Vysoká škola ekonomická v Praze

Fakulta informatiky a statistiky

Proces vývoje mobilní aplikace v React Native

BAKALÁŘSKÁ PRÁCE

ve studijním programu Aplikovaná informatika

Autor: Tomáš Katz

Vedoucí bakalářské práce: Ing. Tomáš Bruckner, Ph.D.

Praha, duben 2019

Prohlášení

Prohlašuji, že jsem bakalářskou práci s názvem Proces vývoje mobilní aplikace v React Native vypracoval samostatně za použití v práci uvedených pramenů a literatury.

V Praze dne ……. …….

Tomáš Katz

Poděkování

Rád bych poděkoval svým kolegům ze společnosti Robeeto za vstřícnost při výběru a zpracování tématu. Dále bych chtěl poděkovat své přítelkyni, Nadije Šlenčakové, za konzultace a korektury během doby psaní této práce.

V neposlední řadě bych rád poděkoval vedoucímu mé práce, Ing. Tomáši Bruckner, Ph.D., za ochotu a velké množství užitečných rad a doporučení, které mi byly poskytnuty a bez kterých by realizace této práce nebyla možná.

Abstrakt

Bakalářská práce se zabývá procesem vývoje mobilní aplikace za pomoci technologie React Native. Cílem práce je poskytnout komplexní přehled v problematice moderního vývoje mobilních aplikací.

Nejprve je popsáno zadání a náležitosti zkoumaného projektu. Další část představuje rozdíly mezi mobilní a webovou platformou, včetně identifikace stávajících trendů. Následně je představeno kompletní technologické řešení. A konečně samotná implementace aplikace, která je doprovázena zhodnocením technologie React Native v praxi.

Poznatky získané během návrhu a tvorby komerční aplikace poskytují potřebná řešení problémů, se kterými se vývojáři velmi pravděpodobně během implementace setkají.

Klíčová slova

React Native, mobilní aplikace, vývoj aplikace, analýza

Abstract

Bachelor thesis is focused on process of development mobile application in React Native. The aim of the bachelor thesis is to provide a complex insight into a modern mobile application development process.

At first, specifications and requirements of analyzed project are defined. The next section describes differences between mobile and web platform, including analysis of modern trends. Consequently, a complete technological stack is introduced. Lastly, the bachelor thesis focus on the application development process, followed by the hands-on evaluation of React Native.

Findings obtained during designing and implementing mobile application can be used as answers to problems that developers will presumably come across to.

Key words

React Native, mobile application, application development, analysis

Seznam obrázků

Obrázek 1: Aplikační schéma, zdroj: (autor 2019) 29
Obrázek 2: Přihlášení, zdroj: (autor 2019) 30
Obrázek 3: Přehled, zdroj: (autor 2019) 31
Obrázek 4: Poptávky, zdroj: (autor 2019) 32
Obrázek 5: Zakázky, zdroj: (autor 2019) 33
Obrázek 6: Profily, zdroj: (autor 2019) 34
Obrázek 7: Detail poptávky, zdroj: (autor 2019) 35
Obrázek 8: Detail zakázky, zdroj: (autor 2019) 36
Obrázek 9: Přidání nabídky, zdroj: (autor 2019) 37
Obrázek 10: Soukromá konverzace, zdroj: (autor 2019) 38
Obrázek 11: Galerie, zdroj: (autor 2019) 39
Obrázek 12: Nastavení, zdroj: (autor 2019) 40

Obsah

1. ÚVOD 9

2. PŘEDSTAVENÍ PROJEKTU 11

2.1 SPOLEČNOST 11
2.2 MODELY FUNGOVÁNÍ 11
2.2.1 Objednávkový systém 12
2.2.2 Poptávkový systém 12
2.3 ZÁKLADNÍ PILÍŘE SLUŽBY 12
2.3.1 Profily 12
2.3.2 Listing 13
2.3.3 Notifikace 13
2.4 BUSINESS PROBLEMATIKA 14
2.5 PŘÍNOS MOBILNÍ APLIKACE 16

