VIŠJA STROKOVNA ŠOLA ACADEMIA

MARIBOR

PRIMERJAVA ZMOGLJIVOSTI IN VARNOSTNIH ZNAČILNOSTI MOBILNIH APLIKACIJ, RAZVITIH V FLUTTERJU IN NEXT.JS KOT PWA

Kandidat: Amar Ibrahimović

Vrsta študija: študent izrednega študija

Študijski program: Računalništvo in informatika

Mentor predavatelj: mag. Ervin Schaff

Mentor v podjetju: mag. Žiga Korošec

Lektor: Urška Prelog, prof. slov. in soc.

Maribor, 2025

IZJAVA O AVTORSTVU DIPLOMSKEGA DELA

Podpisani Amar Ibrahimović sem avtor diplomskega dela z naslovom Primerjava zmogljivosti in varnostnih značilnosti mobilnih aplikacij, razvitih v Flutterju in Next.js kot PWA, ki sem ga napisal pod mentorstvom mag. Ervina Schaffa.

S svojim podpisom zagotavljam, da:

- je predloženo delo izključno rezultat mojega dela,
- sem poskrbel/a, da so dela in mnenja drugih avtorjev, ki jih uporabljam v predloženi nalogi, navedena oz. citirana skladno s pravili Višje strokovne šole Academia Maribor,
- se zavedam, da je plagiatorstvo – predstavljanje tujih del oz. misli, kot moje lastne kaznivo po Zakonu o avtorski in sorodnih pravicah (Uradni list RS, št. 16/07 – uradno prečiščeno besedilo, 68/08, 110/13, 56/15 in 63/16 – ZKUASP); prekršek pa podleže tudi ukrepom Višje strokovne šole Academia Maribor skladno z njenimi pravili,
- skladno z 32.a členom ZASP dovoljujem Višji strokovni šoli Academia Maribor objavo diplomskega dela na spletnem portalu šole.

Maribor, september 2025 Podpis študenta:

ZAHVALA

Rad bi se zahvalil svoji družini, ki me je podpirala na celotni študijski poti. Prav tako bi se rad zahvalil mentorju mag. Ervinu Schaffu, ki me je usmerjal in mi pomagal pri sestavi diplomskega dela.

POVZETEK

Diplomsko delo primerja tehnologiji Next.js in Flutter za razvoj mobilnih aplikacij. Prav tako primerja dva pristopa razvoja mobilnih aplikacij: razvoj PWA (Next.js) in razvoj večplatformskih aplikacij (Flutter).

Na začetku so opisane mobilne aplikacije vključno z zgodovino, operacijskimi sistemi, vrstami in arhitekturo mobilnih aplikacij.

V okviru raziskave so predstavljene značilnosti obeh tehnologij, vključno z arhitekturo, programskimi jeziki ter različno podporo za *native* funkcionalnosti in dostop do strojne opreme. Posebna pozornost je namenjena tudi metodam testiranja in distribucije aplikacij.

V nalogi sta prikazani dve aplikaciji, vsaka izdelana z eno izmed tehnologij. Aplikaciji sta funkcionalno enaki, le da se razlikujeta v uporabljenem pristopu in tehnologiji. Prav tako so razloženi postopek razvoja in uporabljene tehnike ter dodatne tehnologije, ki so pomagale pri razvoju.

Opravljena je analiza skupnosti, razpoložljivosti virov in orodij za razvoj, ki kaže na močno podporo obeh tehnologij, vendar z različno stopnjo razširjenosti in dostopnosti učnih vsebin.

Meritve zmogljivosti vključujejo primerjavo porabe pomnilnika in hitrosti izvajanja HTTP-zahtevkov, pri čemer so se pokazale razlike med aplikacijo Flutter in Next.js PWA.

V diplomskem delu so prav tako predstavljene funkcionalne in tehnološke prednosti ter slabosti obeh pristopov in tehnologij, kar omogoča vpogled v njihovo primernost glede na razvojne in uporabniške potrebe.

Naloga na koncu obravnava varnost aplikacij, zlasti ranljivost za napade *Man-in-the-Middle* ter zaščito pred napadi s povratnim inženiringom (angl. *Reverse Engineering*). Opravljene so bile tudi simulacije vsakega napada na obe aplikaciji. Pri tem so se pri prvem pokazale bistvene razlike, pri drugem pa sploh ne. Prav tako je pojasnjeno, kako lahko povečamo zaščito aplikacij pred temi napadi in ali je to sploh smiselno narediti.

Ključne besede: *PWA, Next.js, Flutter, mobilne aplikacije, kibernetski napadi*

ABSTRACT

Comparison of Performance and Security Features of Mobile Applications Developed in Flutter and Next.js as PWAs.

This thesis compares the technologies Next.js and Flutter for mobile application development. It also compares two approaches to mobile app development: Progressive Web Apps (PWA), development using Next.js and cross-platform application development using Flutter.

The introduction covers mobile applications, including their history, operating systems, types, and architecture.

The study presents the characteristics of both technologies, including their architecture, programming languages, and differing support for native functionalities and hardware access. Special attention is given to testing methods and app distribution.

Two applications are showcased in the thesis, each developed with one of the technologies. The applications are functionally equivalent but differ in the chosen approach and technology. The development process, used techniques, and additional technologies that assisted in development are also explained.

An analysis of the community, availability of resources, and development tools is conducted, revealing strong support for both technologies, though with varying degrees of popularity and availability of learning materials.

Performance measurements include comparisons of memory usage and HTTP request execution speed, highlighting differences between the Flutter app and the Next.js PWA.

The thesis also presents the functional and technological advantages and disadvantages of both approaches and technologies, providing insight into their suitability based on development and user needs.

Finally, the thesis addresses app security, particularly vulnerabilities to Man-in-the-Middle attacks and protection against Reverse Engineering. Simulations of each attack were conducted on both applications, showing significant differences in one case and none in the other. It also discusses how app protection can be enhanced against these attacks and whether implementing such measures is justified.

Keywords: *PWA, Next.js, Flutter, mobile applications, cyber attacks*

KAZALO VSEBINE

KAZALO SLIK

KAZALO TABEL

KAZALO GRAFOV

Razlaga strokovnih izrazov in kratic

- PWA (angl. *Progressive Web App*) je progresivna spletna aplikacija. Spletna stran, ki lahko deluje kot mobilna aplikacija.
- iOS (angl. *iPhone Operating System*) je operacijski sistem podjetja Apple za mobilne telefone iPhone.
- MITM (angl. *Man-in-the-Middle*) je kibernetski napad, pri katerem napadalec poskuša prestreči komunikacijo med odjemalcem in prejemnikom.
- IBM (angl. *International Business Machines*) je ameriško podjetje, ki se ukvarja z razvojem in prodajo tehnologije.
- GPS (angl. *Global Positioning System*) je sistem, ki temelji na satelitih in omogoča navigacijo po celotnem planetu.
- Bluetooth je tehnologija, ki omogoča izmenjavo podatkov na manjši oddaljenosti.
- HTML (angl. *HyperText Markup Language*) je jezik, ki se uporablja za opis strukture spletnih strani.
- CSS (angl. *Cascading Style Sheets*) je jezik, ki se uporablja za oblikovanje spletnih strani.
- JavaScript je programski ali skriptni jezik, ki se uporablja za dodajanje funkcionalnosti na spletno stran.
- React je knjižnica za izdelavo uporabniških vmesnikov, ki temelji na jeziku JavaScript.
- Next.js je ogrodje za izdelavo uporabniških vmesnikov in zalednih delov spletnih aplikacij. Temelji na knjižnici React.
- Xamarin je orodje, ki ga je razvilo podjetje Microsoft in je ustvarjeno za gradnjo večplatformskih aplikacij.
- HTTP (angl. *HyperText Transfer Protocol*) je osnovni protokol za prenos podatkov po spletu.
- HTTPS (angl. *HyperText Transfer Protocol Secure*) je varna različica HTTP-protokola, ki uporablja šifriranje za zaščito podatkov.
- API (angl. *Application Programming Interface*) je vmesnik med različnimi programskimi opremami.

- REST (angl. *Representational State Transfer*) je način gradnje vmesnikov API, pri čemer se za komunikacijo uporablja HTTP.
- C je nizkonivojski programski jezik, ki je vplival na razvoj drugih jezikov.
- C++ je programski jezik, ki je razširitev programskega jezika C.
- Dart je programski jezik, ki ga je razvilo podjetje Google. Uporablja se predvsem v ogrodju Flutter.
- *Rendering* je proces, pri katerem program pretvori podatke v vizualno obliko.
- AOT (angl. *Ahead-of-Time*) je način prevajanja programske kode, pri čemer se celotna koda prevede v strojno kodo pred izvajanjem programa.
- JIT (angl. *Just-In-Time*) je nasprotje AOT. Koda se prevaja med izvajanjem programa.
- *Async* je rezervirana beseda, ki označuje funkcijo kot asinhrono.
- *Await* je rezervirana beseda, ki programu sporoči, naj počaka, da se naloga zaključi.
- *Widgets* so osnovni gradniki vsake aplikacije, pripravljene v Flutterju.
- *StatelessWidget* je gradnik (*widget*), ki nima lastnega stanja in se ne spreminja.
- *StatefulWidget* je gradnik (*widget*), ki ima stanje in se lahko dinamično spreminja.
- Windows je najbolj razširjen operacijski sistem, ki ga je razvilo podjetje Microsoft.
- Linux je odprtokodni operacijski sistem.
- MacOS (angl. *Mac Operating System*) je operacijski sistem za računalnike podjetja Apple.
- DOM (angl. *Document Object Model*) je podatkovna struktura, ki predstavlja vsebino spletne strani kot drevesno strukturo.
- XML (angl. *Extensible Markup Language*) je besedilni format za shranjevanje in prenos podatkov.
- ECMAScript je standard za skriptni jezik, na katerem temelji JavaScript.
- Node.js je okolje za izvajanje kode v JavaScriptu zunaj brskalnika, na strežniku.
- SPA (angl. *Single Page Application*) je spletna aplikacija, ki se naloži kot ena sama stran.
- *Props* so vhodni parametri za komponente v knjižnici React.
- camelCase je način zapisa besed v programiranju.

- GET je HTTP-metoda za pridobivanje podatkov s strežnika.
- POST je HTTP-metoda za ustvarjanje novih podatkov na strežniku.
- PUT je HTTP-metoda za posodobitev obstoječih podatkov na strežniku.
- DELETE je HTTP-metoda za brisanje podatkov na strežniku.
- *Cron job* je opravilo, ki se izvede ob točno določenem času.
- *WebSocket* je tehnologija, ki omogoča stalno dvosmerno komunikacijo med odjemalcem in strežnikom.
- *Compiler* je program, ki prevede kodo, napisano v programskem jeziku, v strojno kodo.
- *Lazy loading* je nalaganje vsebine šele takrat, ko je res potrebna.
- WebP (angl. *Web Picture*) je sodoben format slik, uporabljen na spletnih straneh.
- URL (angl. *Uniform Resource Locator*) je spletni naslov, ki kaže na lokacijo vira.
- *Git* je sistem za upravljanje različic.
- npm (angl. *Node Package Manager*) je orodje za upravljanje paketov in knjižnic v okolju Node.js.
- IDE (angl. *Integrated Development Environment*) je program, ki združuje orodja za pisanje, urejanje in prevajanje kode.
- PIN (angl. *Personal Identification Number*) je kratka številčna koda, ki se uporablja za preverjanje identitete.
- SDK (angl. *Software Development Kit*) je paket orodij za ustvarjanje aplikacij za določeno platformo ali tehnologijo.
- JSON (angl. *JavaScript Object Notation*) je format za shranjevanje in prenos podatkov.
- JWT (angl. *JSON Web Token*) je digitalni žeton za avtentikacijo uporabnika v spletnih aplikacijah.
- MB (angl. *Megabyte*) enota za merjenje velikosti podatkov. Enak je 1,000.000 bajtov.
- ms (angl. *Millisecond*) ali milisekunda je enota za čas, ki je enaka eni tisočinki sekunde.
- CRUD (angl. *Create Read Update Delete*) je kratica za štiri osnovne operacije nad podatki.

- NoSQL (angl. *Not Only SQL*) je skupina podatkovnih baz, ki ne uporabljajo relacijske podatkovne strukture.
- ID (angl. *Identifier*) je unikatna oznaka.
- TLS (angl. *Transport Layer Security*) je varnostni protokol, ki šifrira komunikacijo med dvema napravama v omrežju.
- APK (angl. *Android Package Kit*) je datotečni format, v katerem so zapakirane vse komponente aplikacije za Android.

1 UVOD

Mobilne aplikacije obstajajo že vrsto let in eno izmed vodilnih ogrodij za razvoj je Flutter. Sčasoma so na trg prispele tudi aplikacije PWA (angl. *Progressive Web Apps*). To so spletne strani, ki se lahko uporabijo kot mobilne ali namizne aplikacije. V tem času pa se je razvijal tudi Flutter, ki zdaj omogoča tudi razvoj spletnih in namiznih aplikacij. Obe tehnologiji se torej lahko uporabita za razvoj aplikacij na vseh platformah. Za namene te raziskave se bo uporabil Next.js kot ogrodje za razvoj PWA.

Zaradi številnih omejitev, ki sta jih postavila veliki podjetji Google in Apple, vsako s svojo trgovino za mobilne aplikacije, se številni razvijalci odločijo za uporabo PWA. Tako se izognejo potrebi po distribuciji mobilnih aplikacij v trgovinah Google Play ali App Store in morebitnim stroškom, ki bi s tem nastali.

Namen te raziskave je analizirati prednosti in pomanjkljivosti obeh pristopov razvoja aplikacij. Primerjali bomo razširjenost obeh tehnologij in velikosti njunih skupnosti, enostavnost razvoja in implementacije na drugo platformo, tehnične lastnosti in omejitve, uporabniško izkušnjo ter odpornost proti kibernetskim napadom. Ključni cilj je tudi ugotoviti, kdaj in zakaj je smiselno uporabiti posamezen pristop.

1.1 Opis področja in opredelitev problema

Področje raziskovanja v diplomskem delu se osredotoča na razvoj mobilnih aplikacij, s tem tudi na razvoj spletnih in večplatformskih aplikacij ter na kibernetsko varnost. Pri razvoju mobilne aplikacije lahko izbiramo med številnimi pristopi in orodji. Težava, s katero se sreča večina razvijalcev, je v tem, da niso prepričani, v katero smer je smiselno iti. Velikokrat se odločijo za tisto smer, ki jim je blizu. Za lažjo izbiro bomo primerjali pristop klasičnega razvoja mobilne aplikacije in pristop razvoja PWA. Uporabili bomo tehnologiji Flutter in Next.js. Obe sta zelo priljubljeni na svojem področju, vsaka pa ima prednosti in pomanjkljivosti. Ugotovili bomo, katero je bolj smiselno uporabiti v določenih scenarijih, in tako olajšali izbiro smeri za razvijalce. Poleg tega bomo primerjali značilnosti obeh pristopov na praktičnem primeru dveh aplikacij – ena bo razvita z uporabo Next.js, druga pa v okolju Flutter.

1.2 Namen, cilji in osnovne trditve

Namen diplomskega dela je primerjati pristopa razvoja mobilnih aplikacij in PWA. Konkreten cilj je ugotoviti, kateri pristop je bolj smiseln glede na določen scenarij. Prav tako je cilj prikazati vse dobre in slabe lastnosti obeh pristopov in tehnologij. S tem želimo doseči večjo jasnost pri izbiri pristopa na začetku projekta.

V diplomski nalogi smo postavili hipoteze, ki jih bomo poskušali dokazati. To so:

- H1: Flutter je med razvijalci bolj razširjen kot Next.js za razvoj aplikacij.
- H2: Razvoj mobilne aplikacije s Flutterjem omogoča boljšo zmogljivost in uporabniško izkušnjo kot Next.js PWA.
- H3: Next.js PWA je bolj ranljiv za napad *Man-in-the-Middle* kot Flutter.
- H4: Aplikacija Flutter je bolj odporna proti napadom s povratnim inženiringom (angl. *Reverse Engineering*) kot Next.js PWA.

1.3 Predpostavke in omejitve

Omejitve, ki lahko nastanejo v diplomskem delu, so večinoma povezane z razvojem obeh aplikacij in testiranjem. Menimo, da bo težko povsem enakovredno reproducirati videz na obeh aplikacijah, kar pa ni tako pomembno, saj bo funkcionalnost enaka. Kar zadeva testiranje, menimo, da lahko nastanejo težave predvsem pri enem izmed kibernetskih napadov, zlasti pri napadu s povratnim inženiringom. Predpostavljamo, da se bo pred takim napadom težko zaščititi.

1.4 Uporabljene raziskovalne metode

Za izdelavo diplomskega dela je bil uporabljen primerjalni raziskovalni pristop, saj želimo primerjati dve sodobni tehnologiji za razvoj mobilnih in večplatformskih aplikacij. Najprej smo analizirali vire in literaturo ter tako preučili razširjenost in značilnosti obeh tehnologij. Nato smo razvili eksperimentalna prototipa v obeh tehnologijah, kar je omogočilo izvedbo primerjave. Sledili sta testiranje in eksperimentiranje na obeh prototipih, pri čemer smo merili zmogljivost in simulirali varnostne scenarije. Zbrane podatke iz primerjave smo analizirali in uporabili za preverjanje hipotez.

2 MOBILNE APLIKACIJE

2.1 Definicija

Mobilna aplikacija je programska oprema, ki je razvita za uporabo na manjših napravah, kot so pametni mobilni telefon, tablični računalnik in pametna ura. Glede na to, da si pri uporabi teh naprav večinoma ne pomagamo z miško ali tipkovnico, morajo mobilne aplikacije podpirati upravljanje na dotik. Prav tako morajo biti optimizirane za manjše zaslone in strojno opremo, ki je manj zmogljiva v primerjavi z osebnimi računalniki. (Terrell Hanna & Wigmore, 2025)

2.2 Zgodovina

Čeprav so mobilne aplikacije postale neizmerno priljubljene v prejšnjem desetletju, še ne pomeni, da je bil to njihov začetek. Začetki segajo namreč v prejšnje stoletje.

