ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE

Fakulta elektrotechnická

Katedra počítačů

Bakalářská práce

Multiplatformní aplikace pro sběr georeferencovaných dat

Multiplatform application for geo-referenced data collection

Jan Hrdý

Vedoucí: Ing. David Sedláček, Ph.D.

Studijní program: Softwarové inženýrství a technologie

Květen 2019

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

I. OSOBNÍ A STUDIJNÍ ÚDAJE

Příjmení: **Hrdý** Jméno: **Jan** Osobní číslo: **457800**

Fakulta/ústav: **Fakulta elektrotechnická**

Zadávající katedra/ústav: **Katedra počítačů**

Studijní program: **Softwarové inženýrství a technologie**

II. ÚDAJE K BAKALÁŘSKÉ PRÁCI

Název bakalářské práce:

Multiplatformní aplikace pro sběr georeferencovaných dat

Název bakalářské práce anglicky:

Multiplatform application for geo-referenced data collection

Pokyny pro vypracování:

Navrhněte a implementujte multiplatformní aplikaci pro upozornění na zajímavá místa v okolí.
Aplikace bude umožňovat uživatelům přihlašování a registraci. Aplikace umožní přidávat nové návrhy na zajímavá místa a bude informovat uživatele zajímavými místy v okolí (zobrazená na mapě). Při kliknutí na nějaké zajímavé místo, bude možnost přidat hodnocení. Aplikace bude mít i možnost nahrávat data na server pouze na wifi (tj. jistá forma offline módu) dle preference uživatele.
V návrhu aplikace postupujte podle metodiky UCD (User Center Design).
Výsledná aplikace bude testována z hlediska funkčnosti (například: přidávání dat a GPS lokace) a použitelnosti.
Navrhněte a implementujte vlastní serverové řešení pro sběr dat. Dále navrhněte a vytvořte synchronizační procedury, které budou zajišťovat využití těchto dat produkty ESRI, které jsou využívány pro analýzu geografických dat.

Seznam doporučené literatury:

[1] T. Lowdermilk, User-Centered Design, O'Reilly Media, 2013
[2] B. EisenmanLearning React Native: Building Native Mobile Apps with JavaScript, O'Reilly Media, 2016
[3] B. Fling, Mobile Design and Development, O'Reilly Media, 2009

Jméno a pracoviště vedoucí(ho) bakalářské práce:

Ing. David Sedláček, Ph.D., katedra počítačové grafiky a interakce FEL

Jméno a pracoviště druhé(ho) vedoucí(ho) nebo konzultanta(ky) bakalářské práce:

Datum zadání bakalářské práce: **13.02.2019** Termín odevzdání bakalářské práce: **24.05.2019**

Platnost zadání bakalářské práce: **20.09.2020**

Ing. David Sedláček, Ph.D. podpis vedoucí(ho) práce	podpis vedoucí(ho) ústavu/katedry	prof. Ing. Pavel Ripka, CSc. podpis děkana(ky)

III. PŘEVZETÍ ZADÁNÍ

Student bere na vědomí, že je povinen vypracovat bakalářskou práci samostatně, bez cizí pomoci, s výjimkou poskytnutých konzultací.
Seznam použité literatury, jiných pramenů a jmen konzultantů je třeba uvést v bakalářské práci.

Datum převzetí zadání	Podpis studenta

© ČVUT v Praze, Design: ČVUT v Praze, VIC

Prohlášení

„Prohlašuji, že jsem Bakalářskou práci vypracoval samostatně ve spolupráci s Ing. Davidem Sedláčkem, Ph.D. Prohlašuji, že jsem uvedl všechny použité zdroje v souladu s Metodickým pokynem o dodržování etických principů při tvorbě vysokoškolských závěrečných prací.“

V Praze dne 24. 5. 2019

Jan Hrdý

Poděkování

Děkuji Ing. Davidovi Sedláčkovi, Ph.D. za možnost tvořit mobilní aplikace v React Native. Děkuji za důležité připomínky a návrhy v mobilní aplikaci.

V Praze dne 24. 5. 2019

Jan Hrdý

Abstrakt

Tato bakalářská práce se zabývá návrhem, implementací a testováním mobilní aplikace pro reportování zajímavých míst. Aplikace má pomoci uživatelům k plánování tras nebo například průzkumu památek v okolí. Na začátku práce je představen jemný úvod do světa technologií: React a React Native. Poté se práce zabývá analýzou trhu, dostupných technologií či požadavků. Následně je rozebrán vzhled aplikace s představením možného prototypu. Dále je zpracován vývoj i uživatelské testování. Posledních pár slov se soustředí na to, jak dále postupovat ve vývoji aplikace, přičemž nechybí ani závěr. Součástí bakalářské práce je také tvorba serverového řešení s funkční databází.

Klíčová slova

Android, iOS, mobilní aplikace, React Native, Expo

Abstract

This bachelor thesis deals with design, implementation and testing of mobile application for reporting interesting places. The app is designed to help users plan routes or explore nearby landmarks. At the beginning of the work is a gentle introduction to the world of technology: React and React Native. Then the thesis deals with market analysis, available technologies or requirements. Subsequently, the appearance of the application with the introduction of a possible prototype is discussed. Furthermore, development and user testing are processed. The last few words focus on how to proceed in development, and there is also conclusion. Part of the thesis is also the creation of a server solution with a functional database.

Keywords

Android, iOS, mobile application , React Native, Expo

Obsah

Seznam obrázků

1 ÚVOD

Od historicky prvního telefonu uběhla již pěkná řádka let. Dříve měly telefony pouze ti nejbohatší lidé, avšak v dnešní době má chytrý telefon téměř každý člověk. Mezi dva největší giganty patří Apple s operačním systémem iOS. Dále potom Samsung, Huawei a další s operačním systémem Android.

Důležitou součástí mobilního zařízení je připojení k internetu. První mobilní sít v Československu byla už v roce 1991 a od té doby se rychlost připojení neustále zrychluje (2G, 3G, 4G) [1]. Dnes Vám chytrý telefon s neustálým připojením k internetu otevírá širokou škálu možností. Můžete komunikovat s ostatními prostřednictvím Emailového klienta, obyčejných SMS zpráv, různých sociálních sítí. Další oblíbenou činností je hraní her. Pokud ani to nemáte v plánu, můžete pomocí nativních prohlížečů brouzdat na internetu, popřípadě hledat a číst novinky ze světa. Nedílnou součástí každé moderní aplikace jsou notifikace, zjednodušeně upozornění na nové informace v telefonu.

I v dnešní moderní době, kdy se chytrým elektronickým zařízením věnuje čím dál tím větší pozornost, existují stále tací, kteří rádi tráví čas i jiným způsobem. Někteří z nás si rádi dopřejí sportovní pauzu, jiní například ve volném čase chodí do přírody, navštěvují kulturní místa nebo třeba rádi objevují nové společenské zvyky. Mezi zájmy těchto lidí patří možné procházky či turistické výpravy do nových míst v okolí. I tito lidé mají většinou po ruce chytrý telefon s přístupem internetu, například připojení přes mobilní data. Orientace v cizí krajině může být pro některé velmi obtížná i pokud mají přístup k internetu. Turisti by rádi poznávali nová místa, ale pokud neví kde je hledat, stává se cestování komplikací. Zdá se, že by bylo příhodné mít v telefonu mobilní aplikaci, která by například usnadnila plánování výletů po okolí. Každopádně, na trhu App Store a Google Play právě taková aplikace chybí. Aplikace, která by pomohla v neznámé krajině s orientací, která by poskytovala lidem přístup k mapě s různými zajímavými místy z okolí. Aplikace, která by pomohla lidem v rozhodování, kam se v přírodě vydat, poskytla patřičné informace o zajímavých místech v okolí a navíc nabídla možnost přidat neznámou kulturní památku do okolí.

Vzhledem k této situaci vzniká projekt mobilní aplikace „Chytré stezky“, která bude přístupná na iOS i Android. Bude vyvíjená v jazyce JavaScript, pomocí React Native. Hlavním cílem aplikace bude umožňovat rychlé vyhledávání informací o zajímavých místech z okolí, vzhledem k bodu na mapě, kde se právě nacházíte. Nebude chybět ani možnost přidat důležité místo z Vašeho okolí.

1.1 MOTIVACE

Osobně mě nejvíce na programování baví vývoj mobilních aplikací, ideálně psané v React Native. Líbí se mi ta možnost psát v jednom jazyce pro dva oddělené tábory. Dříve například museli vývojáři psát pro každou platformu zvlášť. Jenže doba se mění a mobilní vývoj se stává jednodušším. Stejně jako nám programátorům se ulehčuje život, stejně tak by bylo vhodné přijít s mobilní aplikací, která by ulehčila život ostatním. Vnitřně cítím motivaci pomoci lidem na cestách a usnadnit tak jejich rozhodování v neznámých krajinách. Je sice pravda, že většinu života trávím u počítače, každopádně i já občas vyrazím do přírody, do neznámých míst a rád bych měl tu možnost jednoduše zapnout aplikaci na telefonu a vybrat si adekvátní výlet na nějaké zajímavé místo v okolí. Bylo by podle mě ideální mít v telefonu aplikaci, která by lidem nabízela patřičné služby se sympatickým uživatelským rozhraním.

2 REACT

Vzhledem k faktu, že vývoj bude psán v React Native, bylo by na místě patřičnou technologii zpočátku řádně představit. Nejprve několik slov o samotném Reactu, dále je potom úvod do React Native a věcí, které s tím úzce souvisí.

Pokud nahlédneme do oblasti webových aplikací odkud React pochází, tak zjistíme, že posledních 10 let vývoje webových stránek se stávají populární Single page applications neboli webová aplikace, kdy se stránka vykreslí pomocí JavaSriptu a dále se už jen stahují data, která se dynamicky mění. Hlavní výhodou Single page applications je fakt, že přináší datovou úsporu. Pro návštěvníka stránky se obsah velice rychle dokáže překreslit, takže se zvyšuje uživatelskou přívětivost. Vzhledem k této skutečnosti začaly vznikat nové frameworky a knihovny jako Angular, VueJS, atd., které se snažily pomoc vývojářům v moderní tvorbě webových stránek [2]. Tyto technologické nástroje se snažili vývojářům poskytnout širokou škálu možností vývoje od začátku až do konce. Jenže právě to přineslo problémy. Byly ohromné a tím pádem bylo těžší přizpůsobit jejich chování dané situaci. Postupem času nastávaly komplikace s jejich zbytečnou složitostí. Toho si všimla i firma Facebook, která přišla s něčím trochu odlišným, přišla s novou technologií pod názvem React.

React se vydal trochu jinou cestou než třeba Angular či VueJS. Vzal si za cíl připravit knihovnu pouze pro tvorbu uživatelských rozhraní, jinak řečeno, React je knihovna jazyka JavaScript, která si z vrstevnaté architektury vzala na starost pouze produkci prezenční vrstvy. Hlavní myšlenkou je tvorba elementů podobných HTML, které mají vlastní funkcionalitu. React se obecně zabývá vykreslováním různých rozhraní, ať už mobilních či webových aplikací. Ze světa Reactu také pramení různé technologické vychytávky jako například JSX, které si představíme později v této kapitole. Postupem času se z pokrokových myšlenek Reactu začaly inspirovat i některé další frameworky či knihovny, kupříkladu Angular, VueJS, a další. React samotný ze začátku používala pouze firma Facebook, každopádně časem začaly React používat i další velké firmy jako například Netflix, Airbnb, a další. Z historického pohledu určitě stojí za zmínku fakt, že React přišel na svět v roce 2013 a o 2 roky později přišel Facebook s React Native.

Naučit se programovat v Reactu, popřípadě v React Native, není příliš těžké, přeci jen je to knihovna jazyka JavaScript, která se stará pouze o uživatelské rozhraní. React navíc má lehce pochopitelné API. Ale to vše je pouze základ, ostatní prvky tvoří jednotlivé stavební bloky tvořené programátory. Navíc každý stavební blok lze jednoduše vyměnit za jiný. To vše řadí React mezi hodně flexibilní knihovny [3].

2.1 REACT NATIVE

React zastává moto: Nauč se jednou, piš všude [4]. To naráží na skutečnost, že v roce 2015 přišel na svět React Native. React Native je framework pro tvorbu nativních aplikací v JavaScriptu. Zjednodušeně řečeno React Native používá celkový koncept Reactu pro tvorbu mobilních aplikací. Tvorba mobilní aplikace v React Native je multiplatformní. To znamená, že vyvíjíte kód v jazyce JavaScript pro iOS i Android zároveň. Není tedy potřeba se pro vývoj mobilní aplikace učit specifický jazyk, jako jsou Java, Kotlin, Objective-C či Swift. Navíc kód napsaný v React Native je velice podobný Reactu, každopádně jsou tam jisté odlišnosti. Celkový ekosystém je stejný, ale některé části ve vykreslování, v sestavení aplikace nebo třeba animace a gesta se liší. Pro představu, v React Native se neobjevují komponenty jako *div*, *paragraf*. React má defaultní stylování do řádků, v React Native stylujeme do sloupců. Navigace se řeší trochu jiným způsobem, atd.