3. ROZDÍLNOST PLATFOREM 17

3.1 PROSTŘEDÍ 17
3.2 UI/UX 17
3.2.1 Material Design 18
3.2.2 Flat Design 18

4. VÝBĚR TECHNOLOGIE 19

4.1 PRIMÁRNÍ KNIHOVNY 19
4.1.1 React 19
4.1.2 React Native 20
4.1.3 Expo 21
4.2 PODPŮRNÉ KNIHOVNY 22

5. IMPLEMENTACE 26

5.1 PROJEKTOVÁ STRUKTURA 26
5.2 PRAVIDLA A SPOLEČNÉ PRINCIPY 28
5.3 APLIKAČNÍ SCHÉMA 29
5.4 PŘIHLÁŠENÍ A REGISTRACE 30
5.5 HLAVNÍ OBRAZOVKY 31
5.5.1 Přehled 31
5.5.2 Poptávky 32
5.5.3 Zakázky 33
5.5.4 Profily 34
5.6 VEDLEJŠÍ OBRAZOVKY 35

5.6.1 Detail poptávky ... 35
5.6.2 Detail objednávky ... 36
5.6.3 Formuláře ... 37
5.6.4 Soukromá konverzace ... 38
5.6.5 Galerie ... 39
5.6.6 Nastavení ... 40
5.7 KOMPONENTY ... 41
5.8 DEPLOYMENT ... 43
6. ZHODNOCENÍ TECHNOLOGIE ... 46
6.1 KLADNÉ STRÁNKY TECHNOLOGIE ... 46
6.2 ZÁPORNÉ STRÁNKY TECHNOLOGIE ... 47
6.3 SHRNUTÍ ... 48
7. ZÁVĚR ... 49
8. ZDROJE INFORMACÍ ... 51

1. Úvod

Ve světě vývoje webových aplikací se profesionálně pohybuji více než 4 roky. Webové stránky, potažmo webové aplikace se staly nedílnou součástí každé společnosti, která se chce úspěšně prezentovat v online světě. Pro samotné frontend developery je dnes naprostou nutností ovládat technologie HTML, CSS a JavaScript.

A právě JavaScript si během posledních let prošel radikálním vývojem, který ho zařadil mezi nejpoužívanější a nejpopulárnější programovací jazyky současnosti. Zároveň jsou pryč časy, kdy si developer vystačil pouze s čistým JavaScriptem, popřípadě knihovnou typu jQuery. Pro účely vývoje kvalitních webových aplikací je nutností komplexní framework. Dnes na trhu dominují především dva hráči – React od Facebooku, nebo Angular od Googlu. A právě s Reactem jsem spojil velkou část svého profesionálního života. [1] [2]

Trhu mobilních aplikací dnes vládnou společnosti s nejrozšířenějšími operačními systémy této platformy – Apple a Google. Apple dnes u svého operačního systému iOS sází na programovací jazyk Swift, který vychází z původního Objective-C. Při současné situaci na trhu je poptávka po iOS vývojářích opravdu velmi vysoká, i proto se tedy jedná o velmi dobře placenou pracovní pozici. Základem Androidu je pro změnu Java, která je oproti řešení Applu daleko rozšířenější a není striktně využívána pouze pro vývoj mobilních aplikací. Z hlediska zastoupení na trhu má Android mnohonásobně větší základnu uživatelů a zařízení, které běží právě na Androidu, avšak z hlediska monetizace aplikací představuje iOS daleko zajímavější platformu.

Vývoj webových a mobilních aplikací byl vždy považován spíše za dva oddělené profesní světy. V minulosti existovaly nástroje, které umožnily webovému vývojáři napsat mobilní aplikaci za pomoci ryze webových technologií – jednalo se o tzv. hybridní aplikace. Mezi nejpopulárnější řešení patří Ionic nebo PhoneGap. Tento druh aplikací se hodil pro tvorbu jednodušších aplikací, kde hlavním benefitem byla rychlost samotného vývoje. Vývojář napsal jeden zdrojový kód, který byl poté zkompilován zvlášť pro iOS a Android. [3] [4]

Primárním problémem byl především výkon a celková nemožnost využívat všech funkcí a nástrojů mobilního zařízení, což při vývoji komplexnějších aplikací na míru představovalo značnou překážku. Právě proto tedy většina vývojářů i společností raději sáhla k nativnímu řešení, které je zpravidla několikanásobně dražší na vývoj, ale umožňuje mít plnou kontrolu nad zdrojovým kódem při použití veškerého softwarového i hardwarového vybavení, kterým daná mobilní platforma disponuje.

