Progresívne webové aplikácie

Matúš Giertl

Bakalářská práce
2019

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky
akademický rok: 2018/2019

ZADÁNÍ BAKALÁŘSKÉ PRÁCE
(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení:	**Matúš Giertl**
Osobní číslo:	**A16080**
Studijní program:	**B3902 Inženýrská informatika**
Studijní obor:	**Softwarové inženýrství**
Forma studia:	**prezenční**
Téma práce:	**Progresivní webové aplikace**
Téma anglicky:	**Progressive Web Applications**

Zásady pro vypracování:

1. **Vypracujte srovnání nativního a webového přístupu k vývoji mobilních aplikací.**
2. **Definujte způsob vývoje aplikací označovaných jako Progresivní webové aplikace [PWA] a popište používané technologie, princip, výhody a nevýhody tohoto přístupu.**
3. **Porovnejte rozdíly ve vývoji PWA na platformách iOS a Android.**
4. **V rámci praktické části stanovte funkční a nefunkční požadavky na reálnou PWA, která by měla sloužit jako informační mobilní web pro uchazeče o studium na UTB.**
5. **Mimo základních webových technologií využijte pro tvorbu praktické aplikace knihovnu React.**

Rozsah bakalářské práce:

Rozsah příloh:

Forma zpracování bakalářské práce: **tištěná/elektronická**

Seznam odborné literatury:

1. **ATER, T., 2017. Building Progressive Web Apps: Bringing the Power of Native to the Browser. O'Reilly Media. ISBN 978-1491961650.**
2. **SHEPPARD, D. 2017. Beginning Progressive Web App Development: Creating a Native App Experience on the Web: Apress.**
3. **HUME, D. A., & OSMANI, A. 2017. Progressive Web Apps: Manning Publications. ISBN 1617294586.**
4. **Google Developers. Progressive Web Apps [online]. Dostupné z: https://developers.google.com/web/progressive-web-apps/**
5. **React Documentation. React: A JavaScript library for building user interfaces [online]. 2018 [cit. 2018-11-26]. Dostupné z: https://reactjs.org/docs/getting-started.html**
6. **KEITH, Jeremy. 2018. Going Offline. A Book Apart. ISBN 1937557650.**

Vedoucí bakalářské práce: **Ing. Radek Vala, Ph.D.**
Ústav informatiky a umělé inteligence

Datum zadání bakalářské práce: **3. prosince 2018**

Termín odevzdání bakalářské práce: **15. května 2019**

Ve Zlíně dne 7. prosince 2018

UNIVERZITA TOMÁŠE BATI VE ZLÍNĚ
Fakulta aplikované informatiky
L.S.

doc. Mgr. Milan Adámek, Ph.D.
děkan

prof. Mgr. Roman Jašek, Ph.D.
garant oboru

Prohlašuji, že

- beru na vědomí, že odevzdáním bakalářské práce souhlasím se zveřejněním své práce podle zákona č. 111/1998 Sb. o vysokých školách a o změně a doplnění dalších zákonů (zákon o vysokých školách), ve znění pozdějších právních předpisů, bez ohledu na výsledek obhajoby;
- beru na vědomí, že bakalářská práce bude uložena v elektronické podobě v univerzitním informačním systému dostupná k prezenčnímu nahlédnutí, že jeden výtisk diplomové/bakalářské práce bude uložen v příruční knihovně Fakulty aplikované informatiky Univerzity Tomáše Bati ve Zlíně a jeden výtisk bude uložen u vedoucího práce;
- byl/a jsem seznámen/a s tím, že na moji bakalářskou práci se plně vztahuje zákon č. 121/2000 Sb. o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon) ve znění pozdějších právních předpisů, zejm. § 35 odst. 3;
- beru na vědomí, že podle § 60 odst. 1 autorského zákona má UTB ve Zlíně právo na uzavření licenční smlouvy o užití školního díla v rozsahu § 12 odst. 4 autorského zákona;
- beru na vědomí, že podle § 60 odst. 2 a 3 autorského zákona mohu užít své dílo – diplomovou/bakalářskou práci nebo poskytnout licenci k jejímu využití jen připouští-li tak licenční smlouva uzavřená mezi mnou a Univerzitou Tomáše Bati ve Zlíně s tím, že vyrovnání případného přiměřeného příspěvku na úhradu nákladů, které byly Univerzitou Tomáše Bati ve Zlíně na vytvoření díla vynaloženy (až do jejich skutečné výše) bude rovněž předmětem této licenční smlouvy;
- beru na vědomí, že pokud bylo k vypracování bakalářské práce využito softwaru poskytnutého Univerzitou Tomáše Bati ve Zlíně nebo jinými subjekty pouze ke studijním a výzkumným účelům (tedy pouze k nekomerčnímu využití), nelze výsledky bakalářské práce využít ke komerčním účelům;
- beru na vědomí, že pokud je výstupem bakalářské práce jakýkoliv softwarový produkt, považují se za součást práce rovněž i zdrojové kódy, popř. soubory, ze kterých se projekt skládá. Neodevzdání této součásti může být důvodem k neobhájení práce.

Prohlašuji,

- že jsem na bakalářské práci pracoval samostatně a použitou literaturu jsem citoval. V případě publikace výsledků budu uveden jako spoluautor.
- že odevzdaná verze bakalářské práce a verze elektronická nahraná do IS/STAG jsou totožné.

Ve Zlíně, dne …….

podpis diplomanta

ABSTRAKT

Práca sa zaoberá analýzou a implementáciou aplikácie pre uchádzačov o štúdium na Univerzite Tomáše Bati ve Zlíně. Výsledná aplikácia úspešne plní svoju funkciu poskytovania dôležitých informácii pre potencionálnych budúcich študentov. Zároveň bola vyvinutá ako overenie konceptu Progresívnych webových aplikácii a implementovaná pomocou najpopulárnejšej knižnice pre tvorbu webových aplikácii – React.

Kľúčové slová: React, JavaScript, Progresívne webové aplikácie, PWA, Firebase, Android, iOS, web, vývoj mobilných aplikácii, hybridný vývoj, Atomický dizajn

ABSTRACT

This thesis deals with the analysis and implenetation of an application, which serves as an information source for TBU applicants. The resulting application is able to deliver on its functionality, which is to inform potential future students. It was also developed as a proof of concept for the technology of Progressive Web Apps and was implemented via the most populat library for creation of web applications - React

Keywords: React, JavaScript, Progressive Web Apps, PWA, Firebase, Android, iOS, web, mobile app development, hybrid development, Atomic Design

Významné poďakovanie patrí vedúcemu mojej práce pánovi Ing. Radkovi Valovi, Ph.D., vďaka ktorému bolo možné z pôvodnej myšlienky overenia konceptu vytvoriť okamžite nasaditeľnú aplikáciu, ktorá má využitie v reálnom svete. Veľká vďaka patrí taktiež za vedenie a podporu v priebehu písania práce. Jeho dôležité rady a pripomienky mi výrazne pomohli v skvalitnení tejto práce.

Ďalej by som sa chcel poďakovať Mgr. Marekovi Jakubovi z marketingového oddelenia univerzity za obsah aplikácie a Bc. Markéte Fajovej za poskytnutý dizajn.

V neposlednom rade patrí vďaka mojim rodičom a priateľke za poskytnutie domácej pohody, príjemného prostredia a podpory počas tvorby práce.

OBSAH

ÚVOD

Propagácia je dôležitou súčasťou marketingu každej univerzity. Ak je vykonávaná dobre, dokáže efektívne presvedčiť uchádzača o podanie prihlášky práve na konkrétnu fakultu. Pod túto kategóriu spadajú rôzne reklamné materiály, dni otvorených dverí, návštevy stredných škôl a mobilné aplikácie, ktoré verejnú mienku ovplyvňujú pravdepodobne najviac.

Jednou z takýchto propagačných „pomôcok“ bola aplikácia „Studuj UTB“, dostupná na všetky významné mobilné platformy. V priebehu času však prestávala plniť svoju primárnu funkciu, ktorou bolo poskytovanie informácii o univerzite a jej rôznych súčastiach nehovoriac o jej zastaralom vzhľade. Výsledkom teda bola zastaralá a neudržiavateľná aplikácia, ktorá poškodzovala vzhľad univerzity ako takej.

Cieľom práce je riešiť tento propagačný problém tvorbou spoľahlivej webovej aplikácie zameranej primárne pre mobilné zariadenia za pomoci moderných webových technológii, implementácie Progresívnej webovej aplikácie a moderného vzhľadu. Jej primárnou úlohou je teda zoznámenie užívateľa s univerzitou a poskytnutie relevantných informácii, ktoré by mohli potencionálneho uchádzača o štúdium zaujímať.

I. TEORETICKÁ ČASŤ

1 VÝVOJ MOBILNÝCH APLIKÁCII

Pod týmto pojmom je možné predstaviť si najmä vývoj aplikácii pre smartfóny, no v dnešnej dobe to nie je tak úplne pravda. Rapídny vývoj technológii v poslednej dekáde znamenal aj príchod nových kategórii "chytrých" zariadení, ako sú napríklad hodinky, multimediálne systémy v autách a tablety. Je možné teda povedať, že pod mobilný vývoj spadá každé zariadenie, ktoré obsahuje mobilný operačný systém [1].

Najrozšírenejším mobilným operačným systémom je Android, open source operačný systém pod záštitou Google, ktorý si môže každý výrobca nainštalovať do svojho zariadenia a, zväčša s ďalšími úpravami pre odlíšenie značky, predávať [2] .

Druhý najrozšírenejší mobilný operačný systém nesie názov iOS z dielne Apple. V čase písania je nainštalovaný na zhruba 23% zariadení (jeho priamy konkurent Android je nainštalovaný na 73% zariadení). Hlavná odlišnosť tohto OS je filozofická. Na rozdiel od Androidu je iOS uzatvorený a je ho možné nájsť iba vo smartfónoch a tabletoch značky Apple [2].

Existuje skutočne pestrá škála systémov a zariadení, ktoré tieto systémy používajú a nie vždy je jednoduché zvoliť správny vývojový prístup. Rozhodovanie samotné ovplyvňuje množstvo faktorov, medzi ktoré je možné najčastejšie zaradiť či už zákazníkov, rozpočet, alebo rýchlosť vývoja a čas potrebný k vydaniu aplikácie na trh. Môžeme si zvoliť natívny, multiplatformový alebo webový prístup k vývoju. Každý prístup so sebou prináša svoje výhody aj nevýhody a je len na zákazníkovi (alebo vývojárovi), ktorý spôsob si vyberie na základe jeho priorít [3].

1.1 Natívny vývoj

Natívne vyvinutá aplikácia je vo svojej podstate program vyvinutý v jazyku navrhnutom tak, aby zo zariadenia vyťažil najvyšší možný výkon. Mnoho ľudí preferuje natívny vývoj pred hociktorým iným. To však znamená, že pre každú platformu je nutné sa naučiť natívny jazyk. Pre Android je to *Java* alebo *Kotlin*, pre iOS je to Objective-C alebo Swift. Znalosť natívneho jazyka pre platformu je síce najlepšia možnosť, ako dosiahnuť najoptimálnejšieho výkonu a kvalitného užívateľského zážitku, avšak to so sebou nesie významnú nevýhodu v podobe času nutného pre vývoj. Ak je potrebné priniesť aplikáciu na viac ako jednu platformu, je vždy nutné naučiť sa rôzne natívne jazyky, čo výrazne predlžuje čas potrebný k vydaniu aplikácie. Je nutné však vziať ohľad aj na nutnú údržbu aplikácie po jej vydaní.

Tento proces je nielen časovo, ale i peňažne náročný. Nielenže je nutné naučiť sa rôzne jazyky, ale aj naučiť sa pracovať s rôznymi vývojovými nástrojmi a platformami, ktoré sa neustále vyvíjajú. Obrovskou výhodou je prísun nových funkcionalít, ktoré sú okamžite dostupné po vydaní novej verzie SDK [3].

1.1.1 Android

Natívne aplikácie pre Android je možné vyvíjať v Jave, C++ a najnovšie pridanom Kotline. Kód je skompilovaný pomocou Android SDK spolu so všetkými dátami a zdrojmi do jediného APK súboru, tzv. Android Package, čo je archivovaný súbor s .apk príponou. Tento súbor v sebe obsahuje všetky súčasti aplikácie a zariadenie ho používa k inštalácii aplikácie samotnej [3].

OS Android je multi-užívateľský operačný systém, v ktorom každá aplikácia je považovaná za jedinečného užívateľa. Každá spustená aplikácia beží vo vlastnom virtuálnom prostredí, chránená viacerými bezpečnostnými prvkami. Systém taktiež implementuje tzv. principle of least privilege. To znamená, že každá aplikácia má predvolene prístup iba ku komponentom, ktoré vyžaduje pre jej spustenie. Bezpečnostnou výhodou tohto prístupu je, že aplikácia sa nemôže dostať ku častiam systému, do ktorých nemá prístup. Môže však nastať situácia, kedy je potrebné, aby spolu určité aplikácie vedeli komunikovať a zdieľať dáta. Toto sa dá docieliť zdieľaným Linux user ID, avšak aplikácie musia byť podpísané rovnakým certifikátom. Aplikácia si taktiež dokáže požiadať o právomoci. Toto sa deje buď pri prvom spustení aplikácie, alebo ak je zvolená funkcionalita aplikácie, na ktorú nemá povolenie, ako napríklad fotoaparát. Medzi ďalšie prístupové právomoci patria napr. užívateľove kontakty, čítanie a zápis do externej pamäte, kamera a Bluetooth. Tieto právomoci udeľuje priamo užívateľ [4].

Práve vďaka svojej open-source povahe je však Android veľmi fragmentovaný. Najnovšiu verziu 8.1 má nainštalovaných zhruba 7.5% všetkých zariadení a najpoužívanejšou verziou je 6.0 Marshmallow s 21.3% [5].

Distribúcia Android aplikácii je primárne pomocou platformy Play Store, ktorá vývojárom umožňuje publikovať, predávať a distribuovať svoju Android aplikáciu užívateľom. Play Store po publikácii aplikácie ponúka množstvo nástrojov zameraných na marketing, metriky a speňaženie. Taktiež je možné kontrolovať, do akých krajín je aplikácia distribuovaná [6].

Android však umožňuje inštaláciu aplikácii aj z iných zdrojov. Druhým najpopulárnejším obchodom je Amazon App Store. Motiváciou za nákupmi z tohto obchodu býva častokrát výrazne nižšia cena a možnosť stiahnutia jednej platenej aplikácie každý deň zdarma [7].

1.1.2 iOS

Počiatky iOS siahajú do roku 2007, kedy firma Apple uviedla na trh svoj prvý smartfón - iPhone. Vtedy však pod jednoduchým názvom iPhone OS. V tom čase nebolo možné vyvíjať pre platformu vlastné aplikácie. Tie mali byť prístupné iba pomocou zabudovaného prehliadača Safari. To sa zmenilo s príchodom prvej veľkej aktualizácie iPhone OS 2, kde Apple na žiadosti vývojárov nielen sprístupnil svoj SDK, ale poskytol aj distribučnú platformu v podobe Apple App Store. Natívnym jazykom po vzore predchádzajúcich softvérových riešení sa stal Objective-C [8].

V roku 2014 na svojej každoročnej konferencii WWDC uviedol Apple pre svoj softvérový ekosystém nový jazyk Swift. Cieľom bolo ponúknuť vývojárom jednoduchú alternatívu (a do budúcna zdanlivo náhradu) k už existujúcemu jazyku Objective-C. Keďže tieto dva jazyky používajú rovnaký kompilátor, je možné v rámci jednej aplikácie používať ako Swift, tak Objective-*C* [9]. Jazyk si za cieľ pokladá stať sa najlepším nástrojom pre tvorbu mobilných, desktopových a cloudových aplikácii. V snahe dosiahnuť tento cieľ bol navrhnutý s troma hlavnými bodmi: bezpečnosť, rýchlosť, expresívnosť. To znamená, že jazyk je často vnímaný ako striktný, čo je nutné pre bezpečnosť. Rýchlostným zámerom je nahradenie jazykov rodiny C (C, C++, Objective-C), čiže výkon musí byť s týmito jazykmi porovnateľný. Swift zároveň benefituje jednoducho dostupnou syntaxou a prvkami moderných jazykov. Výsledkom je teda efektívny, jednoducho čitateľný kód [10]. V súčasnosti sa jedná už o piatu revíziu. Jeho open source nátura a jednoduchá dostupnosť z neho robí vhodného kandidáta pre prvý programovací jazyk [11].