Důležitou zmínkou je fakt, že za pomocí React Native se netvoří žádné hybridní aplikace ani webové, které se transformují na mobilní zařízení. S React Native se tvoří nativní aplikace, které používají nativní komponenty a API, v závislosti na mobilním zařízení. Komponenty v React Native se mapují přímo jedna ku jedné právě na konkrétní zařízení, které právě používáte, ať už Android či iOS. Při vývoji není potřeba se učit nějaké nativní prvky pro danou platformu. Za zmínku stojí určitě fakt, že React Native úzce spolupracuje s technologií pod názvem Expo, která bude představena později v analýze.

2.2 REACT KOMPONENTY

Pokud se dostanete do světa Reactu, tak brzy zjistíte, že téměř vše se točí okolo komponent. Celý koncept programování je navržený tak, aby vývojáři tvořili jednoduché nezávislé komponenty, z kterých poté utvoří složitější komponenty, z kterých poté vytvoří celou aplikaci. Velký důraz je kladen na znovu použitelnost jednotlivých komponent. Při tvorbě komponent vždy píšeme jméno velkým písmenem, stejně jako v HTML. Mezi jednoduché komponenty patří funkce, která přijímá propsy a uvnitř funkce je jednoduchá JSX syntaxe, kterou funkce vrací. Složitější komponenty bývají zpravidla třídy, které obsahují jednoduché metody a důležitou funkci *render*.

Pro určení defaultních hodnot komponenty používáme *defaultProps*. Jsou to statické hodnoty, protože se týkají libovolné stejnojmenné komponenty. Pro určení základních typů v komponentě používáme *propTypes*. Jsou to opět statické hodnoty, které definují datové typy propsů. Využívají se hlavně proto, aby vývojáři předávali do komponent ty správné datové typy.

2.2.1 DĚLENÍ KOMPONENT

Při vývoji bude hrát důležitou roli výběr správného druhu komponenty. Z obecného hlediska můžeme říci, že rozlišujeme jednoduché prezenční komponenty a složité kontejnerové komponenty.

Prezenční komponenty jsou funkce, které nemají žádný vnitřní stav a chybí jim jakékoliv metody. Tyto komponenty se používají nejčastěji. Jsou pro použití na různých místech v aplikaci. Vzhledem k budoucímu testování těchto prezenčních komponent, či případným problémům s transpilací, je snaha nepoužívat pro tvorbu anonymní funkce označované jako Arrow functions.

Kontejnerové komponenty jsou třídy, které obsahují různé vlastní metody a mají vnitřní stav. Tyto komponenty se používají pouze málo. Jsou totiž složitější, tudíž mají nízkou úroveň další použitelnosti.

2.2.2 METODY

Vzhledem k faktu, že *this* (v objektovém světě označení pro vnitřní třídu) se v JavaScriptu chová trochu jinak než v ostatních jazycích, tak se pro psaní metod někdy používají anonymní funkce z ES6 nebo funkce pro nastavení korektního *this*. Zpravidla metody, které používá pouze ta konkrétní komponenta, začínají podtržítkem. Při předávání metody do potomkových komponent je opět výhodné používat anonymní funkce z ES6.

V Reactu existuje jedna skupina specifických metod, tím se myslí životní metody konkrétní komponenty. Tyto metody pracují s tím, v jakém rozpoložení se komponenta nachází. Dalo by se říci, že záleží, jestli se komponenta už vytvořila a připojila nebo ne. Také do této sekce patří metoda *setState*, která dokáže asynchronně měnit vnitřní stav komponenty. Při změně v informaci přidané při vykreslování, se konkrétní komponenta znovu vykreslí neboli zavolá metoda *render*. Za zmínku stojí

jistě fakt, že jediná povinná metoda, která je v každé kontejnerové komponentě, je právě *render,* která vrací JSX syntaxi neboli syntaxi podobnou technologii XML. JSX bude podrobně rozebráno níže v této kapitole.

Při správě vnitřního stavu komponenty budeme jistě využívat konstruktor. Pokud přijde na věc a my budeme chtít mít komponentu s vnitřním stavem, tak potřebujeme v kontejnerové komponentě mít konstruktor. Pro tvorbu konstruktoru můžeme použít kratší ES6, ES7 syntaxi nebo delší syntaxi ze starší verze [5]. V konstruktoru se také mohou objevovat takzvané „bindování“ vnitřních metod, neboli nastavení korektního *this* pro dané metody (doporučená praktika od vývojářů z firmy Facebook) [6].

2.3 STATE A PROPS

V mobilní aplikaci budeme jistě řešit správu dat. Bude nás zajímat, která komponenta má jaký stav a také, jaké parametry komponenta dostane. Samotnou komponentu si v tuto chvíli můžeme představit jako jednu část v naší aplikaci, například navigace, hlavní stránku či animace pro přechod. Nejprve přijde vysvětlení ohledně předávání informací do dalších komponent.

Jednotlivé komponenty mohou přijímat parametry nazývané jako *props*. Maximální počet parametrů pro jednu komponentu není určen, ale většinou se používá pouze pár parametrů. Propsy se píší zpravidla do složených závorek uvnitř komponenty, jako JavaScriptový výraz. Můžeme je také psát do závorek. Každá komponenta v Reactu může přijímat propsy a zároveň předávat propsy. Při práci s propsy používáme referenci *this.props*.

Pro ukládání vnitřního stavu komponenty používáme stav neboli *state*. Na rozdíl od propsů, které nastavuje rodič a jsou neměnné, je *state* snadno měnitelný. Stav komponenty se inicializuje v konstruktoru a dále se dá změnit zavoláním metody *setState*. Pokud zavoláme tuto metodu, zpravidla následuje opětovné vykreslení a připojení patřičné komponenty, proto není vhodné psát metodu *setState* do metody *componentDidMount*.

2.4 STYLOVÁNÍ

Pro stylování uživatelského rozhraní používá React Native čistě JavaScriptový zápis v objektové formě. Všechny komponenty dokáží přijmout parametr *style*. Stylování však používá velice podobné koncepty jako CSS, každopádně všechny názvy ve stylování se píší zpravidla camelCase neboli všechny klíče v objektu píšeme jako jednotnou posloupnost znaků, ale každé druhé a další slovo s velkým písmenem. Dále pak číselné hodnoty se píší pouze čísla. Při vkládání stylů do komponenty můžeme přidat více referencí a vytvořit tak pole stylů, přičemž největší přednost má vždy ten poslední styl. Doporučený zápis při stylování je používat *StyleSheet.create*. Tato metoda odděleně vytvoří nový styl, který lze přidělit do více komponent.

Během přemýšlení, jak rozdělit komponenty v aplikaci, bude hrát hlavní roli Flexbox. Flexbox řeší rozvržení elementů v uživatelském rozhraní. Původní Flexbox pochází z webových technologií. Existují některé známé prvky, které řeší Flexbox ve formě CSS na webu, ale nejsou obsaženy ve stylování v React Native. Při stylování je nastavené rozvržení *Flex* jako výchozí. Není zde jiná možnost, jak stylovat. Při tvorbě aplikace si vývojář vybírá, v jakém směru na sebe komponenty budou navazovat, na jakou se stranu se zarovnají, jak velký prostor zaplní atd.

2.5 JSX

Pro tvorbu komponent bude využita technologie JavaScript extension, zkráceně JSX. V jednoduchosti JSX je syntaxe pro psaní uživatelského rozhraní používaná nejen v Reactu, ale i třeba v React Native. V JSX není definována žádná nová sémantika. JSX obecně vypadá jako XML, ale dokáže používat všemožné JavaScriptové výrazy. Stačí vepsat výraz (pouze výraz, nelze použít nic složitějšího) do složených závorek. Webový prohlížeč si nedokáže poradit s JSX, proto používáme preprocesory pro transformaci do ECMAScriptu (skriptovací jazyk, kde jednou z hlavních implementací je JavaScript) [7].

V JavaScriptu se pro tvorbu elementů v dokumentu používá funkce *createElement*. Ve světě Reactu se používá Reactí funkce *createElement* pro tvorbu elementů, která navíc umí vytvořit elementy reprezentované jako Reactí komponenty. Reactí funkce *createElement* je schopna přijmout libovolné množství argumentů, které reprezentují potomky konkrétního elementu, ze kterých se tvoří stromová strukturu s různými atributy. Jenže tvorba aplikace tímto způsobem není pro vývojáře zcela přívětivá. Je to zbytečně složitá struktura. Naštěstí existuje syntaxe JSX, ve které se také dají použít Reactí Elementy a metody na kolekcích objektů (ideální pro použití metod z funkčního programování: *map*, *filter*, *reduce*). S JSX se dobře pracuje.

V JSX se syntaxe podobá HTML, jsou tam ale určité rozdíly a také doporučení:

- Pro přidání třídy se používá *className*
- Nepoužíváme podmíněný *render* komponent (možné zanoření, které vede k nepřehlednosti v kódu)

2.6 PŘÍSTUP K PROGRAMOVÁNÍ

Poslední informací ze světa Reactu je zmínění přístupu k programování. Z klasického programování v jazyce C známe určitě imperativní přístup. Pokud se kód v určitém jazyku chová spíše imperativně, znamená to, že určuje, jak se má něco stát. Na druhou stranu deklarativní přístup říká, co přesně chceme mít. A teď to hlavní. Při psaní React Native se využívá ve velké míře deklarativní přístup. Jednoznačně určíme, co na obrazovce chceme mít. Kód je poté mnohonásobně čitelnější a také jednodušší na správu. Situaci, jak vše ostatní provést, má na starosti knihovna Reactu neboli řeší imperativní přístup.

3 ANALÝZA

Ve světě mobilních telefonů se technologie dynamicky mění. Je zcela běžné, že software použitý z minulého roku už je zastaralá záležitost. V dnešní době se hodně dbá na to, aby všechny aplikace běžely velice rychle. Byly intuitivní a měly stále moderní design. Ve vývoji se tedy pohybují nejenom vývojáři, ale i testeři či grafici a další. Je také důležité si umět správně vybrat, s čím v budoucnosti pracovat, aby Vám příslušná knihovna či framework spíše pomohly, než uškodily.

Při vývoji mobilní aplikace v React Native nemusíme používat jako vývojáři všichni stejné nástroje. Svět Reactu je hodně flexibilní. Jeden může preferovat čistě JavaScript, Expo [3.6.2] a Redux [3.6.3]. Další může kombinovat různé frameworky pro psaní webových aplikací na mobilní zařízení. Jiný může naopak používat React Native a nativní jazyk pouze pro jednu platformu. Každopádně každý z přístupů s sebou nese patřičné výhody a nevýhody.

V této kapitole prozkoumáme jednotlivé technologie ze světa React Native. Podíváme se na nativní a jiný způsob vývoje. Probereme některé funkční a nefunkční požadavky. Vyzkoušíme si hledání podobného řešení. Zvolíme cílovou skupinu a v neposlední řadě popíšeme různá nebezpečí hrozící při vývoji mobilní aplikace.

3.1 TRH

Za účelem získání výhody na trhu a zjištění potřeb i přání klientů musí být nejdříve provedena analýza trhu. Pokud chceme nabídnout vhodnou aplikaci, je prospěšné projít online obchody, které nabízejí podobné produkty, abychom zjistili, jak je na tom konkurence.

V aktuální chvíli existují na trhu podobné aplikace, ale trh není příliš veliký. Některé aplikace nepodporují český jazyk. Jiné mají starší uživatelské rozhraní nebo se zcela přestaly vyvíjet, což vedlo k jejich poklesu na trhu. Mnohé aplikace nejsou přímo mobilní, ale poskytují velice podobnou funkcionalitu, například aplikace Mapy.cz [8].

Problémem podobných aplikací je komplexní uživatelské rozhraní, nízká míra použitelnosti, jazyková bariéra, obsahují placené reklamy. Chybí jim možnost svobodně přidávat různá zajímavá místa s okamžitým efektem. Tyto důvody označují příčinu vzniku tohoto projektu.

3.2 CÍLOVÁ SKUPINA

Vybrat cílovou skupinu v našem případě není těžké. Mobilní aplikace se soustředí na skupinu lidí, kteří často a rádi chodí do přírody. Tito lidé nemají větší problémy s používáním chytrého telefonu. Průměrný věk těchto lidí je 35 – 50 let. Mezi jejich zájmy patří především příroda a historie. K uspokojení požadavků naší cílové skupiny bude aplikace uživatelsky přívětivá s moderním a jednoduchým designem. Vzhledem ke skutečnosti, že máme pouze jednu cílovou skupinu, bude jednodušší najít případné problémy v aplikaci a také zjistit možná doporučení na zlepšení.

3.3 RIZIKA

Při vývoji aplikace nebo i v okamžiku, kdy bude aplikace v produkci, existují patřičné hrozby, které mohou vyústit v nepříjemná rizika spojená s projektem. To může mít neblahý dopad na vývoj aplikace. Některé problémy mohou vývoj aplikace prodloužit a některé mohou projekt ukončit.