Když v průběhu roku 2013 Facebook představil React Native, způsobil na poli vývoje webových a mobilních aplikací značný rozruch. Technologie si klade za cíl umožnit vývojářům tvorbu mobilních aplikací s využitím JavaScriptu a jeho nejpopulárnější knihovny React.js. Technologie vychází z obecných principů vývoje hybridních aplikací, nicméně výsledný kód aplikace by měl odpovídat kódu nativní aplikace. A právě tento cíl přiměl velkou skupinu vývojářů se o tuto technologii zajímat. Propojení s React.js a dalšími moderními webovými nástroji, které frontend developeři dnes již běžně znají a využívají, představuje zajímavou příležitost, jak se orientovat na úplně nový segment zákazníků. [5]

React Native se v roce 2018 stále nachází v rané fázi vývoje. I přes to je již komerčně využíván několika předními technologickými společnostmi jako je Instagram, Uber nebo Tesla. S rostoucím zájmem o technologii se konstantě rozšiřuje vývojářská komunita, což vede ke vzniku stále nových rozšíření, která umožňují posouvat hranice této technologie dále. Právě na spolupráci s komunitou je React Native do značné míry postaven. [3]

Pro účely této bakalářské práce byly definovány tyto cíle:

1. *Identifikace současných trendů na poli mobilních aplikací, rozdílnosti mezi mobilní a webovou platformou a představení nejpoužívanějších nástrojů pro vývoj mobilních aplikací.*
2. *Výběr konkrétního technologického řešení pro vývoj, odůvodnění jeho použití s ohledem na výkon, rozšířenost nebo zpracování dokumentace a to včetně představení možných alternativ.*
3. *Samotná implementace mobilní aplikace podle stanoveného zadání, která také zahrnuje volbu počáteční projektové struktury, popis funkčnosti jednotlivých sekcí aplikace a celkové shrnutí kladných či záporných stránek React Native.*

2. Představení projektu

Cílem této kapitoly je představit platformu Robeeto, nastínit principy jejího fungování včetně základních pilířů celé služby. Dále budou zanalyzována business problematika, která úzce souvisí se samotným přínosem mobilní aplikace této službě.

2.1 Společnost

Robeeto [6] je internetová platforma, která se snaží změnit způsob, jakým lidé využívají a poskytují služby na internetu. Ve své podstatě se jedná o online tržiště služeb. Za projektem, který vznikl v roce 2015, stojí Simona Kijonková – zakladatelka Zásilkovny, která od základu změnila český logistický trh. [7]

Služba nabízí zákazníkům možnost, najít schopného instalatéra, šikovnou kadeřnici, spolehlivou paní na hlídání nebo nadšeného lektora. V současné době na platformě existuje více než 650 služeb, které jsou rozřazeny do 7 základních kategorií:

- Domov a zahrada
- Domácí mazlíčci
- Fitness a výživa
- Vzdělávání a úprava textů
- Beauty a wellness
- Péče o děti
- Foto a video

Na Robeetu existují dva druhy uživatelů – zákazníci a pracanti. Zákazník je osoba, která na internetu poptává služby. Pracant naopak služby poskytuje a to skrze profily v různých kategoriích služeb. Robeeto však uživatele nesvazuje nutností zastávat pouze jednu z těchto rolí, tedy platí, že jeden uživatel může na Robeetu služby jak poptávat, tak i nabízet.

2.2 Modely fungování

Během více než 3 let vývoje se produkt jako takový a modely jeho fungování měnily a vyvíjely. V současné době na platformě fungují dva funkční modely.

2.2.1 Objednávkový systém

Objednávkový systém reprezentuje původní myšlenku, se kterou Robeeto v roce 2015 přišlo na trh. Pracant si jednoduše vyplnil profily v kategoriích, kde hodlal nabízet své služby. Zákazník si poté vyhledal všechny pracovníky, kteří danou službu nabízejí. Skrze profil si pak lze daného pracanta objednat, včetně místa, času a ceny za objednanou službu. Po uzavření objednávky dochází k propojení obou stran, kdy pracant může zákazníka požádat o poskytnutí dodatečných informací související s výkonem práce. Po úspěšném provedení objednávky má zákazník možnost pracanta ohodnotit. Právě obdržené hodnocení pak pracantovi může dopomoci k dalším zakázkám.

2.2.2 Poptávkový systém

V průběhu roku 2018 na Robeeto dorazil druhý způsob, jak objednávat služby online.
V tomto případě si zákazník sám zadá poptávku na konkrétní službu. Zákazník specifikuje, s čím potřebuje pomoci a jaká je očekávaná cena za poptávanou práci. Pracanti poté sami reagují na vystavené poptávky a prostřednictvím svých profilů nabízejí zákazníkovi zprostředkování služby. Zákazník si z nabídek pracantů vybere tu nejvhodnější. Proces vzniku objednávky a její následná realizace pak funguje shodně jako v případě předchozího modelu.