1.2 Multiplatformové riešenia v rámci natívneho vývoja

Multiplatformové frameworky umožňujú vývojárom vytvárať mobilné aplikácie, ktoré sú kompatibilné s viac ako jedným operačným systémom. Spočiatku tieto nástroje boli nespoľahlivé, vznikalo nečakané správanie sa aplikácii a prichádzali s výkonnostnými problémami. Avšak ich rýchly vývoj v posledných rokoch z nich spravil použiteľného kandidáta pre vývoj aplikácii. Faktory pre výber nástroja sa za poslednú dekádu výrazne zmenili a to, čo z natívneho vývoja robilo z väčšej časti favorita na poli mobilných aplikácii už nemusí byť nutne aktuálne. Postupne vznikli nové požiadavky pre aplikácie z biznisového hľadiska a vývojári boli nútení hľadať nové nástroje, ktoré tieto požiadavky odzrkadľovali. Ich výrazné výhody oproti natívnemu prístupu spočívajú v [12]:

- Znovu použiteľnosť kódu, kde namiesto písania zvlášť kódu pre každú platformu, vývojári môžu použiť rovnaký kód naprieč platformami. Taktiež odstraňuje vysokú mieru repetitívnosti a ťažkopádnej, monotónnej práce [12].
- Lacnejší vývoj, kde firmám stačí investovať do jednorazového vývoja funkcionality a dostanú ju na každej platforme s minimálnymi úpravami [12].
- Rýchlejší vývoj. Nie je nutné použiť separátny jazyk pre každú platformu, namiesto toho je možné použiť jeden jazyk, s ktorým má vývojár skúsenosti a použiť ho naprieč celou aplikáciou. Toto však neznamená iba rýchlejší vývoj, ale i rýchlejší prísun aktualizácii a opráv [12].
- Uniformnosť umožňuje udržiavať rovnaký vzhľad pre aplikáciu a nenúti užívateľov učiť sa nové praktiky, ako dosiahnuť rovnakého výsledku pri jej využívaní vždy, keď použijú iný operačný systém, čo zabraňuje frustrácii užívateľa [12].

Hlavnou výhodou, ktorá je zároveň pre mnohých vývojárov a firmy najpodstatnejším faktorom pri výbere nástroja, je čas potrebný pre vývoj aplikácie, ktorý je oproti natívnym riešením výrazne nižší a tým pádom lacnejší [13].

Napriek ich obrovskému pokroku, stále nie sú akousi "striebornou guľkou", keďže stále so sebou prinášajú radu menších či väčších nevýhod, medzi ktoré patria najmä:

- Limitovaná podpora funkcionalít najnovších verzii operačných systémov. V závislosti od vybraného nástroja môžu byť vývojárom odopreté častokrát kľúčové sady funkcionalít. Tu sa ako hlavný kandidát hlási FaceID v prípade iPhone. V tomto prípade hrá úlohu aj čas. Ak by bol pre vývoj vybraný natívny spôsob, k novým funkcionalitám sa častokrát vývojári dostanú v deň ohlásenia. Pri nástrojoch, ktoré nie sú

natívne je nutné čakať na najbližšiu veľkú aktualizáciu, ktorá tieto nové funkcionality spravidla integruje [14].

- Aplikácie vyvíjané multiplatformovo bývajú spravidla pomalšie. To je spôsobené prekladačmi, kedy je kód priamo prekladaný do natívneho jazyka, čo so sebou v konečnom dôsledku prináša dodatočný kód nutný pre správny beh aplikácie na danej platforme [14].
- Problémom však môže byť aj UI/UX, kde v závislosti od výberu nástroja Častokrát nie sú poskytované natívne komponenty a dosiahnuť správny UX na oboch platformách zvlášť je častokrát veľmi zložitou úlohou. [14]

V rámci natívneho vývoja sa nám dnes predstavujú najmä tri frameworky a to sú: React Native, Xamarin a začínajúci Flutter.

1.2.1 React Native

React Native je open source framework pre mobilný vývoj. Vývojárom umožňuje vyvíjať aplikácie pre Android a iOS pomocou jazyka *JavaScript* a natívnych API [15]. Spustený v roku 2015 pod záštitou Facebooku, umožňuje použiť knižnicu React pre mobilný vývoj [16].

React Native si dáva za cieľ vytvoriť deklaratívny, jednoducho laditeľný nástroj zameraný na vysokú mieru prenositeľnosti a rýchlosť vývoja. Najnižšími podporovanými verziami operačných systémov, pre ktoré je možné vyvíjať, je iOS 9.0 a Android 4.1. alebo novšie. Pre vývoj je možné vybrať si Windows, macOS alebo Linux, avšak spúšťanie iOS aplikácii je možné iba na macOS [17].

Tento framework si získal v posledných rokoch vysokú obľubu a to pre:

- Využitie natívnych UI prvkov, čo nie je u multiplatformových frameworkov bežné. To znamená, že aplikácie budú vyzerať ako natívne a bývajú od nich častokrát ťažko rozlíšiteľné [18].
- Optimálny výkon, vďaka odlišnému návrhu od ostatných nástrojov. Funkcionalita a UI samotné existujú v separátnych vláknach [18].
- Znovu použiteľnosť kódu vďaka pripraveným komponentom. Mnoho komponentov je predpripravených a vývojári ich vďaka podobnosti s klasickou React knižnicou vedia okamžite využívať a ich funkcionalita bude rovnaká na oboch platformách [19].

- Live a Hot Reloading. Bežne je nutné po vykonaní zmien znovu skompilovať celú aplikáciu a nahrať ju do zariadenia čo je proces, ktorý sám o sebe vie zabrať veľké množstvo času. Live a Hot Reloading tento problém rieši a zmeny v aplikácii je možné vidieť takmer okamžite [19].
- Podpora natívnych knižníc tretích strán. RN umožňuje nainštalovať rovnaké balíky, aké by boli využité pri natívnom vývoji a je schopný ich doplniť o JavaScriptové knižnice a knižnice písané pre RN, ako napr. knižnice pre navigáciu [19].

Odradzujúcim faktorom pre vývojárov však môže byť konzistencia aktualizácii frameworku samotného, ktoré sú častokrát mätúce, keďže je nutné vykonať množstvo zmien, aby aplikácia po aktualizácii na novú verziu RN fungovala ako predtým. Vzhľadom k jeho JavaScriptovej povahe taktiež nie je vhodný pre aplikácie založené na zložitých výpočtoch, keďže nie je zvládnutá správa pamäte pre operácie s desatinnými číslami. Taktiež nie je doporučená voľba pri tvorbe bankových, či iných aplikácii, ktoré manipulujú so senzitívnymi dátami užívateľov. [19]

Jeho princípom sú dve vlákna, ktoré má spustené každá React Native aplikácia. Jedno z nich je hlavné jadro, ktoré spúšťa klasickú natívnu aplikáciu, má na starosti zobrazovanie UI elementov a správu užívateľských gest. Druhé vlákno je špecifické pre RN. Jeho úlohou je spúšťanie JavaScriptového kódu, ktorý sa stará o funkcionalitu aplikácie. Taktiež definuje štruktúru a mapovanie UI na funkcionalitu. Tieto dve vlákna spolu nikdy priamo nekomunikujú a nikdy sa neblokujú. Existuje medzi nimi tzv. *bridge*, ktorý je jadrom React Native. *Bridge* má tri podstatné charakteristiky [20]:

- Asynchrónnosť. Umožňuje asynchrónnu komunikáciu medzi vláknami. Toto zaručuje, že sa nikdy nebudú blokovať [17].
- Serializovateľnosť. Dve jadrá nikdy nezdieľajú a nevykonávajú operácie na rovnakých dátach. Namiesto toho medzi sebou zdieľajú serializované správy [17].
- Dávkovateľnosť. Prenos správ z jedného vlákna do druhého a naopak je optimalizovaný, správy sa posielajú "v dávkach" [17].

Aplikácii postavených na React Native sú tisícky. Využitie si našiel u firiem Fortune 500, nových startupoch a nezávislých vývojároch. Medzi najznámejšie aplikácie patria: Facebook, Instagram, Pinterest, Skype, Tesla, Uber, Walmart, Adidas Glitch, Discord, Vogue a mnoho ďalších [18].

1.2.2 Xamarin

Xamarin je multiplatformový framework založený na jazyku C# a natívnych platformových knižníc obalených do .NET vrstvy. Technológia sa skladá z troch hlavných súčastí: Xamarin Platform, Xamarin Cloud, a Xamarin Insights [13].

- Xamarin Platform je považovaný za najpodstatnejšiu súčasť Xamarinu a poskytuje prístup k API, komponentom, bezpečnostným prvkom, virtuálnym strojom atď [13].
- Xamarin Cloud poskytuje platformu pre automatizované testovanie pre zaistenie kvality na rôznych zariadeniach [13].
- Xamarin Insights je monitorovací nástroj ktorý pomáha sledovať pády a výnimky aplikácie [13].

Medzi prednosti Xamarinu patrí výkon, ktorý je porovnateľný s natívnymi aplikáciami. C# je silný jazyk a konkurent na poli mobilného vývoja, ktorý umožňuje konkurovať ostatným, bežne používaným jazykom, ako sú Kotlin, Java a Swift. Podstatné pre platformu sú taktiež *Xamarin.Forms*. Táto technológia umožňuje vývojárom písať UI kód, ktorý je zdieľateľný medzi iOS a Android aplikáciami a ponúka viac ako 40 ovládacích prvkov a rozvržení zobrazenia, ktoré sú počas behu aplikácie mapované na natívne ovládacie prvky. Xamarin má taktiež zabudovanú offline podporu. Vďaka jeho synchronizačným prvkom pre dáta, umožňuje aplikáciám pracovať v offline režime, čo je vlastnosť ktorá bola dávnejšie dostupná iba pre natívne aplikácie [13].

Napriek tomu, že je to multiplatformový nástroj, vývojárov odrádza čas nutný pre vývoj a to najmä z hľadiska prvého nastavenia. Je totiž nutné správne označovať kód naprieč rôznymi operačnými systémami a .NET frameworkom. Toto negatívne ovplyvňuje čas nutný pre spustenie aplikácie a sťahovanie dát. Veľkou nevýhodou je taktiež tvorba UI. *Xamarin.Forms* síce umožňuje zdieľanie niektorých UI komponentov, stále však existujú komponenty, ktoré musia byť zvlášť vyvinuté pre každú platformu. Oproti klasickým webovým technológiám je pre tvorbu UI nutné použiť proprietárny jazyk XAML. Posledným výrazným problémom sú obtiažnosti s kódom samotným. Pre znalosť Xamarinu je potrebné poznať princípy natívneho mobilného vývoja zvlášť pre každú platformu, takže samotný .NET častokrát nestačí. Táto kombinácia znalostí sa hľadá obtiažne, keďže nie každý programátor sa chce učiť rôzne spôsoby, ako pristupovať k vývoju mobilných aplikácii [13].

1.2.3 Flutter

Flutter je open-source SDK vyvinuté pod záštitou Google. Cieľom je rýchla tvorba iOS a Android aplikácii za pomocou zdieľaného kódu. Dokáže komunikovať s natívnymi Android a iOS SDK, avšak pre kompiláciu iOS časti je stále nutný počítač od Apple, rovnako ako je to u ostatných frameworkov a SDK [19].

Pre vývoj používa objektovo orientovaný programovací jazyk Dart. V počiatkoch chcel Google vyvinúť jazyk, ktorý by bol schopný konkurovať JavaScriptu pre front-end aj back-end kód. Ponúka rovnaké nástroje ako JS, na rozdiel od neho však ponúka možnosť využiť princípy objektovo orientovaného programovania, ako napr. dedičnosť. Každý kus kódu, ktorý je napísaný v Darte je možné skompilovať do JS [20].

Aplikácie sú kompilované ahead-of-time do natívneho ARM kódu. Toto oproti klasickým frameworkom ako React Native zvyšuje výkonnosť, keďže uprostred neexistuje žiadny *JS bridge*, ktorý parsuje a spúšťa kód [19].

UI vykresľuje pomocou rýchlej *C++* knižnice zvanej Skia (ktorá je taktiež použitá ako grafický engine pre Google Chrome, Android, Chrome OS a Mozilla Firefox). takže sa nejedná len o akési zaobalenie natívnych UI komponentov, ako je to v prípade RN a Xamarinu. Flutter dokáže výborne replikovať Material Design od Google a komponenty špecifické pre iOS pomocou knižnice Cupertino, no tie nie sú rendrované natívne. Ak by iOS 13 zmenil spôsob vykresľovania niektorého komponentu, pre vývojárov Fluttru by to znamenalo čakanie na aktualizáciu Cupertino knižnice jej tvorcami. Tým pádom by mohla aplikácia vyzerať zastaralo [19].

Podobne ako RN, Flutter je založený na tzv. "reaktívnom programovaní", čo znamená, že aplikácia je schopná reagovať na vstup od užívateľa zmenou "stavu" View, v ktorom sa momentálne nachádza a toto View je znova rendrované na základe nového stavu. Vývojári si, ako aj pri React Native, obľúbili zabudovanú možnosť Hot reloading [19].

Napriek svojím prednostiam je to stále relatívne mladý framework s malou komunitou a vývojové nástroje nie sú úplne dokonalé. Toto sa najviac prejavuje pri odlaďovaní. Je možné použiť print/debugPrin*t* funkcie, prezerať logy a použiť profilačné nástroje. No výpisy z nich môžu byť mätúce a nejasné a nie vždy dokážu naviesť na časť, ktorá chybu skutočne zapríčinila [19].

V čase písania sa teda jedná o dynamicky rozvíjajúci sa nástroj a v budúcnosti bude hlavným vývojovým spôsobom pre chystaný OS Fuchsia [21]. Mnohé korporácie na čele s Google ho taktiež začali adoptovať, medzi najznámejšie je možné spomenúť Alibaba, Google Ads, AppTree, Abbey Road Studios a Tencent [22].

1.3 Hybridný vývoj

Hybridný vývoj je mix natívneho a webového prístupu. Jadro aplikácie je písané za použití klasických webových technológii, ako HTML, CSS a JavaScript. Aplikácia však nie je spustená v prehliadači, ale je zaobalená v tzv. natívnom "wrapperi", ktorý obsahuje vlastný prehliadač, ktorý však užívateľ nevidí. Napríklad aplikácia pre iOS by použila WKWebView k zobrazeniu aplikácie, zatiaľ čo na Androide sa používa WebView k vykonaniu rovnakej funkcie. Tento kód je potom priradený k určitému zaobaleniu za pomoci Apache Cordova alebo Capacitor od Ionic Framework. Toto vytvorí akúsi natívnu schránku, kde je použitý web view danej platformy, v ktorom načíta danú aplikáciu. Vývojárom je umožnené týmto spôsobom vytvárať a publikovať aplikácie buď na Google Play alebo App Store [23].

Cordova aj Capacitor poskytujú plugin systém, ktorý umožňuje prekonať limitácie prehliadača a dostať sa k natívnym funkcionalitám mobilného zariadenia, ako sú TouchID, NFC, Bluetooth [23].

Nespornými výhodami sú rýchlosť vývoja a tým pádom nižšia cena potrebná pre vývoj a kompatibilita naprieč rôznymi platformami, avšak nie je doporučené tento prístup používať pri určitých typoch aplikácie, ako sú napríklad graficky náročne hry a podobné aplikácie [23].

1.3.1 Ionic

Ionic Framework je open source sada UI nástrojov pre vytváranie mobilných a desktopových aplikácii. Zameriava sa na frontend UX a interakciu s UI (zobrazovacie prvky, gestá, animácie), ktorý je postavený na webovom frameworku Angular, môže však byť použitý bez frameworku za použitia skriptu. V aktívnom vývoji je aj podpora pre ostatné webové frameworky ako Vue a React [24].