Ve chvíli, kdy bude aplikace plně v provozu, může nastat situace, že ji lidé nebudou mít důvod nadále používat. Může přijít od konkurence lepší produkt, popřípadě nasazení podobné aplikace s firmou s větším zázemím. Existuje tedy několik příčin možného neúspěchu aplikace. Věříme, že aplikace bude plně funkční a v blízké době (léto 2019) nasazená do provozu.

Dále hrozí riziko na straně vývoje aplikace či testování. Vzhledem k situaci, že nejsou patřiční vývojáři, kteří by produkt v budoucnosti spravovali, může nastat problém s přidáním či odebráním nějaké funkcionality.

3.4 POŽADAVKY

Přejděme nyní na požadovanou funkcionalitu aplikace, která bude sloužit k bližšímu pochopení, proč jsou jednotlivé věci implementovány. Některé požadavky (lze také nalézt pod označením úkol) na aplikaci jsou velice důležité, jako například přidávání míst do mapy či zkoumání zajímavých míst na mapě, avšak některé požadavky nemají tak velkou prioritu. Je dobré dát dohromady vhodné požadavky, které by měla aplikace plnit. Seznam těchto požadavků je rozdělen mezi funkční a nefunkční požadavky. Každý požadavek je možný splnit a zároveň otestovat. Dále by měl být každý požadavek dobře definovaný. Pro mobilní aplikaci byly zvoleny následující požadavky.

3.4.1 FUNKČNÍ POŽADAVKY

1. Přihlášení
    - Uživatel by měl být schopný se přihlásit do aplikace prostřednictvím služby Gmail, popřípadě svým vlastním účtem vytvořeným v aplikaci
    - Uživatel může vstoupit do aplikace jako anonymní uživatel
2. Mapa
    - Uživatel bude mít možnost vyhledávat na mapě významná místa v okolí, bez ohledu na to, zda je přihlášený či nikoliv
    - Uživatel má možnost, při kliknutí na významné místo zjistit více informací o konkrétní lokaci
    - Uživatel je schopný najít na mapě vlastní polohu, popřípadě přesunout pohled mapy na aktuální místo polohy
    - Přihlášený uživatel může přidat při kliknutí na mapu další významné místo s doplňujícími informacemi
    - Přihlášený uživatel má právo smazat vlastní přidané místo na mapě, popřípadě upravovat informace
    - Přihlášený uživatel je schopný přidat libovolné místo do seznamu oblíbených míst.
    - Uživatel dokáže filtrovat zobrazení významných míst na mapě
    - Uživatel si může přečíst vysvětlivky k mapě
3. Vlastní profil
    - Přihlášený uživatel dokáže upravovat vlastní profil

3.4.2 NEFUNKČNÍ POŽADAVKY

1. Bezpečnost
    - K zpřístupnění všech funkcí aplikace je nutné přihlášení. Mobilní aplikace bude zabezpečena za použití unikátních tokenů (informace, které napomáhá k identifikaci uživatelů) při komunikaci se serverem
2. Podpora
    - Mobilní aplikace bude podporovat platformu iOS i Android. Pro Android od verze 6.0, pro iOS od verze 10
3. Spolehlivost
    - I s větším počtem návštěvníků bude aplikace plně schopná reagovat a plnit požadavky uživatelů. Pokud bude aplikaci používat více lidí zároveň, bude to mít pouze minimální dopad. S dobrou spolehlivostí se vždy pojí dobrá škálovatelnost
4. Udržitelnost
    - V případě problémů, popřípadě *bugů* v aplikaci, není obtížné změnit jednu komponentu za jinou. Komponenta tudíž musí mít velice úzké vazby uvnitř sebe, ale zároveň musí být „nezávislá“ na ostatních. Součástí udržitelnosti je i kvalitní dokumentace
5. Rozšiřitelnost
    - V mobilní aplikaci bude snadné přidávat další vychytávky s minimálním rizikem ovlivnitelnosti zbytku fungování aplikace. Bude za potřebí navrhnout kvalitní schéma aplikace. Kód bude přehledný a čitelný, bez nutnosti dlouhého studování, co daná funkce dělá. Vhodné použití *single responsibility principu* neboli jedna funkce bude mít na starost pouze jednu věc a oddělení zodpovědnosti
6. Dostupnost
    - Aplikace bude dostupná na *App Store* a *Google play* 24 hodin denně, 7 dní v týdnu

3.5 MOBILNÍ APLIKACE

Při vývoji mobilní aplikace bude za potřebí pouze React Native. Je dobré vědět něco málo o ostatních způsobech vývoje, znát různé alternativy. S pochopením ostatních principů můžeme totiž jednodušeji pocítit silné a slabé stránky React Native. Pokud porozumíme faktům, čím se od sebe liší jednotlivé technologie, dosáhneme pokroku v mobilním vývoji a začnou nám informace do sebe lépe zapadat. Klíčovou částí jsou informace o tom, jakou má daná technologie podporu, jak velkou má komunitu lidí, jaké organizace danou technologii používají a také, jak těžké je se naučit ji používat.

Nejprve představíme jednotlivé online obchody pro distribuci mobilních aplikací. Dále bude následovat seznam alternativ.

3.5.1 DISTRIBUČNÍ SLUŽBA

V dnešní době jsou na trhu mobilních aplikací 2 giganti, kteří poskytují stažení i instalaci aplikací online. První zmiňovanou je App Store, druhou pak Google Play. U zákaznických firem je však běžným požadavkem mít vlastní aplikaci na obou trzích. Jenže vývoj, nahrání aplikace a správa bývá značně odlišná [12].

App Store je hodně přísný při přijímání aplikací. Někdy zabere vývojářům dlouhé týdny, než dokáží nasadit mobilní aplikaci na App Store. Společnost Apple dlouho kontroluje nahrávání aplikace, například nejsou-li porušeny žádné předpisy. Navíc aplikace musí být dobře otestovaná. Vývojáři musí platit roční členství. Ceny aplikací nebývají nejlevnější.

Google Play není tak přísný na přijímání aplikací. Při nasazení trvá přibližně půl dne, než se aplikace nahraje do obchodu. Navíc Android dává mnohem větší volnost vývojářům. Zároveň ale chybí větší důraz na kvalitu aplikací. Naštěstí pro vývoj na Google Play stačí zaplatit pouze malý finanční obnos a člověk má tak doživotní možnost volného přidávání mobilních aplikací. Navíc cena aplikací nebývá často nijak zvlášť vysoká.

3.5.2 NATIVNÍ NEBO WEBOVÁ APLIKACE

Při vývoji mobilní aplikace nemusí vývojář zůstat pouze v JavaScriptu. React Native je schopný spolupracovat s ostatními jazyky. Dokáže dobře navázat vztah i s jinými frameworky.

Po výběru jedné specifické platformy, ať Android či iOS, můžeme připojit k React Native také část kódu vyvíjeném v nativním jazyce. Pokud zvolíme Android, vybereme, zda použijeme jazyk *Java* nebo *Kotlin*. Jako vývojové prostředí postačí *Android Studio*. Pokud je aplikace cílená pouze pro iPhone a iPad, musíme použít buď *Objective-C* nebo *Swift* a tedy vývojové prostředí *Xcode*. Při nativním vývoji je jednoduší používat schopnosti daných zařízení (kameru, systém, notifikace, …). Další výhodou je vysoká rychlost a přizpůsobivost k reakcím na mobilních zařízení. Je pro ně snažíš pracovat off-line.

Alternativním způsobem vývoje mohou být webové mobilní aplikace. Pro psaní kódu mobilních aplikací stačí použít standardní webové technologie jako například JavaScript, JSON, HTML, … Tyto aplikace běží uvnitř prohlížeče, tudíž uživatel aplikaci může spustit pouze přes webový prohlížeč. Jednoduše se spravují. Je zde možnost použít takřka libovolnou webovou technologii nebo jazyk pro webový vývoj. Rozdíl mezi webovou mobilní aplikaci a webovou aplikací je ten, že webová mobilní aplikace je navržená přímo pro mobilní zařízení. Při používání standardních webových technologií lze také psát takzvané progresivní webové aplikace, popřípadě hybridní aplikace.

3.5.3 OSTATNÍ

Dále následuje seznam technologií, které lze také použít pro tvorbu mobilních aplikací. Tyto technologie mívají slabší komunitu zájemců. Jejich popularita není taková nebo chybí patřičná dokumentace. Je zde také možnost, že tyto technologie jsou teprve na počátku svého působení ve světě mobilních technologií.

Vývoj mobilních aplikací hybridním způsobem umožňuje právě Ionic. Při vývoji se tvoří webová stránka, která používá standardní webové technologie jako JavaScript nebo HTML. Tato aplikace je potom hostována v nativní podobě ve formě *WebView* komponenty. Tvorba aplikace je velice rychlá a jednoduchá. K dispozici jsou nativní vychytávky pro jednotlivé platformy. Ionic je postavený na technologii Cordova (vývoj mobilních aplikací za pomocí HTML, CSS, JavaScript) a JavaScriptu. Ionic je zdarma a open source. Ionic dokáže používat nativní technologii, jako například *Bluetooth*, *TypeScript*, *HealthKit* a další. Ionic z velké části spravují vývojáři z Googlu.

V poslední době velmi roste oblíbenost technologie pod názvem Flutter vytvořená společností Google. Flutter používá podobný technologický přístup při vývoji mobilních aplikací jako React Native. Flutter se snaží při vývoji vytvořit aplikaci, kterou následně zkompiluje do nativního kódu. Flutter používá vlastní vysoce rychlostní mechanismus pro vykreslení. Jeho implementace je v jazyce Dart, což je jazyk s podobnou funkcionalitou jako JavaScript. Společnost Google stojí za technologií Flutter. Poprvé s tímto mobilním SDK přišla společnost Google v roce 2018, verze 1.0.

Poslední popisovanou technologií, která lehce upadá, je Xamarin od společnosti Novell. Xamarin je framework, který stejně jako ostatní zmiňované technologie slouží pro vývoj mobilních aplikací. Xamarin byl později koupen firmou Microsoft. Zpočátku byla možnost tvořit pouze pro iOS, postupem času přibyla i možnost tvořit pro Android. Pro psaní aplikací používá jazyk C#. V Xamarinu je možné tvořit nejen pro Android, iOS, ale také pro Windows Mobile. Pro vývoj v Xamarinu dokonale poslouží vývojové studio *Visual Studio.* Výhodou práce v Xamarinu je možnost sdílení kódu mezi jednotlivými platformami. V Xamarinu hraje důležitou roli framework Xamarin.Forms, který slouží pro tvorbu grafického uživatelského rozhraní.

3.6 TECHNOLOGIE

Další odstavce budou o vývoji mobilní aplikace. Během analýzy byly odhaleny i některé novinky, které při vývoji v React Native určitě nesmějí chybět. Každá další zmíněná technologie dokáže nějakým způsobem efektivně pomoci vývojářům. Jednotlivé technologie se dají charakterizovat jako softwarové nástroje, které mohou výrazně ulehčit práci, zpřehlednit kód nebo nějakým jiným způsobem zpříjemnit vývoj. Některé jsou pouze doporučené řešení, jako například ES6, některé však mají obrovský potenciál a jsou denně využívané u vývojářů React Native.

Pro tvorbu mobilní aplikace v React Native se bude používat software dostupný na internetu, který je zdarma. Po důkladném prozkoumání různých dostupných technologií, které jsou k dispozici, je zde soupis těch nejdůležitějších, které by měl znát každý vývojář React Native.

3.6.1 ES6

ES6 neboli ES2015 je obrovský balíček v JavaScriptu, je to nová verze, která přináší řadu nových věcí. Hlavní důraz je kladen na jednoduchost a srozumitelnost, dále se ES6 snaží adoptovat standardy z ostatních jazyků a také posunout jazyk zase o krok dále. Při implementaci mobilní aplikace by bylo vhodné používat nové nástroje právě ze sady ES6. Celkově lze ES6 chápat jako nový balíček komplexních softwarových nástrojů, které výrazně přispějí k větší efektivitě.

Pokud máte rádi novinky ze světa JavaSciptu, tak React je správná volba. Po delším bádání je snadné zjistit, že právě React úzce pracuje s novými vychytávkami, které přináší ECMAScript. V tomto článku seznam těch nejpoužívanějších částí z ES6 v Reactu.

React je známý pro kombinaci programovacích paradigmat. Proto je normální mixovat ve stejné aplikaci třídy a funkce. I přesto, že React zastává názor programovat spíše funkcionálně, tak je naprosto běžné narazit v Reactu na třídy. Třídy se používají v Reactu pro psaní komponent. Pokud tedy používáme pro tvorbu komponent třídy, tak musí dědit z jiné komponenty. Nejčastěji dědí z *React.Component*.

Velice častým doplňkem je anonymní funkce, která ale nepřináší žádnou novou funkcionalitu. Jsou to obyčejné funkce, které jsou jednodušší na zápis. Používáme jednoduché závorky, šipku (znamínko rovná se, znamínko větší než), složené závorky. V případě jednoho argumentu nemusíme kulaté závorky psát, v případě jednoho výrazu v těle funkce, nemusíme psát složené závorky.