2.3 Základní pilíře služby

Za dobu 2 let, kdy je platforma Robeeto ve vývoji, se služba rozrostla v poměrně komplexní řešení, které disponuje širokým spektrem funkcí. V rámci mobilní aplikace je však velmi zbytečné a nežádoucí, disponovat všemi těmito funkcemi. Mobilní aplikace by měla představovat pohodlné řešení, kde uživatel jednoduše nalezne právě to, co potřebuje. Právě proto budou představeny pouze funkce, které se vývoje mobilní aplikace úzce dotýkají.

2.3.1 Profily

Profil patří mezi stěžejní body celé služby. Základní funkcí profilu je prezentace pracanta zákazníkovi. Právě skrze profil přichází zákazník do styku s pracantem.

Ve svém profilu pracant představuje sám sebe a své zájmy. Pro zákazníky je k dispozici také informace o dosaženém vzdělání, praxi v oboru a samozřejmě také seznam nabízených služeb. Ve vztahu ke službám pracant uvádí cenu, za kterou dané úkony poskytuje, popřípadě jestli je ochoten za danou prací dojet k zákazníkovi. Pro účely objednání pracanta zde také najdeme seznam předchozích hodnocení a volné termíny, kdy je pracant schopen službu vykonat.

Jsou to právě předchozí hodnocení a celková vyplněnost profilu, co rozhoduje o tom, které profily zákazník při hledání uvidí nejvýše. Robeeto se snaží pracanty motivovat k tomu, aby poskytli zákazníkům dostatečné množství informací. Na druhé straně zákazníkovi jsou prezentováni jen ty profily, u kterých lze očekávat, že dokáží jeho maximálně vyhovět požadavkům.

2.3.2 Listing

U každé služby, která nabízí více druhů zboží je nezbytnou funkcionalitou možnost listovat v tomto zboží. Na Robeetu tak místo produktů nacházejí pracanty. V rámci těchto katalogů je možnost filtrovat pracanty podle časové dostupnosti, požadované ceny nebo místa působnosti.

S příchodem poptávkového systému se však tento druh listování dostal do pozadí. Pro potřeby nového systému vznikly nové katalogy speciálně pro poptávky. Jedná se o poptávkové tabulky, které uživatelům poskytují přehled o aktuálních poptávkách v systému. U každé poptávky Zákazník má možnost sledovat stav svých zadaných poptávek. Naopak pracant může na vybrané poptávky reagovat, popřípadě sledovat stav těch, u kterých již v minulosti reagoval.

2.3.3 Notifikace

Pro úspěšné fungování platformy je nezbytné udržovat uživatele informované o relevantních změnách u poptávek, resp. objednávek. Pro tyto účely jsou integrovány tři úrovně notifikací pro zákazníky i pracanty.

E-mailová komunikace

Nejzákladnějším druhem komunikace mezi společností a uživateli je i nadále e-mailová komunikace. Zákazníci obdrží informační zprávu v případě vytvoření poptávky nebo objednávky. Po realizaci služby jsou vyzváni k udělení hodnocení a vyjádření celkové spokojenosti. Pracanti mají navíc možnost přihlásit se k odběru nových poptávek, kdy jsou mu poté zasílány upozornění o vystavění nových poptávek v kategoriích, kde projevil zájem vykonávat služby. I přesto, že se nemusí jednat o nejefektivnější způsob komunikace, představuje emailing způsob, jak oslovit drtivou část uživatelské základny.

Push notifikace

Jedná se krátké informační zprávy, které se uživatelům zobrazují na ploše počítače nebo chytrého telefonu. Vzhledem k tomu, že Robeeto funguje pouze jako webová služba, dochází v současné době pouze k využití na počítačích. Push notifikace představuje jednoduchý

internetový odkaz, který uživateli zobrazí příslušnou webovou stránku. Oproti e-mailu je obsah této zprávy značně zkrácen, aby jasně vystihoval obsah daného odkazu.

Nejčastěji bývají využívány pro upozornění uživatelů na změnu stavu u poptávek – například pokud někdo přidá u poptávky nabídku nebo komentář. Pro uživatele se jedná velmi nenáročný způsob, jak být okamžitě informován o nových událostech, aniž by se musel přihlašovat do mailového klienta. Po registraci na Robeetu je uživatel vyzván k povolení push notifikací v rámci právě používaného webového prohlížeče. Poté již dochází k zasílání upozornění, aniž by musel být prohlížeč vůbec otevřen. Pokud uživatel výzvu o povolení zasílání notifikací odmítne, lze toto rozhodnutí kdykoliv změnit v rámci nastavení daného prohlížeče.