Ionic definuje štyri základné ciele:

- Multiplatformovosť:

- Webové štandardy ako základ
- Nádherný dizajn
- Jednoduchosť

Priniesol so sebou dve hlavné verzie a to Ionic 2/3 a Ionic 4.

Ionic 2/3 Pôvodne bol založený na populárnom frameworku Angular a prístup k natívnym funkcionalitám zariadenia poskytoval za pomoci Apache Cordova. Zároveň ponúkal témy, čo dodávalo komponentom natívny vzhľad a dojem bez zmeny kódu samotného. Spočiatku si získal popularitu vďaka svojej jednoduchosti, avšak mal aj slabú stránku v podobe nízkeho výkonu. Táto verzia pozostávala zo spojenia Angularu, Cordovy a tém [25].

Štvrtá verzia priniesla obrovské vylepšenia pre platformu. Z čisto hybridného frameworku sa preusnula na univerzálny framework, umožňuje totiž vyvíjať hybridné, webové a desktopové aplikácie. Mimo integrácie s Angular 7 bola pridaná podpora pre React a Vue.js a poskytol určité zázemie pre Progresívne Webové Aplikácie. Pre prístup k natívnym funkcionalitám je použitý novovytvorený Capacitor, čo znamená zvýšenie výkonu aplikácii, pričom je spätne kompatibilný s Cordovou [25].

Popularite Ionicu však pomáha aj adopcia známych značiek a korporácii, medzi ktoré patria Comcast, IBM, Target, Mc Donald's a MasterCard [26].

2 PROGRESÍVNE WEBOVÉ APLIKÁCIE

Momentálne neexistuje žiadna presná definícia pre PWA, keďže je to skôr koncept, ako technológia samotná. Tento koncept je však v princípe pomerne jednoduchý. Je to webová aplikácia založená na prehliadači za použitia najnovších webových technológii no hocijakom zariadení, od mobilu, cez desktop až po web. Tento koncept nie je nový a aktívne sa vyvíja už niekoľko rokov [27].

Tento koncept môže byť špeciálne zaujímavý pre firmy, ktoré chcú rýchlym a efektívnym spôsobom budovať svoju značku. Za posledných pár rokov bolo byť "*mobile-first*", avšak s narastajúcim počtom zariadení rôznych rozmerov sa stal klasický mobilný vývoj náročným spôsobom, ako dosiahnuť čo najvyšší počet užívateľov. Preto sa firmy ako Twitter, Starbucks a Forbes rozhodli investovať do PWA [27].

Rozmach popularity ovplyvnilo niekoľko faktorov. Ako prvý významný faktor je možné spomenúť rok 2018, kedy korporácie Google a Microsoft výrazne do technológie zainvestovali [27].

Ako bolo spomenuté vyššie, existuje niekoľko značiek, či už menšie alebo väčšie, ktoré investovali do PWA a uvideli obrovské výhody, ďalej medzi nimi aj:

- **Forbes.** Populárny časopis, ktorý v rámci skvalitnenia UX chcel implementovať PWA. Po nasadení a následnej štúdii došiel k týmto záverom. Opätovné navštívenie stránky sa zvýšilo o 43% na užívateľa, zhliadnutie reklám sa zvýšilo o 20%, 3x vyššia miera scrollovania [27].
- **Flipkart.** Najväčšia e-commerce stránka v Indii zistila, že skoro 60% (!) užívateľov si odinštalovalo ich mobilnú aplikáciu, aby ušetrili miesto [27]. V roku 2015 sa totiž rozhodli adoptovať stratégiu, ktorá podporovala iba natívnu aplikáciu. Flipkart zistil, že je ťažšie a ťažšie dodať rovnaký užívateľský zážitok, ktorý bol rýchly a pútavý. Výsledkom po implementácii PWA bol nárast návštevnosti zo 70 sekúnd na 3.5 minúty a 3x nižšie využitie dát [32].
- **Tinder.** Populárna aplikácia pre randenie, prešla na PWA v roku 2017 a to priviedlo okamžité výsledky. Spozorovali nárast vo "swipovaní" (hlavný koncept aplikácie), v počte odoslaných správ, užívatelia ostávali dlhšie v aplikácii. Zároveň znížili čas potrebný pre načítanie z 11.91 sekúnd na natívnej aplikácii na 4.69. PWA je zároveň o 90% menšia, ako natívna aplikácia [33].

2.1 Princíp

PWA slúži ako ideálne riešenie problémov ako natívnych mobilných aplikácii, tak tradičných webových aplikácii vďaka svojim prednostiam či už pre biznis, vývojára, alebo užívateľa samotného:

- **Rýchlosť**. Podľa Google, PWA majú <1 sek. medián v čase načítania, čo znamená, že sú 4x rýchlejšie pri 10x menej dátach. Zároveň dokážu pracovať offline a v oblastiach so slabým pripojením, čo znamená, že viac užívateľov dokáže pracovať s aplikáciou bez toho, aby boli znevýhodnení svojím pripojením [27].
- **Dostupnosť**. Užívatelia čoraz viac využívajú mobilné vyhľadávanie pre prvý kontakt so značkou. PWA do tohto trendu nádherne zapadajú, pretože sú postavené na webových technológiách, čiže sú aj rovnako indexované, čo má pozitívny vplyv na SEO. Takto môžu byť firmy jednoduchšie vyhľadávateľné. Fakt, že sú inštalovateľné na domácu obrazovku znamená, že dokážu nahradiť natívnu aplikáciu, čím sa užívateľ vyhne App Storom [27].
- **Spoľahlivosť.** Fungujú nezávisle na platforme, pripojení a veľkosti pamäte, ich funkcionalita je výborná nezávisle od externých vplyvov [27].
- **Bezpečnosť.** Podmienkou správnej implementácie PWA je HTTPS protokol čo zaisťuje, že aplikácia a jej obsah sú vždy bezpečne šifrované [27].
- **Zdielatelnosť.** Na rozdiel od aplikácii v obchodoch, ktoré neposkytujú linky pre jednoduché zdieľanie aplikácie, PWA takýto link budú mať vždy, čo umožňuje jednoduché zdieľanie na trhu obyčajným linkom [27].
- **Jednoduchosť implementácie.** K jej vývoju stačí vývojárom použiť webové technológie HTML, CSS, JavaScript. Keďže weboví vývojári sú najpopulárnejším typom vývojárov, firmy a individuálni vývojári môžu využiť existujúce znalosti pre tvorbu tohto typu aplikácie [27].
- **Podoba natívnej aplikácie.** Vďaka modelu aplikačnej ulity, PWA vyzerajú ako natívna aplikácia na hocijakom zariadení s natívnymi funkciami a navigáciou [27].

Existuje mnoho výhod, prečo vyvinúť práve PWA, avšak stále existuje zopár problémov, ktoré musí vývojár riešiť, ak si zvolí tento prístup, ako napríklad:

- **App Store.** Keďže ich popularita rastie očakáva sa, že toto nebude dlhotrvajúci problém. Avšak momentálne, aspoň čo sa Apple App Store týka, PWA nie sú prioritizované. UX pri pridávaní na plochu tiež nie je vyladené, vyžadujúc si dve ťapnutia

namiesto jedného pre Android. Tento problém sa Androidu netýka, keďže s príchodom Chrome 72 bolo umožnené pridanie, do Play Store [27].

- **Kľúčové mobilné vlastnosti nie sú dostupné na všetkých zariadeniach**. V súčasnosti ešte stále existuje určitá limitácia z hľadiska vlastností, s ktorými dokážu vývojári pracovať. GPS a kamera sú síce dostupné, no skener odtlačkov prstu a senzory (ešte) dostupné nie sú [30]. Tu je chyba priamo v JavaScripte, ktorý ako taký beží na jednom vlákne, čo znamená, že ak je vykonávaná nejaká operácia, druhá musí byť pozastavená [34].
- **Nedostatočné oboznámenie**. Napriek obrovskému nárastu v popularite, stále je nutné vzdelávať vývojárov a z časti užívateľov o tom, čo vlastne PWA je a ako ju správne implementovať. Vývojári sa musia naučiť hlavné koncepty technológie ako je optimalizácia výkonu, cachovanie, offline mód. Toto sa však dá pokryť dostatočným množstvom dostupných vzdelávacích materiálov. Na strane užívateľov existuje nedostatočné povedomie o PWA, mnohí nevedia, že aplikáciu je možné pridať na plochu (obzvlášť platné pre iOS). Toto sa mení so značkami ako Pinterest, Tinder, Washington Post a ďalšími, no stále to v súčasnosti nie je dostačujúce [27].

2.2 Vlastnosti platformy a princípy funkcionalít

Progresívne webové aplikácie používajú moderné webové technológie, aby priniesli rýchly, pútavý a spoľahlivý zážitok na mobilnom webe. Obsahuje rôzne architektúry a technológie umožňujúce offline zážitok, synchronizáciu na pozadí a notifikácie. Toto otvára dvere funkcionalitám, ktoré predtým vyžadovali výhradne natívnu aplikáciu. Medzi hlavné vlastnosti patria aplikačný manifest, service workers, offline mód, pridanie na domovskú obrazovku, notifikácie a vývoj pre desktop [35].

2.2.1 App Shell

App Shell sú lokálne zdroje, ktoré webová aplikácia potrebuje k načítaniu kostry UI. V podstate je to balík kódu, ktorý by bol normálne publikovaný do App Store. Tento vývojový prístup sa spolieha na ukladanie App Shell do lokálnej pamäte zariadenia (typicky jednoduché HTML, CSS a JavaScript) potrebnej k zobrazeniu a spusteniu aplikácie. Dynamický obsah sa načítava za pomoci JavaScriptu. Tento prístup je užitočný, keď je potrebné dostať HTML na obrazovku rýchlo a bez pripojenia [36].

App Shell vždy obsahuje HTML, môže obsahovať CSS a JavaScript a niektoré statické zdroje, ktoré dodávajú aplikácii štruktúru, avšak neobsahuje obsah špecifický pre danú stránku. Zhrnutím obsahuje zdroje, ktoré sa menia čo najmenej a môžu byť uložené, aby mohli byť okamžite načítané a nebolo ich potrebné sťahovať znova pri ďalších návštevách. Nie je však vhodné ukladať úplne všetko, keďže toto môže viesť k vyšším načítacím časom a frustrácii užívateľa [36].

Uchováva komponenty lokálne a obsah sťahuje dynamicky pomocou API, pričom však neobetuje objaviteľnosť webu. Vždy, keď užívateľ navštívi web, najnovšia verzia je automaticky zobrazená a nie je potrebné sťahovať vždy novú verziu [36].

2.2.2 App Manifest

Web app manifest je jednoduchý JSON súbor, ktorý informuje prehliadač o tom, ako by sa mala aplikácia správať po inštalácii. Typický manifest zahŕňa informácie o aplikácii, ako napríklad meno, ikony, počiatočnú url, na ktorej sa má aplikácia spustiť a mnohé ďalšie. Manifest je priamo prepojený s HTML stránkami [37].

Medzi ďalšie vlastnosti patria:

- **background_color:** Definuje očakávanú farbu pozadia pre stránku. Táto hodnota je použitá pre definovanie farby pozadia skratky, ak je manifest načítaný. Vďaka tomuto je vytvorený hladký prechod medzi spustením aplikácie a načítaním obsahu [38].
- **description:** Poskytuje všeobecný popis aplikácie [38].
- **display**: Definuje preferované zobrazenie. Možnosti sú standalone, kedy je aplikácia zobrazená ako natívna. Prvky prehliadača nie sú zobrazené, a sú v samostatnom okne. Má vlastnú ikonu na ploche atď. Pri minimal_ui je aplikácia zobrazená ako natívna, obsahuje však určité prvky prehliadačovej navigácie, ktoré sa líšia od prehliadača. Predvolenou možnosťou je browser, kedy je aplikácia zobrazená ako klasický web [38].
- **icons**: Špecifikujú pole obrázkov, ktoré slúžia ako aplikačná ikona. Definuje rozmery, zdroj a typ obrázka [38].
- **related_applications:** Je to pole natívnych aplikácii, ktoré sú inštalovateľné alebo prístupné na danej platforme. Môže tu byť definovaná Play Store aplikácia pre Android alebo App Store aplikácie pre iOS. Tieto aplikácie slúžia ako alternatíva k webstránke, ktorá poskytuje podobnú funkcionalitu. V objekte je nutné definovať platform, url a id aplikácie [38].
- **Splash screens:** Od Chrome verzie 47, pri spustení je zobrazený obrázok, ak je aplikácia spustená cez domovskú obrazovku. Tento obrázok je automaticky generovaný prehliadačom za pomoci niektorých vlastností aplikačného manifestu, špecificky name, background_color, icons [38].

2.2.3 Pridanie na domovskú obrazovku

Užívateľom je umožnené nainštalovať si webovú aplikáciu pomocou prehliadačov, ktoré to podporujú rovnako ako mobilnú aplikáciu. Táto funkcionalita je dosiahnuteľná pomocou aplikačného manifestu a vlastnosti, ktorá sa nazýva "Add to home screen" [39].

Tieto technológie umožňujú aplikácii spustenie priamo z domovskej obrazovky zariadenia namiesto toho, aby užívateľ manuálne do prehliadača zadával URL. Táto webová aplikácia je rovnocenná s natívnymi aplikáciami a je hodná toho byť na domovskej obrazovke, tým pádom je prístupnejšia [39].

O možnosti inštalácie aplikácie informuje tzv. "Add to Home Screen Prompt", ktorý umožňuje jednoduchú inštaláciu aplikácie jedným ťapnutím [40].

K tomu, aby bolo toto upozornenie zobrazené, je nutné splniť niekoľko podmienok, ako sú [39]:

- Manifest, kde musí byť vyplnené meno aplikácie, ikony správnych rozmerov, display a start_url
- Stránka musí byť uložená na HTTPS doméne
- Musí existovať ikona reprezentujúca aplikáciu
- Aplikácia musí mať offline funkcionalitu (toto však vyžaduje iba Chome pre Android

2.2.4 Offline Mód

PWA dokážeme obohatiť o offline mód pri nestabilnom alebo neexistujúcom pripojení za pomoci tzv. "Service Workers" [41].

Service Workers sú virtuálna proxy medzi prehliadačom a sieťou. Primárne riešia problém cachovania prostriedkov a umožňujú ich offline dostupnosť [42].

Sú spustené v samostatnom vlákne a nemajú prístupk DOM. štruktúre. Uvádzajú odlišný prístup k tradičnému webovému vývoju [42]. Aplikáciám umožňujú ovládať sieťové požiadavky, cachovať ich pre zlepšenie výkonu a poskytnúť ich, v prípade potreby, offline [43].

Service Workers závisia na dvoch API, aby mohli umožniť offline funkcionalitu: Fetch (štandardný spôsob získania dát zo siete) a Cache (úložisko pre aplikačné dáta). Táto cache nie je mazaná a je nezávislá od cache prehliadača alebo stavu siete [43].

Slúžia ako základ pre rôzne funkcionality dopomáhajúce k natívnemu zážitku, ako sú napríklad [44]:

- Notifications API: Systém umožňujúci zobraziť a manipulovať s notifikáciami za použitia natívneho notifikačného systému OS [43]
- Background Sync API: Umožňuje odložiť určité akcie, kým nebude mať užívateľ stabilné pripojenie. Toto je užitočné v prípade, kedy je nutné užívateľa uistiť, že to, čo odoslal, sa reálne odoslalo. Serverom táto funkcionalita umožňuje odosielať periodické aktualizácie do aplikácie v momente, kedy bude aplikácia opäť online [43]

Majú svoj vlastný životný cyklus a počas neho prechádzajú troma krokmi: registráciou, inštaláciou a aktiváciou. Aby bolo možné SW nainštalovať, je nutné ho registrovať v hlavnom vlákne JS kódu. Registrácia informuje prehliadač o tom, kde sa SW nachádza a o jeho možnosti inštalácie na pozadí. Ďalším krokom je inštalácia, ktorá sa deje okamžite po registrácii za predpokladu, že prehliadač usúdi, že sa jedná o nový SW. Počas tejto fáze môžeme spustiť vlastný kód, ktorý ovláda udalosti inštalácie. toto je vhodná doba na cachovanie statických prostriedkov aplikácie. Po tom, čo sa úspešne nainštaluje, prechádza do aktivačnej fázy. V momente aktivácie SW ovláda všetky stránky, ktoré sú načítané v jeho rámci a začne načúvať ich udalostiam [43].