Vzhledem k faktu, že neměnnost proměnných je součástí ekosystému Reactu, tak pro tvorbu proměnných používáme zásadně *const*, pokud to však situace vyžaduje, můžeme použít *let*. Další důležitou součástí Reactu jsou modularita. Ve většině JavaScriptových souborů používáme různé importy. Pro nás nejdůležitější balíček, který importujeme, je *react*, dále pak *react-native*.

3.6.1.1 MAP, FILTER, REDUCE

Pokud nepoužíváme pouze primitivní proměnné, ale také složitější proměnné, tak se nabízí opět použít novou funkcionalitu z ES6. Při práci nad polem můžeme používat vestavěné funkce v JavaScriptu, jako například:

- *map* (projít pole hodnot, úprava hodnot, vrátit nové pole hodnot)
- *filter* (projít pole hodnot, filtrovat pouze specifické hodnoty, vrátit nové pole)
- *reduce* (projít pole hodnot, porovnat konkrétní hodnoty, vrátit jednu hodnotu)

a další…

Důležitou součástí každé iterace nad polem v UI, která vrací pole, je umět správně pracovat s klíčem. React umí rychle pracovat nad polem hodnot, kde každá má svůj unikátní klíč, to znamená, že například při používání *map* pro vykreslení jednotlivých elementů je velice vhodné každému elementu předat unikátní klíč do *propsů*.

3.6.2 EXPO

Expo je technologie, která umožňuje jednoduše vyvíjet pro všechny platformy (Android, iOS) v libovolném operačním systému. Expo je zdarma a open-source. Pro sdílení aplikace online poskytuje Expo řadu možností. V dnešní době existuje řada vývojářů, kteří rádi tvoří totožné kopie stávajících aplikací, které pak sdílí přes Expo, například kopie aplikace Instagram [13]. Toto zpracování může sloužit jako inspirace pro vývoj. Pro spuštění sdílené aplikace stačí mít nainstalovanou aplikaci Expo v telefonu. Pro sdílení můžete například použít čárový kód. Expo také poskytuje snadné sestavení aplikace bez rozdílu, jestli chcete používat App Store či Google play. To velice ulehčí vývojářům práci při zveřejňování aplikací. Obrovskou výhodou používání technologie Expo je, že vývojáři nepotřebují používat Android Studio či jiné specifické IDE. Není potřeba mít například vývojové prostředí Xcode pro vývoj na iOS. Expo také poskytuje balíček *expo* (různé komponenty, konstanty …) pro vývoj v React Native.

Pro práci s technologií Expo je vhodné mít stažený a nainstalované Expo XDE, které slouží pro správu lokálních aplikací. Dále je vhodné mít stažený a nainstalovaný balíček *expo*, ať už přes *npm* nebo *yarn* (obě dvě technologie slouží jako správce balíčků z JavaScriptu). Expo také lze používat online. Například je možné vytvořit mobilní aplikaci v prohlížeči, kde je možnost spuštění aplikace online na virtuálním mobilním zařízení.

3.6.3 FLUX

V okamžiku, kdy začneme řešit průběh dat v aplikaci, tak můžeme brát inspiraci z ustáleného vzoru Flux od firmy Facebook. Flux je návrhový vzor v programování, který se zaměřuje na průchod dat v mobilní aplikaci. Jednou z hlavních vlastností této technologie je oddělení zodpovědností při správě dat. Dále řeší problém škálování velkého množství kódu a mění tok dat na jednosměrný průchod neboli *unidirection data flow*.

Flux sám o sobě vypadá dosti užitečně. Vývojáři mají ucelenou představu o průchodu dat. Žádný vývojář se nemusí zbytečně ptát ostatních, zda může poupravit část kódu či nikoliv. V tomto projektu bych ale doporučoval použít knihovnu Redux, která vychází z myšlenky již zmiňovaného vzoru. Základem je ale umět Flux, proto si ho také důkladněji představíme. Redux jako takový se liší od Fluxu tím, že v něm chybí *Dispatcher* neboli krátká část kódu, která posílá zmiňované akce do *Store*. V jednoduchosti *Store* udržuje stav aplikace. *Dispatcher* je nahrazen procesy uvnitř právě zmiňovaného *Store*. Pro Redux ale neplatí, že nejchytřejší částí je *Store*, ale *Reducer*, který rozhodne podle přijetí zprávy z *Action*, co přesně se má provést. V Reduxu také chybí větší míra flexibility, například uvnitř *Storu*. Stav uvnitř *Storu*, který je pouze jeden, je neměnný. Redux velice dobře spolupracuje s Reactem.

3.6.3.1 ZÁKLADNÍ ARCHITEKTURA

Flux se dělí na 4 části:

- **Action** – Naslouchá UI, potom provede nějakou vnitřní logiku (přihlášení, odeslání formuláře,…) a odešle akci do části zvané *Dispatcher*
- **Dispatcher** – Přijímá všechny zprávy z *Action*, vybere ty, které ho zajímají a pošle je do příslušného *Storu*
- **Store** – Aktualizuje data a vyšle informaci do *View*
- **View** – Přijímá upravená data ze *Store* a upravuje UI

3.6.4 MVC

Pro správu dat je možné použít klasický architektonický vzor MVC neboli Model View Controler, který je v IT už přibližně 40 let. Při zvětšujícím se množství řádek kódu, se očekává, že program bude vykonávat více funkcionality. Data se tak začínají používat na více místech v kódu. To přináší mnohé nešvary, například:

- těžší na údržbu
- těžší ladění počítačového programu
- kaskádní efekt
- chybí jasná informace, jak na data flow
- při příchodu dalšího programátora těžký na pochopení
- problém s nejistotou přidávání kódu do kódu jiného programátora

Flux nebo Redux je proto vhodnou alternativní technologií.

3.6.5 REACT NAVIGATION

Bude důležité, aby se uživatel jednoduše orientoval v aplikaci, tudíž není jistě překvapení, že navigace v mobilní aplikaci hraje důležitou roli. K dnešnímu dni existuje velké množství knihoven pro správu navigace v mobilní aplikaci. Nikde není striktně definovaný standard, kterou formu navigace použít. Vzhledem k faktu, že vývoj mobilní aplikace je v React Native, bude přiměřené použít knihovnu vyvíjenou na základě požadavků právě z této komunity. Z toho důvodu použijeme knihovnu React Native psanou v jazyce JavaScript.

Pro použití React Navigation je nutné nainstalovat stejnojmenný balíček. Při práci s React Navigation se často vytvoří odděleně několik souborů, které nastaví navigaci mezi obrazovkami. Pokud chceme zavolat metodu přesunu na jinou obrazovku, použijeme funkci *navigate* z *this.props.navigation*.

3.6.6 HEROKU

Pro lokální použití stačí spustit server na vlastním počítači. Pokud má být aplikace dostupná globálně pro všechny, tak by bylo vhodné spustit server na vzdálené platformě. Uživatelé by tak mohli interagovat přes mobilní aplikaci s tímto vzdáleným serverem. Pokud hledáme jednoduché řešení, které je zcela zdarma, tak dokonale poslouží Heroku.

Heroku je cloudová platforma pro vývojáře, která umožňuje nasadit aplikaci ihned do produkce. Při použití Heroku je jednodušší nastavování, monitorování a správa aplikace online. Heroku podporuje spoustu programovacích jazyků: NodeJS, Ruby, Java, Scala,…

3.6.7 DATABÁZE

Při používání mobilní aplikace bude vhodné rozumně ukládat data. Ideálně se jeví nějaké úložiště dat, které by šlo jednoduše upravovat a později i načítat. Při větším množství uživatelů je dobré zvolit tu správnou databázi pro správu dat. Databáze by měla být schopná pracovat s velkým množstvím dokumentů, měla by být schopná pracovat s formátem JSON, měla by poskytovat jednoduchou integraci dat. Vhodným kandidátem je proto MongoDB.

MongoDB je multiplatformní databáze sloužící jako backendové řešení pro mobilní aplikace. MongoDB nepracuje jako relační databáze, ale jako *NoSQL databáze*. MongoDB je jeden z nejznámějších databázových systému na poli programování.

Pro správu cloudových databází, které by šlo chápat jako úložiště na internetu, by mohl dobře posloužit mLab. Tato služba je zcela zdarma. Poskytuje vývojářům zálohování, škálování a obnovu dat. Služby fungují 24 hodin 7 dní v týdnu a je primárně určená pro práci s databází MongoDB. Velké společnosti jako Amazon, Azure a další používají mLab.

3.7 GEOGRAFICKÁ DATA

Pokud se podíváme na povrch Země z leteckého pohledu, zjistíme, že příroda, živé organismy, infrastruktura a jiné objekty se skládají z různých vrstev. Tyto vrstvy mohou mít různé rozdělení či rozpoložení. Budoucí analýza jednotlivých vrstev nám napomáhá v porozumění, proč jsou objekty umístěné právě tímto způsobem. Dokážeme snáze hledat smysl v jejich propojení. Je třeba si uvědomit, že zkoumání povrchu Země není žádnou novinkou. Již od počátku věků se lidé snažili zmapovat aktuální pohled na svět. Dříve však lidé neměli tak vyspělou technologii, jako máme dnes.

Výstupem projektu bude jednak hotová mobilní aplikace, jednak souhrn dat, který obsahuje různá zajímavá místa z okolí. A právě geografická data jsou pro nás v budoucnosti dosti zajímavá. Po dokončení projektu bude paralelně s fází údržby důležitá fáze zkoumání dat. Bude provedena analýza geografických dat v databázi. Pro naše účely dokonale poslouží zobrazení všech památek na mapě. V tento moment přichází na scénu dopad firmy ESRI, zabývající se vývojem softwaru pro práci s informacemi na mapách. Produkty dané firmy pracují s geografickými informačními systémy, zkráceně označené jako GIS. Nejznámějším produktem firmy je systém ArcGIS, světově nejvýkonnější nástroj pro mapování a analýzu dat.

Klíčovým momentem bude jistě rozhodnutí, jakým způsobem sdílet data. Jedním z vhodných kandidátů bude produkt, vycházející ze systému ArcGIS, pod názvem ArcGIS Online. Tento systém se používá pro přidávání, zobrazení a analýzu geografických dat. Pomocí nástroje ArcGIS online je velice snadné vytvořit jednoduchou mapu složenou z různých vrstev. Informace potom mohou být spravovány na sdílené webové službě. Nástroj samotný není obtížný k běžnému použití, podporuje možnost přidávat uživatelům různá práva k použití. Usnadňuje bližší zkoumání dat a vytváření různých dalších map. Mapy na webové aplikaci mohou obsahovat různé grafy, podpůrné informace a další. V jednoduchosti lze ArcGIS Online popsat jako účinného prostředníka mezi aplikací a daty, který umí jednoduše sbírat data a provádět jejich analýzu.

Vytvoření účtu na ArcGIS Online je pro studenty ČVUT zdarma. Výhodou práce v této webové aplikaci je snadný přístup v rámci organizace. Uživatelé připojené k dané organizaci mohou mít různá přístupová práva. Každý uživatel může spravovat vlastní obsah. Existuje také možnost různého sdílení dat (soukromé, veřejné).

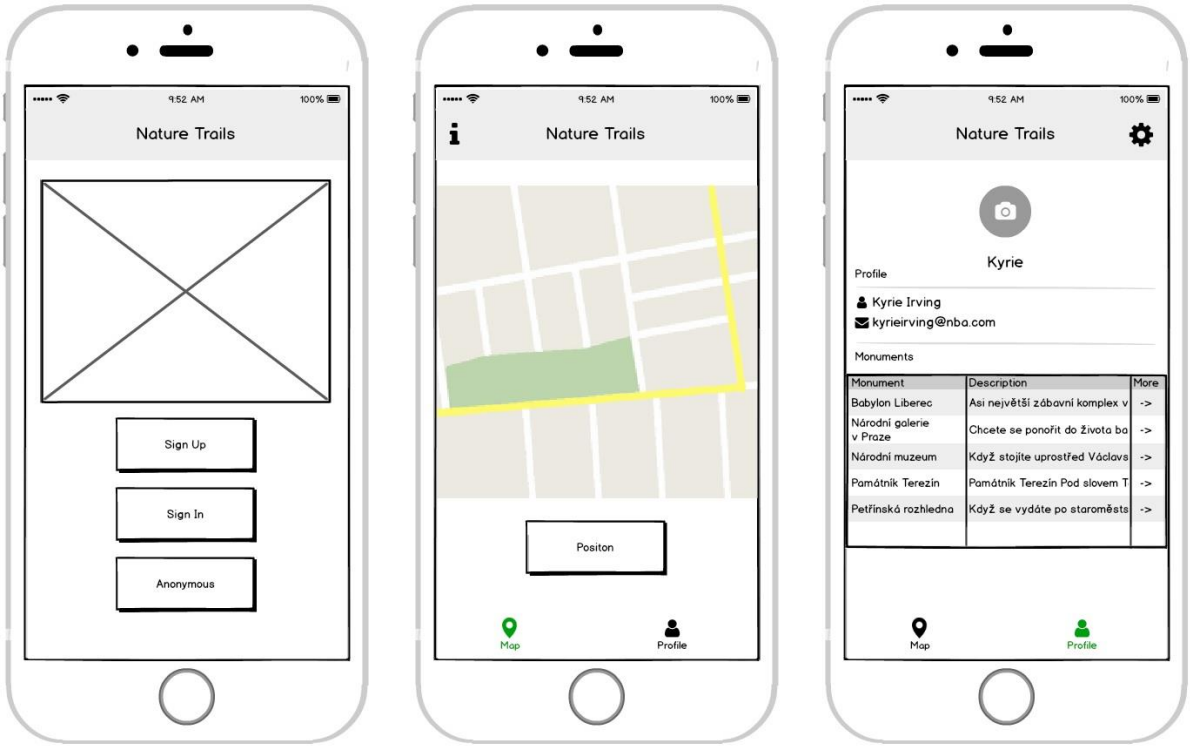
Další ověřenou službou, jak zpracovávat sdílená geografická data je produkt Leaflet, který také mimo jiné podporuje propojení s ArcGIS nástrojem. Leaflet samotný je knihovna jazyka JavaScript, která není nijak rozsáhlá. Její celková velikost je přibližně 40 kB. Leaflet je jednoduchý na použití, značně výkonný, přizpůsobivý různým elektronickým zařízením a hlavně dobře propojitelný s React Native či NodeJS. V našem případě by mohl Leaflet dostat geografická data na vstupu, které později zpracuje. Leaflet očekává data z databáze ve formě JSON, která obsahují pole objektů reprezentující zajímavá místa. Zobrazí jednotlivé body na mapě pomocí ukazatelů označované někdy jako *markers.*