Notifikace v rámci služby

V rámci samotné webové aplikace existuje samostatná záložka s notifikacemi. Jedná se o stejný obsah, který je zasílán v push notifikacích, avšak oproti předchozí variantě, u kde upozornění po zobrazení zmizí, zde jsou upozornění archivovány, aby se k nim uživatelé mohli v případě potřeby kdykoliv vrátit.

2.4 Business problematika

Základním kamenem úrazu na trhu poskytování služeb online je stále jistá nevyzrálost samotných uživatelů. Zatímco většina uživatelů na internetu hojně využívá možnost nakupovat zboží online, u služeb stále panuje jistá skepse. Konkurenční služby se v drtivé většině případů zaměřují pouze na specifickou kategorii služeb, jako například hlídaní dětí, řemeslnické práce nebo doučování. Naproti tomu Robeeto se snaží zákazníkům nabídnout komplexní řešení širokého spektra kategorií.

To s sebou samozřejmě přináší řadu problémů. Z hlediska marketingu není možné cílit na jednu skupinu zákazníků, nýbrž je potřeba oslovovat několik skupin naráz, což představuje zdlouhavý a finančně náročnější úkol. Samotná komunikace se zákazníkem v rámci aplikace je pak také velmi náročná, zejména pokud zákazník poptává několik kategorií najednou, nebo opačně, pokud pracant poskytuje své služby napříč více kategoriemi. Pro účely SEO (Search engine optimalization) je pro každou kategorii sepsána specifická textace, jak u listování pracantů, tak i v rámci vlastních landing page. I přes to, že služba jako taková chce poskytovat obecné řešení, pro potřeby určité kategorie služeb je potřeba komunikaci s uživateli odlišit.

Na základě nasbíraných dat vyšlo najevo několik trendů v chování zákazníků. Pozitivním zjištěním je, že zákazníci v drtivém množství případů zavítají na příslušnou stránku, vzhledem k jejich poptávané službě. Původní princip přímého objednání pracanta je v aktuální verzi spíše upozaděn a přednost je věnována vytvoření poptávky ze strany zákazníka. I zde neexistuje závažnější problém s vytvořením konkrétní poptávky. Počet měsíčně zadaných poptávek na platformě je přímo úměrný prostředkům investovaným do internetové reklamy nebo cold callingu.

První závažný problém tak nastává u pracantů. Je evidován častý jev, kdy u zadaných poptávek zkrátka neexistuje žádná, nebo nedostatečný počet nabídek. Problém je způsobem zejména malou aktivitou nebo ochotou ze strany pracantů. V současné situaci na trhu je mnohdy velmi složité pro konkrétní kategorii služeb zajistit kvalitní pracovní sílu a to jak z důvodů malého množství expertů v odvětví, nebo kvůli jejich časovému vytížení, kdy například řemeslníci nebo domácí kutilové nemají o zakázky nouzi. V těchto případech většinou daní jednotlivci mají svůj pracovní harmonogram plný na několik měsíců dopředu a nejsou ochotní proto uspokojovat další zákazníky na Robeetu.

Druhým zásadním problémem je často neschopnost pracantů řádně komunikovat se zákazníkem. V nejednom případě si pracant neví rady, jak reagovat na konkrétní pracovní příležitost, nebo jaké informace je vhodné zákazníkovi poskytnout. Z druhé strany také často nedochází k poskytnutí potřebných informací ohledně vystavěných zakázek ze strany klientů. Pracant tak zkrátka na potencionálně zajímavou poptávku nereaguje, protože mu chybí specifické informace spojené s výkonem práce.

V neposlední řadě představuje problém snaha některých jedinců obejít systém Robeeta. Platforma v současné verzi nepožaduje po pracantech procentuální provizi za zprostředkování zakázky, i přes to, že v celém procesu zaujímá roli prostředníka a zprostředkovatele mezi poptávajícím a nabízejícím. Nicméně i tak se nezanedbatelné procento uživatelů snaží s druhou stranou domluvit mimo platformu. To představuje nepříjemnost v situaci, kdy se obě strany dohodnou na poskytnutí služby, ale vystavená poptávka zůstává i nadále otevřená, navzdory tomu, že zákazník již o zprostředkování služby nestojí. Samotný pracant pak samozřejmě nemá nárok na ohodnocení, což na kvalitě pracovníků nepřidá. Snahou Robeeta tedy je, aby se zákazníci a pracanti naučili, jak spolu komunikovat a následně uzavírat objednávky a aby měli motivaci ochotu se do aplikace vracet a nadále ji využívat.