Mimo offline funkcionality poskytujú aj mnohé ďalšie vlastnosti, vrátane správy notifikácii, vykonávanie náročnejších výpočtov v separátnom vlákne, ovládať sieťové žiadosti a modifikovať ich [42].

Vzhľadom na to, ako sú SW mocné, je možné spustiť ich iba v rámci bezpečného kontextu. Preto je nutná komunikácia pomocou HTTPS protokolu [42].

2.2.5 Push Notifikácie

S príchodom PWA bolo vývojárom umožnené odosielať na webe tzv. push notifikácie. Vo svojej podstate sú to správy, ktoré sú zobrazené na užívateľovom zariadení. Môžu byť spustené lokálne v otvorenej aplikácii, alebo môžu byť odoslané zo servera aj keď nie je aplikácia aktívne spustená [45].

Funkcionalita ako taká je zložená z Notifications a Push API. Notifications API umožňuje zobraziť užívateľovi systémové notifikácie. Push API umožňuje service workerom spracovávať správy zo servera na pozadí a "odovzdať" ich aplikácii [45].

Notifikácie samotné sú jednoduchým a mocným spôsobom, ako komunikovať s užívateľom. Upozorňujú na udalosti a lákajú užívateľa, aby znova použil aplikáciu. Dokážu zobraziť ikonku a krátky text a po kliknutí otvoria aplikáciu. Taktiež je možné integrovať tlačidlá pre rýchlu odpoveď na určitú udalosť bez toho, aby bola aplikácia spustená [45].

3 KOMPATIBILITA PWA NA JEDNOTLIVÝCH PLATFORMÁCH

Jedna z charakteristík PWA sa nazýva "*Progressive Enhancement*". Je to spôsob vývoja webu, ktorý sa zameriava na najzásadnejšiu funkcionalitu. Tento spôsob potom progresívne nabaľuje ďalšie, technologicky náročnejšie vrstvy v závislosti od prostredia, kde je aplikácia spustená. Tým pádom je každému umožnené získať prístup k aplikácii na hocijakom prehliadači a pri hocijakom pripojení, zatiaľ čo je poskytovaná aj vylepšená verzia aplikácie pre užívateľov so silnejším hardvérom alebo pripojením. Vďaka tomuto je možné dosiahnuť optimálny užívateľský zážitok a zaisťuje stabilitu [46].

PE stratégia pozostáva z niekoľkých princípov [46]:

- Obsah stránky by mal byť prístupný naprieč všetkými prehliadačmi
- Základná funkcionalita by mala byť dostupná naprieč všetkými prehliadačmi
- Štýlovanie je poskytnuté pomocou externe napojeného CSS
- Rozšírená funkcionalita by mala byť poskytovaná externe napojeným JS
- Nastavenia užívateľovho prehliadača sú rešpektované

Pri tomto prístupe je nutné začať vývoj s jednoduchou funkcionalitou dostupnou všade. V prípade využitia najnovších frameworkov a webových vlastností môže dôjsť k nestabilite a tým pádom zlému užívateľskému zážitok z používania aplikácie [46].

Pri webovom vývoji poznáme tri hlavné vrstvy. Sémantickú (HTML), vizuálnu (CSS) a interaktívnu (JS). Pre každú vrstvu je nutné vziať v úvahu evolúciu spojenú s daným jazykom a poskytnúť alternatívnu alebo ochudobnenú verziu tých vlastností aplikácie, ktoré nie sú podporované [47].

Napriek tomu, že PWA nenasledujú túto metodológiu striktne, hlavne čo sa nezávislosti JavaScriptu týka, z veľkej časti ovplyvnila špecifikácie a vlastnosti pre PWA, od čoho dostalo PWA svoj prvý prívlastok: "progresívne" [47].

3.1 iOS

Podpora PWA pre iOS prišla s verziou 11.3 neumožňovala však prístup ku všetkým funkcionalitám a prvotné vlastnosti boli výrazne obmedzené. Medzi povolené vlastnosti patrili: prístup ku geolokácii a akcelerometru a iným, menej významným vlastnostiam, ktorý výrazne ovplyvňovali kvalitu vývoja a tým pádom priniesol viac obmedzení, ako užitočných vlastností, medzi ktoré patria [48]:

- Offline dáta mohli mať veľkosť maximálne 50 Mb
- Po nevyužití aplikácie podobu niekoľko týždňov, iOS automaticky vymaže všetky aplikačné dáta
- Bluetooth, Touch ID, Face ID a iné natívne vlastnosti nie sú povolené
- Nemožnosť vykonávania kódu na pozadí
- Prístup ku kontaktom a lokácii na pozadí bol odopretý
- Nemožnosť použiť služby typu Apple Pay
- Špecificky pre iPad nebolo možné využiť multitasking
- Neboli podporované notifikácie
- Chýbala podpora pre Web App Install Banner

Rok po uvedení prvotnej podpory pre PWA na iOS, Apple vydal verziu 12.2 pre iPhone a iPad, ktorá znamenala významný krok vpred v podobe nových funkcionalít. V zásade ide o [49]:

- PWA odteraz majú nový životný cyklus, kedy je stav aplikácie uložený medzi rôznymi návštevami
- Pridané navigačné gestá typické pre Safari
- Prehliadač zabudovaný do okna PWA. Pred iOS 12.2 všetky linky otvorili prehliadač Safari, čím užívateľa odlákali od aplikácie samotnej
- Web Share prvej úrovne je odteraz podporovaný pre standalone mód

Podpora pre manifest je doteraz slabá a nezdokumentovaná, PWA pre iOS nevyužívajú mnohých manifestových atribútov, ako sú icons, orientation, theme_color a iné, avšak táto funkcionalita je v aktívnom vývoji. Príchod najnovšej aktualizácie však so sebou stále nepriniesol podporu pre funkcionality, ako sú: web push a background sync [49].

3.2 PWA pre desktop

Od verzie Chrome 73 bola spustená podpora PWA na všetkých desktopových platformách, vrátane Chrome OS, Linux, Mac a Windows [50].

Tieto aplikácie je možné taktiež nainštalovať na plochu, sú rýchle a integrované, keďže sa spúšťajú rovnako, ako hocijaká iná aplikácia a sú spustené v samostatnom okne, bez UI prehliadača [50].

Implementácia je rovnaká, ako pre mobilné PWA, je možné zobraziť App Install Banner, i pracovať offline za pomoci Service Workers, podpora sa však týka iba prehliadača Chrome [51].

II. PRAKTICKÁ ČASŤ

4 FUNKCIONÁLNA ANALÝZA

Aplikácia je určená pre uchádzačov o štúdium na Univerzite Tomáše Bati v Zlíne. Účel tejto novej aplikácie je nahradenie staršej, neaktuálnej verzie a v tomto procese odstrániť jej nedostatky a ponúknuť nový, moderný a v neposlednom rade spoľahlivý zážitok z jej používania. Pôvodná aplikácia trpela nedostatkami najmä z hľadiska udržateľnosti a rozšíriteľnosti, čo zapríčinila nesprávna kombinácia technológii, zvolených pre implementáciu. Odrážajúc sa teda od pôvodnej funkcionality a nedostatkov som bol schopný vytvoriť ako požiadavky funkcionálne, tak i nefunkcionálne. Vďaka nim som získal presnú predstavu o celej funkcionalite a z týchto znalostí nasledovne vytvoril prípady použitia, ktoré pokrývajú i alternatívne scenáre pre rôzne platformy. Na základe tohto prvotného návrhu boli zvolené najvhodnejšie technológie, ktoré pokrývajú požiadavky na túto novú aplikáciu, uľahčujú jej vývoj, a dopĺňajú ekosystém Progresívnych webových aplikácii.

4.1 Funkcionálne požiadavky

Funkcionálne požiadavky tvoria popis zamýšľanej funkcionality aplikácie a jej súčastí. Jedná sa o sadu základnej funkcionality alebo želaného chovania, ktoré musí byť jasne popísané.

R1. Aplikácia musí byť schopná zobrazovať všeobecné informácie o univerzite

R2. Aplikácia musí byť schopná zobrazovať informácie o fakultách

R3. Aplikácia musí byť schopná zobrazovať informácie o meste Zlín

R4. Aplikácia musí byť schopná zobrazovať informácie o možnostiach návštevy

R5. Aplikácia musí byť schopná zobrazovať informácie o všetkých ponúkaných odboroch v rámci jednotlivých fakúlt

4.2 Nefunkcionálne požiadavky

Nefunkcionálne požiadavky definujú vlastnosti aplikácie, ako sú bezpečnosť, spoľahlivosť, výkonnosť, udržateľnosť, rozšíriteľnosť a použiteľnosť. Slúžia aj ako obmedzenia alebo nároky.

NR1. Aplikácia bude cielená na najnovšie mobilné zariadenia za pomoci webových technológii

NR2. Aplikácia bude vytvorená pomocou knižnice React

NR3. Aplikácia bude mať vzhľad natívnej mobilnej aplikácie

NR4. Aplikácia bude fungovať bez dátového pripojenia

NR5. Aplikáciu bude možné nainštalovať na užívateľovu domovskú obrazovku

NR6. Aplikácia bude jednoducho rozšíriteľná o ďalšiu funkcionalitu

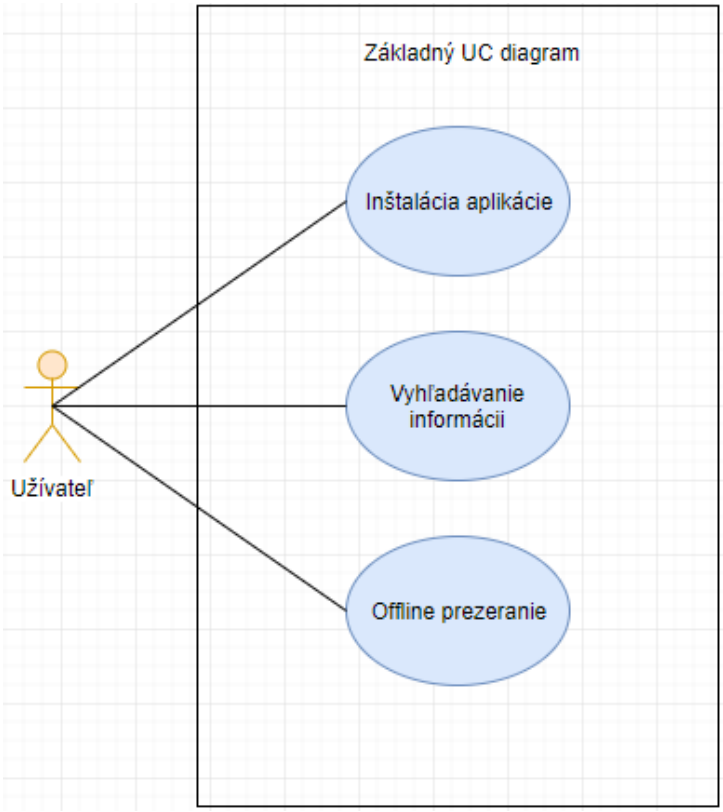
NR7. Aplikácia bude zabezpečená pomocou HTTPS protokolu

4.3 Diagram prípadu použitia a scenáre

Prípady použitia opisujú, ako užívateľ aplikácie dosiahne určitého cieľa. Vďaka tomu je možné detailne popísať, aké výstupy môžeme od aplikácie očakávať na základe rôznych akcii.

Scenár je postupnosť krokov, ktoré popisujú interakciu aplikácie a užívateľa, ktorý ju používa.

Keďže z hľadiska funkcionality neexistujú rozdiely medzi užívateľmi, aktéra má aplikácia iba jedného, teda užívateľa, ktorý aplikáciu používa.



Obrázok 1: Základný diagram prípadu použitia

4.3.1 UC01: Inštalácia aplikácie

<table>
<tr><td colspan="3">Názov: Inštalácia aplikácie OS Android</td></tr>
<tr><td colspan="3">ID: UC01</td></tr>
<tr><td colspan="3">Charakteristika:
Užívateľ je schopný nainštalovať si aplikáciu na svoju domovskú obrazovku</td></tr>
<tr><td colspan="3">Primárny aktér:
Užívateľ</td></tr>
<tr><td colspan="3">Vedľajší aktéri:
Aplikácia</td></tr>
<tr><td colspan="3">Vstupné podmienky:
Užívateľ musí mať spustený prehliadač Google Chrome na platforme Android</td></tr>
<tr><td colspan="3">Výstupné podmienky:
Užívateľ si úspešne nainštaluje aplikáciu na svoju domovskú obrazovku</td></tr>
<tr><td colspan="3">Hlavný scenár:</td></tr>
<tr><td>Krok</td><td>Aktér/Sys-tém</td><td>Popis</td></tr>
<tr><td>1</td><td>Užívateľ</td><td>Zadá adresu aplikácie</td></tr>
<tr><td>2</td><td>Aplikácia</td><td>Vyzve užívateľa k inštalácii aplikácie</td></tr>
<tr><td>3</td><td>Užívateľ</td><td>Klikne na tlačidlo „Nainštalovať aplikáciu“</td></tr>
<tr><td>4</td><td>Aplikácia</td><td>Otvorí dialógové okno so základnými informáciami o aplikácii.</td></tr>
<tr><td>5</td><td>Užívateľ</td><td>Potvrdí zobrazené informácie</td></tr>
<tr><td>6</td><td>Aplikácia</td><td>Vytvorí spustiteľného zástupcu na užívateľovu plochu</td></tr>
<tr><td colspan="3"></td></tr>
<tr><td colspan="3">Alternatívne scenáre:
UC01a – Manuálna Inštalácia aplikácie OS Android
UC01a2 – Inštalácia aplikácie pre platformu iOS</td></tr>
</table>

Tabuľka 1: UC01 Inštalácia aplikácie OS Android

Názov – Alternatívny scenár: Manuálna Inštalácia aplikácie OS Android		
ID: UC01a		
Charakteristika: Užívateľ je schopný nainštalovať si aplikáciu na svoju domovskú obrazovku manuálne		
Primárny aktér: Užívateľ		
Vstupné podmienky: Užívateľ musí mať spustený prehliadač Google Chrome na platforme Android		
Výstupné podmienky: Užívateľ si úspešne nainštaluje aplikáciu na svoju domovskú obrazovku		
Hlavný scenár:		
Krok	Aktér/Systém	Popis
1	Užívateľ	Zadá adresu aplikácie
2	Užívateľ	Klikne na nastavenia prehliadača
3	Užívateľ	Klikne na tlačidlo „Pridať na domovskú obrazovku“
4	Aplikácia	Otvorí dialógové okno so základnými informáciami o aplikácii.
5	Užívateľ	Potvrdí zobrazené informácie
6	Aplikácia	Vytvorí spustiteľného zástupcu na užívateľovu plochu

Tabuľka 2: Manuálna Inštalácia aplikácie OS Android

Názov – Alternatívny scenár: Inštalácia aplikácie pre platformu iOS
ID: UC01a2
Charakteristika: Užívateľ je schopný nainštalovať si aplikáciu na svoju domovskú obrazovku na platforme iOS
Primárny aktér: Užívateľ
Vstupné podmienky: Užívateľ musí mať spustený prehliadač Safari na platforme iOS
Výstupné podmienky: Užívateľ si úspešne nainštaluje aplikáciu na svoju domovskú obrazovku
Hlavný scenár:

Krok	Aktér/Sys-tém	Popis
1	Užívateľ	Zadá adresu aplikácie
2	Užívateľ	Klikne na tlačidlo „zdieľať“ v menu prehliadača
3	Užívateľ	Klikne na tlačidlo „Pridať na domovskú obrazovku“
4	Aplikácia	Otvorí dialógové okno so základnými informáciami o aplikácii.
5	Užívateľ	Potvrdí zobrazené informácie
6	Aplikácia	Vytvorí spustiteľného zástupcu na užívateľovu plochu