4 NÁVRH

Při návrhu aplikace uživatelského rozhraní nebyly předem definované požadavky. Rozložení elementů či výběr barev byl vytvořen právě během návrhu, podle kterého se bude vyvíjet pozdější implementace mobilní aplikace. Zvolený vzhled bude představen v podobě Wireframů neboli návrhů jednotlivých obrazovek, za použití softwarového nástroje Balsamiq mockups [10]. Poslední část je rozdělena na část přístupnou běžnému uživateli a část, která běží na pozadí neboli na *frontend* a *backend*.

Síťová architektura, která se pro celý projekt nabízí, je určitě klient-server. Jedná se o klasickou dvouvrstvou architektu, kde si mezi sebou obě vrstvy vyměňují data. První část (klientská část) bude reprezentovat uživatelské rozhraní mobilní aplikace. Součástí klientské části bude také aplikační logika, například Redux. Pomocí vestavěné metody *fetch* budou posílány na server jednotlivé dotazy ve formě JSON. Součástí JSON požadavků posílaných směrem na server bude také unikátní uživatelský token. Návrh klientské části bude vhodné přizpůsobit stylu Model-View-ViewModel (MVVM), která je pro React Native typická. Celková velikost klientské části bude tvořit zhruba 70 % celého projektu. Serverovou část navrhuji rozdělit na dvě části, práce s databází a zbytek. V databázové části budou jednotlivé moduly rozdělené na významná místa, kategorie, uživatele. Druhá část bude obsahovat autentizaci, konfiguraci a jiné.

4.1 OBRAZOVKY



Obrázek 4-1: Návrh obrazovek

4.2 FRONTEND

Aplikace pro uživatelské rozhraní je psaná v jazyce JavaScript za použití knihovny React Native. Při vývoji bude psán kód použitelný pro platformu iOS i Android. React Native se umí velice dobře postarat o vykreslování, to nám umožní hladce zpracovat různé animace, například při načítání. Většina kódu

bude psaná čistě deklarativním způsobem. To přináší následující výhody: přehledný a čitelný kód, jednodušší tvorba uživatelského rozhraní atd. React Native je schopný používat nativní gesta a nativní komponenty v mobilní aplikaci. Navíc React Native poskytuje službu pod názvem *hot reloading* neboli možnost úpravy kódu, která se zobrazí na mobilní aplikaci bez nutnosti spuštění opětovné kompilace.

Během vývoje bude vhodné používat knihovnu pro práci s navigací v aplikaci. Běžný uživatel bude chtít možnost přecházet z jedné obrazovky na druhou. Tuto funkcionalitu dokáže výborně zastat React Navigation. Pro naše účely dokonale poslouží základní 3 možnosti přesouvaní mezi obrazovkami. Bude vhodné použít *switchNavigator* pro jednoduchou výměnu obrazovek. Dále se nabízí *bottomTabNavigator* pro navigaci v hlavním menu. V případě, že se uživatel bude chtít dostat do různých konkrétních částí, bude ideální použít *stackNavigator*.

Jako technologický pomocník pro lokální zkoušení aplikace na vlastním mobilním zařízení dokonale poslouží Expo. Je možné používat i důležité komponenty, konstanty a další z balíčku *expo*. Pro naše účely se například bude hodit konstanta pro *statusbar*. Expo navíc poskytuje možnost sdílení aplikace online, nebude tedy problém, pokud někdo bude chtít spustit aplikaci na svém Android zařízení.

Technologii Redux je možné použít pro správu dat. Je to nejčastěji používaná knihovna, která udává přehlednost celé aplikaci. Redux je jednoduchý na použití a má jasnou strukturu, jak pracovat s daty.

4.3 BACKEND

Hlavním úkolem bude zvolit adekvátní technologii běžící na pozadí a také zvolit vhodnou komunikaci se serverem. V mobilní aplikaci například záleží, zda je uživatel přihlášený či nikoliv. Aplikaci je totiž možné rozběhnout v anonymním režimu. Uživatel má poté omezená práva při jejím použití. Důležitou roli bude hrát dynamické zpracování informací a určitá bezpečnost.

Část aplikace běžící na pozadí, konkrétně serverová část, bude psaná v NodeJS. Právě systém NodeJS byl vybrán, protože je zaměřený pro psaní škálovatelných aplikací s primárním účelem tvorby serverové části a také proto, že je to JavaScript. Pro komunikaci mezi backendem a frontendem poslouží API zaměřené na distribuci dat. Jako vhodný kandidát se jeví REST API. Součástí backendu bude také psaní pokročilých JavaScriptových technik, proto je nutné použít vhodný kompilátor pro kompilaci z ES6+ na starší široce podporovanou ES5. Kompilátor je program, který přeloží kód z nové verze do starší. Pro autentizaci použijeme kvalitní software, využívající tajný unikátní klíč. Za běhu aplikace bude uživatelská strana komunikovat se serverovou stranou pomocí tokenů. Mezi možné technologie se řadí JWT. Pro rozsáhlou serverovou komunikaci poslouží framework pod názvem *Express.* Jednou z možností pro serverové úložiště patří například technologie zvaná Heroku, které je zdarma. Není nutné řešit placení za serverové služby. Databáze použitá v aplikace by měla umět práci s dokumenty typu JSON, měla by poskytovat jednoduchou komunikaci dat.

Pro zpracování dat budou vytvořeny synchronizační procedury, které usnadní práci s geografickými informacemi. Pro jednoduchou komunikaci dat s produkty ESRI doporučuji použít JavaScriptovou technologii, která byla odhalena v analýze. Knihovna Leaflet dokonale poslouží k propojení dat. Navrhuji tvorbu veřejně dostupné mapy, která bude sloužit jako prostředník mezi geografickými informacemi na pozadí a přívětivým online uživatelským rozhraním. Výsledný produkt bude navržený pro spolupráci s ArcGIS produkty za pomoci sady nástrojů pod názvem ESRI Leaflet [9].

5 IMPLEMENTACE

V této kapitole bude postupně představena implementace mobilní aplikace. Není možné rozebrat do detailu každý kousek, proto bude z každé části popsaná ta nejdůležitější část implementace. Nejprve rozeberu vizuální část implementace a potom část kódu na pozadí. Implementace vychází z návrhu popsaném ve stejnojmenné kapitole. Celá aplikace je dostupná online [11].

Na začátku vývoje byl vytvořen jednoduchý funkční server komunikující s databází. Ve chvíli, kdy bylo možné využívat základní služby serverové části, začal vývoj nativní aplikace v React Native. Postupem času se začaly tvořit nové a nové obrazovky, které využívaly úzké spojení se serverem. Hlavní důraz byl kladen na naprogramování obrazovky s mapou památek. S přibývajícím kódem v React Native se začalo i mírně upravovat serverové řešení. Veškerý kód, který byl napsán, je v jazyce JavaScript.

JavaScript je asynchronní klientský, mírně objektově orientovaný skriptovací jazyk, který se podílí na tvorbě moderních webových aplikací. JavaScript není předurčen k tvorbě univerzálních aplikací, jako například Java, ale stejně jako Java hojně využívá prvky funkcionálního programování. Většina kódu z JavaScriptu reprezentuje jednotlivé metody různých komponent. Klasický přístup během programování React Native je přidat na začátek souboru odkazy, vybrat správný druh komponenty a nakonec vytvořit JSX obohacené o různé metody. JavaScript byl standardizován společností ECMA pod názvem ECMAScript, zkráceně ES. V roce 2015 přišla zásadní změna. Jazyk byl obohacen o velké množství nových technologií ve verzi 6 označované jako ECMAScript 2015 neboli ES6. Během vývoje byli přidány i některé knihovny (za pomocí správce balíčků) psané v JavaScriptu.

5.1 VÝVOJOVÉ PROSTŘEDÍ

Pro vývoj jsem používal vývojové prostředí Visual Studio Code, zkráceně VS Code, které maximálně pomáhá k implementaci v React Native. Součástí IDE VS Code od společnosti Microsoft jsou vývojářské nástroje, které zlehčují implementaci. Programátor má možnost použít nástroj pro ladění programu, nástroj pro analýzu kódu upozorňující na chyby v programování nebo například možnost přidání různých vývojářských balíčků (Prettier, Babel …). Navíc VS Code podporuje propojení se systémem pro správu verzí. Alternativním vývojovým prostředím může být například Atom, Sublime Text nebo pro opravdové labužníky třeba Vim.

5.1.1 UKÁZKA

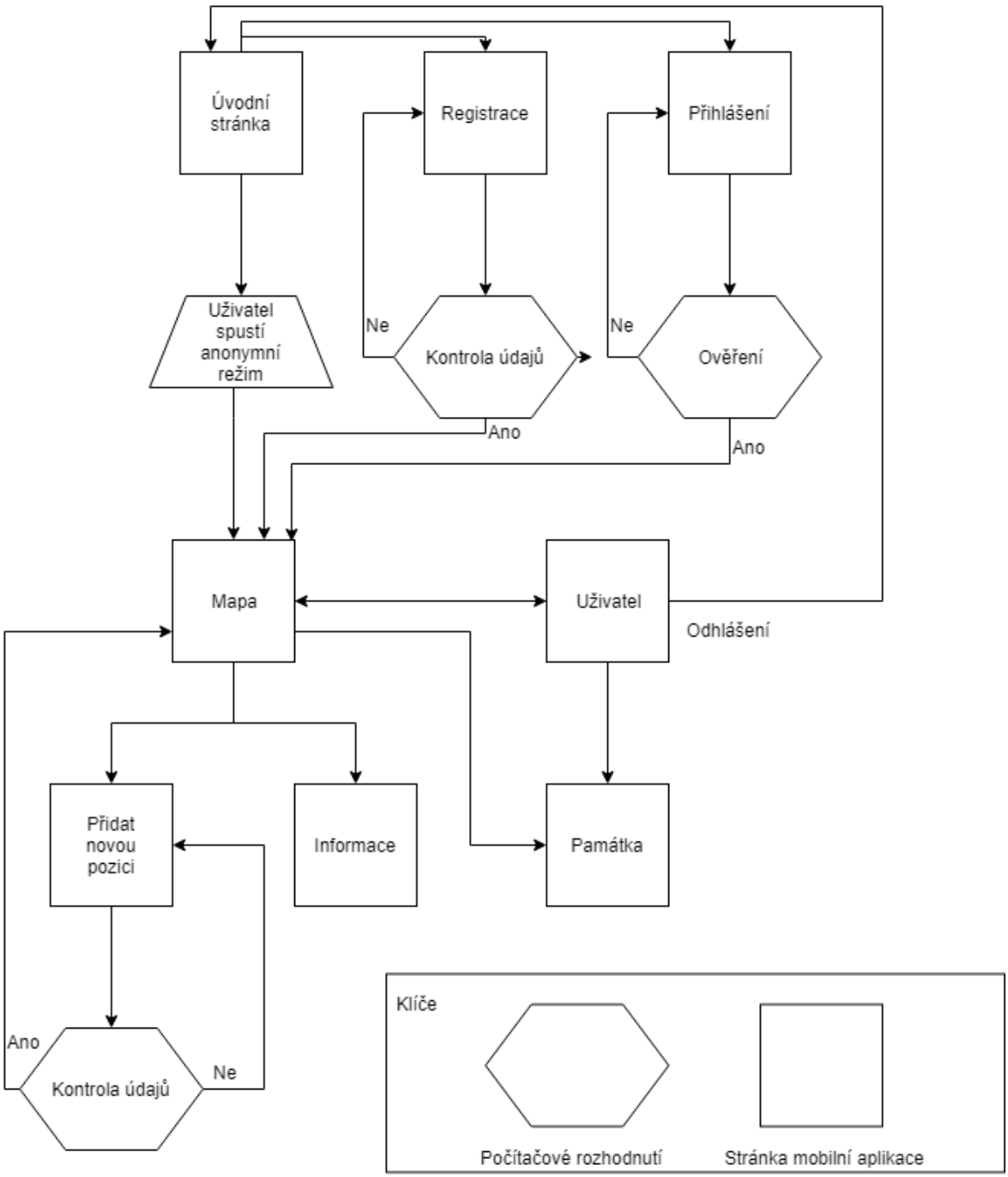
Ukázka kódu napsaná pro mobilní aplikaci v React Native. Tato část představuje prezenční komponentu používanou pro zobrazení bodů na mapě. Celkový kód je mírně modifikovaný, aby poskytl lepší představu o tom, jak vypadá čisté JSX. Za povšimnutí stojí podobný styl zápisu jako například HTML či XML souborů.

```
export default class Markers extends Component {

  // Missing few lines of code

  render() {
     return (
        <Marker
        key={monument._id}
        pinColor={newCategory.color || "#fff"}
        coordinate={{
          latitude: monument.latitude,
          longitude: monument.longitude
        }}
      >
        <MapView.Callout>
          <View>
            <Text>{monument.title}</Text>
            <Text>{monument.description}</Text>
            <Text>Category : {newCategory.title}</Text>
            <OdinaryButton
              name={i18n.t("home.like")}
              method={() => likeSingleMonument(monument)}
            />
            <OdinaryButton
              name={i18n.t("home.more")}
              method={() => navigateToSingleMonument(navigate, monument)}
            />
          </View>
        </MapView.Callout>
      </Marker>
    );
  }

}
```