2.5 Přínos mobilní aplikace

Mobilní aplikace představuje logický krok v evoluci původně webové aplikace. Webové prostředí je dnes prostor, ve kterém se pohybuje drtivá část populace. Díky oblibě mobilních telefonů však webové rozhraní nemusí být dostatečné. Mobilní aplikace umožnují jejím uživatelům využívat specifické funkce a vlastnosti, které v rámci webových prohlížečů zkrátka nejsou možné.

Primárním cílem mobilní aplikace je zejména poskytnout uživatelům Robeeta takový komfort, který je přiměje službu řádně a aktivně využívat. Z hlediska uživatelské základny bude mobilní aplikace zaměřená pouze na pracanty, jelikož se jedná o více problematickou skupinu uživatelů. Snahou je poskytnout pracantům takové rozhraní, které jim pomůže najít pracovní příležitost daleko rychleji a jasněji než skrze webovou verzi. Samotná aplikace by měla působit velmi minimalisticky a poskytovat pracantovi pouze relevantní informace. Veškerá komunikace by měla být daleko jednodušší a oproštěná od redundantních údajů. Díky přímému cílení na pouze jednu část uživatelského spektra může být samotná komunikace daleko efektivnější.

Díky integraci nativních notifikací se poté otevírá příležitost uživatele kontaktovat ještě rychleji. U webových notifikací jsou uživatelé vázáni na to, mít s sebou svůj počítač. U mobilního zařízení tato bariéra prakticky odpadá, jelikož většina majitelů mobilních zařízení je má po většinu dne neustále po ruce. Samotná interakce uživatele s mobilní aplikací je také prokazatelně daleko příjemnější, než práce na desktopu.

3. Rozdílnost platforem

Tato kapitola se zaměřuje na rozdílnosti mezi mobilní a webovou platformou a také identifikaci současných trendů a problémů na poli mobilních aplikací.

3.1 Prostředí

Vývoj webové aplikace je úzce spjat s využitím zdrojů, které aplikaci poskytuje webový prohlížeč. Proces optimalizace se tak zaměřuje na to, aby stránka nebo aplikace fungovala především v těch nejpoužívanějších prohlížečích na trhu. Mezi ty dnes patří především Google Chrome, Mozilla Firefox, Internet Explorer, Edge nebo Safari. Napříč verzemi těchto prohlížečů může docházet k rozdílné podpoře některých funkcí zejména v kontextu CSS nebo JavaScriptu.

Na místo prohlížeče je vývoj mobilní aplikace vázán na druh a verzi operačního systému. Je třeba brát v potaz, zda verze operačního systému danou funkcionalitu podporuje. Toto je problém zejména u zařízení s Androidem, jelikož tento systém oproti iOS trpí velkou roztříštěností z hlediska používaných verzí operačního systému. Zejména starší mobilní zařízení pak často pohání několik let stará verze Androidu. Při vývoji je třeba některé funkce pro tyto verze opravovat nebo dokonce je kompletně vypouštět.

Při vývoji hybridní mobilní aplikace nebo s použitím webových technologií obecně, jako v případě React Native, dochází k nedostupnosti některých funkcí, které fungují ve webovém prostředí, nikoliv však v mobilním. Prvním příkladem je například možnost provést refresh dané stránky, například za účelem aktualizace dat z API (Application Programming Interface). Vhodnou a masivně rozšířenou metodou pro stejný účel je v mobilním světě pull to refresh. V neposlední řadě je třeba upozornit, že při práci s JavaScriptem nelze využívat webové objekty document, window nebo LocalStorage - možnost ukládat data z webu do počítače uživatele.

3.2 UI/UX

Mobilní aplikace dnes není záležitost pouze dobrého nápadu a později i správné implementace funkcionality. Nedílnou součástí je bezesporu také vizuální zpracování. Právě implementace vhodného a uživatelsky příjemného uživatelského rozhraní (User Interface) společně s celkovým uživatelským prožitkem (User Experience) je základním stavebním kamenem úspěšné aplikace.

Mobilní rozhraní má oproti tomu webovému svá specifika. Oproti desktopu má uživatel k dispozici menší pracovní plochu, což by se mělo při tvorbě UI projevit přeskládáním

jednotlivých prvků aplikace, změnou práce s prostorem na obrazovce nebo přizpůsobením velikosti písma. V závislosti na platformě se mohou lišit i některé specifické možnosti interakce uživatele, kdy na dotykových zařízeních například nelze implementovat tzv. hover state – stav, kdy uživatel přes daný prvek najede kurzorem myši. Na druhou stranu základní možnosti interakce, například click nebo scroll, zůstávají napříč platformami téměř stejné.