Tabuľka 3: Inštalácia aplikácie pre platformu iOS

4.3.2 UC02: Vyhľadávanie informácii

Názov: Vyhľadávanie informácii		
ID: UC02		
Charakteristika: Užívateľ je schopný získať si detailné informácie o jednotlivých sekciách		
Primárny aktér: Užívateľ		
Vedľajší aktéri: Aplikácia		
Vstupné podmienky: Aplikácia musí byť spustená Užívateľ sa nachádza v prvej časti menu: Univerzita		
Výstupné podmienky: Užívateľ si efektívne a rýchlo nájde požadované informácie		
Hlavný scenár:		
Krok	Aktér/Systém	Popis
1	Užívateľ	Vysunie sťahovateľnú ponuku
2	Užívateľ	Dotykom zvolí položku z vysunutej ponuky
3	Aplikácia	Presmeruje užívateľa na detail zvolenej položky
4	Užívateľ	Je schopný navigovať sa gestami na stránke detailu
Alternatívne scenáre: UC02a – Vyhľadávanie informácii o štúdiu		

Tabuľka 4: Vyhľadávanie informácii

Názov – Alternatívny scenár: Vyhľadávanie informácii o štúdiu		
ID: UC01a2		
Charakteristika: Užívateľ je schopný získať detailné informácie o štúdiu		
Primárny aktér: Užívateľ		
Vedľajší aktéri: Aplikácia		
Vstupné podmienky: Užívateľ musí mať spustený prehliadač Safari na platforme iOS Užívateľ sa musí nachádzať v sekcii „Štúdium“		
Výstupné podmienky: Užívateľ si efektívne a rýchlo nájde požadované informácie		
Hlavný scenár:		
Krok	Aktér/Sys-tém	Popis
1	Užívateľ	Vysunie sťahovateľnú ponuku
2	Užívateľ	Dotykom zvolí požadovanú fakultu
3	Užívateľ	Si zvolí požadovaný stupeň štúdia
4	Užívateľ	Si zvolí požadovaný odbor
5	Aplikácia	Zobrazí detailné informácie o danom odbore

Tabuľka 5: Vyhľadávanie informácii o štúdiu

4.3.3 UC03: Prezeranie obsahu aplikácie so slabým alebo neexistujúcim pripojením

Názov: Prezeranie obsahu aplikácie s neexistujúcim pripojením
ID: UC03
Charakteristika: Užívateľ je schopný čerpať obsah aplikácie napriek tomu, že je nemá dostupné pripojenie na internet
Primárny aktér: Užívateľ
Vedľajší aktéri:

Aplikácia		
Vstupné podmienky: Aplikácia bola prvýkrát spustená so sieťovým pripojením Pri prvej návšteve bol uložený statický obsah aplikácie Nie je dostupné dátové pripojenie		
Výstupné podmienky: Užívateľ dokáže získať informácie rovnako, akoby bol online		
Hlavný scenár:		
Krok	Aktér/Sys-tém	Popis
1	Užívateľ	Spustí aplikáciu
2	Aplikácia	Načíta z pamäte statické zdroje
3	Aplikácia	Upozorní užívateľa o momentálnom stave pripojenia
4	Užívateľ	Je schopný navigovať sa naprieč aplikáciou tak, akoby bola online

Tabuľka 6: Prezeranie obsahu aplikácie s neexistujúcim pripojením

5 NÁVRH ARCHITEKTÚRY

Pôvodná aplikácia „Studuj UTB“ bola postavená na multiplatformovej technológii pre tvorbu natívnych mobilných aplikácii Xamarin. Podporované platformy v tej dobe boli Android, iOS a Windows Phone a dáta získavala z univerzitného systému STAG. Z dlhodobého hľadiska toto riešenie nebolo ideálne, keďže každá zmena v dátovej štruktúre STAGu dokázala znefunkčniť celú aplikáciu. Výsledkom po dlhodobej prevádzke bola aplikácia, ktorá:

- Nedokázala pracovať bez internetového pripojenia
- Bola náchylná k pádom
- Nebola udržateľná
- Nedokázala zobrazovať aktuálne informácie
- Bola odstránená z App Store pre iOS
- Bola nespustiteľná pre Windows Phone
- Získavala zlé hodnotenia pre Android

Pri výbere vhodných technológii som teda musel zvážiť všetky tieto nedostatky a berúc v úvahu požiadavky, som sa rozhodol postaviť novú aplikáciu na moderných webových technológiách, najpopulárnejšej JS knižnice React a v neposlednom rade na titulnej technológii PWA.

5.1 React + Web

Z požiadaviek na aplikáciu vyplýva, že je nutné vytvoriť multiplatformnú, jednoducho udržateľnú a rozšíriteľnú aplikáciu. Vzhľadom na túto skutočnosť sa voľba webových technológii javí ako ideálny kandidát.

5.1.1 Tvorba základnej kostry projektu

Pre vytvorenie základnej projektovej štruktúry pre React bolo potrebné nainštalovať NodeJS, ktorý slúži ako lokálny aplikačný server. Pre správu závislostí bol použitý manažér Yarn, známy pre svoju rýchlosť a bezpečnosť.

Potom je možné použiť skript *create-react-app*, ktorý automaticky vytvorí kostru projektu.

5.1.2 Štruktúra komponentov a Atomic Design

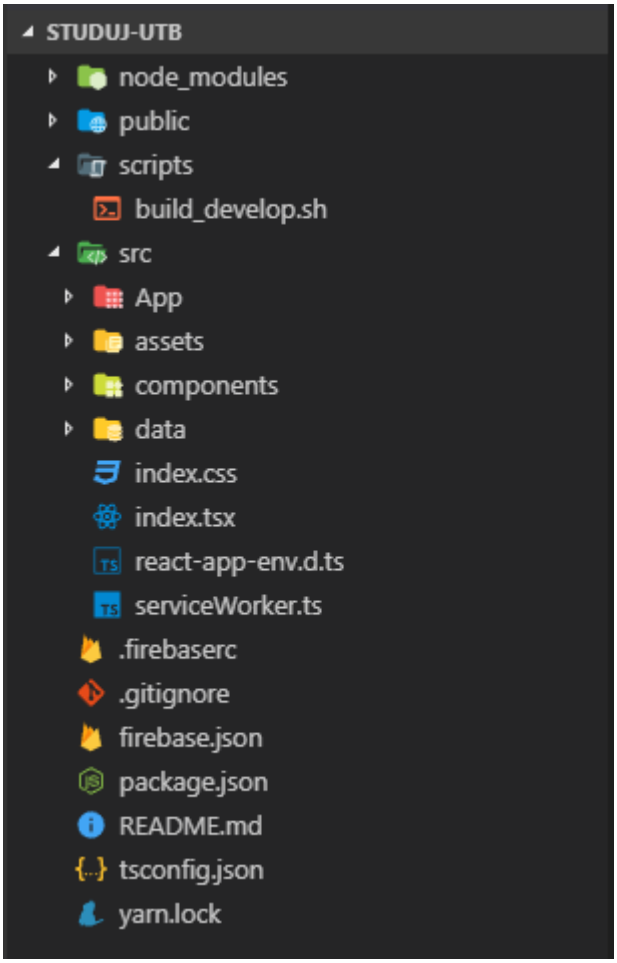
Po vytvorení základného projektu som existujúcu šablónu ďalej upravil pre použitie s atomickým dizajnom. Ten poskytuje jednoducho zrozumiteľnú metodológiu pre tvorbu a štruktúru aplikácie. Keďže sa odráža od reálneho sveta, je jednoducho vysvetliteľný a najmä zrozumiteľný.

Tento spôsob delenia teda radí komponenty do viacero kategórii:

- *Atoms:* Jedná sa o najmenšie a najzákladnejšie komponenty, ktoré v sebe neuchovávajú aplikačnú logiku a slúžia ako stavebné prvky väčších komponentov.
- *Molecules:* Sú zložené z viacerých atómov, aplikačnú logiku si opäť však preberajú z väčších komponentov.
- *Organisms:* Tvoria najväčšie časti stránky, bežne sa jedná o rôzne formuláre a pod.
- *Templates:* Určujú rozloženie organizmov, molekúl a atómov na stránke, je tu definované, ako bude celková stránka vyzerať.
- *Pages:* Najvyšší komponent, ktorá sa stará o aplikačnú logiku, výpočty a preberanie dát. Tieto potom presúva do príslušnej template, ktorá dáta rozdeľuje nižším komponentom.

Medzi ostatné významné zložky v projektovom adresári patria:

- *node_modules:* obsahujú externé knižnice, ktoré dopĺňajú základnú funkcionalitu
- *Public:* obsahuje verejne dostupné zdroje, v rámci tejto aplikácie je to index.html, kde je vložený skript pre spustenie Reactu, logo UTB, aplikačný manifest, ikony a spúšťacie obrazovky.
- *scripts*: obsahuje skript pre vytvorenie buildu z vývojového prostredia
- *src:* v seb eobsahuje *App* a *Routes. App* je najzákladnejší komponent, ktorá v sebe obsahuje *Routes,* umožňujúce navigáciu v rámci aplikácie a využitie ostatných komponent.
- *assets:* Obsahujú obrázky použité v aplikácii.
- *components:* Zložka obsahujúca komponenty podľa atomického dizajnu
- *data:* Tu sú uložené statické aplikačné dáta vo formáte JSON.
- *serviceWorker.*ts: Nastavenie pre service worker
- *tsconfig.json:* nastavenie pre TypeScript
- *package.json:* obsahuje zoznam všetkých externých balíčkov a knižníc



Obrázok 2: Stromová štruktúra aplikácie

5.1.3 Knižnice

Pre dodanie potrebnej kvality bolo nutné pridať ďalšie podporné knižnice, ktoré značne uľahčujú vývoj, ako sú:

- *Emotion:* Štýlovavia knižnica, vďaka ktorej jej možné separátne štýlovať jednotlivé komponenty a tieto štýly je potom ďalej možné využívať na iných miestach v aplikácii. Princípom je moderné „CSS in JS“, kde sa nevytvárajú samostatné CSS súbory, ale komponenty sú štýlované priamo pomocou JS.
- *FortAwesome:* Podporná knižnica pre zdarma dostupné ikony.
- *Typescript:* V aplikáciách postavených na JavaScripte je častokrát problém s neočakávaným chovaním v rámci aplikácie. Pre minimalizáciu týchto problémov a zaistenie typovej bezpečnosti dát, som zvolil TypeScript, čo je nadstavba nad JS, umožňujúca pokročilé techniky typovania dát.

- *Nuka-Carousel:* Pridáva gestá do rôznych častí aplikácie, najviac viditeľné v stránkach obsahujúce detaily určitej sekcie.
- *React Router:* Umožňuje navigáciu v rámci Reactu, výhodou je, že sa nenačítava nový index, iba sa prekresľuje ten existujúci, čo so sebou prináša výkonnostné výhody.

5.1.4 Použité komponenty

Základom každej komponenty sú tzv. *Props*, ktoré určujú, aké dáta komponent prijme. Neskôr je použitý React, pre tvorbu komponentu samotného.

5.1.4.1 Atómy

Atómy tvoria najzákladnejšiu stavebnú jednotku pre ostatné komponenty, spravidla sa jedná a tlačidlá, texty a podobné nižšie prvky. Atómy môžu taktiež tvoriť abstraktnejšie elementy ako farebné témy a štýly písma. V rámci aplikácie sú použité:

- *Icon:* Komponent implementujúci ikony z fortawesome balíka, v rámci nej je možné nastaviť názov, rozmer, farbu a odsadenie. Jedná sa o akúsi nadstavbu predvoleného komponentu *FontAwesomeIcon*, ktorý zjednodušuje jej použitie a obohacuje ho o novú funkcionalitu.
- *Layout:* je zaobalenie klasického elementu *div,* obohatený o CSS vlastnosť *display: flex.*
- *Text:* Je komponent, ktorý je využitý pre textový obsah. Nastaviteľná je farba, veľkosť a odsadenie.
- *Ribbon:* Jedná sa o obdĺžnikové rozdelenie stránky, jeho základné využitie je v hlavičke ako predeľovač a v pätičke, ako kontajner pre navigačné menu. Je možné dynamicky určiť šírku a štýl rámčeku, ktorý toto rozdelenie ohraničuje.

```tsx
/** @jsx jsx */
import { jsx } from '@emotion/core';
import styled from '@emotion/styled';
import { SFC } from 'react';
import {
  ColorProperty,
  FontSizeProperty,
  PaddingTopProperty,
  PaddingRightProperty,
} from 'csstype';
jsx;

type TextProps = {
  color?: ColorProperty;
  fontSize?: FontSizeProperty<string>;
  paddingTop?: PaddingTopProperty<string>;
  paddingRight?: PaddingRightProperty<string>;
};

const Text: SFC<TextProps> = styled.p(
  {
    margin: 0,
    padding: 0,
  },
  (props: TextProps) => ({
    color: props.color ? props.color : 'black',
    fontSize: props.fontSize,
    paddingTop: props.paddingTop,
    paddingRight: props.paddingRight,
  }),
);

export default Text;
```

Obrázok 3: Ukážka typického atómu, v hornej časti je inicializovaná CSS knižnica Emotion, import nutných súčastí. Pomocou TypeScriptu sú definované vlastnosti a v dolnej časti je vytvorený samotný atóm, ktorý ma prednastavené vlastnosti a zároveň dynamické vďaka vlastnostiam definovaným pomocou TS

5.1.4.2 Molekuly

Molekuly vznikajú kombináciou dvoch a viacerých atómov a sú najnižšou jednotkou organizmu. V rámci práce boli vytvorené a použité:

- *Card:* Zložená z najzákladnejších existujúcich atómov, jedná sa o sekciu, ktorá v sebe väčšinou združuje popisy s informáciami o fakultách atď. Skladá sa z nadpisu a textu.
- *ModuleButton:* Molekula, ktorá je použitá v rámci navigácie pre bakalárske, magisterské a doktorské odbory.
- *ModuleInfo:* V sebe obsahuje krátky zoznam sstručných informácii o študijnom odbore.
- *UniversityBigText:* Veľký text na úvodnej obrazovke
- *TabBarItem:* Je jedinou položkou menu, je možné určiť ikonu, popis a kam treba užívateľa po kliknutí/dotyku presmerovať. Štrukturálne sa jedná a spojenie navigačného komponentu a *IconWithLabel.*

```tsx
/** @jsx jsx */
import { jsx } from '@emotion/core';
import { SFC } from 'react';
import Layout from '../atoms/Layout';
import Text from '../atoms/Text';
jsx;

export type UniversityBigTextProps = {
  firstHeadingText: string;
  secondHeadingText: string;
};

const UniversityBigText: SFC<UniversityBigTextProp
  firstHeadingText,
  secondHeadingText,
}) => (
  <Layout
    css={{
      flexDirection: 'column',
      marginTop: '20px',
      marginLeft: '35px',
    }}
  >
    <Text
      css={{
        fontSize: '1.5rem',
        fontWeight: 'bold',
      }}
      color="white"
    >
      {firstHeadingText}
    </Text>
    <Text color="white" css={{ fontWeight: 'bold',
      {secondHeadingText}
    </Text>
  </Layout>
);

export default UniversityBigText;
```

Obrázok 4: Ukážka typickej molekuly, ktorá je v tomto prípade vytvorená z dvoch unikátnych atómov. Spolu tvoria jeden celok

5.1.4.3 Organizmy

Vznikajú spojením dvoch a viacerých molekúl, organizmy umožňujú tvorbu väčších celkov aplikácie, ktoré sú spravidla komplexnejšie. V rámci práce boli vytvorené a použité tieto organizmy:

- *Header:* Hlavička, ktorá tvorí hornú časť stránky. V rámci jej prispôsobenia je možné nastaviť spätné navigačné tlačidlo a spodný rámik.
- *NavExcerpt:* Predstavuje jednu položky sťahovacieho menu, skladá sa z obrázku, nadpisu a textu.
- *PageFooter:* Slúži ako kontajner pre spodnú časť stránky. Je možné doňho vložiť akýkoľvek iný komponent.
- *ModuleCard:* Karta, ktorá je použitá v rámci detailu fakulty.
- *PageLayoutWrapper:* Určuje základné štýlovanie hlavných stránok, prijímané dáta sú obrázok, použitý na hlavnej stránke a text, použitý pre popis tejto stránky. Implementuje v sebe viacero sekcii, ktoré slúžia ako hlavička, hlavný obsah a pätička.
- *Tabbar:* Kontajner pre jednotlivé prvky navigácie

```
import { Component } from 'react';
//@ts-ignore
import React from 'react';

import TabBarItem from '../molecules/TabbarItem';
import TabBar from './Tabbar';

class PageFooter extends Component {
  render() {
    const { children } = this.props;

    return (
      <>
        {children}
        <TabBar>
          <TabBarItem label="Univerzita" icon="university" exact to="/univerzita" />
          <TabBarItem label="Studium" icon="user-graduate" to="/studium" />
          <TabBarItem label="Město" icon="city" to="/mesto" />
          <TabBarItem label="Navštiv nás" icon="university" to="/navstiv-nas" />
        </TabBar>
      </>
    );
  }
}

export default PageFooter;
```

Obrázok 5: Ukážka typického organizmu, zloženého z viacerých molekúl, tvorí väčšiu časť obrazovky.

