.simicart.com/blog/progressive-web-apps-examples/ (2023-08-13).

13. Aplikacija. URL: https://www.enciklopedija.hr/natuknica.aspx?ID=3306 (2023-08-13).

14. BEGINNER'S GUIDE TO APPS: NATIVE VS HYBRID VS PWA, 2021. URL: https://makelemonapp.com/beginners-guide-to-apps-native-vs-hybrid-vs-pwa/ (2023-08-13).

15. Brown, R. Progressive Web Apps (PWAs): Definition, How They Work, and Why You Need One? URL: https://ionic.io/resources/articles/what-is-a-progressive-web-app-and-why-you-need-one (2023-08-13).

16. Cruz, A. Progressive Web Apps: An in-depth look at the past, the present, and the future, 2021. URL: https://www.truewindglobal.com/progressive-web-apps-an-in-depth-look-at-the-past-the-present-and-the-future/ (2023-08-13).

17. Darya, N. PWA VS native app: what to choose in 2023?, 2022. URL: https://solveit.dev/blog/progressive-web-app-vs-native-app (2023-08-13).

18. Dharmwan, S. 10 Examples of Hybrid Apps that have taken Businesses to the next Level, 2023. URL: https://cynoteck.com/blog-post/hybrid-apps-that-have-taken-businesses-to-the-next-level/#Advantages_of_Hybrid_Apps (2023-08-13).

19. Gillis, A. S. Native app. URL: https://www.techtarget.com/searchsoftwarequality/definition/native-application-native-app (2023-08-13).

20. How to format datetime in ReactJS, 2023. URL: https://www.altcademy.com/blog/how-to-format-datetime-in-reactjs/ (2023-09-05).

21. IQUII. Progressive Web App (PWA): what they are, pros and cons and the main examples on the market, 2019. URL: https://medium.com/iquii/progressive-web-app-pwa-what-they-are-pros-and-cons-and-the-main-examples-on-the-market-318f4538c670 (2023-08-13).

22. Korolov, S. Mobile App Development Approaches Explained, 2021. URL: https://railsware.com/blog/native-vs-hybrid-vs-cross-platform/#Progressive_web_app_-_as_a_bonus_to_mention (2023-08-13).

23. Kremer, M. What is a PWA?, 2023. URL: https://onesignal.com/blog/what-is-a-pwa/ (2023-08-13).

24. Lighthouse. URL: https://developer.chrome.com/docs/lighthouse/overview/ (2023-08-13).

25. Microsoft. Screen sizes and breakpoints, 2023. URL: https://learn.microsoft.com/en-us/windows/apps/design/layout/screen-sizes-and-breakpoints-for-responsive-design (2023-08-13).

26. Nyakundi, H. What is a PWA? Progressive Web Apps for Beginners, 2023. URL: https://www.freecodecamp.org/news/what-are-progressive-web-apps/ (2023-08-13).

27. Progressive Web Apps. URL: https://web.dev/progressive-web-apps/ (2023-08-13).

28. PWA. URL: https://web.dev/what-are-pw (2023-08-13).

29. PWA vs. Native App vs. Hybrid vs. SPA vs. MPA vs. Responsive Website, 2023. URL: https://www.gomage.com/blog/pwa-vs-native-vs-hybrid-vs-responsive-website/ (2023-08-13).

30. Sagawara, R. Serviceworker-storage, 2017. URL: https://github.com/RyotaSugawara/serviceworker-storage (2023-09-05).

31. Simplelearn. Demystifying Progressive Web Apps: What They Are, How They Work, and Why They Matter, 2023. URL: https://www.simplilearn.com/progressive-web-apps-article (2023-08-13).

32. Staničić, O. Kraljević, S. ANALIZA ZNAČAJKI I PERFORMANSI PREOGRESIVNIH WEB APLIKACIJA. // Politehnika i dizajn 7, 4(2019). str. 240-247. URL: https://hrcak.srce.hr/236123 (2023-08-13).

33. Tran, J. PWA vs Native App and how to choose between them, 2022. URL: https://www.magestore.com/blog/pwa-vs-native-app-and-how-to-choose-between-them/ (2023-08-13).

34. What is a progressive web app?. URL: https://developer.mozilla.org/en-US/docs/Web/Progressive_web_apps/Guides/What_is_a_progressive_web_app#pwas_and_single-page_apps (2023-08-13).

35. Wieckowska, S.; Bracisiewicz, M. PWA vs. MPA vs. SPA – What's the Best Choice for Your App?, 2022. URL: https://neoteric.eu/blog/pwa-vs-mpa-vs-spa-whats-the-best-choice-for-your-app/ (2023-08-13).

36. WWW. URL: https://www.enciklopedija.hr/natuknica.aspx?ID=66413 (2023-08-13).