Jak Apple, tak i Google při vývoji svých operačních systémů dospěly k názoru, že aplikace na jejich platformě je třeba vizuálně normalizovat a proto pro vývojáře připravily velmi obsáhlé dokumentace, které představují základní vizuální principy daného ekosystému. I přesto, že vývojáři nejsou striktně nuceni tyto principy následovat, je třeba brát v potaz, že koneční uživatelé aplikací si tyto návyky používáním naprosto základních funkcí jejich mobilních zařízení osvojili, a proto jejich dodržování napomáhá samotnému uživatelskému prožitku.

3.2.1 Material Design

Material Design od Google je souhrnný návod pro tvorbu uživatelského rozhraní pro Android. Je třeba zmínit, že v dnešní době se Googlu již podařilo tuto metodiku rozšířit i do jeho dalších produktů a to nejenom na mobilní platformě. Principy Material Design jsme schopni pozorovat i na webových verzích služeb jako je Gmail nebo YouTube. [8]

Mezi základní charakteristiky patří jednotné a zjednodušené rozhraní využívající tzv. hloubku designu. Často dochází k využití jasných barev a tučnosti, inspiraci zde Google čerpá z papíru a inkoustu. Material Design ve své podstatě tlačí na jednoduchost a minimalismus.

3.2.2 Flat Design

Apple s příchodem iOS 7 v roce 2013 představil nový vizuální směr, kterým se v rámci svého softwarového řešení vydal. Mezi grafickými odborníky tento směr získal označení Flat design a v současné době se jedná o zřejmě nejpoužívanější metodiku při tvorbě UI pro webové i mobilní aplikace. Principy Apple důkladně sepsal v publikaci Human Interface Guidlines. [9]

Flat design by mohl být považován za opak skeumorfismu (objekt, který imituje funkční prvky originálu). Podobně jako v případě Material Designu i zde dochází k nahrazení reálných obrazů dvourozměrnými vizuálními detaily. Oproti řešení od Googlu však nepoužívá stíny a jakýkoliv druh rozmazání. Zakládá si na vysoké čitelnosti s jasnou typografií. Ve srovnání s Material Designem také často využívá daleko méně kontrastní barvy nebo grafické prvky.

4. Výběr technologie

Před samotným napsáním zdrojového kódu aplikace je třeba si jasně vymezit, jaké technologie budou pro účely daného projektu použity. Jednotlivé knihovny lze rozdělit do dvou skupin. Primárními knihovnami se rozumí takové technologie, které jsou nezbytně nutné pro vývoj mobilní aplikace za pomoci React Native. Podpůrné knihovny naproti tomu reprezentují knihovny nebo balíčky, které rozšiřují možnosti práce s primárními knihovnami, nebo zkrátka napomáhají zlepšovat kvalitu vyvíjené aplikace nebo služby.

4.1 Primární knihovny

4.1.1 React

React je masivně rozšířená JavaScript knihovna pro budování uživatelského rozhraní v rámci webových aplikací. V roce 2018 se stal nejpopulárnějším frameworkem pro JavaScript. React se jako knihovna neustále technologicky vyvíjí a inženýři ve Facebooku tak velmi frekventovaně zásobují komunitu novými verzemi. Právě na komunitě je tato knihovna do značné míry postavena. V souvislosti s již zmíněnou popularitou se React těší velkému množství rozšiřujících balíčků, které umožňují vytvářet velmi komplexní aplikace. Veškeré principy a postupy jsou velmi pečlivě sepsány v rámci oficiální dokumentace Thinking in React. Pro pochopení základního paradigmatu je potřeba chápat tyto pojmy a principy:

JSX

JSX je velmi užitečné rozšíření syntaktických pravidel JavaScriptu, které v rámci tohoto jazyku umožňuje psát konstrukce založené na značkovacím jazyce HTML. I přesto, že pro práci s Reactem není striktně vyžadován, JSX se již dnes stal standardem, který je využíván téměř bez výjimky. [1]

Komponenta

Funkcí komponenty je rozdělit uživatelské rozhraní na nezávislé a znovupoužitelné části. Jedná se o základní stavební prvky při práci se Reactem. Z technologického hlediska může být komponenta definována jako:

- Funkce (stateless component)
- Třída (stateful component)

Jakožto součást UI musí mít každá komponenta návratovou hodnotu. Tou může být prostý text, JSX struktura, jiná komponenta nebo hodnota null.