5.1.4.4 Pages – stránky a Templates – šablóny

Stránka je hierarchicky najvyšším komponentom. Ich účelom je spracovávanie aplikačnej logiky. Výsledky výpočtov / prebraných dát potom presúvajú do šablóny. V mojej práci sú to najpodstatnejšie komponenty, keďže umožňujú testovanie dát, vďaka ktorému je jednoducho možné upravovať nižšie komponenty.

Každá šablóna je odvodená od názvu stránky, ku ktorej patrí. Určuje rozloženie organizmov, molekúl a dodáva im kontext.

V rámci práce boli vytvorené a použité tieto stránky a k nim priradené šablóny:

- *MainPage* a *MainPageTemplate:* sa starajú o zobrazovanie obsahu pre hlavné stránky, Spracúvajú obrázok na pozadí, zobrazovaný text a obsah rolovacej ponuky. Dáta sú jednoducho mapované pomocou tzv. *route params*, kde si stránka preberie kontext, kde sa nachádza a podľa toho dynamicky mapuje obsah. Nie je teda nutné písať zvlášť nový kód pre každú stránku, čo výrazne zmenšuje objem nutného kódu a zvyšuje výslednú efektivitu a výkonnosť aplikácie.
- *ExcerptDetailPage* a *ExcerptDetailTemplate:* Ako každá stránka, svoje dáta mapujú pomocou *route params*. Stránka slúži pre zobrazenie dynamického detailu, ak užívateľ zvolí jednu možnosť rolovacej ponuky.
- *FacultyInfoPage* a *FacultyInfoTemplate:* Zastrešujú pod sebou rýchly prehľad pre jednotlivé fakulty.
- *ModulesListPage* a *ModulesListTemplate:* Zobrazujú jednotlivé študijné odbory v závislosti od toho, aká bola zvolená fakulta a stupeň štúdia (bakalársky, magisterský, doktorský).
- *FieldDetail* a *FieldDetailTemplate:* Slúžia k zobrazovaniu detailných informácii a zvolenom odbore, poskytujú rýchly náhľad o forme a type štúdia, popis odboru, predmety a možné uplatnenie po vyštudovaní.
- *NotFoundPage:* Je predvolenou stránkou v prípade, že sa užívateľ omylom (alebo zámerne) naviguje na stránku, ktorá neexistuje. V rámci dobrého UX je užívateľovi ponúknutá možnosť ísť o krok naspäť.

```
// @ts-ignore
import React, { Component } from 'react';
import MainPageTemplate from '../templates/MainPageTemplate';

import { RouteComponentProps } from 'react-router';

import mainPageData from '../../data/MainPageData';
import excerpts from '../../data/NavExcerptData';

type Props = {
  tab: string;
};

class MainPage extends Component<RouteComponentProps<Props>> {
  render() {
    const { tab } = this.props.match.params;

    return (
      <MainPageTemplate
        firstHeadingText={mainPageData[tab].firstHeadingText}
        secondHeadingText={mainPageData[tab].secondHeadingText}
        backgroundImage={mainPageData[tab].backgroundImage}
        excerpts={excerpts[tab]}
      />
    );
  }
}

export default MainPage;
```

Obrázok 6: Ukážka typického stránkového komponentu, kde je mapovaná logika a predávaná nižšie do šablóny.

```
class MainPageTemplate extends Component<Props> {
  render() {
    const {
      firstHeadingText,
      secondHeadingText,
      backgroundImage,
      excerpts,
    } = this.props;

    const renderExcerpts = excerpts.map((excerpt: Excerpt) => (
      <NavExcerpt
        key={excerpt.heading}
        heading={excerpt.heading}
        description={excerpt.description}
        thumbnail={excerpt.thumbnail}
        to={excerpt.to}
      />
    ));

    return (
      <PageLayoutWrapper
        firstHeadingText={firstHeadingText}
        secondHeadingText={secondHeadingText}
        backgroundImage={backgroundImage}
      >
        {renderExcerpts}
      </PageLayoutWrapper>
    );
  }
}

export default MainPageTemplate;
```

Obrázok 7: Ukážka typickej šablóny, preberá sformátované dáta zo stránky a určuje rozloženie ostatných komponentov na stránke.

5.2 Statické zdroje

Obsiahnuté v zložke *assets*, sú rozdelené podľa sekcii, kde sa využívajú. Jedná sa o:

- *Backgrounds:* ďalej rozdelené na *excerpt-detail-pages,* v ktorých sa nachádzajú obrázky pre detaily a *mainpage* obsahujúce obrázky vo vyššom rozlíšení pre hlavné stránky
- *Thumbnails:* Jedná sa o zdroje pre navigačné kartičky komponentu *NavExcerpt.* Ich štruktúra zodpovedá konkrétnej hlavnej stránke a sekcii, kde sú tieto obrázky využité.
- *UTB Logo:* je nezaradenou položkou a jedná sa o logo viditeľné vo vrchnej časti hlavnej stránky

5.3 Dáta

Aplikácia samotná je stavaná tak, aby bola v budúcnosti obohatená o akýkoľvek CMS. Tomu teda zodpovedá aj štruktúra dát, ktoré sú statické, jednoducho mapovateľné a rozdelené podľa sekcii, kde sú využité. To umožňuje jednoducho upravovať obsah aplikácie bez zásahu do samotného kódu. Je teda jednoduché pridávať, meniť, či vymazávať obsah a nepoškodiť tým nijakým spôsobom aplikáciu samotnú. Dáta boli poskytnuté marketingovým oddelením UTB a nachádzajú sa v zložke *data*. Jedná sa konkrétne o:

- *ExcerptDetailData:* Obsahujú v sebe importy statických zdrojov podstatných pre danú detailnú stránku. Samotná premenná *ExcerptDetailData* je potom štruktúrovaná do rôznych objektov, rozdelených podľa hlavných stránok a ich príslušných sekcii. Konečný obsah jednej položky sa skladá z nadpisu, textu a pozadia, ktoré sú dynamicky mapované.

```
const ExcerptDetailPageData = {
  univerzita: {···
  },
  studium: {···
  },
  mesto: {···
  },
  'navstiv-nas': {
    'dny-otevrenych-dveri': [
      {
        header: 'Navštiv nás!',
        text:
          'Nejlepší příležitostí k návštěvě univer
        background: DodDatesBackground,
      },
      {
        header: 'Dny otevřených dveří',
        text:
          'Den otevřených dveří na UTB (na FMK pou
        background: DodBackground,
      },
```

Obrázok 8: Ukážka dát pre detail sekcie „mesto“ a prvku „navstiv-nas“

- *MainPageData*: Obsahujú dáta pre hlavné stránky, skladajú sa z textu a príslušného obrázka pre danú sekciu

```
const MainPageData = {
  univerzita: {
    firstHeadingText: 'PROČ STUDOVAT NA',
    secondHeadingText: 'UTB?',
    backgroundImage: UniversityBackground,
  },
```

Obrázok 9: Ukážka dát pre hlavnú stránku „univerzita“

- *NavExcerptData:* Tvoria obsahovú štruktúru pre samotnú rolovaciu ponuku, dáta v premennej sú rozdelené podľa stránky, kde sa užívateľ nachádza. Samotné objekty, majú nasledovné vlastnosti: *isOnTop:* určuje zaoblenie rohov, *heading:* nadpis, *description:* text, *thumbnail:* obrázok, *to:* určuje kam bude komponent užívateľa presmerovávať

```
const NavExcerptData = {
  univerzita: [
    {
      isOnTop: true,
      heading: 'NABÍDKA STUDIA',
      description: 'Vyber si z bohaté nabídky stud
      thumbnail: StudyOfferThumbnail,
      to: '/univerzita/nabidka-studia',
    },
    {
      isOnTop: false,
      heading: 'ATRAKTIVNÍ PROSTŘEDÍ',
      description: 'Vzdělávej se v moderním a špič
      thumbnail: AttractiveEnvironmentThumbnail,
      to: '/univerzita/atraktivni-prostredi',
    },
```

Obrázok 10: Ukážka dát pre rolovaciu ponuku sekcie „univerzita“

- *StudyModules:* Zoznam všetkých odborov, ktoré je možné študovať na UTB. Štrukturálne rozdelenie je podľa fakulty, a stupni štúdia. Po tomto mapovaní je už možné dostať sa k jednotlivým odborom, ktoré v sebe uchovávajú názov, popis, predmety, možné uplatnenie a rýchly detail daného odboru.

```
const studyModules = {
  'fakulta-technologicka': {
    'bakalarske-studium': [
      {
        title: 'Chemie a analýza potravin',
        to: 'chemie-a-analyza-potravin',
        description:
          'Nestačí vám jen „mít rád jídlo“, vy ho
        subjects: [
          'Obecná a anorganická chemie',
          'Organická chemie I a II',
          'Produkce potravinářských surovin',
          'Biochemie',
          'Základy senzorické analýzy',
          'Chemie bioaktivních látek',
          'Obecná a potravinářská mikrobiologie',
          'Legislativa v potravinářství',
          'Chemie potravin',
          'Potravinářské technologie a biotechnol
          'Chemická informatika',
          'Instrumentální metody v analýze potrav
        ],
        jobs: [
          'Potravinářský průmysl',
          'Státní správa',
          'Kontrolní instituce',
          'Výzkumné a vývojové instituce',
        ],
        additionalInfo: {
          faculty: 'Fakulta technologická',
          programme: 'Technologie a hodnocení pot
          type: 'Bakalářské',
          language: 'Čeština',
          form: 'Prezenční, Kombinovaná',
        },
```

Obrázok 11: Ukážka dát pre technologickú fakultu, bakalársky stupeň štúdia a odbor „Chemie a analýza potravin“

5.4 Architektúra PWA

Knižnica React a spôsob, akým je navrhnutá štruktúra dát pokrývajú rozšíriteľnosť a jednoduchú údržbu. K pokrytiu zvyšných požiadaviek, ako natívnosť a offline podpora bolo nutné navrhnúť architektúru v rámci PWA. Toto bolo dosiahnuté pomocou správneho návrhu aplikačného manifestu a *Service Worker* implementácie.

5.4.1 Aplikačný manifest

Je jednou z podmienok správnej implementácie PWA. V mojej práci som ho prispôsobil potrebám konkrétnej aplikácie a splnil tak podmienky inštalovateľnosti, menovite názov aplikácie, rôzne veľkosti ikon, počiatočná url, *standalone display* mód, orientácia a farba pozadia. Kompletná implementácia manifestu je na obr. 13.

Keďže iOS v dobe písania tento manifest plne nepodporuje, bolo nutné do úvodného indexu vložiť určité html *meta* značky, ktoré túto funkcionalitu nahrádzajú, ako je uvedené na obr. 12. Jedná sa zväčša o ikony, názov aplikácie a štýl zobrazenia, ak je aplikácia spustená z domovskej obrazovky.

```
<meta
  name="viewport"
  content="width=device-width, initial-scale=1, shrink-to-fit=yes"
/>
<meta name="theme-color" content="#141B2C" />
<meta
  name="apple-mobile-web-app-status-bar-style"
  content="black-translucent"
/>
<meta name="apple-mobile-web-app-title" content="Studuj UTB" />
<meta name="apple-mobile-web-app-capable" content="yes" />
```

Obrázok 12: Ukážka niektorých „meta“ značiek potrebných k správnej funkcionalite PWA pre iOS. Ďalej sú doplnené o ikony a spúšťacie obrazovky.

```
{
  "short_name": "Studuj UTB",
  "name": "",
  "icons": [
    {
      "src": "icons/android/android-launchericon-512-512.png",
      "sizes": "512x512"
    },
    {
      "src": "icons/android/android-launchericon-192-192.png",
      "sizes": "192x192"
    },
    {
      "src": "icons/android/android-launchericon-144-144.png",
      "sizes": "144x144"
    },
    {
      "src": "icons/android/android-launchericon-96-96.png",
      "sizes": "96x96"
    },
    {
      "src": "icons/android/android-launchericon-72-72.png",
      "sizes": "72x72"
    },
    {
      "src": "icons/android/android-launchericon-48-48.png",
      "sizes": "48x48"
    }
  ],
  "start_url": ".",
  "display": "standalone",
  "theme_color": "#141B2C",
  "background_color": "#141B2C",
  "orientation": "portrait-primary"
}
```


Obrázok 13: Aplikačný manifest novej aplikácie Studuj UTB

5.4.2 Service Workers

Poslednou významnou požiadavkou je podpora offline funkcionality, ktorá je umožnená práve vďaka novej technológii *Service Workers.* Od novších verzi skriptu *Create React App* je podpora pre SW zabudovaná a v *App* komponente je nutné zavolať funkciu *sw.register().* To spustí životný cyklus SW, počas ktorého sa nainštaluje, aktivuje, kontroluje, či už nejaký SW existuje a v neposlednom rade ukladá statické zdroje aplikácie do lokálnej pamäte zariadenia. Stratégia pre ukladanie, ktorú som zvolil pre túto aplikáciu je ukladanie všetkých zdrojov. Vo väčšine prípadov toto nie je ideálne z hľadiska veľkosti a náročnosti, avšak vzhľadom na povahu aplikácie a veľkosť zdrojov, ktoré mi boli poskytnuté som sa pre čo najvyššiu efektivitu rozhodol zvoliť predvolenú stratégiu a tým pádom uložiť všetky zdroje.

Kód samotného SW zistí, či je prehliadač schopný podporovať túto funkcionalitu. Ak áno, začne inštaláciu, počas ktorej ukladá statické zdroje aplikácie. V konečnej fázy skontroluje, či nie je dostupný novší obsah a podľa toho je tento obsah buď ponechaný, alebo aktualizovaný za podmienky, že neexistuje ďalšia inštancia aplikácie.

```
function registerValidSW(swUrl: string, config?: Config) {
  navigator.serviceWorker
    .register(swUrl)
    .then(registration => {
      registration.onupdatefound = () => {
        const installingWorker = registration.installing;
        if (installingWorker == null) {
          return;
        }
        installingWorker.onstatechange = () => {
          if (installingWorker.state === 'installed') {
            if (navigator.serviceWorker.controller) {
              console.log(
                'New content is available and will be used when all ' +
                  'tabs for this page are closed. See http://bit.ly/CRA-PWA.'
              );
              if (config && config.onUpdate) {
                config.onUpdate(registration);
              }
            } else {
              console.log('Content is cached for offline use.');
              if (config && config.onSuccess) {
                config.onSuccess(registration);
              }
            }
          }
        };
      };
    })
    .catch(error => {
      console.error('Error during service worker registration:', error);
    });
}
```

Obrázok 14: Ukážka registračnej funkcie SW, počas ktorej sa ukladajú statické zdroje.