Leta 1984 je bil izumljen prvi praktični žepni računalnik Psion Organiser, ki je imel aplikacije, kot sta ura in kalkulator. (OutSystems, 2025)

Deset let pozneje se je pojavil prvi pametni telefon, kot ga številni danes imenujejo. Digitalni asistent Simon, ki ga je razvilo podjetje IBM (angl. *International Business Machines*), je lahko pošiljal telefaks in elektronsko pošto. Prav tako je imel imenik, koledar in urnik v obliki mobilnih aplikacij. (OutSystems, 2025)

Naslednji večji premik v tej industriji se je zgodil leta 2002. To je bil pametni telefon podjetja Blackberry, ki je poslovnežem olajšal opravljanje klicev s slušalkami in pošiljanje elektronske pošte. (Mobile Phone Museum, 2025)

Leta 2007 je podjetje Apple dalo na trg svoj iPhone, ki so ga na začetku promovirali kot iPod (predvajalnik glasbe), s funkcijami navadnega telefona in dostopom do internetnega omrežja. (Apple, 2025)

Ena izmed glavnih prodajnih prednosti prvotnega iPhona je bila, da je integriral iPod z mobilnim telefonom, zato ni bilo treba nositi dveh ločenih naprav. (Newport, 2019)

Eno leto za tem je podjetje Apple objavilo svojo trgovino za mobilne aplikacije, imenovano App Store. Po objavi operacijskega sistema Android in trgovine Google Play je priljubljenost mobilnih telefonov in aplikacij začela eksponentno rasti. (Volle, 2025)

2.3 Operacijski sistemi na mobilnih napravah

Operacijski sistem je sistemska programska oprema, ki upravlja vse vire računalniške naprave. Je vmesnik med programsko in strojno opremo naprave. (GeeksforGeeks, 2025)

Od leta 2007 do danes v svetu pametnih telefonov prevladujeta operacijska sistema Android in iOS. (OutSystems, 2025)

2.3.1 iOS

Za operacijskim sistemom stoji podjetje Apple, ki ga uporablja na svojih mobilnih napravah iPhone. iOS je znan po tem, da je posebej prilagojen za naprave iPhone in precej zaprt sistem. Posledici tega sta zelo dobra zmogljivost naprave in povečana varnost. Za nameščanje mobilnih aplikacij se na takem sistemu uporablja App Store. (Volle, 2025)

2.3.2 Android

Android je odprtokodni operacijski sistem podjetja Google, ki se uporablja na različnih mobilnih napravah. Tako ni omejen le na eno vrsto naprave, kot je to značilno za iOS. Zaradi svoje odprtokodne narave je precej manj omejen in veliko bolj prilagodljiv. V primerjavi s sistemom podjetja Apple je torej lahko manj varen. Na napravah z Androidom se za nameščanje mobilnih aplikacij uporablja trgovina Google Play. (BasuMallick, 2025)

2.4 Vrste mobilnih aplikacij

Poznamo več vrst mobilnih aplikacij. Najpogosteje jih delimo glede na način razvoja oziroma način delovanja na napravah. Takšna delitev vpliva na zmogljivost, uporabniško izkušnjo, razvojne stroške in vzdrževanje aplikacij.

2.4.1 *Native* aplikacije

Native aplikacije so razvite posebej za določeno platformo. Mobilne aplikacije so razvite posebej za operacijski sistem Android ali iOS. Pri razvoju se uporabljajo programski jeziki in

orodja, ki so značilna za posamezno platformo. Za Android se najpogosteje uporablja Java ali Kotlin, za iOS pa Swift ali Objective-C. (King, 2025)

Glavna prednost takih aplikacij je najvišja mogoča zmogljivost in popoln dostop do vseh funkcionalnosti naprave, kot so senzorji, GPS, kamera, bluetooth idr.

Pomanjkljivost pa je lahko višji strošek razvoja, saj je treba eno aplikacijo razviti za vsako platformo posebej.

2.4.2 Večplatformske aplikacije

Večplatformske aplikacije omogočajo razvoj ene skupne kodne baze, ki jo je mogoče izvajati na različnih platformah, kot sta Android in iOS. Tak pristop bistveno zmanjša čas in posledično stroške razvoja. Zmogljivost večplatformske aplikacije je pogosto primerljiva z zmogljivostjo *native* aplikacije. (JetBrains, 2025)

Med najbolj priljubljena orodja za ta pristop spadajo Flutter, React Native in Xamarin. (King, 2025)

2.4.3 Progresivne spletne aplikacije

PWA so aplikacije, ki se izvajajo znotraj brskalnika in niso nameščene na napravi. V bistvu so to torej spletne strani, ki se lahko dodajo na začetni zaslon. Razvite so z uporabo standardnih spletnih tehnologij, kot so HTML, CSS, JavaScript. Omogočajo delovanje brez povezave, potisna obvestila in druge značilnosti *native* aplikacije.

Tehnologija *service worker* omogoča delovanje brez povezave.

Prednost takih aplikacij je v tem, da jih ni treba nameščati iz trgovine za aplikacije. Za razvijalca to pomeni bistveno olajšavo, saj se izogne stroškom in dodatnemu delu. Prav tako je takšno aplikacijo lažje vzdrževati in posodabljati v primerjavi z *native* aplikacijo.

Čeprav so se progresivne spletne aplikacije od svojega nastanka bistveno izboljšale, še vedno nimajo popolnega dostopa do vseh funkcionalnosti naprave. (King, 2025)

2.4.4 Hibridne aplikacije

Hibridne aplikacije so razvite z uporabo spletnih tehnologij, kot so HTML, CSS in JavaScript, nato pa se zaženejo v t. i. *native* »kontejnerju«. V bistvu so to spletne aplikacije, ki se izvajajo v *native* okolju. To jim omogoča dostop do večine funkcij naprave. Lahko rečemo, da združujejo značilnosti PWA in *native* aplikacij. Med znana orodja za razvoj hibridnih aplikacij spadata Ionic in Apache Cordova.

Prednosti so hitrejši razvoj in ena skupna kodna baza za več platform.

Med pomanjkljivostmi pa sta slabša zmogljivost v primerjavi z *native* aplikacijami in omejen dostop do nekaterih funkcij naprave. (King, 2025)

2.5 Arhitektura mobilnih aplikacij

Arhitektura mobilne aplikacije opisuje, kako so sestavljeni posamezni deli aplikacije in kako med sabo komunicirajo. Dobro načrtovana arhitektura je ključna za boljšo zmogljivost, večjo varnost in tudi lažje vzdrževanje.

Arhitektura mobilne aplikacije je sestavljena iz več delov.

2.5.1 Uporabniški vmesnik (predstavitvena plast)

Uporabniški vmesnik oz. predstavitvena plast je del aplikacije, ki ga uporabnik vidi in s katerim komunicira. Vključuje elemente, kot so gumbi, navigacija, zasloni in obrazci. Pomembno je, da je prilagojen manjšim zaslonom in da omogoča dovolj dobro izkušnjo z uporabo na dotik. Njegova osnovna funkcija je sprejemanje vnosov uporabnika in prikazovanje podatkov iz nižjih plasti aplikacije. (Dhaduk, 2025)

2.5.2 Plast logike poslovanja

Logika poslovanja vsebuje osnovno logiko aplikacije, ki opravlja naloge, kot so obdelava podatkov, računanje, validacija, obvestila ipd. Pomembno je, da je ta plast v arhitekturi ločena od uporabniškega vmesnika, saj to omogoča ponovno uporabo. (Dhaduk, 2025)

2.5.3 Podatkovna plast

Podatkovna plast je odgovorna za povezovanje z zalednimi sistemi in podatkovnimi bazami. Povezovanje se pogosto izvaja z uporabo API-jev (angl. *Application Programming Interface*), ki omogočajo komunikacijo z zunanjimi strežniki, najpogosteje v obliki REST (angl. *Representational State Transfer*). Podatkovna plast omogoča tudi shranjevanje podatkov v lokalne baze podatkov.

Ta plast se običajno uporablja za avtentikacijo uporabnikov in sinhronizacijo podatkov. (Dhaduk, 2025)

3 FLUTTER

3.1 Zgodovina ogrodja

Prva različica ogrodja Flutter je bil t. i. projekt Sky, ki ga je podjetje Google predstavilo leta 2015. Cilj projekta je bil ustvariti visokozmogljive aplikacije z uporabo programskega jezika Dart.

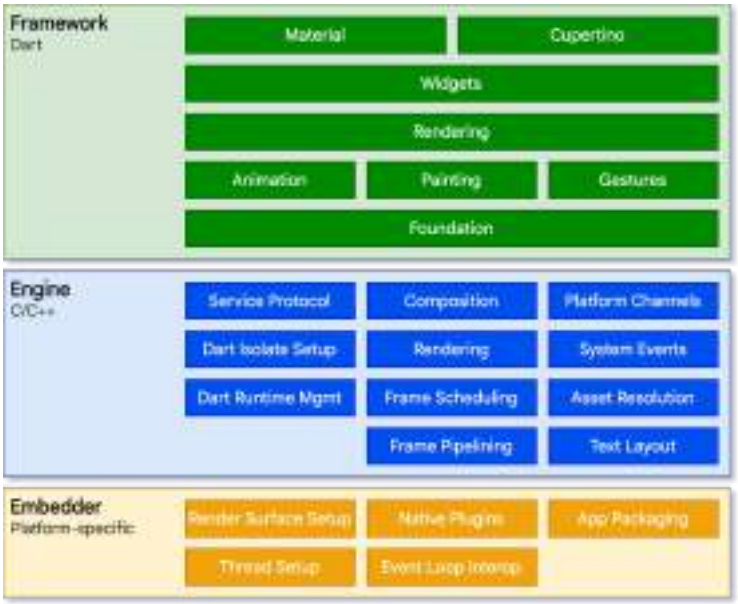
Leta 2017 je podjetje Google preimenovalo Sky v Flutter in objavilo njegovo prvo testno različico.

Leta 2018 je bila objavljena prva stabilna različica ogrodja. Flutter je zaživel med razvijalci zaradi naprednih funkcionalnosti in zaradi tega, ker je bila ena kodna baza dovolj za platformi iOS in Android.

Od leta 2019 je podjetje Google nenehno posodabljalo Flutter in omogočilo razvoj spletnih ter namiznih aplikacij. (Alpondith, 2025)

3.2 Arhitektura

Arhitektura ogrodja Flutter je razdeljena na tri dele oz. plasti: Framework, Engine, Embedder. Vsak del je sestavljen iz knjižnic, ki so opcijske. (Pérez, 2025)



Slika 1: Arhitektura ogrodja Flutter

(Vir: https://somniosoftware.com/blog/flutters-architecture-explained)

3.2.1 Framework

Framework je plast, s katero razvijalci največ komunicirajo, saj jim omogoča, da gradijo vizualno bogate in odzivne vmesnike. V tej plasti so med drugim vključeni tudi osnovni gradniki uporabniškega vmesnika, sistem za prikazovanje in animacije.

Ta plast je napisana v programskem jeziku Dart. (Flutter, 2025)

3.2.2 Engine

Engine je srednji del arhitekture in je napisan v programskem jeziku C++ ter skrbi za nizkonivojske funkcije. Medtem ko plast Framework gradi uporabniški vmesnik, Engine skrbi za njegov videz, kar je ena izmed njegovih osnovnih nalog. Z grafično knjižnico, imenovano Skia, omogoča prikaz grafik, animacij ipd. Zagotavlja tudi nizkonivojsko implementacijo osnovnega Flutterjevega API-ja. (Flutter, 2025)

3.2.3 Embedder

Embedder je del arhitekture, ki deluje kot vmesnik med aplikacijo, razvito v Flutterju, in operacijskim sistemom. Narejen je za vsak operacijski sistem posebej in tako omogoča dostop do različnih storitev, ki bi jih aplikacija lahko potrebovala. Embedder omogoča, da se aplikacija, razvita v Flutterju, lahko izvaja na različnih napravah. (Flutter, 2025)

3.3 Jezik Dart

Podjetje Google je razvilo programski jezik Dart kot alternativo za JavaScript na področju spletnega programiranja. Pozneje se je uveljavil kot ključni del ogrodja Flutter za razvoj aplikacij, saj celoten uporabniški vmesnik, logika aplikacije in interakcije s sistemom potekajo v tem jeziku. (Nukumizu, 2025)

3.3.1 Tipizacija

Dart omogoča strogo preverjanje tipov med prevajanjem programa, kar imenujemo statična tipizacija. To razvijalcu omogoča preverjanje napak pred izvedbo programa. Pri dinamični

tipizaciji tip spremenljivke ni fiksno določen, temveč se avtomatsko določi med izvajanjem programa. Takrat lahko nastanejo težave, če razvijalec ni dovolj pozoren. V svetu programiranja je načeloma statična tipizacija bolj priljubljena, zlasti pri razvoju večjih projektov.

Dart, kot tudi vsi drugi programski jeziki, ima osnovne podatkovne tipe:

- Number (številka): *int* za celo število in *double* za decimalna števila,
- String: uporablja se za shranjevanje nizov znakov,
- Bool: uporablja se za shranjevanje logične vrednosti *true* ali *false*.

Poleg osnovnih podatkovnih tipov ima tudi strukturirane podatkovne tipe, kot so *records* (objekti), *lists* (polja), *functions* (funkcije) ipd. (Nukumizu, 2025)

3.3.2 Kontrolne strukture in funkcije

Dart omogoča izvajanje pogojnih izrazov, zank in pisanje funkcij tako kot vsak programski jezik. Sintaksa je zelo podobna drugim programskim jezikom.

Spodaj je primer kode za pogojni stavek.

```
if (age >= 18) {
    print('Adult');
} else {
    print('Child');
}
```

(Dart, 2025)

Spodaj je primer kode za eno izmed zank.

```
for (int i = 1; i <= 5; i++) {
    print('Number: $i');
}
```

(Dart, 2025)

Spodaj je primer kode funkcije.

```
int square(int number) {
  return number * number;
}
```

(Dart, 2025)

3.3.3 Objektno usmerjeno programiranje

Dart je objektno usmerjen programski jezik. To pomeni, da je vse v Dartu objekt, tudi podatkovni tipi. Temelji na razredih (angl. *classes*) in objektih ter omogoča osnovne koncepte objektno usmerjenega programiranja, kot so dedovanje, polimorfizem, enkapsulacija in abstrakcija. (Dart, 2025)

Spodaj je primer uporabe objektno usmerjenega programiranja v jeziku Dart. V primeru se uporabi razred Person (oseba) s konstruktorjem in metodo, nato se v programu ustvari objekt iz razreda in požene funkcija za pozdrav (angl. *greet*).

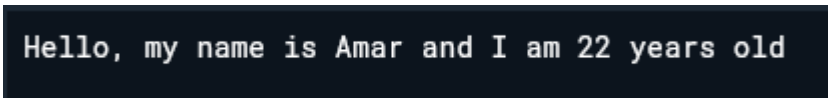
```
class Person {
  String name;
  int age;

  Person(this.name, this.age);

  void greet() {
    print('Hello, my name is $name and I am $age years old');
  }
}

void main() {
  var personObj = Person('Amar', 22);
  personObj.greet();
}
```

Funkcija za pozdrav v konzolo izpiše stavek z imenom in starostjo, ki sta bila nastavljena pri ustvarjanju objekta. Naslednja slika prikazuje izpis v konzoli.



Slika 2: Dart funkcija izpis v konzoli

(Vir: Lasten vir)

3.3.4 Asinhrono programiranje

Koda, ki se izvaja v Dartu, je sinhrona, kar pomeni, da se vsaka njena vrstica izvede šele, ko prejšnja vrstica konča izvedbo. Če pa želimo, da določene operacije potekajo v ozadju, program pa nadaljuje delovanje, lahko uporabimo ključni besedi *async* in *await*. (Dart, 2025)

To imenujemo asinhrono programiranje. Ta princip uporabljamo, kadar vemo, da bo določena operacija potekala dlje časa. Primer uporabe je pošiljanje zahtevka na zaledni sistem in čakanje na odgovor. (Bevans, 2025)

V Dartu je rezultat operacije, ki je asinhrona, objekt razreda Future. (Dart, 2025)

Spodaj je primer funkcije, ki uporabi rezervirani besedi *async* in *await* za izvedbo asinhrone operacije ter vrne objekt razreda Future.

```
Future<String> fetchData() async {
  await Future.delayed(Duration(seconds: 2));
  return 'Data has been fetched successfully';
}
```

3.3.5 Paketi in knjižnice

Dart omogoča dodajanje paketov in knjižnic iz repozitorija, imenovanega pub.dev. To omogoča razširitev funkcionalnosti, npr. dodajanje HTTP-paketa za ustvarjanje HTTP-zahtevkov. (pub.dev, 2025)

3.3.6 Način izvajanja kode v Dartu

Dart lahko izvaja kodo na dva glavna načina.

AOT (angl. *Ahead-of-Time*) je način, pri katerem se koda v Dartu prevede v nativno strojno kodo pred zagonom aplikacije. To omogoča hiter zagon in visoko zmogljivost. Tak način se uporablja v produkciji.

JIT (angl. *Just-in-Time*) je način, pri katerem se koda v Dartu prevede v nativno strojno kodo ob zagonu aplikacije. Ta način se uporablja med razvojem aplikacije, saj omogoča hitrejše testiranje, ker ni treba za vsako spremembo ponovno zagnati aplikacije. (Obinna, 2025)

3.4 Uporabniški vmesnik

Da bi zgradili uporabniški vmesnik v okolju Flutter, moramo uporabiti osnovne gradnike aplikacije (*widgets*). To je lahko vizualni element (npr. gumb), struktura (npr. vrstica ali stolpec) ali celotna logična komponenta (npr. navigacija).

Poznamo dve vrsti gradnikov. (Flutter, 2025)

3.4.1 StatelessWidget

StatelessWidget je nespremenljiv gradnik, ki ne ohranja stanja. To pomeni, da se ne bo spreminjal med uporabo aplikacije in da uporabnik nima interakcije z njim. To so lahko gradnik za besedilo, razmik med elementi idr. (Flutter, 2025)

3.4.2 StatefulWidget

StatefulWidget je gradnik, ki ima stanje in se lahko spreminja. To pomeni, da se lahko zgodi interakcija med gradnikom in uporabnikom. Takrat se mu bo spremenilo stanje, npr. števec se poveča vsakič, ko uporabnik pritisne na gumb. (Flutter, 2025)

3.4.3 Cupertino Widgets

Cupertino Widgets se uporabljajo pri razvoju aplikacij za iOS. Sledijo smernicam podjetja Apple in zagotavljajo izkušnjo, ki je bolj skladna z videzom in delovanjem iOS. (Flutter, 2025)

3.4.4 Material Widgets

Material Widgets se uporabljajo pri razvoju aplikacij za Android. Temeljijo na sistemu Material Design, ki ga je razvilo podjetje Google. Trenutno se za Flutter uporablja Material 3 kot privzeti oblikovalski jezik, ki postavlja pravila za animacije, barve, oblike ipd. (Flutter, 2025)

3.5 Večplatformski razvoj

Ena izmed ključnih prednosti ogrodja Flutter je podpora za razvoj večplatformskih aplikacij z eno samo kodno bazo. Flutter je začel kot ogrodje za mobilne aplikacije, zdaj pa omogoča razvoj za Android, iOS, splet in namizne aplikacije (Windows, Linux in macOS).

S tem pristopom se zmanjšajo čas in stroški razvoja, saj ni treba večkrat razvijati enake aplikacije za več platform.

Pomanjkljivost je v tem, da ima vsaka platforma svoje posebnosti in je to skoraj nemogoče pokriti z eno kodno bazo. (BrowserStack, 2025)

4 NEXT.JS

Next.js je ogrodje za gradnjo *front-end* in zalednega dela spletnih aplikacij. Razvilo ga je podjetje Vercel, ki ima svojo platformo za postavitev spletnih aplikacij na strežnik in gostovanje. Next.js je ogrodje, ki v osnovi uporablja knjižnico React, vendar doda veliko dodatnih funkcionalnosti in optimizacij. (Next.js, 2025)

4.1 React

React ali ReactJS je ena izmed najbolj priljubljenih knjižnic za gradnjo *front-end* dela spletnih aplikacij. Razvilo ga je podjetje Facebook, zdaj imenovano Meta. React temelji na programskem jeziku JavaScript, ki je standard za razvijalce, ki se ukvarjajo z razvojem spletnih strani. Omogoča gradnjo spletnih aplikacij, ki so razširljive, vzdržljive in imajo visoko zmogljivost. (History of ReactJS, 2025)

Za pisanje HTML- ali XML-značk uporablja posebno sintakso, imenovano JSX (angl. *JavaScript XML*), ki ustvari elemente React. Posebnost JSX je v tem, da se lahko uporabi JavaScript in se tako pripravi dinamična postavitev spletne strani. (GeeksforGeeks, 2025)

4.1.1 JavaScript

JavaScript je standardni skriptni oz. programski jezik za dodajanje funkcionalnosti na spletnih straneh. Leta 1995 ga je ustvaril Brendan Eich z namenom, da bi ga uporabil kot skriptni jezik za brskalnik Netscape Navigator.