Obrázek 5-1: Ukázka kódu v React Native (Markers.js)

5.2 FRONTEND

5.2.1 NAVIGACE



Obrázek 5-2: Navigace

Vizuální průchod mobilní aplikací je znázorněn pomocí schématu (Obrázek 5-2). Schéma obsahuje seznam klíčů, které usnadní orientaci. Při samotném spuštění aplikace se uživatel ocitne na úvodní stránce, kde si může vybrat druh přístupu. Dále pak následuje několik možností, které si uživatel může zvolit v závislosti na zvoleném přístupu, například může přejít na stránku s mapou.

Samotná navigace v mobilní aplikaci je realizována pomocí knihovny React Navigation. Knihovna vznikla na popud vývojářů, kterým chybělo rozumné a standardizované řešení navigace v mobilních aplikacích [14]. React Navigation podporuje základní gesta pro navigaci nebo třeba použití klasických stylů navigace v mobilních aplikacích. Součástí hlavičky v horní části aplikace bývá název stránky s případnými tlačítky, například tlačítko „jít zpět“.

V implementaci jsou použité tři různé technologie pro správu navigace mobilní aplikace. První z nich je klasická zásobníková navigace. Stránky se na sebe postupně nabalují a v případě potřeby se uživatel může vrátit na předposlední navštívenou stránku. Výhodou této techniky je jednoduchý způsob použití mobilní aplikace pro uživatele. Zásobníková navigace se používá například na uživatelské stránce. Druhou technologií je základní přepínací navigace, která slouží k obyčejnému přepínání mezi dvěma celky. V okamžiku, kdy chceme dosáhnout pouze jednoduchého přesunu mezi dvěma obrazovkami, reprezentuje tento způsob ideální řešení. V mobilní aplikaci je použita například pro přesun mezi úvodní stránkou a stránkou s mapou. Posledním způsobem navigace v mobilní aplikaci je tabulátorová navigace, konkrétně spodní tabulátorová navigace. V dolní části aplikace jsou umístěny ikony sloužící pro přesun mezi základními obrazovkami. Výhodou je zjednodušení průchodu aplikací pro uživatele, protože v libovolném okamžiku bude mít člověk přehled, v jaké části se právě nachází.

5.2.2 OBRAZOVKA S MAPOU

Obrázek 5-3: Obrazovka s mapou

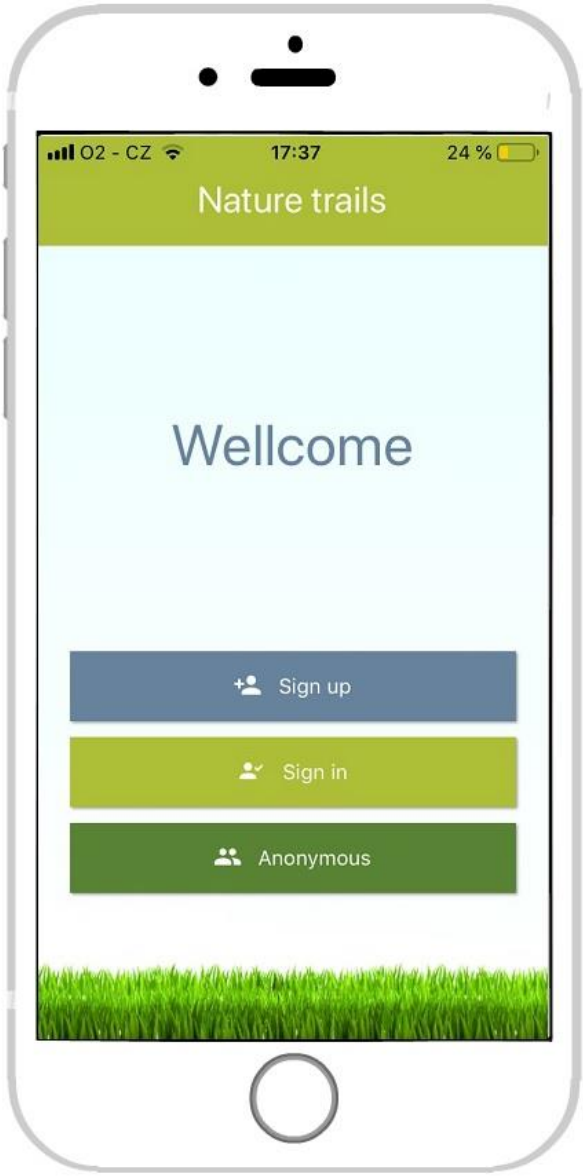
První stránka, která se po přihlášení uživatele či anonymním přístupu uživatele zobrazí, je obrazovka s mapou. Největší a zároveň hlavní část obrazovky tvoří mapa od společnosti Google, která obsahuje značky reprezentující zajímavá místa v okolí. Uživatel má možnost přejíždět prstem po mapě a zkoumat různé oblasti. Ve chvíli, kdy se bude chtít uživatel vrátit na svoji aktuální pozici, stačí kliknout na přesun na vlastní pozici. Pokud bude někdo chtít přidat vlastní bod do mapy, stačí využít funkce dlouhého stisknutí stejného místa na mapě, popřípadě kliknout na ikonku v pravém horním rohu, která přijme jako místo přidání aktuální polohu. Pro přidávání památky musí být uživatel přihlášený. Každá památka na mapě má svoji kategorii, která se odlišuje barevně od ostatních. V případě bližších informací stačí kliknout na ikonku informace. Pokud se uživateli líbí libovolné místo na mapě, je možné, dát památce „to se mi líbí“. Ve chvíli, kdy se chce uživatel dozvědět více informací o konkrétním bodě na mapě, stačí rozkliknout tlačítko s nápisem více informací.

5.2.3 UŽIVATELSKÁ OBRAZOVKA

Obrázek 5-4: Uživatelská obrazovka

Druhá obrazovka, na kterou lze přejít ze spodní navigace je uživatelská obrazovka. Součástí jsou uživatelské informace a také seznam památek přidaných přihlášeným uživatelem. Seznam památek je seřazený podle data přidání. Pokud je uživatel přihlášený přes Gmail, je na obrazovce jeho profilová fotografie. V okamžiku, kdy nemá nastavenou fotografii, bude uživatelský profil reprezentovaný jeho iniciály. V seznamu památek lze každou památku jednoduše rozkliknout a zjistit tak více informací. V okamžiku, kdy uživatel klikne na památku, je zobrazeno její jméno, popis, kategorie, kam se řadí, jméno autora, který památku přidal a počet „to se mi líbí“. Uživatel má také možnost smazat konkrétní památku. Uživatelská obrazovka poskytuje mimo jiné možnost odhlásit se. Ve chvíli, kdy není nikdo přihlášený, vypadá uživatelská stránka jinak. Zobrazí se pouze informace, že nikdo není přihlášený a také možnost se dodatečně přihlásit.

5.2.4 ÚVODNÍ OBRAZOVKA



Obrázek 5-5: Úvodní obrazovka

Úvodní stránka poskytuje 3 základní možnosti přístupu k aplikaci. První možností je nejlehčí způsob základního použití, což znamená přejít do aplikace jako anonymní uživatel. Ten je omezen pouze k prohlížení památek z okolí. Při výběru přihlášení či registrace se uživatel dostane do aplikace s rozšířenou možností použití. Dostane tak šanci přidávat památky ze svého okolí, popřípadě odebírat vlastní. V poslední řadě je zde možnost přístupu do aplikace přihlášením přes Gmail. Při přechodu mezi obrazovkami může nastat situace, kdy stránka není ještě připravena k načtení. V tu chvíli nahradí aktuální obrazovku stránka s načítáním. Stránka pracuje s jednoduchou animací implementovanou JavaScriptem pomocí techniky zvané Sprite, což znamená sebrat řadu obrázků a pustit je ve smyčce rychle za sebou.

5.2.5 OFF-LINE REŽIM

Obyčejní uživatelé se během používání aplikace mohou dostat do situace, kdy nebudou mít připojení k internetu. Jejich telefon například nemusí mít dostatečný signál nebo mohou mít vypnuté datové spojení. Vzhledem k faktu, že připojení k internetu v přírodě není většinou kvalitní, je v hodné počítat i s režimem off-line. Ve chvíli, kdy uživatel není připojen k internetu, tak není možné zprovoznit připojení k serveru a tudíž není možné ukládat aktuální data do databáze.

Řešením je implementace jisté formy off-line režimu. Uživatel má tedy možnost přidávat a odebírat různá, zajímavá místa do mapy bez ohledu na připojení. Při kliknutí na tlačítko „Přidat novou památku“ se pošlou JSON data obohacená uživatelským *id* a *tokenem* do databáze. Paralelně s tím se památka uloží do Store přes *monumentReducer*. Data jsou tedy dostupná v databázi, ale i lokálně u uživatele na telefonu. Dalším prvkem je lokální ukládání dat v mapě. Stačí jednou načíst data do mapy a potom už při prohlížení nemusí být uživatel přihlášený k internetu.

5.2.6 GUI

Vzhled mobilní aplikace nebyl od začátku vývoje ze strany klienta žádným způsobem určený. Hlavním požadavkem bylo implementovat prostředí, které poskytuje zajímavá místa v okolí s možností zjištění více informací. Vhodným zadostiučiněním byla implementace mapy zobrazující památky podle jejich polohy. Zbytek aplikace neměl jasně danou strukturu.

Mobilní design je implementovaný jako kombinace ověřené technologie a vlastní kreativity. Mezi technické věci patří například jasný postup při výběru velikostí či zarovnání písma. Používání předepsaných, standardizovaných pravidel [15]. Při implementaci vzhledu jsem proto často vycházel z pokynů Material Design pro tvorbu grafického uživatelského rozhraní. Dalším využitím Material Design bylo například použití jejich ikonek v mobilní aplikaci. Pokud se na vzhled podíváme z kreativního pohledu, tak například součástí mobilního designu je unikátní Sprite [16] implementovaný v JavaScriptu, který se používá pro načítání aplikace. Další kreativní výsadou je ojedinělá kombinace barev, vytvořená podle vlastního cítění. Mobilní aplikace používá 3 dominantní barvy (Obrázek 5-6).

Obrázek 5-6: Hlavní barevná kombinace

5.2.7 KNIHOVNY

5.2.7.1 REDUX

V celém projektu mobilní aplikace je hlavním řešením správy dat právě Redux. Základem této technologie je Store, v našem případě je Store implementovaný v jednom souboru, který má zhruba 20 řádek kódu. Store klasicky drží celkový stav aplikace. Většina mobilních komponent je právě spojená s tímto úložištěm. V okamžiku, kdy se změní stav, v tu chvíli se komponenty pracující s konkrétní informací překreslí. Pro změnu dat v úložišti jsou implementované jednotlivé objekty akce. Každá akce se volá pomocí metody *dispatch*. V mobilní aplikaci je implementováno hned několik souborů pracujících s akcí, například akce pracující s památkami či akce pracující s uživateli. Celkový chod technologie Redux uzavírá *reducer*, který upravuje data uložená ve *store*. Reducer dostane aktuální stav aplikace a provede patřičnou úpravu. V mobilní aplikaci je rozdělen *reducer* na 3 části:

1. Uživatelská část
2. Významné místo v okolí
3. Kategorie

Technologie Redux je poměrně robustní nástroj, který může někdy přivodit více škody než užitku. V situacích, kdy pracujeme s menší aplikací, není vhodné tuto technologii vůbec používat. Přináší zbytečnou komplexitu do jednoduché aplikace. Naopak ve chvílích, kdy máme v rukou větší projekt, jako je například implementace této mobilní aplikace, dokáže Redux zpřehlednit a usnadnit práci i pro budoucí použití.