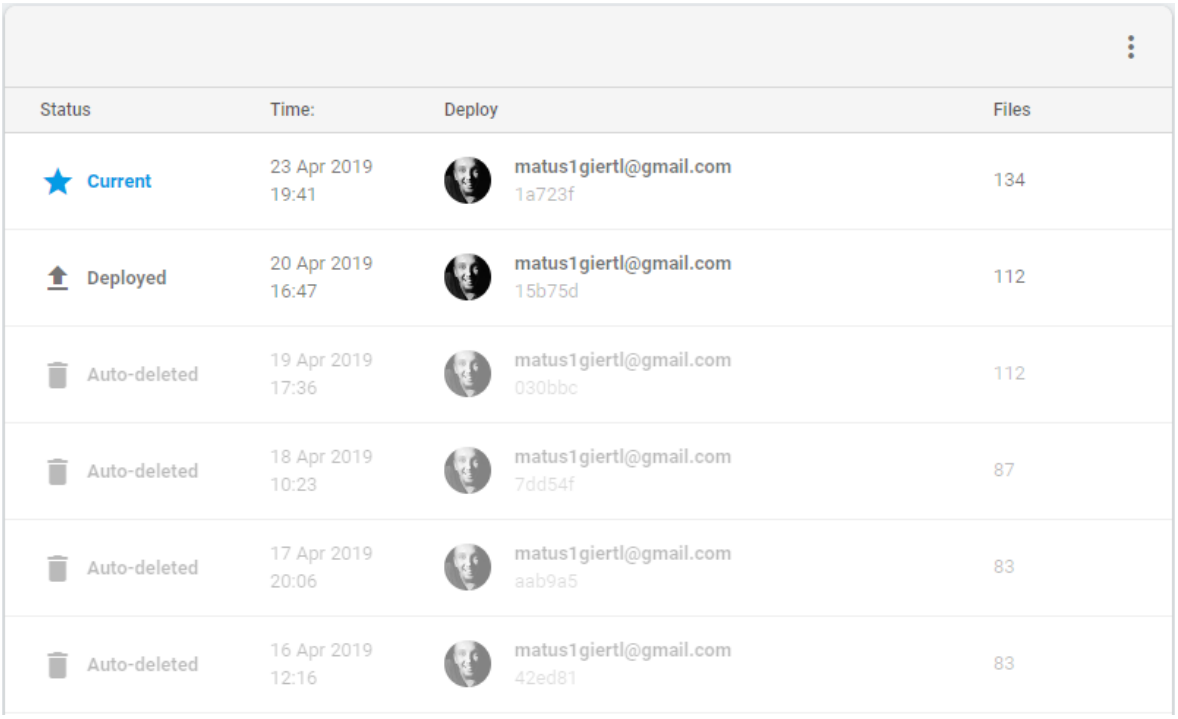
5.5 Deployment a hosting

Aplikácia bola počas svojho vývoja verzovaná pomocou systému Git a zdrojový kód bol nahraný na úložisko GitHub, ktoré umožňuje, mimo úložiska, jednoduchú manipuláciu s týmto verzovacím systémom. Je teda možné prezerať históriu vývoja od jeho počiatku, až po koniec. Kód, ktorý je odovzdaný v rámci tejto bakalárskej práce sa nachádza vo vetvi *bachelors.*
Aktuálny repozitár je dostupný na tejto adrese: https://github.com/MattGiertl/studuj-utb

Hosting bol vyriešený pomocou Google služby *Firebase.* Primárne určený pre mobilné aplikácie, avšak jednoducho použiteľný aj pre web, poskytuje množstvo nástrojov pre správu a deployment. V súčasnosti je využitá iba hosting služba.

Firebase je integrovaný za pomoci inštalačneho skriptu a Firebase CLI. Aktuálna vývojová verzia. Firebase umožňuje jednoduchú správu nad jednotlivými verziami aplikácie.
V rámci súčasného nastavenia je po každej tretej verzii posledná mazaná v rámci šetrenia dostupného priestoru a je možné vrátiť sa o verziu späť v prípade nepodareného release. Tieto nastavenia sa dajú ľubovoľne meniť bez obmedzenia.
Najnovšia vývojová verzia je dostupná na adrese: https://studujutb.firebaseapp.com/#/univerzita

Pre release aktuálnej verzie na hosting som vytvoril vlastný skript *build_develop.sh* v zložke *scripts,* ktorého účelom je aktivácia vetvy *development*, stiahnutie najnovších zmien, inštalácia závislostí, spustenie buildu React aplikácie a odoslanie výslednej zložky na hosting pomocou príkazu *firebase deploy.*



Obrázok 15: Ukážka posledných troch verzii aplikácie.

```
1  git checkout develop
2  git pull
3  yarn
4  yarn build
5  firebase deploy
```

Obrázok 16: Skript build_develop.sh používaný k odoslaniu aplikácie na server.

5.6 Zabezpečenie

Aplikácia nepoužíva žiadny vstup od užívateľa (zadávanie dát) k správnemu pracovaniu a nevyužíva ani senzitívne informácie. Bezpečnosť je však stále jednou z priorít ako knižnice React, tak PWA.

React samotný obsahuje ochranu pred XSS útokmi. Predtým, ako zobrazí element na stránke, vložené html je dynamicky kontrolované.

Poslednou podmienkou správnej implementácie PWA je HTTPS protokol. Bez neho aplikácia nebude považovaná za PWA a nebude podporovať užívateľa v tom, aby si ju nainštaloval na svoju domovskú obrazovku. To je najmä z dôvodu Service Workers. Ak by ich prípadný útočník napadol a odstránil, prípadne výrazne modifikoval, mohol by s adostať k citlivým informáciám užívateľa, prípadne by dokázal znefunkčniť celú aplikáciu.

6 NÁVRH UŽÍVATEĽSKÉHO ROZHRANIA

Z potreby obnovy pôvodnej aplikácie vznikla aj nutnosť nového, pútavého dizajnu. Tento vznikal zapojením študentky FMK Bc. Markéty Fajovej.

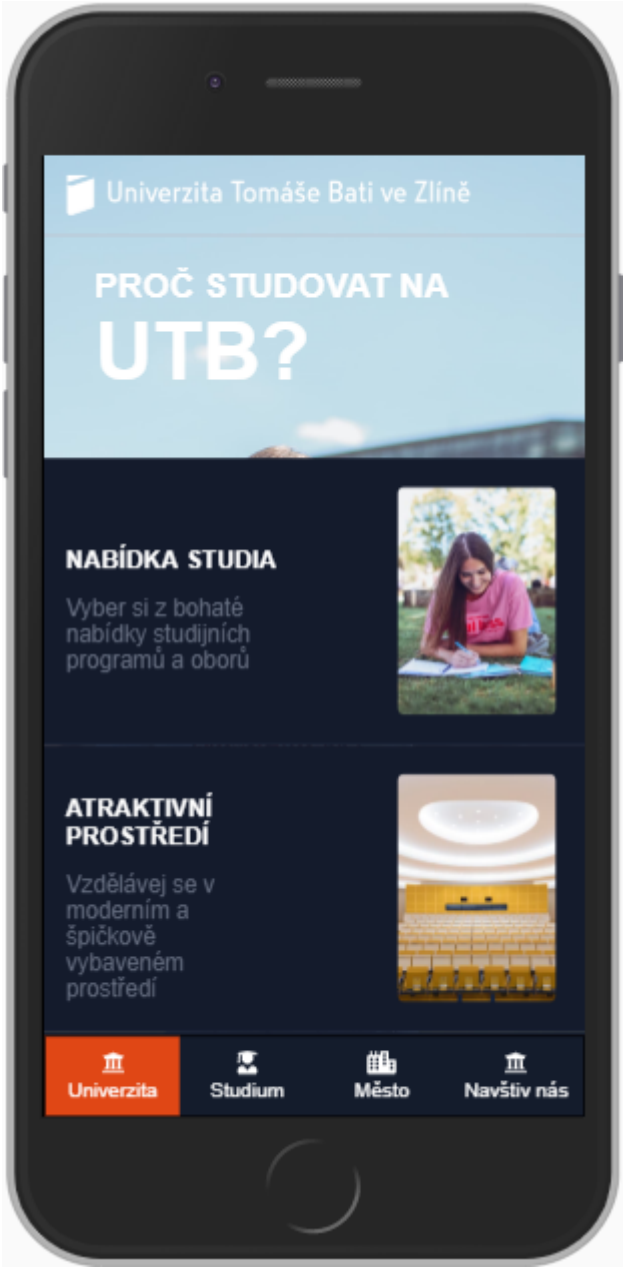
6.1 Hlavná stránka

Slúži ako úvodná obrazovka každej sekcie, Užívateľ je pútaný obrázkom, zodpovedajúcim sekcie a pomocou navigačnej šípky je pozývaný k vysunutiu rolovacej ponuky. Každá stránka zároveň obsahuje logo UTB a určitú formu nadpisu. V rámci spodnej navigácie existujú štyri hlavné stránky, poskytujúce vlastný obsah a informácie o univerzite, štúdiu, meste a možnostiach návštevy.

Obrázok 17: Ukážka hlavnej stránky „Univerzita“

6.2 Rolovacia ponuka

Nachádza sa na hlavnej obrazovke, užívateľovi sa zobrazí po posunutím hlavnej obrazovky nahor, obsahuje navigačné karty informujúce o rôznych prednostiach, vlastnostiah a možnostiach, ktoré daná sekcia ponúka. Je to, okrem spodnej navigácie, primárny spôsob získavania informácii pomocou tejto aplikácie.



Obrázok 18: Rolovacia ponuka sekcie „Univerzita“

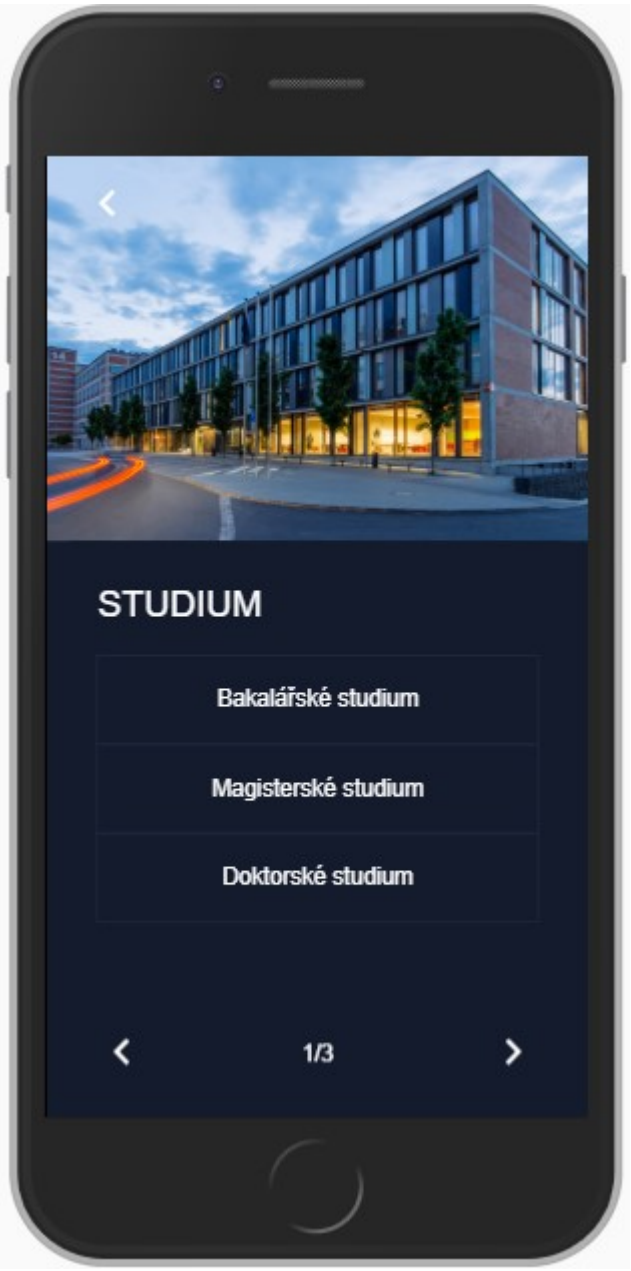
6.3 Stránka informačného detailu

Je stránkou, ktorá obsahuje detailné informácie o navigačnej karte, ktorú si užívateľ zvolil. Samotná stránka má štruktúru akejsi prezentácie, ktorou je užívateľ schopný navigovať sa nielen pomocou gest, ale aj navigačných šípok, ktoré buď posúvajú užívateľa do ďalšej časti, alebo vracajú o stránku späť. Pre prehľadnosť bol taktiež pridaný indikátor počtu stránok daného detailu.

Obrázok 19: Ukážka detailu jednej z navigačných kariet.

6.4 Stránka fakulty

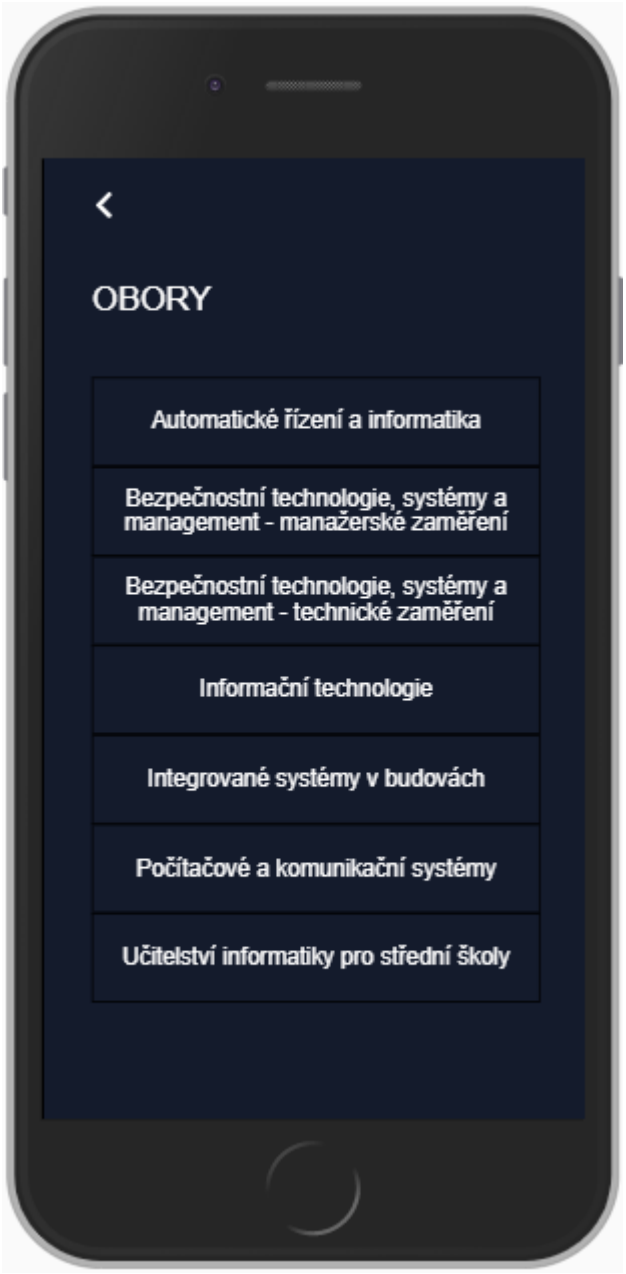
Je akousi špeciálnou úpravou predošlej stránky, kedy okrem detailu o jednotlivých fakultách sú pridané tzv. „fakultné tlačidlá“, ktoré poskytujú prístup k odborom na rôznych stupňoch štúdia, v rámci zvolenej fakulty. Užívateľ sa taktiež môže navigovať gestami, prípadne navigačnými šípkami.



Obrázok 20: Ukážka stránky Fakulty technologickej s „fakultnými tlačidlami“

6.5 Stránka zoznamu odborov

Na tejto stránke je zobrazený zoznam odborov v rámci zvolenej fakulty a stupňa štúdia. Každé tlačidlo vedie k detailnejšiemu popisu jednotlivých odborov.



Obrázok 21: Ukážka odborov na FAI pre magisterské štúdium.

6.6 Stránka detailu odboru

Poskytuje užívateľovi všetky podstatné informácie o zvolenom odbore, medzi ktoré patrí rýchly náhľad, popis odboru, predmetov a možností zamestnania po štúdiu.

Obrázok 22: Krátky náhľad detailu odboru „Informačné technológie“

ZÁVER

V rámci mojej práce bola navrhnutá a následne implementovaná aplikácia určená uchádzačom o štúdium na Univerzite Tomáše Bati ve Zlíně. Táto aplikácia slúži ako prvé zoznámenie potencionálnych uchádzačov o štúdium a zároveň poskytuje dôležité a rýchlo dostupné informácie.