JavaScript temelji na standardu ECMAScript, ki je specifikacija za skriptne jezike. (LaunchSchool, 2025)

JavaScript je visokonivojski dinamično tipiziran programski jezik, kar pomeni, da se tipi spremenljivk določijo med izvajanjem programa. (MDN, 2025)

Je temelj za številne knjižnice in ogrodja, kot sta React in Next.js. Uporabi pa se lahko tudi drugje, npr. za gradnjo zalednih sistemov z Node.js, ali za ustvarjanje namiznih aplikacij z ogrodjem Electron. (NSB App Studio, 2025)

4.1.2 TypeScript

TypeScript je nadgradnja programskega jezika JavaScript, ki omogoča strogo tipiziranje. Uporablja se za vse večje projekte, saj se napake hitreje odkrijejo, ker se vidijo že v razvijalskem okolju.

Brskalnik ne razume kode TypeScript, zato se pred izvajanjem pretvori v kodo JavaScript. (TypeScript, 2025)

4.1.3 JSON

JSON (angl. *JavaScript Object Notation*) je zapis za izmenjavo podatkov med sistemi. Čeprav je po sintaksi zelo podoben JavaScriptu in ima njegovo ime, je neodvisen od programskega jezika ter se uporablja za izmenjavo podatkov med odjemalcem in strežnikom. Je novejši standard in nadomešča XML.

Uporablja pare ključ-vrednost in podpira strukture, kot so objekti in seznami.

Spodaj je primer zapisa JSON.

```
{
  "name": "John Doe",
  "age": 29,
  "isMember": true,
  "address": {
    "street": "123 Main Street",
    "city": "New York",
    "zip": "10001"
  },
  "hobbies": ["reading", "cycling", "coding"]
}
```

(JSON, 2025)

4.1.4 Virtualni DOM

Virtualni DOM (angl. *Document Object Model*) je ključni del knjižnice React, saj bistveno poveča hitrost spletne aplikacije.

V tradicionalnih spletnih aplikacijah je HTML-dokument predstavljen v obliki drevesa, kjer je vsak element vozlišče. Ko se spremeni del vsebine, brskalnik posodobi dejanski DOM. Ta postopek je lahko zelo počasen, saj mora za vsako spremembo posodobiti celoten DOM in ponovno izrisati vsebino. To je lahko pri večjih aplikacijah zelo naporen proces.

Virtualni DOM ima drugačen pristop. Najprej si ustvari lahkotno kopijo dejanskega DOM-a v pomnilniku. Ko se v aplikaciji zgodi sprememba vsebine, se najprej posodobi v virtualnem DOM-u, ki je v pomnilniku. Nato algoritem primerja staro in novo različico virtualnega DOM-a in določi minimalne spremembe, ki jih je treba izvesti v dejanskem DOM-u. Torej se ne posodobi celotno drevo, temveč le določena vozlišča. (GeeksforGeeks, 2025)

4.1.5 Enostranske aplikacije

SPA (angl. *Single-Page Applications*) je sodoben pristop za razvoj spletnih aplikacij, pri katerem se celotna aplikacija naloži ob prvem obisku spletne strani. Vse nadaljnje spremembe se izvajajo dinamično, brez ponovnega nalaganja strani.

Namesto da bi brskalnik ob vsaki spremembi strani pridobil celotno novo HTML-stran s strežnika, se potrebna koda naloži vnaprej in se spremembe izvajajo z uporabo jezika JavaScript.

Prednost je v tem, da:

- je uporabniška izkušnja boljša, saj se prehodi med stranmi zgodijo skoraj takoj;
- je pri vmesniku občutek delovanja kot pri mobilnih aplikacijah;
- je strežnik manj obremenjen, saj ni treba vsakič poslati zahtevka za novo HTML-stran.

Pomanjkljivosti so lahko daljši začetni čas nalaganja in izzivi SEO, saj se stran nalaga dinamično in jo iskalniki težje indeksirajo. (GeeksforGeeks, 2025)

4.1.6 Komponentni pristop

Komponentni pristop je sodobna metoda razvoja programske opreme, pri kateri je uporabniški vmesnik razdeljen na manjše dele, imenovane komponente. Namen tega je, da se lahko ena

komponenta uporabi večkrat, kar omogoča pisanje kode, ki je berljiva in se ne ponavlja brez dobrega razloga. Vsaka komponenta ima svojo logiko, podatke in vizualno predstavitev.

Pogosto so organizirane v hierarhijo, pri čemer nadrejene komponente (angl. *parent*) vsebujejo podrejene komponente (angl. *child*). Komponente so lahko manjše enote, npr. gumbi, vnosna polja, ali pa celotne strani in moduli. (GeeksforGeeks, 2025)

V Reactu so komponente standard in skupaj s konceptom React Hooks omogočajo upravljanje stanja in življenjskega cikla komponente. V Reactu so komponente lahko funkcijske ali razredne.

Razredne komponente so starejši pristop in so danes redkeje uporabljene. Prav tako ne podpirajo React Hooks, ki je pomemben koncept za gradnjo zmogljivih spletnih aplikacij.

Funkcijske komponente so novejši pristop in podpirajo React Hooks. Velikokrat so bolj berljive in zahtevajo manj pisanja kode.

Props so podatki, ki se lahko pošljejo iz nadrejene komponente v podrejeno. Delujejo kot argumenti funkcije in se lahko uporabijo v komponenti kot atribut. (W3Schools, 2025)

Lahko rečemo, da so komponente enak koncept kot gradniki (*widgets*) v Flutterju. Vsak element v Flutterju je *widget*, ki se lahko združuje in prilagaja po potrebi.

4.1.7 JSX

JSX je razširitev programskega jezika JavaScript in omogoča pisanje strukture uporabniškega vmesnika v obliki, podobni HTML-kodi. Integriran je v Reactu in opisuje, kako naj se elementi izrišejo na zaslonu. Ker brskalniki ne razumejo JSX-kode, jo React spremeni v kodo JavaScript z uporabo orodja, kot je Babel, ki je prevajalnik JavaScripta (angl. *compiler*).

Koda je skoraj popolnoma enaka kot pri HTML-u, le da so nekatere izjeme, npr. imena atributov, ki se v JSX pišejo z uporabo notacije camelCase.

Pri pisanju JSX-kode je pomembno, da je prisoten le en sam korenski element. Vsi drugi elementi morajo biti znotraj tega elementa.

Uporaba izrazov JavaScripta je omogočena z uporabo zavitih oklepajev. To pomeni, da lahko vstavimo dinamične vrednosti, spremenljivke in funkcije.

Datoteke React je mogoče poimenovati s končnico .jsx, namesto z .js. To omogoča samodejno dopolnjevanje JSX-kode v tej datoteki. (GeeksforGeeks, 2025)

4.1.8 React Hooks

React Hooks so funkcije, ki omogočajo uporabo stanja (angl. *state*) in drugih funkcionalnosti Reacta znotraj funkcijskih komponent. Funkcij React Hooks ni mogoče uporabiti v razrednih komponentah.

Najpogosteje uporabljene so useState, useEffect, useContext in useMemo.

Za dodajanje in upravljanje stanja lahko uporabimo useState. Zahteva spremenljivko in funkcijo za spremembo te spremenljivke. Razlog za uporabo useState je v tem, da so spremembe spremenljivke useState vidne v uporabniškem vmesniku. Pri uporabi navadne spremenljivke to ni mogoče.

Spodaj je primer uporabe useState. V tem primeru je *count* spremenljivka, *setCount* pa funkcija za spremembo vrednosti spremenljivke.

```
import { useState } from 'react';

function Counter() {
  const [count, setCount] = useState(0);

  return (
    <div>
      <p>Count: {count}</p>
      <button onClick={() => setCount(count + 1)}>Increment</button>
    </div>
  );
}
```

Hook useContext omogoča izvajanje stranskih učinkov (angl. *side effects*).

Sledi primer uporabe useEffect, pri kateri se ob vsaki spremembi useState spremenljivke *counter* izpiše njena vrednost v konzolo. V oglatih oklepajih so odvisnosti, ki sprožijo useEffect. V tem primeru je to spremenljivka *counter*, torej se useEffect sproži vsakič, ko se spremenljivka spremeni. Odvisnosti je lahko več, lahko so pa oglati oklepaji tudi prazni, kar pomeni, da se useEffect sproži le enkrat.

```jsx
import { useState, useEffect } from 'react';

function CounterLogger() {
  const [counter, setCounter] = useState(0);

  useEffect(() => {
    console.log(`Counter value: ${counter}`);
  }, [counter]);

  return (
    <div>
      <p>Counter: {counter}</p>
      <button onClick={() => setCounter(counter + 1)}>Increment</button>
    </div>
  );
}
```

Hook useContext omogoča dostop do podatkov iz React Context API. Je minimalistična rešitev za upravljanje globalnega stanja (angl. *global state management*). Omogoča enostavno deljenje podatkov med komponentami, brez potrebe po prenašanju podatkov v obliki *props*. Če morajo *props* biti podani skozi veliko komponent, da pridejo do cilja, je useContext odlična rešitev. (GeeksforGeeks, 2025)

4.2 Struktura

Next.js ima specifično strukturo datotek in map, ki omogoča napredne funkcionalnosti.

V novejših različicah Next.js so ključne te mape:

- mapa *app*, v kateri so datoteke, ki predstavljajo strani;
- mapa *api*, ki je v mapi *app*, je mesto, kjer so vse končne točke (angl. *endpoints*) za zaledni del aplikacije;
- mapa *public*, v kateri so vse statične datoteke, npr. slike, ikone ipd.

V Next.js je usmerjanje (angl. *routing*), urejeno s pomočjo strukture map in datotek.

Če želimo, da je datoteka v mapi *app* stran, jo moramo poimenovati *page*, npr. page.js ali page.jsx. Za ureditev postavitve strani se uporablja *layout*, npr. layout.js ali layout.jsx. S tem bomo ustvarili stran, ki bo dostopna na poti »/«, npr. http://localhost:3000/. Na primer, če želimo ustvariti pot »/users«, moramo v mapi *app* ustvariti novo mapo *users* in v njej datoteko page.js ali page.jsx. (Next.js, 2025)

4.3 Zaledni del

Čeprav je Next.js predvsem namenjen razvoju uporabniških vmesnikov na podlagi Reacta, omogoča tudi razvoj zalednega dela spletnih aplikacij, brez potrebe po ločenem zalednem strežniku. Torej celotno aplikacijo, uporabniški vmesnik in zaledni del, lahko razvijemo v enem projektu.

API-poti (angl. *routes*) so glavna funkcionalnost za zaledni del v Next.js. Omogočajo ustvarjanje strežniških končnih točk (angl. *endpoints*) znotraj projekta. Vsaka strežniška končna točka lahko obdeluje HTTP-zahteve, kot so GET, POST, PUT, DELETE itn.

V mapi *api* je pomembno, da so datoteke poimenovane *route*, torej route.js. Če želimo ustvariti API-pot *users*, moramo v mapi *api* ustvariti mapo *users* in znotraj nje datoteko route.js, v kateri se bodo obdelovali HTTP-zahtevki. (Next.js, 2025)

Prednost gradnje zalednega dela v Next.js je v tem, da ni treba imeti ločenega zalednega strežnika, torej je razvoj hitrejši in gostovanje enostavnejše, saj moramo gostiti le en projekt namesto dveh.

Pomanjkljivost je v tem, da lahko projekt postane nasičen z vsemi datotekami in mapami na zaledni in *frontend* strani, kar posledično pomeni, da je projekt manj berljiv.

Ogrodju Next.js v primerjavi s klasičnim zalednim sistemom, npr. Node.js, manjka nekaj funkcionalnosti. Predvsem zaradi omejitev z gostovanjem je težje integrirati povezave WebSocket. Prav tako Next.js nima vgrajenih mehanizmov za izvajanje opravil v ozadju, kot so *cron jobs*.

4.4 Izris strani

Next.js ponuja več pristopov k izrisu strani (angl. *rendering*), ki omogočajo boljšo zmogljivost, uporabniško izkušnjo in optimizacijo za SEO. Ti pristopi so SSR (angl. *Server-Side Rendering*), SSG (angl. *Static Site Generation*), CSR (angl. *Client-Side Rendering*) in ISR (angl. *Incremental Static Regeneration*). (Cezar, 2025)

4.4.1 SSR

Pri SSR se HTML-vsebina generira na strežniku ob vsakem zahtevku. To pomeni, da brskalnik prejme že ustvarjeno stran, kar bistveno zmanjša čas do prvega prikaza in izboljša SEO. (Cezar, 2025)

4.4.2 SSG

Pri SSG se strani generirajo v času gradnje projekta, torej vnaprej. Takšne strani so dostavljene kot statične datoteke, kar omogoča hitro nalaganje. Statične strani so enake za vse uporabnike in jih uporabnik ne more spremeniti. (Cezar, 2025)

4.4.3 CSR

Pri CSR se generiranje strani izvaja v brskalniku. To običajno pomeni, da se stran nalaga dlje časa in je čas nalaganja odvisen od odjemalca. Tak pristop je značilen za čisti React. (Cezar, 2025)

4.4.4 ISR

ISR je hibridni pristop, ki omogoča osveževanje statično generiranih strani po določenem času, brez potrebe po ponovni gradnji celotnega projekta. Strani so v osnovi statične, vendar se po nekem času osvežijo na strežniški strani. (Cezar, 2025)

4.5 Zmogljivost in optimizacija

Next.js ponuja več orodij in tehnik za optimizacijo spletnih aplikacij v primerjavi z osnovno knjižnico React.

4.5.1 Optimizacija slik z Next Image

Velike in neoptimizirane slike lahko upočasnijo nalaganje strani. Next.js ponuja komponento Next Image, ki nadomesti HTML-značko <img> in avtomatsko optimizira sliko.

Slike se prilagodijo velikosti zaslona, kar pomeni, da odjemalec prejme slike optimiziranih mer.

Lazy loading je tehnika, pri kateri se slike naložijo šele, ko se uporabnik pomakne do njih. To bistveno zmanjša čas začetnega nalaganja.

Next.js izbere optimalen format slike za prikaz v brskalniku. Ta format je običajno WebP (angl. *Web Picture*). (Turingvang, 2025)

4.5.2 Dinamični uvoz

Next.js samodejno razdeli aplikacijo na manjše dele (angl. *chunks*), ki se naložijo le po potrebi. To zmanjša začetno velikost datotek v JavaScriptu in pospeši nalaganje strani.

Dinamični uvoz omogoča nalaganje komponent pozneje, npr. ko uporabnik klikne gumb. (Farihatul, 2025)

4.5.3 Optimizacija kode v JavaScriptu in CSS-u

Next.js ima svoj prevajalnik (angl. *compiler*), ki je hitrejši od standardnega prevajalnika Babel. Samodejno izvaja pomanjševanje (angl. *minification*) datotek v JavaScriptu in CSS-u. Naloži le potrebne dele kode, odvečna koda v JavaScriptu in CSS-u pa se samodejno odstrani. Tako se zmanjša velikost datotek in poveča hitrost nalaganja. (Next.js, 2025)

4.6 SEO

Poleg metod za izris strani SSR in SSG je v Next.js mogoče dobro poskrbeti za optimizacijo SEO.

4.6.1 Upravljanje metapodatkov

Next.js uporablja komponento Next Head, ki omogoča preprosto upravljanje metaoznak, kot so naslov strani, opis in ključne besede. Pravilna uporaba metapodatkov je pomembna za SEO, saj iskalnikom omogoča pravilno predstavitev vsebine. (Next.js, 2025)

4.6.2 Podpora za sitemap in robots.txt

Integracija datotek sitemap.xml in robots.xml je v aplikaciji Next.js najpogosteje omogočena s paketom *next-sitemap*.

Sitemap je XML-datoteka, ki vsebuje seznam pomembnih URL-jev (angl. *Uniform Resource Locator*), skupaj z dodatnimi informacijami. Namen tega je, da iskalnikom olajša odkrivanje in indeksiranje vseh podstrani na spletni strani.

Robots.txt je besedilna datoteka, ki iskalnikom sporoča, katere dele spletne strani naj indeksirajo in katere naj preskočijo. Tako se prepreči, da iskalniki dostopajo do zasebnih ali začasnih delov strani. (npm, 2025)

4.6.3 Hitrost nalaganja in optimizacija

Hitrost nalaganja strani je eden izmed ključnih vidikov za SEO. Next.js z omenjenimi orodji za optimizacijo in metodami za izris pomaga doseči hitro nalaganje strani. Prav tako je zelo pomembno, da je spletna stran optimizirana za različne zaslone (angl. *responsiveness*).

4.7 Next.js kot osnova za PWA

Next.js že od začetka podpira funkcionalnosti, ki so pomembne za PWA. Večina aplikacij Next.js teče na strežnikih, ki podpirajo HTTPS, kar je pogoj za PWA. Prav tako se lahko z dodatnimi paketi preprosto integrirajo *service workerji*, ki omogočajo delovanje brez spletne povezave.

Za izdelavo PWA z Next.js je treba dodati datoteko manifest.js, ki opisuje aplikacijo z atributi, kot so ime, ikona, barve ipd. Prav tako je *service workerje* treba implementirati s paketom *next-pwa*.

Ko je vse to urejeno, se ikona za namestitev PWA običajno prikaže na vrhu brskalnika v naslovni vrstici, odvisno od brskalnika in naprave. Ko se namesti, lahko aplikacijo uporabljamo kot spletno stran brez brskalnika. (npm, 2025)

4.8 Vercel kot platforma za gostovanje

Vercel je platforma za gostovanje spletnih aplikacij. Glede na to, da je platformo razvilo podjetje, ki je razvilo tudi Next.js, je gostovanje zelo preprosto in dobro optimizirano. Vercel je prva izbira razvijalcev za gostovanje aplikacij Next.js.

Vercel avtomatsko podpira Next.js funkcionalnosti, kot so SSR, SSG in ISR. Prav tako za vse API-poti omogoča funkcije brez strežnika (angl. *serverless*), kar pomeni, da strežniška logika poteka brez upravljanja lastnih strežnikov.

Glede na to, da Next.js ne podpira *cron jobs*, jih Vercel omogoča. To je urejeno tako, da se v Vercelu nastavi čas za izvedbo funkcije, v Next.js pa se le izbere funkcija, ki se mora izvesti.

Proces gostovanja je zelo preprost, treba je le povezati Git repozitorij projekta. Tako se ustvari spletna aplikacija, ki je enaka tisti iz repozitorija. Ko Vercel zazna, da se je zgodila sprememba v repozitoriju, se aplikacija znova naloži. (Next.js, 2025)

5 PROCES RAZVOJA APLIKACIJ

Aplikacija Barber Booking je namenjena frizerjem, ki želijo imeti organiziran seznam terminov za striženje. Omogoča avtentikacijo, ustvarjanje strank, pregled terminov po dnevih in rezervacijo termina z možnostjo izbire stranke. Prav tako je omogočeno brisanje termina in prav tako stranke.

Aplikacija bo ustvarjena v ogrodjih Next.js in Flutter, z namenom primerjave tehničnih značilnosti, uporabniške izkušnje, zmogljivosti ter varnosti.

Za upravljanje z različicami (angl. *version control*) bosta uporabljena Git in platforma GitHub.