5.2.7.2 REACT NATIVE MAPS

Pravděpodobně nejdůležitější komponentou v celé mobilní aplikace je *MapView*, z doplňkového balíčku *react-native-maps*. Zprovoznění probíhá klasickou cestou stažení balíčku pomocí technologie *npm* nebo *yarn*. React-native-maps spravuje společnost Airbnb. *MapView* poskytuje mapu pro mobilní aplikaci s možností přidání jednotlivých bodů. Mapu různých významných památek v okolí tvoří výchozí mapa neboli mapa od společnosti Google. Komponenta *Marker*, která se předává jako potomek komponenty *MapView*, a která pochází ze stejného balíčku, označuje různá zajímavá místa z okolí.

5.2.7.3 I18N

I18n slouží jako balíček pro překlad mobilní aplikace do různých jazyků. Do paměti se texty ukládají v podobě JSON objektů. I18n nejčastěji používá metodu *t* sloužící pro přeložení do aktuálně zvoleného jazyka. Aplikace podporuje češtinu a angličtinu.

5.3 BACKEND

Pro tvorbu backendu byl zvolen framework NodeJS, který umožňuje spouštět kód v jazyce JavaScript bez pomoci webového prohlížeče. NodeJS se běžně používá pro psaní serverové části. Hlavní výhodou NodeJS je fakt, že umí výborně komunikovat s více klienty zároveň. Vzhledem k faktu, že aplikace je předurčená pro víceuživatelské využití, je součástí kódu několik asynchronních volání. Pro pohodlnější implementaci, například používáním pokročilých asynchronních operací, byl použit kompilátor Babel.

Pro navázání komunikace se serverem, neboli uvedení do ostrého provozu, je použito cloud hosting Heroku (datové úložiště na internetu) [17]. Výhodou Heroku je fakt, že použití je zcela zdarma. Při založení projektu na Heroku se vytvoří název aplikace, v našem případě nese název *https://nature-heroku.herokuapp.com/*. Veškerá komunikace je implementována přes REST API rozhraní. Sice NodeJS není přímo spojený s REST API, ale i tak se společně v hojné míře používají. V implementaci aplikace jsou použité HTTP metody, jako například GET, DELETE, POST. Součástí REST API implementace je také *middleware* neboli funkce, které se vykonají s každým HTTP požadavkem. Ve stručnosti serverové řešení funguje tak, že každé API, až na jedno (dotaz na Mapu), které dostane požadavek, vrací data ve formě JSON. Návratový kód zpravidla bývá 200, což reprezentuje úspěch, popřípadě 400, což značí neúspěch.

Veškeré URL použité pro komunikaci začíná vždy */api* a pokračuje v závislosti na dotazu. Například pokud by nás zajímalo, jak získat všechny kategorie, stačí poslat dotaz ve tvaru: *https://nature-heroku.herokuapp.com/api/CultureCategory*, který vrací JSON. Vzhledem k implementaci architektury REST je následné jednodušší testování aplikace.

5.3.1 BEZPEČNOST

Při použití aplikace hraje důležitou roli, zda je uživatel přihlášený či nikoliv. V implementaci je k těmto účelům použit nástroj Passport, který slouží jako *middleware* pro NodeJS. Passport podporuje autentizaci v podobě jména a hesla přes Facebook, Gmail a mnohé další. V mobilní aplikaci je implementována autentizace přes email od společnosti Google. Pro ověření přihlášení uživatele se v aplikaci používá JSON Web Token zkráceně JWT, který se postará při přihlášení uživatele o přidělení nově vytvořeného tokenu. Ve chvíli, kdy uživatel bude chtít přidat nějaké významné místo z okolí do aplikace, tak se pošle v hlavičce autorizace právě vytvořený token, který ověří, zda je uživatel přihlášený či nikoliv.

5.3.2 REST API

Spolupráce server-klient probíhá pomocí služby REST. Na vzdálená data se pokaždé ptá React Native pomocí metody *fetch.* Součástí dotazu vždy bývá výchozí URL, v našem případě je to: *https://nature-heroku.herokuapp.com/api* obohacené o konkrétní příponu, například: */CultureCategory*. Nyní adresa vypadá následovně: *https://nature-heroku.herokuapp.com/api/CultureCategory*. Dále je vybrána vhodná metoda. Pokud se ptáme na výčet kulturních kategorií, přidáme metodu *GET*. Pokud použijeme metodu *POST*, tak součástí hlavičky bývá *Authorization: userToken* neboli přidáme unikátní uživatelský token, který na druhé straně vyhodnotí server jako validní či nevalidní. V jednoduchosti to znamená, že uživatel může přidat patřičné údaje, protože jeho token byl vytvořený při registraci/přihlášení nebo to server jednoduše zamítne. Poslední součástí metody *fetch* je *body*. V případě metody *POST* bývá v *body*, někdy také označováno jako tělo dotazu, umístěna přenášená data určená pro zapsání.

5.3.3 DATABÁZE

Pro komunikaci mezi NodeJS a NoSQL databází MongoDB je implementován modul Mongoose, který usnadňuje přístup k jednotlivým objektům. Mongoose umožňuje klasické operace pro tvorbu, smazání, úpravu a čtení různých objektů. Pro tvorbu jednotlivých schémat objektů neboli modelů, jako je například památka, se používá právě Mongoose. Každé schéma definuje přibližnou podobu jednotlivých modelů. Při práci s MongoDB není nutné se učit jednotlivé příkazy. S MongoDB se komunikuje pomocí jazyka JavaScript a navíc práce s MongoDB je mnohonásobě flexibilnější než třeba použití relační databáze.

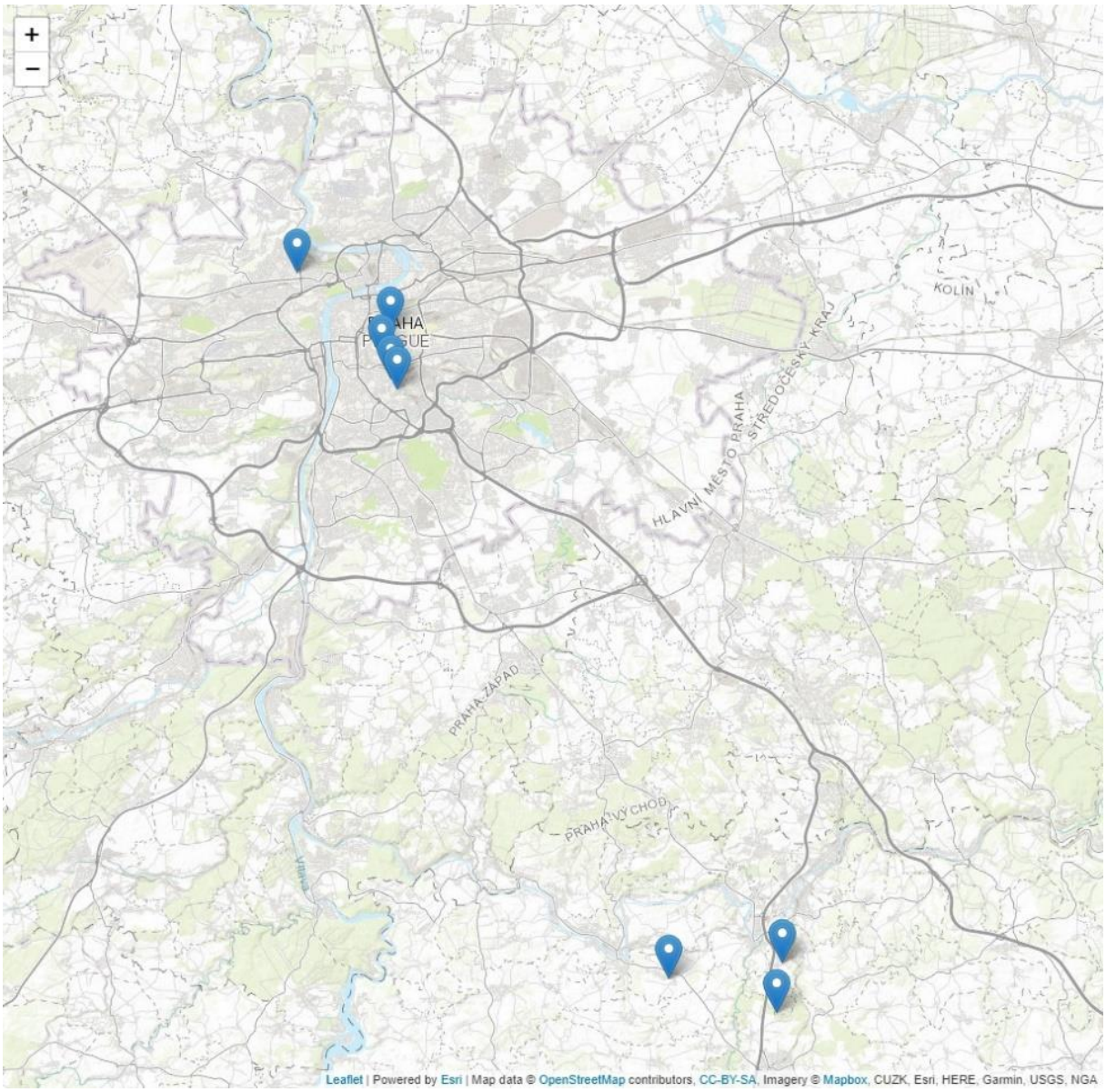
5.3.4 KNIHOVNY

5.3.4.1 EXPRESS

Společně s NodeJS je implementován framework Express, který byl při svém vývoji inspirován jiným jazykem, než je JavaScript a to konkrétně skriptovacím, objektově orientovaným jazykem Ruby. Express je nejpoužívanější framework ve světě NodeJS, proto část implementovaného kódu tvoří právě on. Express samotný poskytuje robustní sadu funkcí pro webové a mobilní aplikace. Hlavní předností je *router*, který přesměruje URL dotaz na přesně daný kontrolér. Router rozeznává různé HTTP metody, jako jsou například GET, DELETE, POST a jiné. Každý kontrolér je potom spojený s asynchronní akcí.

5.3.4.2 LEAFLET

Pro vizualizaci geografických dat byl použit již dříve představený Leaflet. V serverové části je nastavený *router* na dotaz (*…/api/CultureMonumentsMap*), který zobrazí geografické informace v podobě mapy na webové stránce. Soubor HTML ze serveru je obohacený při vykreslování o data z databáze. Celkový výstup je spojením souboru *map.html* a souhrn památek v podobě JSON. Výsledný soubor je nastavený za pomocí nástroje ESRI Leaflet na komunikaci s ESRI produktem ArcGIS. Při následné změně hodnot v databázi se při znovuotevření dané webové stránky aktualizují data.



Obrázek 5-7: Leaflet mapa

6 TESTOVÁNÍ

Ve světě Reactu v oblasti testování hraje hlavní roli knihovna Jest. Jedná se o nástroj používaný převážně pro psaní Unit testů. Tento druh testování je pro programátory poměrně snadný. Stačí vytvořit soubor s koncovkou test a napsat několik řádků kódu. Testy jako takové probíhají po spuštění asynchronně. Vývoj mobilní aplikace byl už od začátku zaměřený na uživatele. Jednou z hlavních priorit bylo navrhnout aplikaci, která bude uživatelsky přívětivá a intuitivní. Cílem tedy není tvorba automatických testů, ale testování použitelnosti aplikace. Uživatel je ve středu mobilní aplikace. Testování má například pomoci k lepší orientaci v aplikaci.

Vzhledem k situaci, že vyvíjená mobilní aplikace není zatím dostupná na Google Play ani App Store, bylo nejsnazším řešením přimět uživatele k instalaci aplikace Expo. Ta umožňuje spuštění aplikace na vlastním telefonu. Stačí přes Expo vytvořit trvalou URL projektu a sdílet tuto adresu s ostatními uživateli, kteří budou mobilní aplikaci testovat.

Aplikace byla otestována 3 různými uživateli. Podle studie doktora Nielsena [18] stačí 5 uživatelů k odhalení 75 % problémů souvisejících s použitelností aplikace. Uživatelé testovali pouze prototyp mobilní aplikace, který splňuje hlavní funkcionalitu aplikace. Jedna z vlastností prototypu je fakt, že heslo je psané jako obyčejný text, není nijak skryto při psaní. Každý uživatel má rozdílné zkušenosti s mobilními aplikacemi. Testování proběhlo u každého uživatele ve dvou fázích. V první fázi byla připravena sada testovacích případů, které uživatel musel projít.

1. Spustit aplikaci jako anonymní uživatel.
2. Najít nejbližší památku v okolí z kategorie Town.
3. Přihlásit se do aplikace pomocí emailu od společnosti Google.
4. Přidat dvě zajímavá místa ze svého okolí.
5. Smazat libovolné nově přidané místo
6. Odhlásit se z aplikace.