Po niekoľkých sedeniach s členom marketingového oddelenia UTB som bol schopný efektívne zozbierať ako funkcionálne, tak i nefunkcionálne požiadavky a vytvoriť prípady použitia. Medzi najvýznamnejšie patrilo: poskytovanie užitočných informácii uchádzačom, multiplatformnosť, offline podpora, škálovateľnosť, udržiavateľnosť a vzhľad natívnej mobilnej aplikácie. Väčšina týchto požiadaviek vznikala najmä z nedostatkov pôvodnej aplikácie, postavenej na technológii Xamarin a dátovej vrstve poskytovanej pomocou systému STAG. Kombinácia týchto riešení sa ukázala byť v priebehu času nevhodná a vznikla potreba vyvinúť novú aplikáciu postavenú na rovnakej myšlienke, no s moderným vzhľadom a vlastnosťami popísanými vyššie.

Pre riešenie týchto problémov som okrem PWA technológii zvolil populárnu knižnicu React, ktorá dokonale dopĺňa prednosti PWA a umožnila jednoduchú udržiavateľnosť a škálovateľnosť aplikácie. V rámci zrýchlenia vývoja a zvýšenia kvality boli pridané ďalšie podporné knižnice. Architektúra projektu je postavená tak, aby boli jednotlivé komponenty čo najviac dynamické. Aplikáciu som teda rozdelil na päť typov obsahových stránok, ktoré si dokážu dynamicky mapovať dáta podľa toho, v akej časti aplikácie sa užívateľ nachádza. Vďaka tomuto prístupu som dokázal zefektívniť od počiatku výkonnosť aplikácie a znížiť celkový objem dát, ktorý je nutný stiahnuť pri prvom spustení a tým som dokázal výrazne zvýšiť výkonnosť aplikácie a užívateľský zážitok. Týmto som teda dokázal pokryť škálovateľnosť a automaticky aj multiplatformnosť.

Návrh dátovej vrstvy pozostáva zo statických súborov, ktoré sú v rámci projektu delené podľa sekcie aplikácie, kde sú využívané. Od počiatku bola ich štruktúra navrhovaná tak, aby boli v budúcnosti bezbolestne nahradené systémom pre správu obsahu.

Pokrytím väčšiny požiadaviek som bol pripravený implementovať PWA funkcionalitu, ktorá zaručuje natívnosť a offline podporu. Aby bola aplikácia považovaná za PWA, bolo nutné splniť viacero podmienok, medzi ktoré patria súčasti ako správne vyplnený aplikačný manifest a Service Workers, ktoré zaručujú ukladanie statických zdrojov a v prípade tejto aplikácie i obsahu do lokálnej pamäte zariadenia.

Poslednou súčasťou implementácie bolo nasadenie aplikácie. Pre toto bola využitá služba Firebase poskytujúca pokročilé nástroje pre správu webovej aplikácie. Pre zjednodušenie nasadenia na server bol vytvorený skript, ktorý automaticky aktualizuje obsah, doinštaluváva chýbajúce závislosti, pripraví aplikáciu a konečne ju nasadí.

Aplikácia bola v priebehu svojho vývoja pravidelne verzovaná a zálohovaná pomocou verzovacieho systému Git a repozitár bol uložený v online službe GitHub.

Kombináciou týchto technológii a služieb som bol teda schopný vytvoriť aplikáciu, ktorá spĺňa všetky požiadavky zadané marketingovým oddelením UTB. Usúdil som teda, že voľba technológie PWA za sprievodu knižnice React bola ideálnym kandidátom pre tento typ aplikácie, keďže sa dokázala efektívne vyvarovať existujúcim nedostatkom pôvodnej aplikácie a v konečnom dôsledku sa stal aplikáciou hodnou miesta na domovskej obrazovke užívateľa.

Aplikácia bola stavaná mobile-first prístupom a tým pádom je pripravená byť rozšírená o desktopovú verziu. Desktopová verzia teda bude vychádzať z rovnakého zdrojového kódu, ako mobilná a to vďaka technikám responzívneho vývoja webových aplikácii. Bude teda možné osloviť užívateľov nielen mobilných zariadení, ale i užívateľov „klasického“ desktopu, či tabletu na ľubovoľnej platforme.

Dátová štruktúra a dynamické mapovanie dát v rámci jednotlivých komponent taktiež pripravujú aplikáciu na budúce rozšírenie o systém správy obsahu, ako je Flamelink, Drupal, Joomla!, alebo na univerzite používaný WordPress.

Výsledkom mojej práce je teda aplikácia, ktorá úspešne rieši doterajší problém marketingového oddelenia UTB a spĺňa všetky požiadavky nastolené vo fáze analýzy. Úspešne dokáže reprezentovať univerzitu, vyzdvihuje jej prednosti a poskytuje relevantné informácie pre uchádzačov. Vďaka dynamickej architektúre a modernému vzhľadu ponúka výrazne vyšší užívateľský zážitok, ako ponúkala pôvodná aplikácia a je teda schopná ju okamžite nahradiť. Splnením všetkých požiadaviek a berúc v úvahu konečný výsledok, som teda schopný s istotou tvrdiť, že zadanie bolo úspešne splnené.

ZOZNAM POUŽITEJ LITERATÚRY

1. SONMEZ, John. What is Mobile Development?. *Simple Programmer* [online]. [cit. 2019-05-07]. Dostupné z: https://simpleprogrammer.com/what-is-mobile-development/#native-development

2. *Mobile Operating System Market Share Worldwide* [online]. 2016 [cit. 2019-05-07]. Dostupné z: http://gs.statcounter.com/os-market-share/mobile/worldwide

3. *Hybrid vs Native app development* [online]. 2019 [cit. 2019-05-07]. Dostupné z: https://medium.com/@ScaffoldDigital/hybrid-vs-native-app-development-b3467ee101d9

4. *Application Fundamentals* [online]. 2019 [cit. 2019-05-07]. Dostupné z: https://developer.android.com/guide/components/fundamentals

5. *Distribution dashboard* [online]. 2019 [cit. 2019-05-07]. Dostupné z: https://developer.android.com/guide/components/fundamentals

6. *Publish your app* [online]. 2019 [cit. 2019-05-07]. Dostupné z: https://developer.android.com/about/dashboards

7. YADAV, Gopal. Best Google Play Store Alternatives. *Medium* [online]. 2018 [cit. 2019-05-07]. Dostupné z: https://medium.com/pen-bold-kiln-press/best-google-play-store-alternatives-30c759de1c26

8. DAVENPORT, Alba. The History of iOS App Development. *Sutori* [online]. [cit. 2019-05-07]. Dostupné z: https://www.sutori.com/story/the-history-of-ios-app-development--hzFfwkD2KYLaa5WrxsrUFGMh

9. LARDINOIS, Frederic. Apple Launches Swift, A New Programming Language For Writing iOS And OS X Apps. *TechCrunch* [online]. 2014 [cit. 2019-05-07]. Dostupné z: https://techcrunch.com/2014/06/02/apple-launches-swift-a-new-programming-language-for-writing-ios-and-os-x-apps/

10. *About Swift* [online]. [cit. 2019-05-07]. Dostupné z: https://swift.org/about/

11. *Swift* [online]. [cit. 2019-05-07].

12. GOYAL, Arun. 5 Advantages of Cross Platform Mobile App Development. *Digital Doughnut*[online]. 2017 [cit. 2019-05-07]. Dostupné z: https://www.digitaldoughnut.com/articles/2017/april/5-advantages-of-cross-platform-mob-app-development

13. *Flutter vs React Native vs Xamarin for Cross Platform Development* [online]. 2018 [cit. 2019-05-07]. Dostupné z: https://hackernoon.com/flutter-vs-react-native-vs-xamarin-for-cross-platform-development-5f92cfb178ff

14. MANCHANDA, Amit. Where Do Cross-Platform App Frameworks Stand in 2019?. *Net Solutions*[online]. 2019 [cit. 2019-05-07]. Dostupné z: https://www.netsolutions.com/insights/cross-platform-app-frameworks-in-2019/

15. *React Native Repository* [online]. [2015] [cit. 2019-05-07]. Dostupné z: github.com/facebook/react-native

16. RAJPUT, Mehul. The Advantages and Disadvantages of Using React Native as Cross-Platform App Development. *Mind Inventory* [online]. 2018 [cit. 2019-05-07]. Dostupné z: www.mindinventory.com/blog/pros-cons-using-react-native/

17. EVKOVSKI, Blagoja. React Native: What it is and how it works. *Medium* [online]. 2017 [cit. 2019-05-07]. Dostupné z: https://medium.com/we-talk-it/react-native-what-it-is-and-how-it-works-e2182d008f5e

18. *React Native Showcase* [online]. [cit. 2019-05-07]. Dostupné z: http://facebook.github.io/react-native/showcase

19. BELLINASO, Marco. Flutter: the good, the bad and the ugly. *Medium* [online]. 2018 [cit. 2019-05-07]. Dostupné z: https://medium.com/asos-techblog/flutter-vs-react-native-for-ios-android-app-development-c41b4e038db9

20. KEPPLE, Dru. What Is Dart, and Why Should You Care?. *Envatotuts+* [online]. 2012 [cit. 2019-05-07]. Dostupné z: https://code.tutsplus.com/articles/what-is-dart-and-why-should-you-care--active-11233

21. AMADEO, Ron. Google's "Fuchsia" smartphone OS dumps Linux, has a wild new UI: Taking a look at Google's mysterious third operating system. *Ars technica* [online]. 2017 [cit. 2019-05-07]. Dostupné z: https://code.tutsplus.com/articles/what-is-dart-and-why-should-you-care--active-11233

22. *Flutter Showcase* [online]. [cit. 2019-05-07]. Dostupné z: https://flutter.dev/showcase

23. GRIFFITH, Chris. What is Hybrid App Development?. *Ars techniva* [online]. 2019 [cit. 2019-05-07]. Dostupné z: https://ionicframework.com/enterprise/resources/articles/what-is-hybrid-app-development

24. *What is Ionic Framework?* [online]. [cit. 2019-05-07]. Dostupné z: https://ionicframework.com/docs/intro

25. RAMOS, Javier. Ionic 4: All you need to know. *Medium* [online]. 2019 [cit. 2019-05-07]. Dostupné z: https://medium.com/@javier.ramos1/ionic-4-all-you-need-to-know-d2b9627aaf03

26. *Ionic Framework* [online]. [cit. 2019-05-07]. Dostupné z: https://ionicframework.com/

27. BROWN, Rachel. What is a Progressive Web App and Why You Need One?. *Ionic Framework*[online]. [cit. 2019-05-07]. Dostupné z: https://ionicframework.com/enterprise/resources/articles/what-is-a-progressive-web-app-and-why-you-need-one

28. PEREZ, Sarah. Majority of U.S. consumers still download zero apps per month, says comScore. *TechCrunch* [online]. 2017 [cit. 2019-05-07]. Dostupné z: https://techcrunch.com/2017/08/25/majority-of-u-s-consumers-still-download-zero-apps-per-month-says-comscore/

29. SUTTER, Biran. Vrooom! Why Website Speed Matters. *Enterpreneur* [online]. 2017 [cit. 2019-05-07]. Dostupné z: https://www.entrepreneur.com/article/281986

30. ENGE, Eric. Mobile vs Desktop Traffic in 2019. *Stone Temple* [online]. 2019 [cit. 2019-05-07]. Dostupné z:

https://www.stonetemple.com/mobile-vs-desktop-usage-study/

31. WONG, Jason. Progressive Web Apps Will Impact Your Mobile App Strategy. *Gartner* [online]. 2017 [cit. 2019-05-07]. Dostupné z: https://www.gartner.com/en/documents/3645344

32. WONG, Jason. Flipkart triples time-on-site with Progressive Web App. *Google Developers*[online]. [cit. 2019-05-07]. Dostupné z: https://developers.google.com/web/showcase/2016/flipkart

33. OMSANI, Addi. A Tinder Progressive Web App Performance Case Study. *Medium* [online]. 2017 [cit. 2019-05-07]. Dostupné z: https://medium.com/@addyosmani/a-tinder-progressive-web-app-performance-case-study-78919d98ece0

34. *3 LIMITATIONS OF PROGRESSIVE WEB APPS* [online]. 2018 [cit. 2019-05-07]. Dostupné z: https://www.strv.com/blog/3-limitations-of-progressive-web-apps

35. *Google Developers* [online]. [cit. 2019-05-07]. Dostupné z: Introduction to Progressive Web App Architectures

36. *The Web App Manifest* [online]. [cit. 2019-05-07]. Dostupné z: https://developers.google.com/web/fundamentals/web-app-manifest/

37. *Web App Manifest* [online]. [cit. 2019-05-07]. Dostupné z: https://developer.mozilla.org/en-US/docs/Web/Manifest

38. *How to make PWAs installable* [online]. [cit. 2019-05-07]. Dostupné z: https://developer.mozilla.org/en-US/docs/Web/Progressive_web_apps/Installable_PWAs

39. WONG, Peter. Add to Home Screen. *Google Developers* [online]. [cit. 2019-05-07]. Dostupné z: https://developers.google.com/web/fundamentals/app-install-banners/

40. *Add to Home screen* [online]. [cit. 2019-05-07]. Dostupné z: https://developer.mozilla.org/en-US/docs/Web/Progressive_web_apps/Add_to_home_screen

41. *Offline mode* [online]. [cit. 2019-05-07]. Dostupné z: https://docsify.now.sh/pwa

42. *Making PWAs work offline with Service workers* [online]. [cit. 2019-05-07]. Dostupné z: https://developer.mozilla.org/en-US/docs/Web/Progressive_web_apps/Offline_Service_workers

43. *Introduction to Service Worker* [online]. [cit. 2019-05-07]. Dostupné z: https://developer.mozilla.org/en-US/docs/Web/Progressive_web_apps/Offline_Service_workers

44. *Introduction to Service Worker* [online]. [cit. 2019-05-07]. Dostupné z: https://developers.google.com/web/ilt/pwa/introduction-to-service-worker

45. *Introduction to Push Notifications* [online]. [cit. 2019-05-07]. Dostupné z: https://developers.google.com/web/ilt/pwa/introduction-to-push-notifications

46. DUBEY, Praaven. What is Progressive Enhancement, and why it matters. *Medium*[online]. 2018 [cit. 2019-05-07]. Dostupné z: https://medium.freecodecamp.org/what-is-progressive-enhancement-and-why-it-matters-e80c7aaf834a

47. *Progressive enhancement* [online]. [cit. 2019-05-07]. Dostupné z: https://github.com/sylvainpolletvillard/pwa-cookbook/blob/master/static/pages/en/progressive-enhancement.md

48. FIRTMAN, Maximiliano. Progressive Web Apps on iOS are here. *Medium* [online]. 2018 [cit. 2019-05-07]. Dostupné z: https://medium.com/@firt/progressive-web-apps-on-ios-are-here-d00430dee3a7

49. FIRTMAN, Maximiliano. What's new on iOS 12.2 for Progressive Web Apps. *Medium* [online]. 2019 [cit. 2019-05-07]. Dostupné z: https://medium.com/@firt/whats-new-on-ios-12-2-for-progressive-web-apps-75c348f8e945

50. LEPAGE, Peter. Desktop Progressive Web Apps. *Google Developers* [online]. [cit. 2019-05-07]. Dostupné z: https://developers.google.com/web/progressive-web-apps/desktop

51. LOVE, Chris. Google Chrome Desktop Progressive Web App Add to Homescreen Capabilities. *Google Developers* [online]. 2019 [cit. 2019-05-07]. Dostupné z: https://love2dev.com/blog/chrome-desktop-pwa/

ZOZNAM POUŽITÝCH SYMBOLOV A SKRATIEK

API Application Programming Interface

CLI Command Line Interface

CMS Content Management System

CSS Cascading Style Sheets

DOM Documnet Object Model

FAI Fakulta aplikovanej informatiky

FMK Fakulta multimediálnych komunikácii

HTTPS Hypertext Transfer Protocol

JS JavaScript

PE Progressive Enhancement

PWA Progressive Web Apps

RN React Native

SEO Search Engine Optimisation

SDK Software Development Kit.

UI/UX User Interface/User Experience

UTB Univerzita Tomáše Bati

WWDC Worldwide Developers Conference.

XSS Cross Site Scripting

ZOZNAM OBRÁZKOV

ZOZNAM TABULIEK

ZOZNAM PRÍLOH

P1 CD s diplomovou prácou a súbory obsahujúce zdrojové kódy