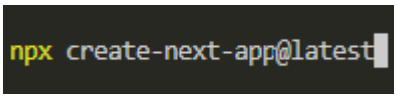
5.1 Aplikacija Next.js

5.1.1 Nalaganje potrebnih programov in ustvarjanje projekta

Če gradimo Next.js projekt od začetka, moramo naložiti Node.js. Skupaj z Node.js se bo naložil tudi npm (angl. *Node Package Manager*). Node.js je izvajalno okolje za programski jezik JavaScript. Uporablja se za izvedbo ogrodij v JavaScriptu. Za nalaganje paketov in knjižnic se uporablja npm. (Node.js, 2025)

Za IDE (angl. *Integrated Development Environment*) smo izbrali Visual Studio Code, saj je to standard za razvoj spletnih aplikacij in tudi drugih programov. (Visual Studio Code, 2025)

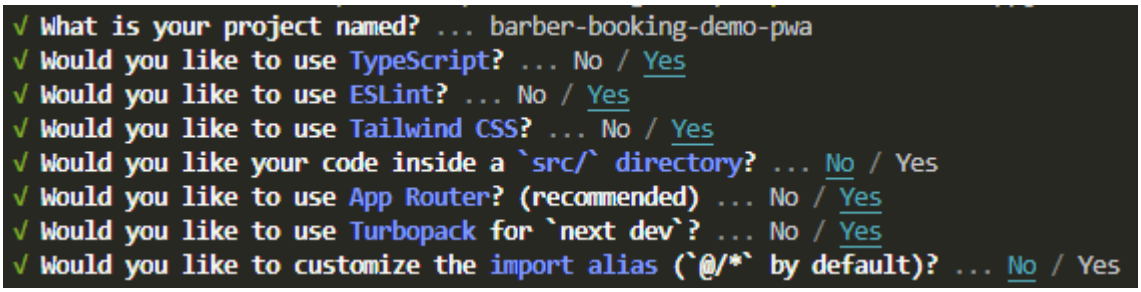
Ko je vse pripravljeno, si izberemo mapo, v kateri želimo ustvariti projekt. Odpremo terminal v tisti mapi in napišemo ukaz:



Slika 3: Ukaz za kreiranje projekta Next.js

(Vir: Lasten vir)

Nato bo treba odgovoriti na vprašanja:



Slika 4: Vprašanja pri ustvarjanju projekta Next.js

(Vir: Lasten vir)

Tako se bo ustvaril projekt barber-booking-demo-pwa. Tukaj smo izbrali jezik TypeScript, saj želimo podporo za statično tipiziranje. (Next.js, 2025)

5.1.2 Komponente

Aplikacija je razdeljena na strani in komponente zaradi boljšega pregleda in uporabe modernih praks.

Imamo tri strani: domača stran, *booking* in *customers*. Pri tem se domača stran uporablja za avtentikacijo s PIN (angl. *Personal Identification Number*).

Komponent je nekaj več, najpomembnejše pa so Calendar, Customers, CustomerAutocomplete in Navigator. Imamo tudi komponenti za izboljšanje uporabniške izkušnje: Alert in Loader.

5.1.3 Zaledni del

Za zaledni del se v obeh aplikacijah uporablja ločen zaledni strežnik, ustvarjen z Node.js in Express.js. Aplikacija Next.js ima enake API-poti, kot strežnik Node.js, vendar bosta za namene primerjave obe aplikaciji komunicirali z zalednim sistemom Node.js.

Edina izjema je avtentikacija, pri kateri bo aplikacija Next.js uporabljala lastno API-pot in paket next-auth za PIN-avtentikacijo. Ta izjema je prav tako za namene primerjave, saj lahko primerjamo varnost pri uporabi lastnih API-poti in varnost pri uporabi ločenega zalednega sistema.

5.1.4 Knjižnice, paketi in drugi pripomočki

Kot že omenjeno, se za namene avtentikacije uporablja paket next-auth, ki omogoča preprosto vzpostavitev avtentikacije v aplikacijah Next.js. (NextAuth.js, 2025)

MUI (angl. *Material User Interface*) je knjižnica React za gradnjo uporabniških vmesnikov, ki jo je ustvarilo podjetje Google. Omogoča uporabo številnih komponent, ki implementirajo Googlov Material Design. Ključne komponente uporabljene v projektu so Typography, Button, Box, Modal in Input. (MUI, 2025)

Za videz aplikacije se uporablja tudi Tailwind CSS, ki omogoča pisanje kode v CSS-u v obliki HTML-razredov. (Tailwind CSS, 2025)

Za gradnjo koledarja se uporablja FullCalendar, ki je knjižnica, ustvarjena za aplikacije v JavaScriptu. (FullCalendar, 2025)

Za izvajanje HTTP-zahtevkov se uporablja knjižnica axios, ki se lahko uporabi v brskalniku ali v zalednem delu Node.js. (Axios, 2025)

Day.js je knjižnica v JavaScriptu, ki se uporablja za upravljanje datumov. (Day.js, 2025)

5.1.5 Priprava za PWA

Priprava za PWA poteka s knjižnico next-pwa.

Najprej je treba ustvariti datoteko next.config.js:

```
const withPWA = require("next-pwa")({
  dest: "public",
});

module.exports = withPWA({
  scope: "/app",
});
```

Slika 5: Datoteka next.config.js

(Vir: Lasten vir)

Nato je treba ustvariti datoteko manifest.js v mapi app:

```
import type { MetadataRoute } from "next";

export default function manifest(): MetadataRoute.Manifest {
  return {
    name: "Barber booking",
    short_name: "BBDPWA",
    description: "Barber booking demo PWA app",
    start_url: "/",
    display: "standalone",
    background_color: "#000000",
    theme_color: "#000000",
    icons: [
      {
        src: "/icon-192x192.png",
        sizes: "192x192",
        type: "image/png",
      },
      {
        src: "/icon-512x512.png",
        sizes: "512x512",
        type: "image/png",
      },
    ],
  };
}
```

Slika 6: Datoteka manifest.js

(Vir: Lasten vir)

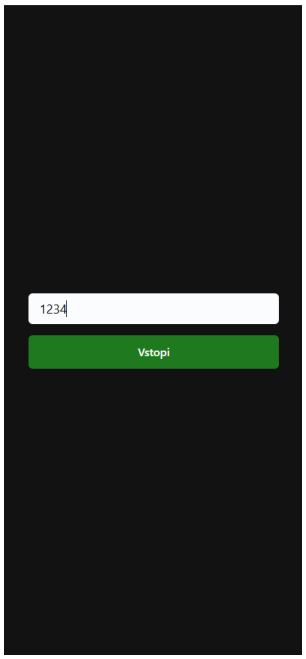
Tako smo omogočili *service worker* in posledično delovanje brez povezave. Prav tako je omogočeno dodajanje aplikacije na začetni zaslon mobilnega telefona ali namiznega računalnika. (npm, 2025)

5.1.6 Končna aplikacija in primer uporabe

Aplikacija Barber Booking omogoča frizerju preprosto upravljanje in pregled terminov po dnevih in urah. Prav tako omogoča preprosto dodajanje in izbiro strank pri ustvarjanju termina. Koledar prikazuje termine po dnevih in vsebuje gumbe, ki omogočajo menjavo dneva za en dan ali sedem dni.

Aplikacija bo prikazana s pomočjo zaslonskih slik, prav tako bo razloženo delovanje na primeru.

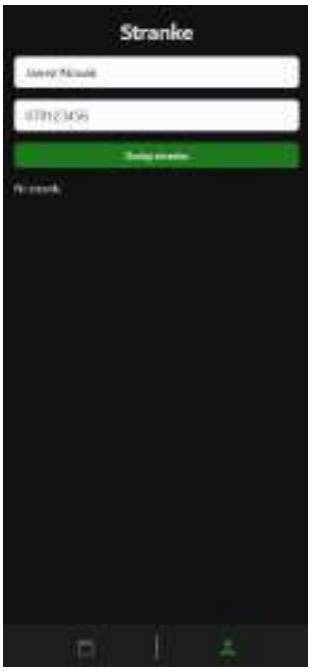
Prva stran, ki jo uporabnik vidi, je stran za vnos PIN-a. V vnosno polje vnese štirimestno številko in pritisne gumb »Vstopi«.



Slika 7: Zaslon za prijavo v PWA

(Vir: Lasten vir)

Na zaslonu za stranke lahko dodamo novo stranko z vpisom imena in priimka ter telefonske številke v vnosna polja in s pritiskom na gumb »Dodaj stranko«.



Slika 8: Ustvarjanje stranke v PWA

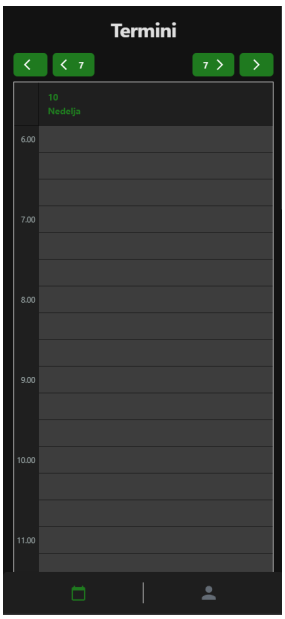
(Vir: Lasten vir)

Stranka je ustvarjena, lahko jo vidimo na seznamu strank.

Slika 9: Seznam strank v PWA

(Vir: Lasten vir)

Če v navigaciji spodaj levo pritisnemo na ikono koledarja, se bomo prestavili na zaslon za termine. Z uporabo gumbov zgoraj nad koledarjem lahko izberemo dan.



Slika 10: Zaslon za termine v PWA

(Vir: Lasten vir)

S klikom na prazno vrstico v koledarju se odpre dialog, v katerem lahko izberemo stranko in dodamo termin.

Slika 11: Dialog za dodajanje termina v PWA

(Vir: Lasten vir)

Po dodajanju par terminov bo koledar videti tako:

Slika 12: Seznam terminov v PWA

(Vir: Lasten vir)

S stiskom na termin se odpre dialog, ki izpiše podrobnosti in omogoča izbris izbranega termina.

Slika 13: Podrobnosti termina v PWA

(Vir: Lasten vir)

Če se vrnemo na zaslon za stranke, s stiskom na ikono koša za smeti lahko izbrišemo stranko. S tem se bodo izbrisali vsi termini, ki so vezani na to stranko.

Slika 14: Brisanje stranke v PWA

(Vir: Lasten vir)

5.2 Aplikacija Flutter

5.2.1 Nalaganje potrebnih programov in ustvarjanje projekta

Za ustvarjanje aplikacij v Flutterju moramo najprej naložiti Flutter SDK (angl. *Software Development Kit*). Glede na to, da bomo uporabljali Visual Studio Code, je najlažje, da uporabimo vtičnik v Flutterju.

Slika 15: Flutterjev vtičnik

(Vir: Lasten vir)

Prav tako imamo naložen Dartov vtičnik za podporo jeziku Dart.

Zdaj stisnemo tipke na tipkovnici: Ctrl, Shift in P. S tem se odpre spustni seznam, s katerega izberemo »Flutter: New project«

Slika 16: Spustni seznam za ustvarjanje projekta v Flutterju

(Vir: Lasten vir)

Če že nimamo naloženega Flutter SDK, bomo morali izbrati lokacijo na disku, kamor se bo samodejno naložil. Po tem se bo prikazal še en spustni seznam za izbiro vrste projekta. Tukaj izberemo »Application«.

Slika 17: Spustni seznam za izbiro vrste projekta v Flutterju

(Vir: Lasten vir)

Po vpisu imena projekta in lokacije na disku se bo trenutno okno zaprlo in odprlo se bo novo okno Visual Studio Code, v katerem bo projekt, ki smo ga pravkar ustvarili.

Lahko opazimo, da ima Visual Studio Code pri uporabi Flutterja določene dodatke v uporabniškem vmesniku. Čisto na levi strani so dodane tri ikone, ki se uporabljajo kot orodja za razvijalce.

Slika 18: Orodja v Flutterju za razvijalce

(Vir: Lasten vir)

Prav tako je čisto spodaj dodan gumb za izbiro naprave, na kateri se bo izvaja aplikacija med procesom razvoja. To je lahko fizična ali virtualna naprava.

Slika 19: Gumb za izbiro naprave

(Vir: Lasten vir)

Če želimo imeti virtualno napravo, je treba naložiti Android Studio, ki omogoča izvajanje virtualne naprave na sistemu. (Flutter, 2025)

5.2.2 Widgets

V aplikaciji smo ustvarili štiri gradnike (*widgets*): MyApp, LoginScreen, BookingScreen in CustomerScreen. MyApp je osnovni gradnik, torej najvišji v hierarhiji, iz njega izhajajo vsi drugi.

5.2.3 Zaledni del

Aplikacija pošilja HTTP-zahtevke na zaledni strežnik, ustvarjen z Node.js in Express.js. Tukaj se avtentikacija ureja s pomočjo zalednega strežnika, saj Flutter ne omogoča lastnih API-poti tako kot Next.js.

5.2.4 Paketi

Paketi se v projekt v Flutterju dodajo tako, da se dodata ime paketa in želena različica v datoteko pubspec.yaml, v razdelek dependencies.

Slika 20: Datoteka pubsec.yaml

(Vir: Lasten vir)

Nato moramo v terminalu izvesti ukaz *flutter pub get*. Ta ukaz prenese in namesti vse pakete, ki so našteti v datoteki pubsec.yaml. Paketi se prenesejo iz repozitorija pub.dev. (Dart, 2025)

V tem projektu so uporabljeni štirje paketi: http, cupertino_icons, shared_preferences in syncfusion_flutter_calendar.

Paket http se uporablja za pošiljanje HTTP-zahtevkov na zaledni strežnik in njihovo obdelavo. (pub.dev, 2025)

Paket cupertino_icons omogoča uporabo ikon Apple iOS v aplikaciji. (pub.dev, 2025)

Paket shared_preferences omogoča shranjevanje manjših količin podatkov lokalno na napravi. (pub.dev, 2025)

V tem projektu se za lokalno shranjevanje uporablja JWT (angl. *JSON Web Token*) pri avtentikaciji. To aplikaciji omogoča, da ve, ali je uporabnik prijavljen ali ni. Vsak zaslon je zaščiten tako, da ne omogoča uporabe, če uporabnik ni prijavljen. Če se to vseeno zgodi, je uporabnik preusmerjen na zaslon za prijavo. (jwt.io, 2025)

Paket syncfusion_flutter_calendar je uporabljen za prikaz koledarja v aplikaciji na zaslonu BookingScreen. Tako je omogočeno tudi upravljanje terminov. (pub.dev, 2025)

5.2.5 Končna aplikacija

Glede na to, da je funkcionalnost aplikacij Flutter in Next.js enaka, bomo tukaj pokazali le videz aplikacije, saj je funkcionalnost s primerom že prikazana v prejšnjih poglavjih.

Slika 21: Zaslon s prijavo v Flutter

(Vir: Lasten vir)

Slika 22: Zaslon s termini v Flutterju

(Vir: Lasten vir)

Slika 23: Dodajanje stranke v Flutter

(Vir: Lasten vir)

Slika 24: Dodajanje termina v Flutter

(Vir: Lasten vir)

Slika 25: Podrobnosti termina v Flutterju

(Vir: Lasten vir)

Slika 26: Brisanje stranke v Flutterju

(Vir: Lasten vir)

5.3 Zaledni sistem

Zaledni sistem je ustvarjen s pomočjo okolja Node.js in ogrodja Express.js. Vsebuje API-poti za CRUD (angl. *Create, Read, Update, Delete*) strank in terminov. Prav tako omogoča avtentikacijo s pomočjo JWT.

Uporablja programski jezik TypeScript za strogo tipiziranje.

MongoDB je baza, ki je uporabljena za namene shranjevanja podatkov o strankah in terminih. To je dokumentna podatkovna baza NoSQL (angl. *not only SQL*), ki shranjuje podatke v obliki, podobni JSON, imenovani BSON (angl. *Binary JSON*). Pogosto se uporablja s tehnologijami, ki temeljijo na programskem jeziku JavaScript, saj je prijazen za uporabo s tem jezikom. (GeeksforGeeks, 2025)

Za povezavo z bazo se uporablja knjižnica Mongoose, ki omogoča preprosto povezavo in strukturo podatkov. (mongoose, 2025)

5.4 Stroški uporabe

Stroškov razvoja obeh aplikacij in zalednega sistema pri osnovni uporabi ni, razen če uporabljamo dodatne plačljive knjižnice oz. pakete. Razlike v stroških nastanejo pri gostovanju Next.js PWA in objavi aplikacije Flutter.

Next.js PWA je mogoče gostiti na platformi Vercel, kjer začetnih stroškov ni. Namreč začetni Hobby plan je zastonj. Če pa v prihodnosti aplikacija dobi veliko uporabnikov in začetni plan ni dovolj, obstaja Pro plan, ki stane vsaj ~ 17,22 evra na mesec, saj se lahko zaračunajo dodatni stroški glede na dodatno porabo. (Vercel, 2025)

Pri aplikaciji Flutter je nekoliko drugače, saj jo je treba objaviti v trgovini za aplikacije. Če jo hočemo imeti na obeh operacijskih sistemih, Androidu in iOS-u, je objava nujna v trgovinah Google Play in App Store. Google Play zahteva enkraten strošek v višini ~ 21,52 evra. (Google, 2025) App Store pa zahteva strošek v višini ~ 85,22 evra na leto. (Apple, 2025)

Tabela 1: Primerjava stroškov gostovanja/objave aplikacij

	Next.js PWA (Vercel)	Aplikacija Flutter (Google Play in App Store)

Začetni stroški	0 (Vercel Hobby Plan)	Google Play: ~ 21,52 EUR App Store: ~ 85,22 EUR/leto
Obratovalni stroški	Vercel Hobby Plan: 0 Pro Plan: od ~ 17,22 EUR/mesec	Google Play: 0 App Store: ~ 85,22 EUR/leto

(Vir: Lasten vir)

6 ANALIZA SKUPNOSTI IN PODPORE

Priljubljenost tehnologij, ki jih uporabljamo, je zelo pomembna, saj je to kazalnik, da je tehnologija kakovostna in da bo najverjetneje še dolgo podprta. V tem poglavju si bomo pogledali statistične podatke in trende, ki prikazujejo, kako široko sta tehnologiji Flutter in Next.js uporabljeni v industriji.

Vsi statistični podatki so nazadnje pridobljeni 11. avgusta 2025 in se v prihodnosti lahko spremenijo.

6.1 Skupnosti razvijalcev

6.1.1 GitHub

Next.js ima na platformi GitHub 30.172 sprememb (angl. *commits*) in 3648 razvijalcev, ki so naredili spremembe. Okoli 134.000 uporabnikov si je repozitorij označilo kot priljubljen in okoli 29.000 uporabnikov si je iz tega repozitorija ustvarilo svojega (angl. *fork*). Ima okoli 2300 prijav napak in nalog (angl. *issues*). V celoti ima okoli pet milijonov uporabnikov. (GitHub, 2025)

Flutter ima 85.439 sprememb, kar je bistveno več, vendar ima veliko manj razvijalcev, le 1598. Veliko več uporabnikov si je označilo repozitorij kot priljubljen, okoli 172.000. Število *forkov* je pa enako, torej okoli 29.000. Število GitHub Issues je okoli 5000, kar je bistveno več. Nimajo navedenega podatka o celotni bazi uporabnikov, tako da tega ne moremo primerjati. (GitHub, 2025)

6.1.2 Stack Overflow

Stack Overflow je platforma za postavljanje vprašanj glede programiranja, na kateri si lahko uporabniki med seboj pomagajo in odgovarjajo na vprašanja.

Next.js ima na tej platformi 42.641 vprašanj (Stack Overflow, 2025), Flutter pa 179.717 (Stack Overflow, 2025). Razlika je tukaj precej večja, kot je bila na platformi GitHub. Razlaga je lahko, da je Flutter težji za učenje in da uporabniki naletijo na več težav, ki jih ne znajo sami

rešiti. Prav tako je Next.js bolj specifična tehnologija od Flutterja, zato si bomo še pogledali številke vprašanj za React, saj na tej tehnologiji temelji Next.js.