V druhé fázi mohl uživatel využívat aplikaci podle svého uvážení. Nebyly stanoveny žádné cíle, které musí uživatel splnit. Na konci druhé fáze byly pro každého uživatele připraveny otázky:

1. Jaké má uživatel zkušenosti s podobnou mobilní aplikací?
2. Má aplikace skutečně nějakou užitečnou hodnotu?
3. Je aplikace snadná pro běžné použití?
4. Líbí se uživateli design aplikace?
5. Jaké změny navrhuje uživatel v aplikaci udělat?

Výsledky testování budou sloužit pro budoucí úpravu aplikace.

6.1 UŽIVATEL 1

První uživatel má základní zkušenosti s elektrotechnikou. Dotykové zařízení jako mobilní telefon je pro něj stále ještě svět plný záhad. S touto mobilní aplikací se setkal poprvé při testování. Již dříve se několikrát podílel na testování, ale nikdy ne v oblasti IT. Testování proběhlo na zařízení Xiaomi mi a2 dne 14. 3. 2019.

Průběh testování v první fáze probíhal hladce až do chvíle, kdy měl uživatel místo smazat. Hledal tlačítko na odstranění zajímavého místa v okolí na špatné obrazovce. V druhé fázi uživatel přidal ještě několik dalších míst. Aplikace se mu velice líbila. Hlavní předností byla příjemná barevná kombinace. Uživatel používá pro hledání zajímavých míst mapy od společnosti Seznam. V aplikaci vidí velký přínos. Velice rád cestuje a aplikace mu přijde dosti vhodná i pro ostatní cestovatele. Používání bylo snadné, až na drobné detaily. Jedno z navrhovaných zlepšení bylo přidat přesun prstem mezi obrazovkami. Další poznámkou bylo přidat jinou funkcionalitu do aplikace. Dále zmínil fakt, že není jasné, která obrazovka ze spodní navigace je zapnutá. Poslední připomínkou bylo zjištění, že jedno ze dvou tlačítek přihlášení přes Gmail nefunguje.

6.2 UŽIVATEL 2

Tento uživatel má velké zkušenosti s mobilním zařízením, které získal dlouhodobou praxí. Uživatel aplikaci již zná z prvotní fáze vývoje. Aplikaci nikdy dříve nepoužíval. Nemá žádné předchozí zkušenosti s testováním, ale má drobné zkušenosti s programováním. Testování proběhlo na zařízení iPhone 7 dne 14. 3. 2019.

Během první fáze uživatel neměl žádný problém s testovacím scénářem. Při testování necítil potřebu ptát se na dodatečné informace. Vše proběhlo podle očekávání v poměrně rychle. Jediné drobné zdržení bylo najít zpětně obrazovku přihlášení. Ve druhé části uživatel po krátkém používání aplikace sdělil, že nemá zkušenosti s podobnou aplikací. Jeho zájem není chodit do přírody či hledat zajímavá místa z okolí. Aplikace mu přijde jednoduchá a intuitivní. Nemyslí si, že se bude v budoucnu hojně využívat. Vzhled aplikace se mu zdál být průměrný a dodatečně navrhl přidat možnost vyhledávání památek prostřednictvím vyhledávacího panelu.

6.3 UŽIVATEL 3

Poslední uživatel má mírné zkušenosti jak s operačním systémem Android, tak s operačním systémem od společnosti Apple. Svůj mobilní telefon používá velmi málo. Uživatel pracuje převážně se základními aplikacemi, které telefon poskytuje. Je pro něj aplikace zcela neznámá. Zkušenosti s testováním nemá. Test proběhl na zařízení iPhone 5s dne 14. 3. 2019.

Na začátku první fáze měl uživatel problémy s přednastaveným cizím jazykem, proto byla mobilní aplikace optimalizována a přepnuta do češtiny. Během testování došlo k problému, kdy uživatel nevěděl, jak přidat místo na mapě. Zbytek testovacích případů proběhl v pořádku. Uživatel nemá zkušenosti s podobnou službou. Aplikace mu přijde vhodná pro lidi, kteří mají větší technologické znalosti. Používání prototypu bylo zpočátku dosti zmatené, ale ke konci uživatel pociťoval větší jistotu v ovládání. Vzhled aplikace mu přišel vyhovující. Případné nedostatky pocítil během registrace, neboť není vidět při psaní na klávesnici heslo. V budoucnu by zavedl podporu pro více jazyků.

7 DALŠÍ POSTUP

Výsledná aplikace splňuje požadovanou funkcionalitu. Vzhledem k tomu, že je aplikace zatím prázdná, neboli chybí základní množství zajímavých míst po okolí, bude vhodné, doplnit databázi o některé všeobecně známé památky. Zpočátku se zaměřením na Českou republiku. Je třeba ještě provést další testy mobilní aplikace před nasazením do ostrého provozu. Testování naznačilo několik možných vlastností, které je možné do aplikace implementovat. Zároveň se objevilo několik komplikací, které by bylo vhodné odstranit. V budoucnu to znamená sejít se společně s ostatními, kteří s aplikací přichází do kontaktu a prodiskutovat možné změny či rozšíření aplikace. Následně provést výsledné transformace. Aplikace potom projde procesem finálních úprav a bude připravena k vypuštění do světa.

Pro nasazení aplikace na Google Play a App Store bude nutné vytvořit balíčky aplikací, například pomocí technologie, kterou poskytuje Expo. Dále je třeba zajistit vývojářský účet v každém online obchodě zvlášť. Ve finále se nahrají oba balíčky na příslušný server. Po spuštění se začne pracovat na případném zlepšení či možné rozšiřitelnosti aplikace.

8 ZÁVĚR

Cílem bakalářské práce bylo vytvořit mobilní aplikaci pro reportování přírodovědeckých informací a GPS lokace. Aplikace byla úspěšně napsaná v jazyce JavaScript. Frontendová část byla psaná v React Native. Backend byl vytvořen v NodeJS. Na začátku byl představen React, React Native a příslušné technologie, které s vývojem souvisí. Před samotnou implementací byla provedena analýza a návrh aplikace. Klíčovou složkou analýzy bylo vyřešit cílovou skupinu, prozkoumat různá rizika a také malé množství informací ohledně mobilních aplikací a nativního vývoje. Nedílnou součástí analýzy bylo zkoumání různých dostupných technologií a práce s geografickými daty. Pro aplikaci vypracován návrh ohledně frontendu a backendu. Během implementace bylo nutné už od začátku spojit funkční serverovou část s mobilní aplikací. Při vývoji byl hlavní důraz kladen na to, umístit uživatele do středu aplikace [19]. Důležitou roli hrála snadná přehlednost, jednoduchost, čitelnost a uživatelsky přívětivý design. Nakonec bylo provedeno testování mobilní aplikace. Jednotlivé testovací případy prokázali patřičnou funkcionalitu celé aplikace.

Ve chvíli, kdy bude aplikace hotová, tak projde procesem finálních úprav. Potom bude následovat nasazení do ostrého provozu na Google Play a App Store.

Použité zkratky

2G, 3G, 4G Generace bezdrátové telefonní technologie mobilního telefonu.

API Application Programming Interface

CSS Cascading Style Sheets

ECMA European Computer Manufacturers Association

ES2015 ECMAScript 2015

ES6 ECMAScript 6

GPS Global Positioning System

GUI Graphical User Interface

Gmail Google Mail

HTML Hypertext Markup Language

HTTP Hypertext Transfer Protocol

IT Informační technologie

JSON JavaScript Object Notation

JSX JavaScript XML

JWT JSON Web Tokens

MVC Model-view-controller

REST Representational state transfer

SPA Single-Page Application

SQL Structured Query Language

UCD User Centered Design

UI User interface

VS Code Visual Studio Code

Wi-Fi Wireless Fidelity

XML Extensible Markup Language

ČVUT České vysoké učení technické

Seznam literatury

1. 20 let mobilů v Česku. Vzpomínáte na mobilní pravěk? – MobilMania.cz. MobilMania.cz – O mobilech víme vše [online]. Dostupné z: *https://www.mobilmania.cz/clanky/20-let-mobilu-v-cesku-vzpominate-na-mobilni-pravek/sc-3-a-1317211/default.aspx*
2. Attention Required! | Cloudflare. Attention Required! | Cloudflare [online]. Dostupné z: *https://geekflare.com/bet-javascript-frameworks/*
3. Road to learn React by Robin Wieruch [Leanpub PDF/iPad/Kindle]. Leanpub: Publish Early, Publish Often [online]. Copyright © 2010 [cit. 04.05.2019]. Dostupné z: *https://leanpub.com/the-road-to-learn-react*
4. React Native and the New Dream of Learn Once, Write Anywhere - The New Stack. Home - The New Stack [online]. Copyright © 2019 The New Stack. All rights reserved. [cit. 04.05.2019]. Dostupné z: *https://thenewstack.io/react-native-learn-write-anywhere/*
5. Why Do We Write super(props)? — Overreacted. Overreacted — A blog by Dan Abramov [online]. Dostupné z: *https://overreacted.io/why-do-we-write-super-props/*
6. Handling Events – React. React – A JavaScript library for building user interfaces [online]. Copyright © 2019 Facebook Inc. [cit. 14.05.2019]. Dostupné z: *https://reactjs.org/docs/handling-events.html*
7. JSX In Depth – React. React – A JavaScript library for building user interfaces [online]. Copyright © 2019 Facebook Inc. [cit. 04.05.2019]. Dostupné z: *https://reactjs.org/docs/jsx-in-depth.html*
8. Mapy.cz. Mapy.cz [online]. Dostupné z: *https://mapy.cz/*
9. Esri Leaflet. Esri GitHub | Open Source and Example Projects from the Esri Developer Platform [online]. Dostupné z: *https://esri.github.io/esri-leaflet/*
10. Balsamiq Wireframes | Balsamiq. Balsamiq. Rapid, effective and fun wireframing software. | Balsamiq [online]. Copyright © their respective owners [cit. 16.05.2019]. Dostupné z: *https://balsamiq.com/wireframes/*
11. Jan Hrdý / nature · GitLab. [online]. Dostupné z: *https://gitlab.com/hrdyjan1/nature/*
12. Apple AppStore vs. Google Play - souboj není vyrovnaný | Dotekomanie.cz. Dotekomanie.cz - vše o mobilech a tabletech [online]. Dostupné z: *https://dotekomanie.cz/2018/04/appfigures-zanalyzovali-rozvoj-aplikaci-na-app-store-a-google-play/*
13. GitHub - jamland/instabyte: Clone of Instagram made with React Native. The world's leading software development platform · GitHub [online]. Copyright © 2019 [cit. 04.05.2019]. Dostupné z: *https://github.com/jamland/instabyte*
14. Navigation - Lecture 6 - CS50's Mobile App Development with React Native - YouTube. YouTube [online]. Dostupné z: *https://www.youtube.com/watch?v=QHorNUuEXc0*
15. Android Developers. Android Developers [online]. Dostupné z: *https://developer.android.com/design*
16. Create a Sprite Animation with HTML5 Canvas and JavaScript { William Malone }. William Malone [online]. Dostupné z: *http://www.williammalone.com/articles/create-html5-canvas-javascript-sprite-animation/*
17. Cloud Application Platform | Heroku. Cloud Application Platform | Heroku [online]. Copyright © [cit. 17.05.2019]. Dostupné z: *https://www.heroku.com/*

18. NIELSEN, Jakob.. Usability inspection methods. New York [u.a.]: Wiley, 1994. ISBN 04-710-1877-5.
19. UCD – Wikipedie. [online]. Dostupné z: *https://cs.wikipedia.org/wiki/UCD*
20. [online]. Dostupné z: *https://expo.io/@hrdyjan1/nature*
21. Up and Running - Expo Documentation. [online]. Dostupné z: *https://docs.expo.io/versions/latest/workflow/up-and-running/*

Příloha

Spuštění mobilní aplikace

Pro spuštění aplikace existují dvě možné cesty. Obě ve výsledku poskytují stejnou funkci, spuštění aplikace, ale každá na to jde trochu jinak. První cesta vede k rychlému spuštění aplikace. Stačí si na telefon nainstalovat aplikaci Expo. Zkopírovat čárový kód, popřípadě odkaz z webových stránek [20] a spustit. V tuto chvíli máte pouze možnost aplikaci používat. Pokud chcete právo modifikovat, musíte zvolit druhou variantu, která je poněkud komplexnější. Pro sestavení a spuštění budou za potřebí zdrojové kódy k aplikaci [11]. Dalším krokem je opatřit si vhodnou kolekci nástrojů k programování aplikací v React Native, ideální variantou se jeví již dříve zmiňované Expo. K serverové komunikaci poslouží NodeJS. V případě absence NodeJS v PC je nutné požadovanou technologii doinstalovat. Právě NodeJS nám dále umožní instalace balíčku Expo. Dalším krokem je stažení aplikace Expo z online obchodu pro instalaci aplikací (Google Play, App Store), přičemž nezáleží na tom, zda má požadovaný telefon operační systém Android, či iOS. Po stažení aplikace se připojí obě zařízení (PC, mobilní telefon) do stejné sítě. Dále už jen stačí vlézt do správné složky v příkazové řádce a spustit aplikaci pomocí definovaných pravidel [21].