React ima 476.726 vprašanj, kar pojasni tako majhno število vprašanj za Next.js. (Stack Overflow, 2025)

6.1.3 Reddit

Reddit ni platforma, ki bi bila namenjena samo razvijalcem in programiranju, temveč je splošno namenska. Kljub temu je skupnost razvijalcev na tej platformi zelo velika.

Next.js ima na tej platformi svojo skupnost (angl. *subreddit*) z imenom r/nextjs in ima okoli 142.000 članov. Ustvarjena je bila 25. oktobra 2016. (Reddit, 2025)

Flutter ima svojo skupnost z imenom r/FlutterDev in ima okoli 160.000 članov. Ustvarjena je bila 30. aprila 2013, kar je zanimivo, saj je Flutter izšel leta 2017. Mogoče je, da je bila ta skupnost prej uporabljena za kaj drugega ali za različico Flutterja, ki še ni izšla. (Reddit, 2025)

6.1.4 Primerjava skupnosti razvijalcev

Tabela 2: Primerjava skupnosti razvijalcev obeh tehnologij

	Next.js PWA	Aplikacija Flutter
Število *commitov* na GitHubu	30.172	85.439
Število razvijalcev na GitHubu	3648	1598
Število oznak »priljubljeno« na GitHubu	Okoli 134000	Okoli 172.000
Število *forkov* na GitHubu	Okoli 29.000	Okoli 29.000
Število prijav napak na GitHubu	Okoli 2300	Okoli 5000
Število uporabnikov na GitHubu	Okoli 5,000.000	/
Število vprašanj na Stack Overflowu	42.641	179.717
Število članov na Redditu	Okoli 142.000	Okoli 160.000

(Vir: Lasten vir)

6.2 Viri

6.2.1 Uradna dokumentacija in videoposnetki

Obe tehnologiji imata uradno dokumentacijo, ki je zelo dobra in razumljiva. Nekdo, ki se še ni ukvarjal s temi tehnologijami, bo lahko z uporabo uradne dokumentacije zelo hitro začel razvijati aplikacije. Osebno ne bi rekel, da je katera izmed dokumentacij boljša, obe sta izjemni.

Za tiste, ki imajo raje videoposnetke, sta na voljo Youtubova kanala z vsebinami, ki lahko pomagajo prav vsakemu razvijalcu.

Flutter ima uradni kanal, na katerem objavljajo videoposnetke o tem, kako Flutter deluje, novosti, konference, tutoriale ipd. (Youtube, 2025)

Čeprav Next.js nima uradnega kanala, obstaja kanal »leerob« (Youtube, 2025). Vodi ga Lee Robinson, ki je pomemben del podjetja Vercel. Na svojem kanalu objavlja videoposnetke o Next.js, platformi Vercel, Reactu in drugih spletnih tehnologijah. (leerob, 2025)

Rečemo lahko, da Flutterjev uradni kanal objavlja pogosteje, saj ima več ustvarjalcev. Prav tako ga je lažje najti na Youtubu.

6.2.2 Neuradni tečaji

Platforma Udemy je namenjena učenju s spletnimi tečaji, ki jih ustvarijo znani razvijalci. Vsebine so večinoma plačljive, vendar so običajno vredne svoje cene, saj so razlage zelo dobre in sestava tečaja je običajno odlična. Primerjali bomo po en tečaj vsake tehnologije, ki ima največ ocen.

Next.js ima tečaj s 23.400 ocenami in njegova povprečna ocena je 4,7. Čeprav ni mogoče videti števila vseh tečajev, je število strani s tečaji 417. (Udemy, 2025)

Flutter ima tečaj s 86.981 ocenami in njegova povprečna ocena je 4,6. Število strani je 150. (Udemy, 2025)

Tukaj lahko opazimo, da ima Flutter precej več ocen, vendar ima Next.js precej več tečajev. Število tečajev pri Next.js je tako veliko, ker so na ta seznam vključeni tudi tečaji React.

7 PRIMERJAVA ZMOGLJIVOSTI

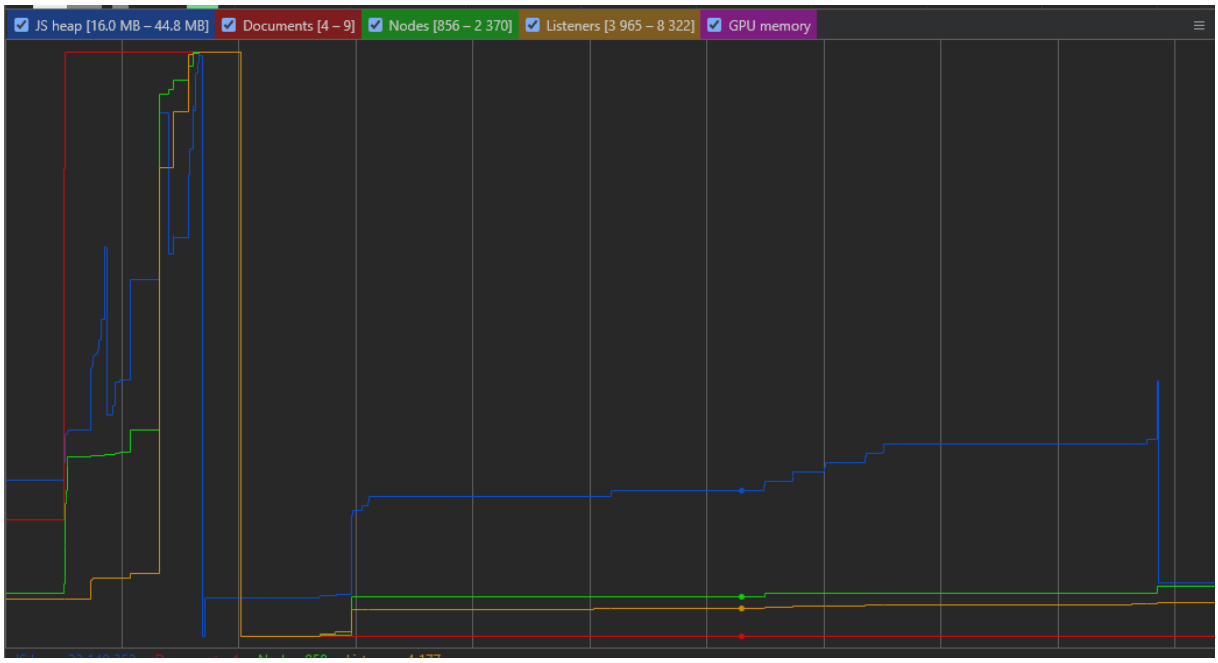
Primerjali bomo obe aplikaciji. Za ustrezno in pošteno primerjavo ju bomo testirali na isti napravi in istem omrežju ter povezani na isti zaledni sistem. Zaledni sistem se bo izvajal na računalniku, ki je priklopljen na isto omrežje kot mobilni telefon, na katerem bomo testirali obe aplikaciji. Aplikacijo Next.js bomo testirali na brskalniku Google Chrome, kjer bomo meritve dobili s pomočjo orodij za razvijalce. Meritve Flutterja bomo dobili s Flutterjevimi orodji za razvijalce. Testiranje bo potekalo ročno.

Testni scenarij je:

- zagon aplikacije,
- navigacija na stran za stranke in nato nazaj na stran za termine,
- ustvarjanje desetih dogodkov zaporedno.

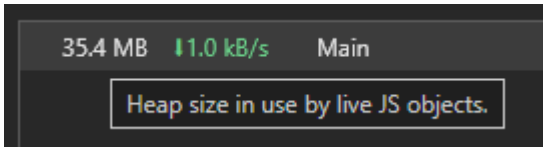
7.1 Merjenje porabe pomnilnika

7.1.1 Poraba pomnilnika Next.js PWA



Slika 27: Poraba pomnilnika v PWA

(Vir: Lasten vir)

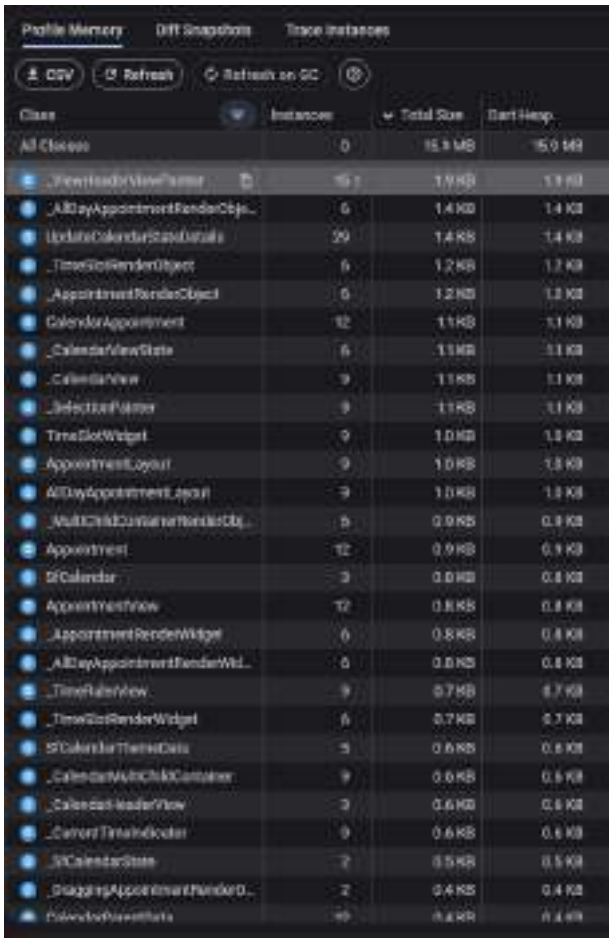


Slika 28: Velikost kopice v PWA

(Vir: Lasten vir)

Vidimo, da PWA porabi najmanj 16 MB (angl. *megabyte*) in največ 44,8 MB. Povprečje je 35,4 MB.

7.1.2 Poraba pomnilnika v Flutterju



Slika 29: Poraba pomnilnika v Flutterju

(Vir: Lasten vir)

Iz priložene slike vidimo, da Flutter porabi 15,9 MB, kar je bistveno manj kot PWA.

7.1.3 Primerjava porabe pomnilnika

Tabela 3: Primerjava porabe pomnilnika obeh aplikacij

	Next.js PWA	Aplikacija Flutter
Minimalna poraba pomnilnika	16 MB	/
Maksimalna poraba pomnilnika	44,8 MB	/
Povprečna poraba pomnilnika	35,4 MB	15,9 MB

(Vir: Lasten vir)

7.2 Merjenje časov HTTP-zahtevkov

7.2.1 Čas HTTP-zahtevkov Next.js PWA

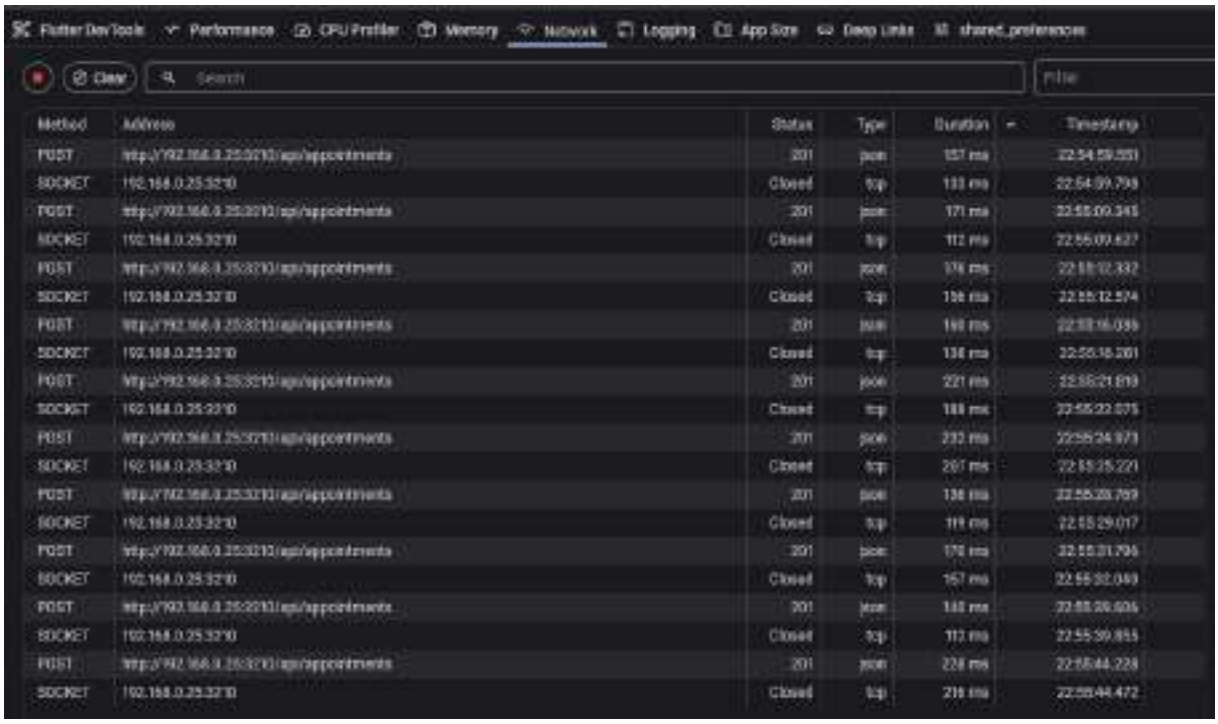
Name	Status	Type	Initiator	Size	Time
customers	200	xhr	CustomerAutocomplete.tsx:16	612 B	52 ms
customers	200	xhr	CustomerAutocomplete.tsx:16	211 B	80 ms
appointments	204	preflight	Preflight	0 B	33 ms
appointments	201	xhr	Calendar.tsx:122	649 B	154 ms
customers	200	xhr	CustomerAutocomplete.tsx:16	211 B	37 ms
customers	200	xhr	CustomerAutocomplete.tsx:16	211 B	74 ms
appointments	204	preflight	Preflight	0 B	23 ms
appointments	201	xhr	Calendar.tsx:122	649 B	92 ms
customers	200	xhr	CustomerAutocomplete.tsx:16	211 B	58 ms
customers	200	xhr	CustomerAutocomplete.tsx:16	211 B	88 ms
appointments	204	preflight	Preflight	0 B	20 ms
appointments	201	xhr	Calendar.tsx:122	651 B	89 ms
customers	200	xhr	CustomerAutocomplete.tsx:16	211 B	84 ms
customers	200	xhr	CustomerAutocomplete.tsx:16	211 B	120 ms
appointments	204	preflight	Preflight	0 B	27 ms
appointments	201	xhr	Calendar.tsx:122	651 B	187 ms
customers	200	xhr	CustomerAutocomplete.tsx:16	211 B	46 ms
customers	200	xhr	CustomerAutocomplete.tsx:16	211 B	76 ms
appointments	201	xhr	Calendar.tsx:122	649 B	108 ms
customers	200	xhr	CustomerAutocomplete.tsx:16	211 B	49 ms
customers	200	xhr	CustomerAutocomplete.tsx:16	211 B	78 ms
appointments	204	preflight	Preflight	0 B	20 ms
appointments	201	xhr	Calendar.tsx:122	651 B	178 ms
customers	200	xhr	CustomerAutocomplete.tsx:16	211 B	61 ms
customers	200	xhr	CustomerAutocomplete.tsx:16	211 B	92 ms
appointments	201	xhr	Calendar.tsx:122	649 B	126 ms
customers	200	xhr	CustomerAutocomplete.tsx:16	211 B	53 ms
customers	200	xhr	CustomerAutocomplete.tsx:16	211 B	81 ms
appointments	204	preflight	Preflight	0 B	39 ms
appointments	201	xhr	Calendar.tsx:122	651 B	93 ms
customers	200	xhr	CustomerAutocomplete.tsx:16	211 B	69 ms
customers	200	xhr	CustomerAutocomplete.tsx:16	211 B	98 ms
customers	200	xhr	CustomerAutocomplete.tsx:16	211 B	46 ms
customers	200	xhr	CustomerAutocomplete.tsx:16	211 B	74 ms
appointments	204	preflight	Preflight	0 B	23 ms
appointments	201	xhr	Calendar.tsx:122	651 B	94 ms
customers	200	xhr	CustomerAutocomplete.tsx:16	211 B	70 ms
customers	200	xhr	CustomerAutocomplete.tsx:16	211 B	99 ms
appointments	204	preflight	Preflight	0 B	39 ms
appointments	201	xhr	Calendar.tsx:122	649 B	99 ms

Slika 30: HTTP-zahtevki v PWA

(Vir: Lasten vir)

Iz priložene slike vidimo, da je narejenih deset HTTP-zahtevkov za ustvarjanje termina. Povprečen čas je 122 ms.

7.2.2 Čas HTTP-zahtevkov v Flutterju



Slika 31: Čas HTTP-zahtevkov v Flutterju

(Vir: Lasten vir)

Iz priložene slike vidimo HTTP-zahtevke. Povprečen čas je 179,1 ms, kar je bistveno več v primerjavi z Next.js PWA.

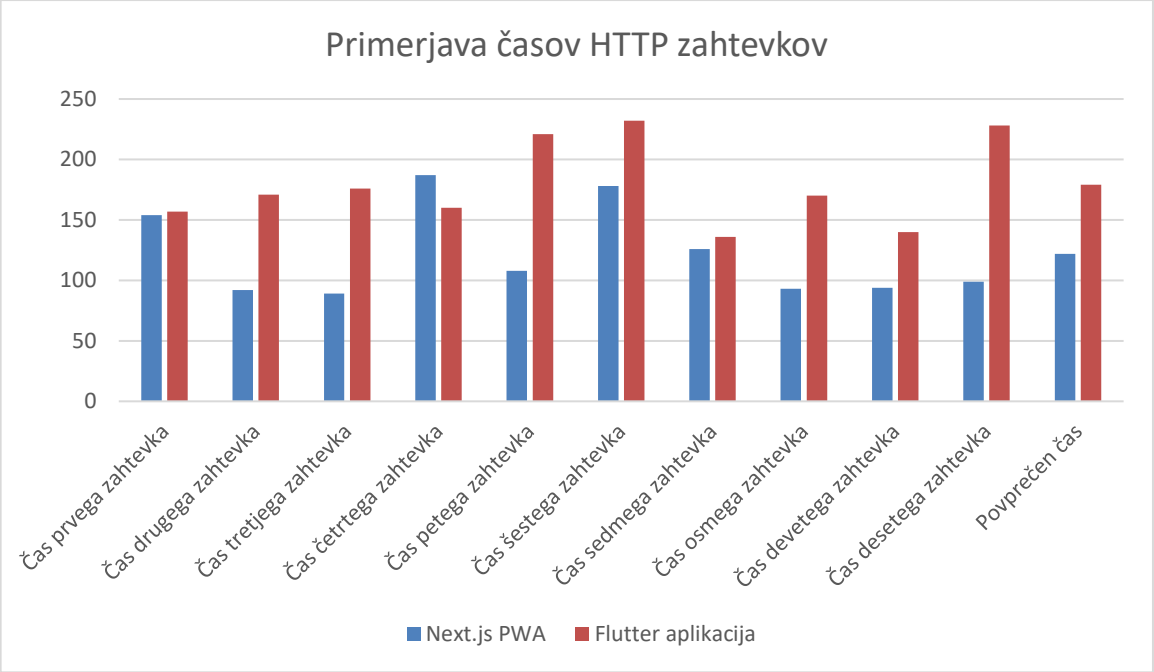
7.2.3 Primerjava časov HTTP-zahtevkov

Tabela 4: Primerjava časov HTTP-zahtevkov v obeh aplikacijah

	Next.js PWA	Aplikacija Flutter
Čas prvega zahtevka	154 ms	157 ms
Čas drugega zahtevka	92 ms	171 ms
Čas tretjega zahtevka	89 ms	176 ms
Čas četrtega zahtevka	187 ms	160 ms
Čas petega zahtevka	108 ms	221 ms

Čas šestega zahtevka	178 ms	232 ms
Čas sedmega zahtevka	126 ms	136 ms
Čas osmega zahtevka	93 ms	170 ms
Čas devetega zahtevka	94 ms	140 ms
Čas desetega zahtevka	99 ms	228 ms
Povprečen čas	122 ms	179,1 ms

(Vir: Lasten vir)



Graf 1: Primerjava časov HTTP-zahtevkov obeh aplikacij

(Vir: Lasten vir)

8 PRIMERJAVA UPORABNIŠKE IZKUŠNJE IN TEHNOLOŠKIH ZNAČILNOSTI

8.1 Dizajn in odzivnost

8.1.1 Next.js PWA

Dizajn je odvisen od HTML, CSS in odzivnosti spletne strani. Prav tako je odvisen od brskalnika, saj starejši brskalniki ne podpirajo določenih funkcionalnosti CSS.

Animacije so na novejših telefonih dokaj hitre, ampak vseeno niso tako tekoče kot na *native* aplikacijah. (Damir, 2025)

8.1.2 Flutter

Uporabniški vmesnik je risan z lastnim mehanizmom Skia, zato je videti enako na vseh platformah.

Odzivnost je visoka in animacije so dobro optimizirane tudi na srednje zmogljivih napravah. (Damir, 2025)

Uporabiti je mogoče gradnika Material in Cupertino za boljšo integracijo s platformo ter posledično boljšo uporabniško izkušnjo.

8.2 Native funkcionalnosti

8.2.1 Flutter

Flutter ima neposreden dostop do strojne opreme, torej dostop do kamere, GPS, bluetootha in senzorjev. Prav tako lahko izvaja opravila v ozadju, pošilja potisna obvestila in shranjuje velike količine podatkov lokalno na napravi. (Damir, 2025)

8.2.2 Next.js PWA

Dostop do strojne opreme je zelo omejen in odvisen od brskalnika, ki ga uporabnik uporablja. PWA se večinoma izklopijo, kadar se ne uporabljajo, zato opravila v ozadju niso mogoča. Pošiljanje potisnih obvestil ni mogoče na vseh operacijskih sistemih, večinoma le na novejših. Shranjevanje velike količine podatkov ni mogoče, saj je PWA omejena s kapaciteto brskalnika. (Damir, 2025)

8.3 Funkcionalne razlike za razvijalce

Aplikacijo Flutter je mogoče namestiti iz trgovin Google Play ali App Store. Obe trgovini sta plačljivi in imata svoja merila. PWA je dostopna prek URL-ja, prav tako jo je mogoče dodati na začetni zaslon. Tako se razvijalec izogne stroškom trgovin z aplikacijami, zlasti če uporablja brezplačen paket gostovanja, kot je mogoče na platformi Vercel. (Vercel, 2025)

Tabela 5: Primerjava stroškov različnih platform za aplikacije

	Cena
Google Play	25 $
App Store	99 $ na mesec
Vercel (Hobby plan)	0 $

(Vir: https://appradar.com/blog/google-play-apple-app-store-fees)

Načeloma je posodobitev Next.js PWA lažja in hitrejša, zlasti če se uporablja platforma kot Vercel, ki samodejno zazna spremembe v repozitoriju in znova zgradi projekt. Pri aplikaciji Flutter je treba za vsako spremembo objaviti novo različico v trgovini z aplikacijami.

Razvoj Next.js PWA zna biti preprostejši, saj je osnova jezik JavaScript, ki se uporablja povsod v spletnih aplikacijah. Menim, da je spletnemu razvijalcu bistveno lažje ustvariti PWA kot *native* aplikacijo. Prehod na Flutter zna biti zahtevnejši, saj je Dart zelo specifičen jezik, ki se drugje kot v Flutterju skoraj ne uporablja. Po drugi strani pa ima Flutter veliko uporabnih orodij za gradnjo mobilnih aplikacij, ki so v pomoč razvijalcem.

Testiranje v okolju Flutter vključuje emulacijo različnih naprav in platform. Pri PWA je testiranje mogoče neposredno v brskalniku z orodji za razvijalce, pri tem ni treba nameščati aplikacije na različnih napravah. (Damir, 2025)

9 VARNOSTNA ANALIZA

9.1 Kibernetska varnost aplikacij

Kibernetska varnost aplikacij se ukvarja z zaščito programske opreme in podatkov pred nepooblaščenim dostopom, zlorabo in napadi. Pri aplikacijah je varnost še posebej pomembna, saj pogosto obdelujejo osebne podatke, omogočajo avtentikacijo in dostopajo do strojne opreme. (Cisco, 2025)

9.2 Napad Man-in-the-Middle

MITM (angl. *Man-in-the-Middle*) napad se zgodi, ko napadalec prestreže komunikacijo med uporabnikom in strežnikom, ne da bi katera stran to zaznala. (Fortinet, 2025)

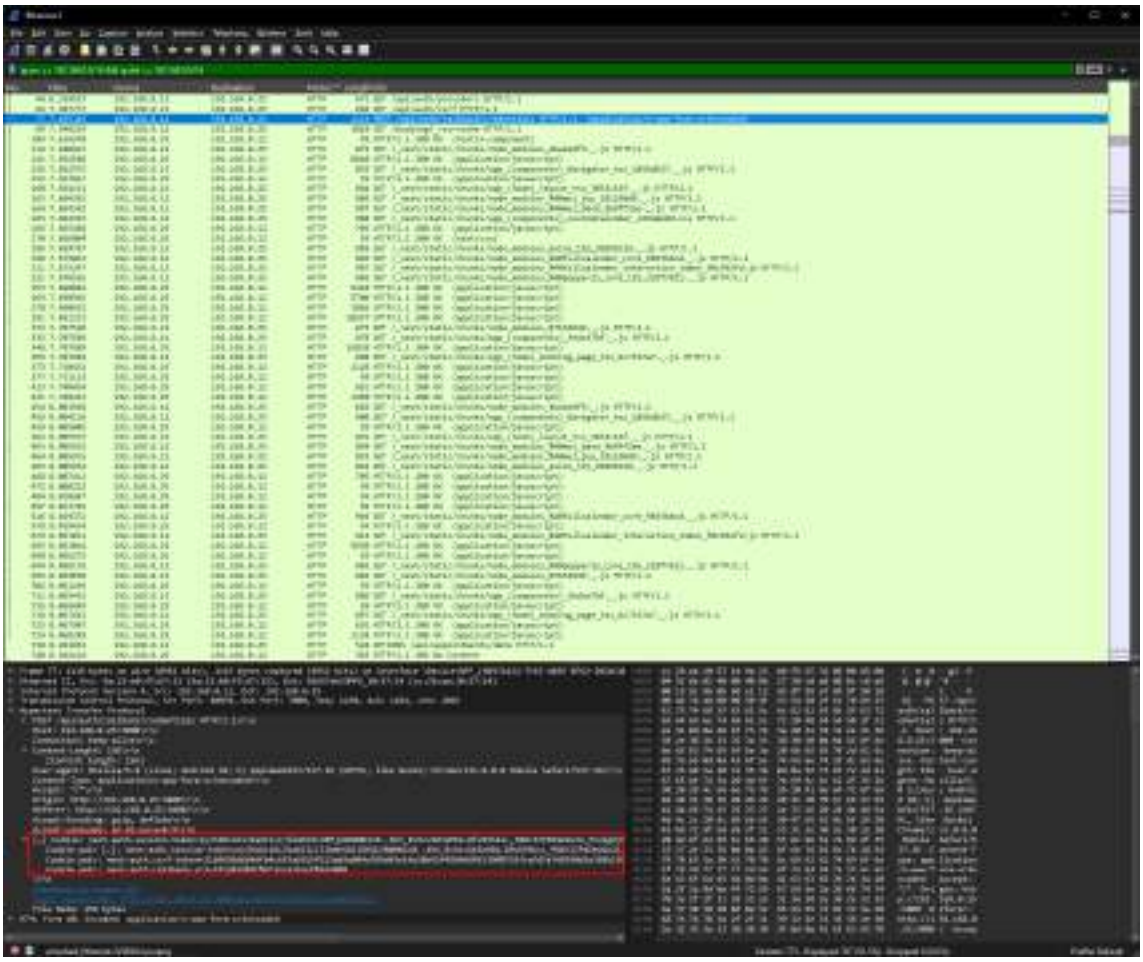
Ta napad bomo izvedli na obeh aplikacijah in primerjali učinkovitost. Za simulacijo napada bomo uporabili program Wireshark, ki analizira promet v omrežju. Promet v omrežju bo filtriran tako, da bo prikazan le tisti, kjer je pošiljatelj in prejemnik lokalna naprava. (Wireshark, 2025)

Testni scenarij v aplikaciji je:

- prijava v aplikacijo,
- ustvarjanje termina.

9.2.1 Napad na PWA

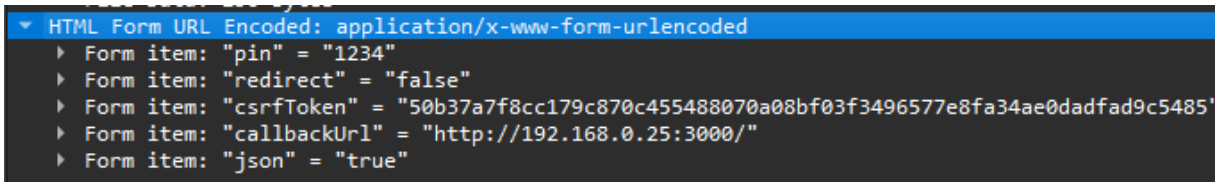
Na sliki 32 lahko vidimo POST HTTP-zahtevek za avtentikacijo z uporabo lastne Next.js API-poti. Wireshark je zaznal ta zahtevek in omogočil vpogled v podatke. Opazimo, da je viden piškotek za avtentikacijo next-auth.



Slika 32: Wiresharkova avtentikacija v PWA

(Vir: Lasten vir)

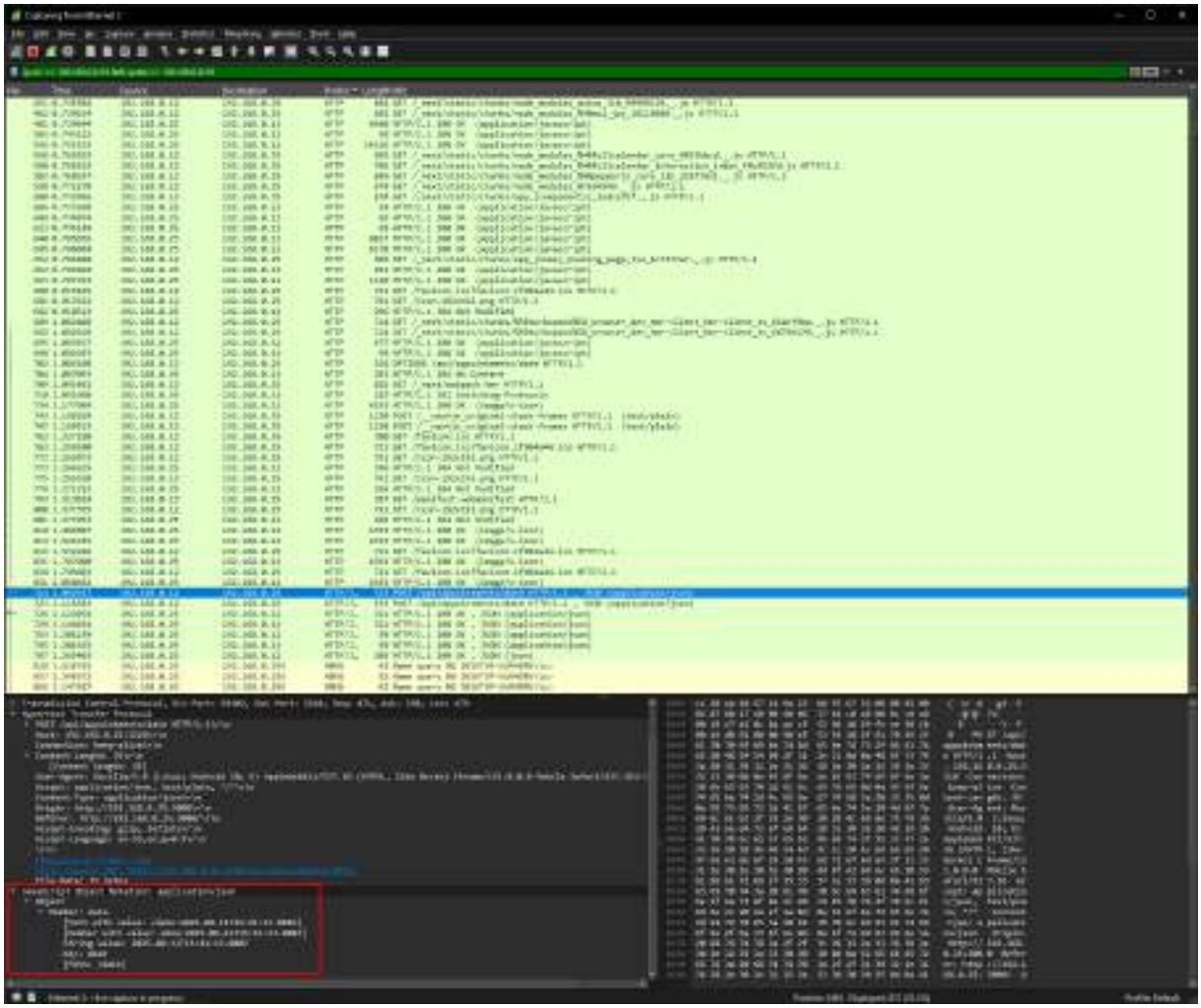
V HTML-formi je PIN, ki je prijavni podatek. S tem bi napadalec ugotovil način za prijavo v aplikacijo. Čeprav je tukaj prikazan PIN, bi bilo enako z elektronsko pošto in geslom.



Slika 33: Wiresharkovi prijavni podatki v PWA

(Vir: Lasten vir)

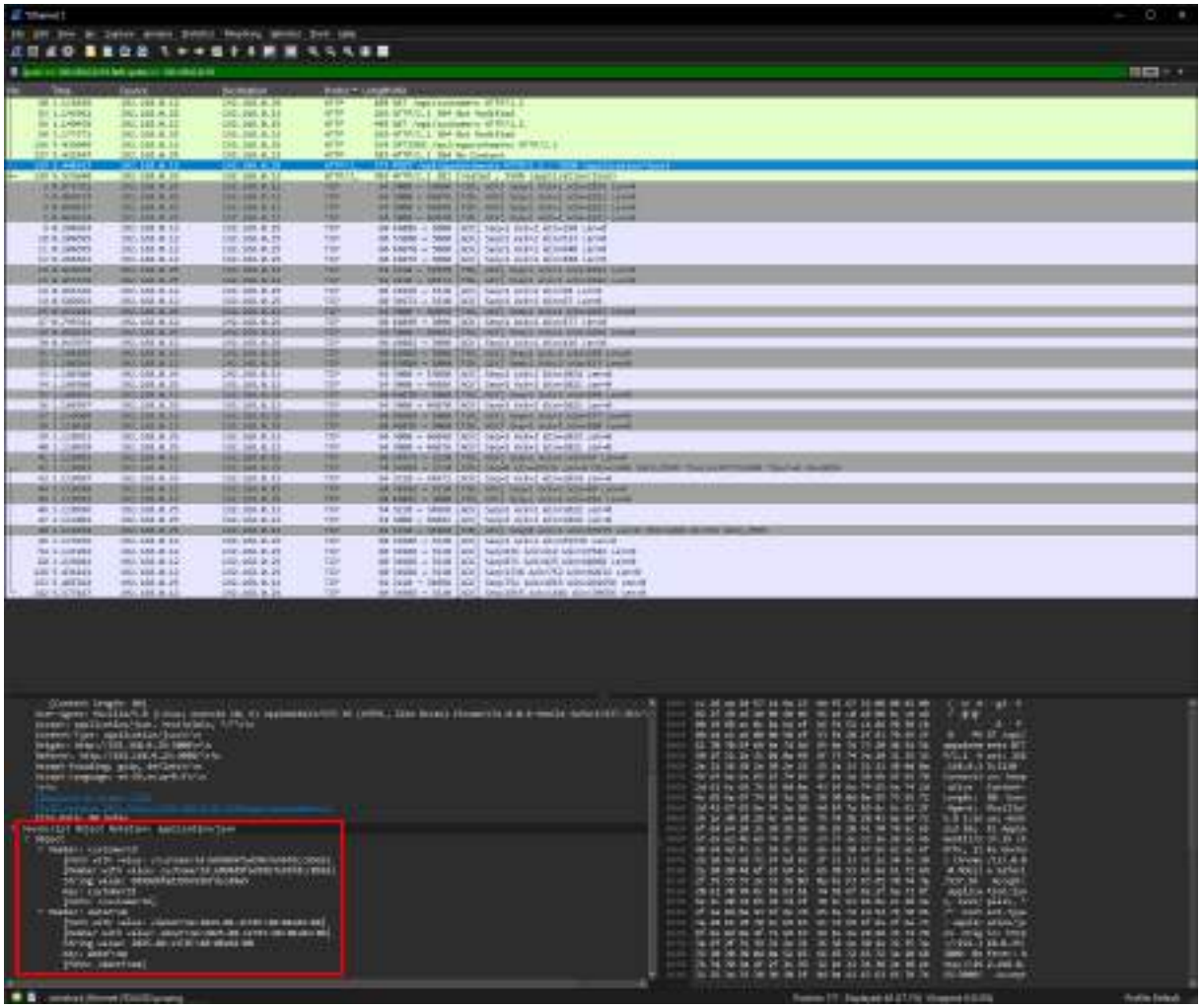
Na sliki 34 je prikazan HTTP-zahtevek za pridobitev terminov na določen datum. Tudi ta zahtevek je uspešno prestrežen, čeprav je v nasprotju z avtentikacijo tukaj uporabljen ločen strežnik.



Slika 34: Wiresharkova pridobitev terminov v PWA

(Vir: Lasten vir)

Prav tako se pri ustvarjanju termina lahko ugotovi ID (angl. *identifier*) stranke, kar zna biti uporaben podatek za napadalca.

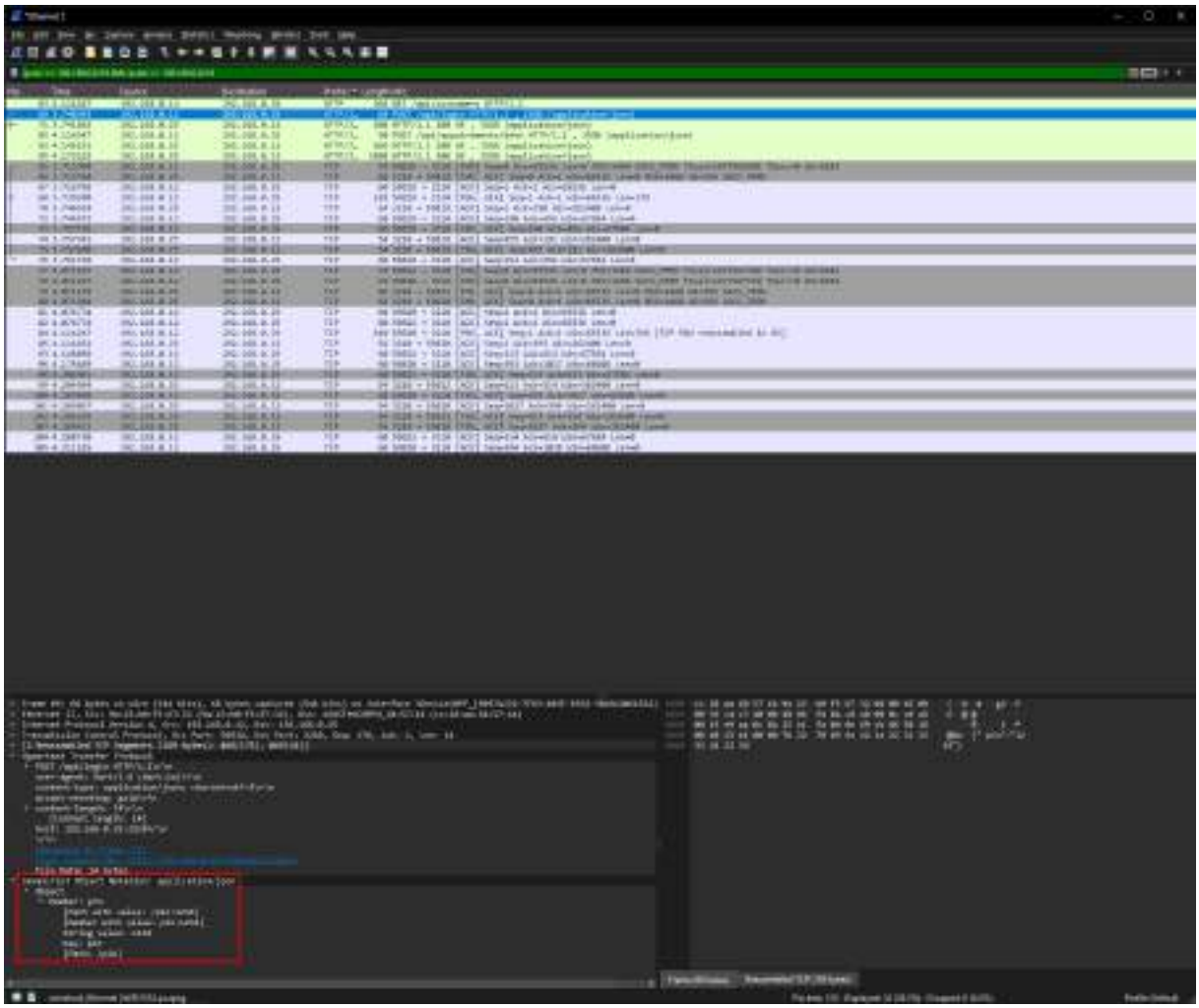


Slika 35: Wiresharkovo ustvarjanje termina v PWA

(Vir: Lasten vir)

9.2.2 Napad na aplikacijo Flutter

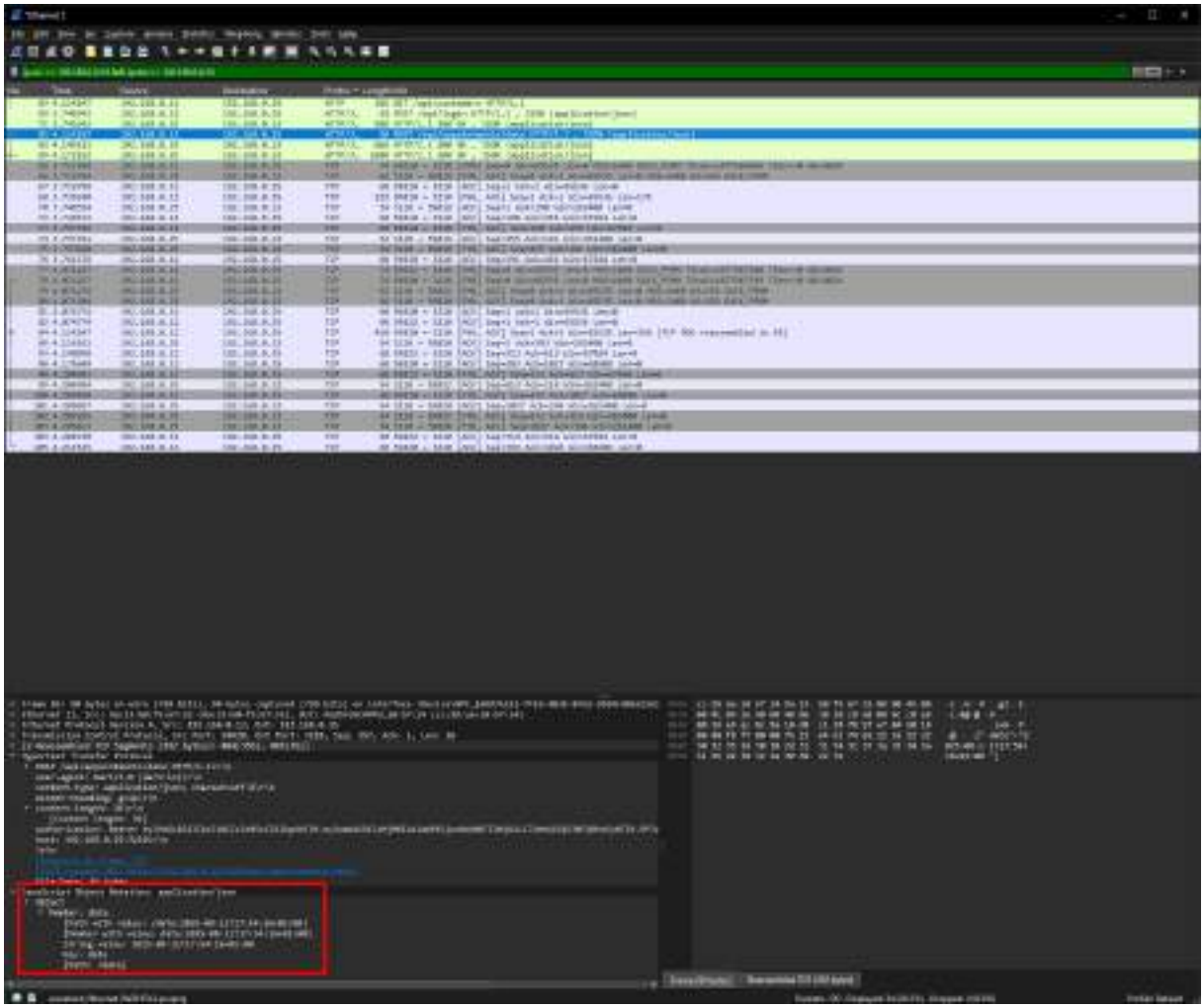
Pri aplikaciji Flutter ni nič drugače, čeprav uporablja ločen strežnik za avtentikacijo. Edina razlika je v tem, da se PIN prikaže na drugem mestu.



Slika: Wiresharkova avtentikacija v aplikaciji Flutter

(Vir: Lasten vir)

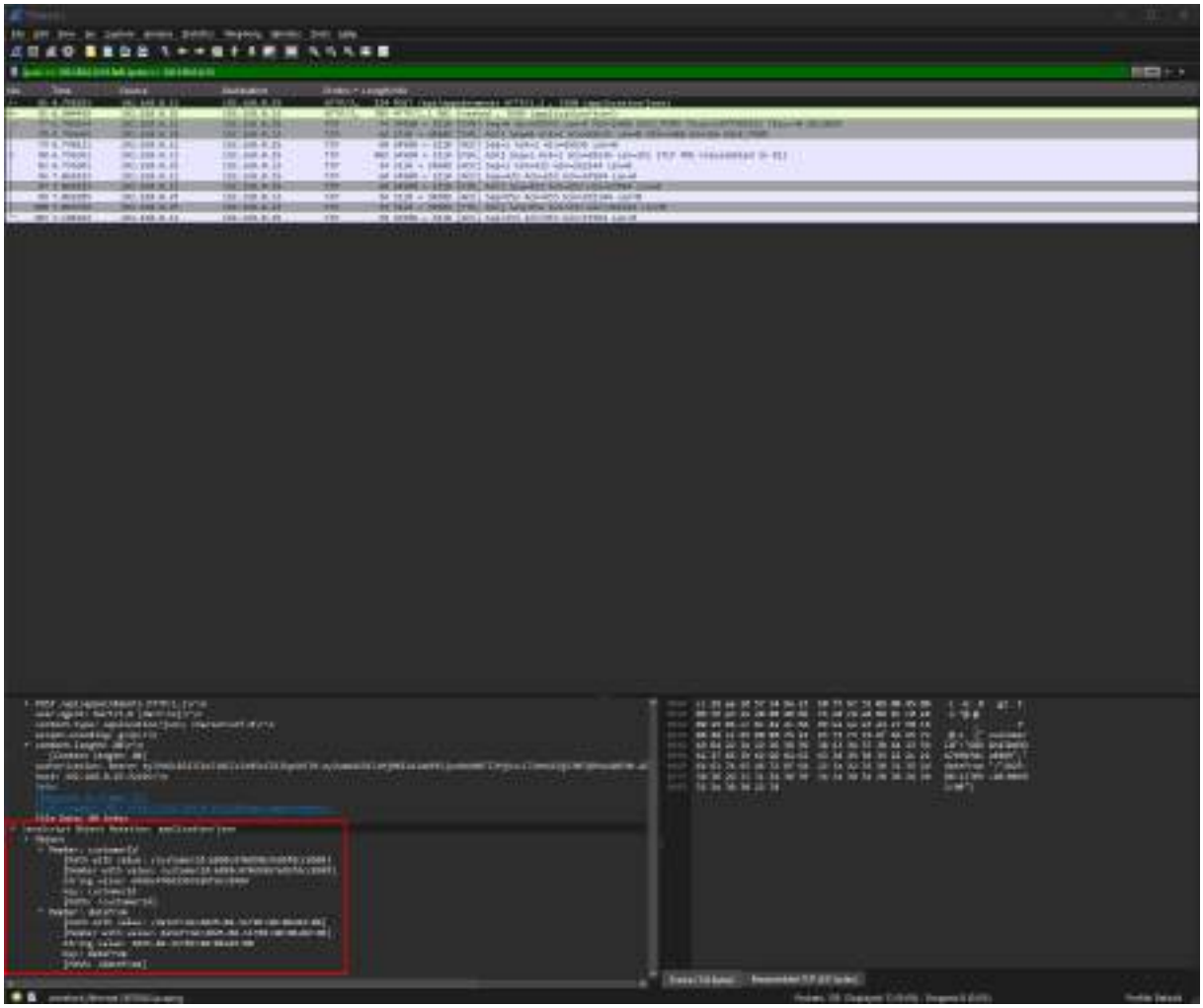
Tukaj je isti rezultat kot pri PWA, zahtevek za pridobitev terminov je jasno viden.



Slika 36: Wiresharkova pridobitev terminov v aplikaciji Flutter

(Vir: Lasten vir)

Zahtevek za ustvarjanje termina je viden tako kot pri PWA.



Slika 37: Wiresharkovo ustvarjanje termina v aplikaciji Flutter

(Vir: Lasten vir)

9.2.3 Zaščita pred napadom

Kot smo videli, izvedba napada ni bila zahtevna. Razlog je v tem, da se uporablja HTTP, ki je nezaščiten. Pri HTTP se podatki ne šifrirajo, temveč se pošiljajo po omrežju v navadni tekstovni obliki.

Če bi se želeli zaščititi pred takimi napadi, bi morali strežnik prilagoditi za uporabo HTTPS. V tem primeru bi napadalec bistveno težje naredil napad MITM. Nastaviti bi moral ponarejen TLS-certifikat in prisiliti napravo, da mu zaupa. (Fortinet, 2025)

9.3 Povratni inženiring

Povratni inženiring (angl. *Reverse Engineering*) je proces, kjer se poskuša pridobiti izvorna koda ali logika delovanja aplikacije. Tako lahko napadalci pridobijo občutljive podatke, razumejo logiko aplikacije in iščejo varnostne ranljivosti. (Siemens, 2025)

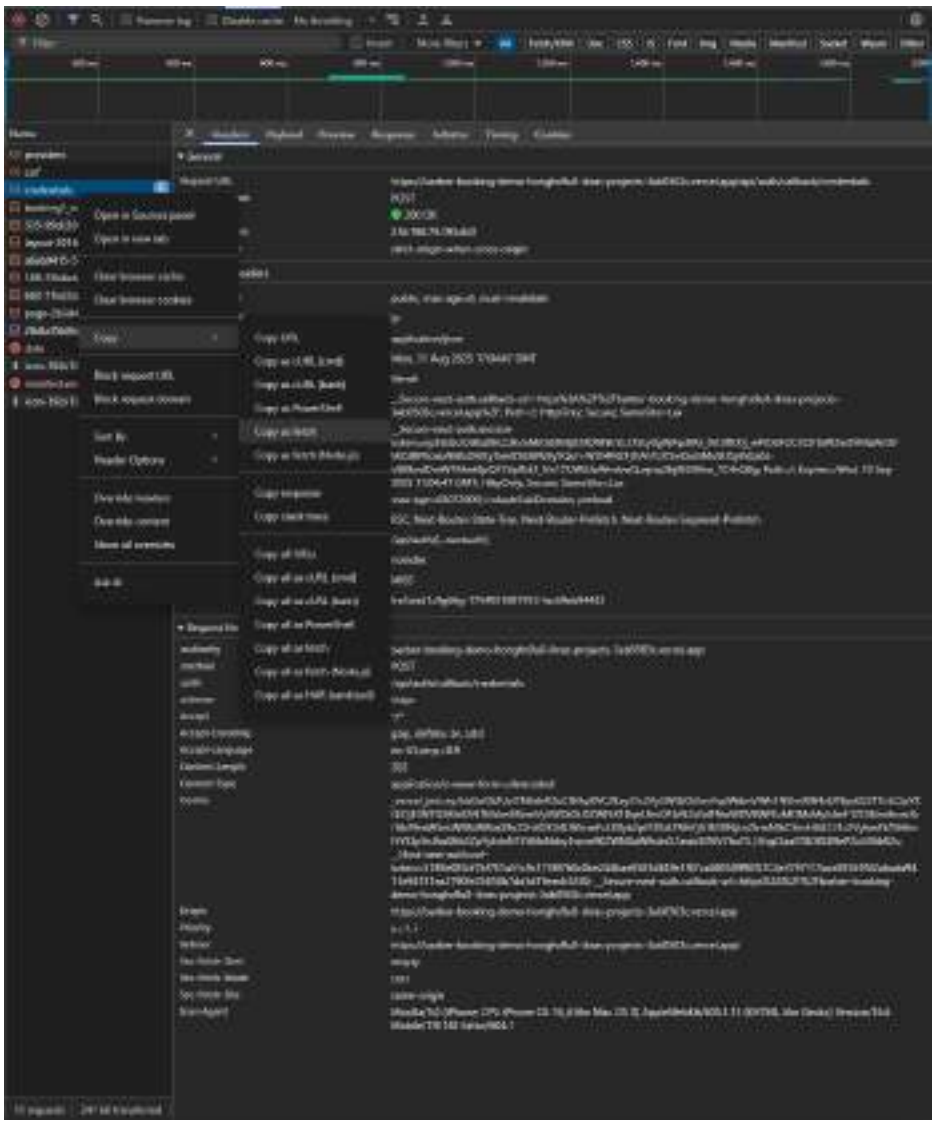
9.3.1 Next.js PWA

Če odpremo razvijalska orodja v brskalniku, lahko izberemo zavihek Sources, na katerem bomo našli kodo v JavaScriptu, ki je razdeljena po delih. Ta koda je pomanjšana in bistveno spremenjena od izvorne kode. Razbiranje logike bo zelo težko, vendar je v teoriji mogoče.

Slika 38: Pomanjšana koda Next.js PWA

(Vir: Lasten vir)

Naslednja stvar, ki jo lahko naredimo, je kopiranje HTTP-zahtevkov. Ponovno odpremo razvijalska orodja in zavihek Network. Izberemo HTTP-zahtevek in izberemo »Copy as fetch«.



Slika 39: Kopiranje HTTP-zahtevkov Next.js PWA

(Vir: Lasten vir)

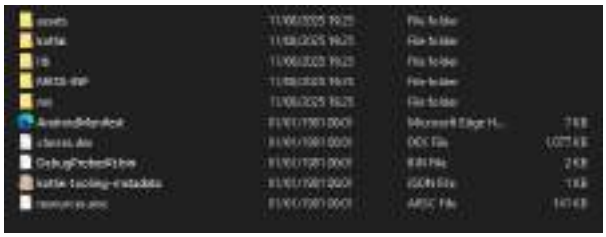
Ko to naredimo, se bo v odložišče prekopirala spodnja koda. To je JavaScript funkcija *fetch*, ki naredi HTTP-zahtevek. Tako imamo kodo z vsemi podrobnostmi.

```
fetch("https://barber-booking-demo-honghx9u8-ibras-projects-
3ab0503c.vercel.app/api/auth/callback/credentials", {
  "headers": {
    "accept": "*/*",
    "accept-language": "en-US,en;q=0.9",
    "content-type": "application/x-www-form-urlencoded",
    "priority": "u=1, i",
    "sec-fetch-dest": "empty",
```

```
    "sec-fetch-mode": "cors",
    "sec-fetch-site": "same-origin"
  },
  "referrer": "https://barber-booking-demo-honghx9u8-ibras-projects-
3ab0503c.vercel.app/",
  "body":
"pin=1234&redirect=false&csrfToken=5189e083cf7bf701a51c9c175f0768c0be244bae058
1d849e1f87ca68058ff98&callbackUrl=https%3A%2F%2Fbarber-booking-demo-honghx9u8-
ibras-projects-3ab0503c.vercel.app%2F&json=true",
  "method": "POST",
  "mode": "cors",
  "credentials": "include"
});
```

9.3.2 Aplikacija Flutter

Postopek pri aplikaciji Flutter je nekoliko drugačen. Najprej je treba pridobiti APK (angl. *Android Package Kit*). To se na sistemu Android naredi z ukazom adb pull, s čimer mu podamo pot do aplikacije. APK je stisnjena datoteka, zato jo moramo razširiti. Dobimo datoteke, prikazane na sliki 40.



Slika 40: Razširjena APK-datoteka

(Vir: Lasten vir)

Če odpremo mapo lib in izberemo katerokoli mapo, bo tam datoteka libapp.so. Ker se koda v Dartu prevede, je del nje v tej datoteki. Prevajanje te kode ponovno v kodo v Dartu je praktično nemogoče.

Tukaj lahko najdemo tudi konfiguracijske JSON-datoteke, slike, fonte, ikone ipd.

9.3.3 Zaščita

V teoriji se je nemogoče popolnoma zaščititi pred napadom s povratnim inženiringom. Vendar obstajajo tehnike, s katerimi je mogoče napadalcu otežiti delo. Prav tako je treba premisliti, ali je sploh upravičeno vlagati čas in trud v zaščito proti temu napadu.

Aplikacijo Flutter je mogoče zaščititi z uporabo metode *obfuscation*, ki naredi binarno kodo aplikacije neberljivo. Simbole zamenja z drugimi simboli. (Flutter, 2025)

Next.js PWA je spletna aplikacija, kar jo bolj izpostavlja temu napadu, ker je koda poslana uporabnikovemu brskalniku. Zaradi tega jo je lažje pregledati in analizirati. Next.js samodejno poskrbi za pomanjšanje kode, kar pomaga pri zmanjšanju berljivosti. Vendar je to še vedno koda v JavaScriptu, ki je bolj berljiva od *native* kode.

10 SKLEP

Z diplomskim delom smo izvedli primerjavo dveh sodobnih tehnologij za razvoj uporabniških aplikacij: Next.js in Flutter. Pri tem smo se predvsem osredotočili na mobilne aplikacije in primerjavo med PWA ter večplatformskimi aplikacijami. Namen raziskave in primerjave je bil ugotoviti, kako se tehnologiji razlikujeta z vidika uporabniške izkušnje, zmogljivosti, razvoja, varnosti, priljubljenosti in podpore skupnosti.

Analiza skupnosti in podpore je pokazala, da sta obe tehnologiji med razvijalci zelo priljubljeni, le da Next.js nekoliko zaostaja zaradi svoje specifičnosti. Obe tehnologiji imata dobro spisano dokumentacijo, številne uporabnike in razvijalce ter aktivne razvijalske forume. Vse to lajša učenje in reševanje težav.

Testiranje zmogljivost je pokazalo, da aplikacija Flutter zasede manj pomnilnika, medtem ko Next.js PWA hitreje izvaja HTTP-zahtevke.

Primerjava uporabniške izkušnje in tehnoloških značilnosti je pokazala, da ima Next.js PWA številne prednosti pri dostopnosti, preprosti distribuciji in gostovanju ter integraciji z obstoječimi spletnimi rešitvami. Flutter pa ponuja bolj konsistentno uporabniško izkušnjo na različnih platformah in lažji dostop do strojne opreme ter *native* funkcionalnosti mobilnih naprav. Prav tako je očitno, da je PWA zelo omejen na brskalnik in operacijski sistem mobilne naprave.

Analiza funkcionalnih razlik za razvijalce pokaže, da moramo dobro premisliti, katero tehnologijo in kateri pristop uporabiti. Eno izmed meril je lahko objava aplikacije. Flutter zahteva objavo aplikacije prek trgovin za aplikacije, kar prinaša dodatne stroške razvoja. PWA tega ne zahteva, dostopna je neposredno prek URL-ja.

Prav tako je vzdrževanje Next.js PWA lažje in hitrejše, zlasti pri uporabi sodobnih platform za gostovanje, kot je Vercel, ki samodejno znova zgradi projekt. Pri Flutterju bo to nekoliko težje, saj je za čisto vsako spremembo treba narediti novo različico in jo objaviti v obeh trgovinah z aplikacijami.

Spletni razvijalci bodo lažje pristopili k razvoju PWA zaradi znanega programskega jezika JavaScript in preprostejšega testiranja v brskalniku. Flutter pa zahteva posebno znanje jezika Dart, ki se ne uporablja pogosto. Prav tako je testiranje nekoliko težje, vendar je nabor orodij za mobilni razvoj zelo dober in je razvijalcem v veliko pomoč.

H1: Flutter je med razvijalci bolj razširjen kot Next.js za razvoj aplikacij.

Hipotezo lahko potrdimo. Po analizi skupnosti, virov, tečajev in podpore lahko opazimo, da je Flutter bolj priljubljen med razvijalci. Flutter ima boljše statistike na platformah GitHub, Stack Overflow in Reddit. Prav tako je na neuradnih tečajih bolje ocenjen. Razlog je najverjetneje v tem, da je bolj splošna tehnologija v primerjavi z Next.js, ki temelji na knjižnici React in je bolj specifičen.

H2: Razvoj mobilne aplikacije s Flutterjem omogoča boljšo zmogljivost in uporabniško izkušnjo kot Next.js PWA.

Hipotezo lahko potrdimo. Flutter uporablja lasten grafični mehanizem Skia, ki zagotavlja enoten in odziven uporabniški vmesnik z gladkimi animacijami, neodvisno od platforme. PWA pa je odvisen od zmogljivosti brskalnika, kar lahko vpliva na animacije. Flutter omogoča neposreden dostop do strojne opreme, medtem ko je PWA na teh področjih omejen zaradi brskalnika. Flutter porabi manj pomnilnika, kar izboljšuje zmogljivost. Next.js PWA je boljši na področju izvajanja HTTP-zahtevkov, na katerem je hitrejši, kar pospeši prenos podatkov.

H3: Next.js PWA je bolj ranljiv za napad *Man-in-the-Middle* kot Flutter.

Hipotezo lahko zavržemo. Varnostna analiza in simulacija napada kažeta, da sta obe tehnologiji enako ranljivi za napade MITM. Učinkovitost napada je bila veliko bolj odvisna od zalednega dela, v katerem so se uporabljale nezavarovane HTTP-povezave. Odpornost obeh aplikacij bi lahko povečali z uporabo HTTPS.

H4: Aplikacija Flutter je bolj odporna proti napadom s povratnim inženiringom (angl. *Reverse Engineering)* kot Next.js PWA.

Hipotezo lahko potrdimo. Glede na to, da Next.js PWA teče v brskalniku, je lažje dostopen z uporabo razvijalskih orodij. Koda je pomanjšana, vendar je še vedno koda v JavaScriptu. Koda v Dartu, ki ga uporablja Flutter, se prevede v *native* kodo, ki je bistveno zahtevnejša za preoblikovanje in branje. Prav tako je postopek pridobivanja kode daljši, ker je treba pridobiti APK-datoteko, jo prenesti na računalnik in razširiti.

11 VIRI IN LITERATURA

Alpondith. (10. avgust 2025). *The History of Flutter : From Origins to Global Adoption*. Pridobljeno iz Medium: https://alpondith.medium.com/the-history-of-flutter-from-origins-to-global-adoption-ddd5b1873dd3

Apple. (10. avgust 2025). *Apple Reinvents the Phone with iPhone*. Pridobljeno iz Apple: https://www.apple.com/newsroom/2007/01/09Apple-Reinvents-the-Phone-with-iPhone/

Apple. (27. Avgust 2025). *Choosing a Membership*. Pridobljeno iz Apple Developer: https://developer.apple.com/support/compare-memberships/

Axios. (10. avgust 2025). *Getting Started*. Pridobljeno iz Axios: https://axios-http.com/docs/intro

BasuMallick, C. (10. avgust 2025). *What Is Android OS? History, Features, Versions, and Benefits*. Pridobljeno iz Spiceworks: https://www.spiceworks.com/tech/tech-general/articles/android-os/

Bevans, D. (10. avgust 2025). *Asynchronous vs. Synchronous Programming: Key Similarities and Differences*. Pridobljeno iz Mendix: https://www.mendix.com/blog/asynchronous-vs-synchronous-programming/

BrowserStack. (10. avgust 2025). *What is Flutter: Definition, Benefits, and Limitations*. Pridobljeno iz BrowserStack: https://www.browserstack.com/guide/what-is-flutter

Cezar, F. (10. avgust 2025). *Understanding All Types of Page Rendering in Next.js*. Pridobljeno iz DEV Community: https://dev.to/felipecezar01/understanding-all-types-of-page-rendering-in-nextjs-1fbi

Cisco. (11. avgust 2025). *What is cybersecurity?* Pridobljeno iz Cisco: https://www.cisco.com/site/us/en/learn/topics/security/what-is-cybersecurity.html

Damir. (11. avgust 2025). *PWA vs. Native: Which Approach is Right for Your Project?* Pridobljeno iz COBE: https://www.cobeisfresh.com/blog/pwa-vs-native-which-approach-is-right-for-your-project#:~:text=Better%20experience%3A%20Unlike%20PWAs%2C%20native,apps%20have%20the%20upper%20hand.

Dart. (10. avgust 2025). *Asynchronous programming: futures, async, await*. Pridobljeno iz Dart: https://dart.dev/libraries/async/async-await

Dart. (10. avgust 2025). *Classes*. Pridobljeno iz Dart: https://dart.dev/language/classes

Dart. (10. avgust 2025). *Functions*. Pridobljeno iz Dart: https://dart.dev/language/functions

Dart. (10. avgust 2025). *Loops*. Pridobljeno iz Dart: https://dart.dev/language/loops

Dart. (10. avgust 2025). *Operators*. Pridobljeno iz Dart: https://dart.dev/language/operators

Dart. (10. avgust 2025). *The pubspec file*. Pridobljeno iz Dart: https://dart.dev/tools/pub/pubspec

Day.js. (10. avgust 2025). *Day.js*. Pridobljeno iz Day.js: https://day.js.org/

Dhaduk, H. (10. avgust 2025). *Mobile Application Architecture: Layers, Types, Principles, Factors*. Pridobljeno iz Simform: https://www.simform.com/blog/mobile-application-architecture/

Farihatul, M. (10. avgust 2025). *What is code splitting in Next.js? How does it improve performance?* Pridobljeno iz Medium: https://medium.com/@farihatulmaria/what-is-code-splitting-in-next-js-how-does-it-improve-performance-bccd4c8eda58

Flutter. (10. avgust 2025). *Flutter architectural overview*. Pridobljeno iz Flutter: https://docs.flutter.dev/resources/architectural-overview

Flutter. (11. avgust 2025). *Obfuscate Dart code*. Pridobljeno iz Flutter: https://docs.flutter.dev/deployment/obfuscate

Flutter. (10. avgust 2025). *Start building Flutter Android apps on Windows*. Pridobljeno iz Flutter: https://docs.flutter.dev/get-started/install/windows/mobile

Flutter. (10. avgust 2025). *Widget catalog*. Pridobljeno iz Flutter: https://docs.flutter.dev/ui/widgets

Flutter. (10. avgust 2025). *Widget class*. Pridobljeno iz Flutter: https://api.flutter.dev/flutter/widgets/Widget-class.html#:~:text=Widgets%20are%20the%20central%20class,their%20fields%20must%20be%20final).

Fortinet. (11. avgust 2025). *Man-in-the-Middle Attack: Types And Examples*. Pridobljeno iz Fortinet: https://www.fortinet.com/resources/cyberglossary/man-in-the-middle-attack

FullCalendar. (10. avgust 2025). *FullCalendar*. Pridobljeno iz FullCalendar: https://fullcalendar.io/

GeeksforGeeks. (10. avgust 2025). *Component-Based Architecture - System Design*. Pridobljeno iz GeeksforGeeks: https://www.geeksforgeeks.org/system-design/component-based-architecture-system-design/

GeeksforGeeks. (10. avgust 2025). *Difference Between Virtual DOM and Real DOM*. Pridobljeno iz GeeksforGeeks: https://www.geeksforgeeks.org/reactjs/difference-between-virtual-dom-and-real-dom/

GeeksforGeeks. (10. avgust 2025). *MongoDB - Working and Features*. Pridobljeno iz GeeksforGeeks: https://www.geeksforgeeks.org/mongodb/what-is-mongodb-working-and-features/

GeeksforGeeks. (10. avgust 2025). *React Hooks*. Pridobljeno iz GeeksforGeeks: https://www.geeksforgeeks.org/reactjs/reactjs-hooks/

GeeksforGeeks. (10. avgust 2025). *React JSX*. Pridobljeno iz GeeksforGeeks: https://www.geeksforgeeks.org/reactjs/reactjs-jsx-introduction/

GeeksforGeeks. (10. avgust 2025). *What is an Operating System?* Pridobljeno iz GeeksforGeeks: https://www.geeksforgeeks.org/operating-systems/what-is-an-operating-system

GeeksforGeeks. (10. avgust 2025). *What is Single Page Application?* Pridobljeno iz GeeksforGeeks: https://www.geeksforgeeks.org/javascript/what-is-single-page-application/

GitHub. (11. avgust 2025). *flutter*. Pridobljeno iz GitHub: https://github.com/flutter/flutter

GitHub. (11. avgust 2025). *next.js*. Pridobljeno iz GitHub: https://github.com/vercel/next.js

Google. (27. avgust 2025). *Get started with Play Console*. Pridobljeno iz Play Console Help: https://support.google.com/googleplay/android-developer/answer/6112435?hl=en#zippy=%2Cstep-sign-up-for-a-play-console-developer-account%2Cstep-pay-registration-fee

History of ReactJS. (10. avgust 2025). Pridobljeno iz Great Learning: https://www.mygreatlearning.com/react-js/tutorials/history-of-reactjs

JetBrains. (10. avgust 2025). *What is cross-platform mobile development?* Pridobljeno iz JetBrains: https://www.jetbrains.com/help/kotlin-multiplatform-dev/cross-platform-mobile-development.html#cross-platform-mobile-development-definition-and-solutions

JSON. (10. avgust 2025). *Introducing JSON*. Pridobljeno iz JSON: https://www.json.org/json-en.html

jwt.io. (10. avgust 2025). *JSON Web Tokens*. Pridobljeno iz jwt.io: https://www.jwt.io/

King, M. (10. avgust 2025). *Types of Mobile Apps (2025)*. Pridobljeno iz Business of Apps: https://www.businessofapps.com/app-developers/research/types-of-mobile-apps/

LaunchSchool. (10. avgust 2025). *Introduction*. Pridobljeno iz LaunchSchool: https://launchschool.com/books/javascript/read/introduction

leerob. (11. avgust 2025). *leerob*. Pridobljeno iz leerob: https://leerob.com/

MDN. (10. avgust 2025). *JavaScript*. Pridobljeno iz MDN: https://developer.mozilla.org/en-US/docs/Web/JavaScript

Mobile Phone Museum. (10. avgust 2025). *BlackBerry 5810*. Pridobljeno iz Mobile Phone Museum: https://www.mobilephonemuseum.com/phone-detail/blackbarry-5810

mongoose. (10. avgust 2025). *Mongoose*. Pridobljeno iz Mongoose: https://mongoosejs.com/

MUI. (10. avgust 2025). *MUI*. Pridobljeno iz MUI: https://mui.com/

Newport, C. (2019). *Digital Minimalism.* Penguin Random House LLC.

Next.js. (10. avgust 2025). *Head*. Pridobljeno iz Next.js: https://nextjs.org/docs/pages/api-reference/components/head

Next.js. (10. avgust 2025). *How to use Next.js as a backend for your frontend*. Pridobljeno iz Next.js: https://nextjs.org/docs/app/guides/backend-for-frontend

Next.js. (10. avgust 2025). *Installation*. Pridobljeno iz Next.js: https://nextjs.org/docs/app/getting-started/installation

Next.js. (10. avgust 2025). *Next.js Compiler*. Pridobljeno iz Next.js: https://nextjs.org/docs/architecture/nextjs-compiler

Next.js. (10. avgust 2025). *Next.js Docs*. Pridobljeno iz Next.js: https://nextjs.org/docs

Next.js. (10. avgust 2025). *Next.js on Vercel*. Pridobljeno iz Next.js: https://vercel.com/docs/frameworks/full-stack/nextjs

Next.js. (10. avgust 2025). *Project structure and organization*. Pridobljeno iz Next.js: https://nextjs.org/docs/app/getting-started/project-structure

NextAuth.js. (10. avgust 2025). *NextAuth.js*. Pridobljeno iz NextAuth.js: https://next-auth.js.org/

Node.js. (10. avgust 2025). *Node.js*. Pridobljeno iz Node.js: https://nodejs.org/en

npm. (10. avgust 2025). *next-pwa*. Pridobljeno iz npm: https://www.npmjs.com/package/next-pwa

npm. (10. avgust 2025). *next-sitemap*. Pridobljeno iz npm: https://www.npmjs.com/package/next-sitemap

NSB App Studio. (10. avgust 2025). *Using Node and Electron to build Desktop Apps*. Pridobljeno iz NSB App Studio: https://wiki.appstudio.dev/Using_Node_and_Electron_to_build_Desktop_Apps

Nukumizu, S. (10. avgust 2025). *The History And Rules of Dart Language*. Pridobljeno iz Medium: https://medium.com/@author2000.1225/the-history-and-rules-of-dart-language-f25e09a58530

Obinna, O. (10. avgust 2025). *How does JIT and AOT work in Dart?* Pridobljeno iz Medium: https://onuoha.medium.com/how-does-jit-and-aot-work-in-dart-cab2f31d9cb5

OutSystems. (10. avgust 2025). *What are mobile applications? Getting started with mobile apps*. Pridobljeno iz OutSystems: https://www.outsystems.com/application-development/mobile-apps-overview

Pérez, P. (10. avgust 2025). *Flutter's Architecture Explained*. Pridobljeno iz Somnio Software: https://somniosoftware.com/blog/flutters-architecture-explained

pub.dev. (10. avgust 2025). *cupertino_icons*. Pridobljeno iz pub.dev: https://pub.dev/packages/cupertino_icons

pub.dev. (10. avgust 2025). *http*. Pridobljeno iz pub.dev: https://pub.dev/packages/http

pub.dev. (10. avgust 2025). *pub.dev*. Pridobljeno iz pub.dev: https://pub.dev/

pub.dev. (10. avgust 2025). *shared_preferences*. Pridobljeno iz pub.dev: https://pub.dev/packages/shared_preferences

pub.dev. (10. avgust 2025). *syncfusion_flutter_calendar*. Pridobljeno iz pub.dev: https://pub.dev/packages/syncfusion_flutter_calendar

Reddit. (11. avgust 2025). *r/FlutterDev*. Pridobljeno iz Reddit: https://www.reddit.com/r/FlutterDev/

Reddit. (11. avgust 2025). *r/nextjs*. Pridobljeno iz Reddit: https://www.reddit.com/r/nextjs/

Siemens. (11. avgust 2025). *Reverse engineering*. Pridobljeno iz Siemens: https://www.sw.siemens.com/en-US/technology/reverse-engineering/

Stack Overflow. (11. avgust 2025). *Newest 'flutter' Questions*. Pridobljeno iz Stack Overflow: https://stackoverflow.com/questions/tagged/flutter

Stack Overflow. (11. avgust 2025). *Newest 'next.js' Questions*. Pridobljeno iz Stack Overflow: https://stackoverflow.com/questions/tagged/next.js

Stack Overflow. (11. avgust 2025). *Newest 'react' Questions*. Pridobljeno iz Stack Overflow: https://stackoverflow.com/questions/tagged/react

Tailwind CSS. (10. avgust 2025). *Rapidly build modern websites without ever leaving your HTML*. Pridobljeno iz Tailwind CSS: https://tailwindcss.com/

Terrell Hanna, K., & Wigmore, I. (10. avgust 2025). *What is a mobile app (mobile application)?* Pridobljeno iz TechTarget: https://www.techtarget.com/whatis/definition/mobile-app

Turingvang. (10. avgust 2025). *Nextjs Image*. Pridobljeno iz Medium: https://medium.com/@turingvang/nextjs-image-2522985a5188

TypeScript. (10. avgust 2025). *TypeScript*. Pridobljeno iz TypeScript: https://www.typescriptlang.org/

Udemy. (11. avgust 2025). *Search results*. Pridobljeno iz Udemy: https://www.udemy.com/courses/search/?src=ukw&q=flutter&sort=relevance

Udemy. (11. avgust 2025). *Search results*. Pridobljeno iz Udemy: https://www.udemy.com/courses/search/?src=ukw&q=next.js&sort=relevance

Vercel. (16. avgust 2025). *Find a plan to power your projects.* Pridobljeno iz Vercel: https://vercel.com/pricing

Visual Studio Code. (10. avgust 2025). *Visual Studio Code*. Pridobljeno iz Visual Studio Code: https://code.visualstudio.com/

Volle, A. (10. avgust 2025). *iOS*. Pridobljeno iz Britannica: https://www.britannica.com/technology/iOS

W3Schools. (10. avgust 2025). *React Components*. Pridobljeno iz W3Schools: https://www.w3schools.com/react/react_components.asp

Wireshark. (11. avgust 2025). Pridobljeno iz Wireshark: https://www.wireshark.org/

Youtube. (11. avgust 2025). *Flutter*. Pridobljeno iz Youtube: https://www.youtube.com/@flutterdev/videos

Youtube. (11. avgust 2025). *leerob*. Pridobljeno iz Youtube: https://www.youtube.com/@leerob/videos

12 PRILOGE

Vsi repozitoriji ustvarjenih aplikacij so na platformi GitHub.