Props

Každá komponenta může být rozšířena vlastními atributy, které se nazývají properties, zkráceně props. Tyto rozšiřující vlastnosti mají, z hlediska datového typu, tvar objektu. Základním pravidlem je, že v rámci komponenty nelze měnit jejich hodnotu. Komponenta je může využít například jako určující pravidlo pro vykreslení nebo je předat komponentám na nižší úrovni.

State

State lze chápat jako vnitřní a privátní stav dané komponenty. Stejně jako v případě props bývá definován jako objekt. Základním rozdílem je, že state lze v rámci komponenty měnit. Je třeba brát v úvahu, že state může být definován pouze v rámci třídy, u prosté funkce to možné není.

Lifecycle

Jedná se o sadu metod, které jsou automaticky vyvolány změnami v komponentě. Skrze ně lze definovat chování, které nastane například při prvotním zobrazení komponenty nebo při změně state či props.

4.1.2 React Native

Základním předpokladem použití React Native je alespoň základní znalost Reactu a jeho principů. React Native představuje řešení, které přenáší stěžejní benefity původní knihovny do prostředí mobilních aplikací. Pro návrh uživatelského rozhraní jsou využívány nativní komponenty, které nahrazují původní webové bloky. Veškeré HTML značky jsou tak nahrazeny novými, speciálně navrženými značkami pro mobilní zařízení. [5]

Každá nativní komponenta disponuje seznamem dostupných atributů nebo metod, které lze nastavit a použít. Od toho se pak odvíjí například schopnost komponenty reagovat na události nebo spravovat obsah. S každou další verzí se seznam dostupných komponent rozšiřuje, nebo se rozšiřují možnosti jeho použití. Mezi ty elementární a nejčastěji používané lze zařadit:

Text

Text je základní a zároveň jediná komponenta pro práci s textem. Tam, kde se ve webovém prostředí používaly značky jako h1, h2, h3 nebo p, bude použita tato komponenta. Pro účely rozlišení jednotlivých textových komponent je nutné použít stylování.

View

View představuje nejuniverzálnější stavební blok. Jedná se o jakýsi ekvivalent HTML značky div. Nejčastěji bývá využíváno jako kontejner s podporou stylování nebo detekce událostí. Při

použití je však potřeba brát v úvahu omezení, které této komponentě neumožňuje provádět scrolling jejího obsahu.

ScrollView

Výše zmíněný problém lze jednoduše vyřešit použitím komponenty ScrollView. Kromě toho tato komponenta disponuje dalšími specifickými funkcemi pro kontrolu ovládání a událostí spojených se scrollingem.

FlatList

Pro účely výpisu seznamů nebo položek stejného druhu lze využít komponentu FlatList. Jedná se o užitečnou komponentu, která disponuje velmi podobnými funkcemi jako ScrollView. Kromě trochu jiného zápisu je jejím hlavním rozdílem schopnost efektivně vykreslovat pouze ty položky, které jsou v daný okamžik viditelné na obrazovce. Jedná se o obdobu webové technologie lazy loadingu.

StyleSheet

Velkým specifikem React Native je naprostá absence stylování pomocí kaskádových stylů. Tradiční technologie CSS je nahrazena komponentou StyleSheet, která umožnuje definovat jednotlivé stylizační třídy. Z hlediska syntaxe se jedná o téměř totožný zápis, jako v případě kaskádových stylů. Vše ale probíhá na úrovni JavaScriptu.

4.1.3 Expo

Expo představuje komplexní nástroj, postavený speciálně pro React Native, který umožňuje v rámci aplikace spravovat proces vývoje a distribuce na jednotlivé mobilní platformy. Přestože za tímto projektem nestojí vývojový tým React Native, je tento nástroj oficiálně doporučován.

V prvé řadě Expo umožnuje založení projektu s předdefinovanou strukturou a přítomností základních knihoven. To velmi urychluje proces samotného vývoje. Bez ohledu na fakt, že implementace jádra projektu je ve výchozím nastavení skryta, lze daný projekt rozšiřovat o další knihovny dle specifických požadavků. Důležité je ovšem brát zřetel na aktuálně používané verze hlavních knihoven. Expo velmi striktně definuje s jakými verzemi React Native, potažmo Reactu, pracuje a některé rozšiřující knihovny tak nemusí být nutně kompatibilní s tímto nastavením.

Pro účely vývoje mobilní aplikace poskytuje Expo řadu užitečných a nezbytných funkcí. Vývojový proces lze řídit pomocí příkazové řádky nebo v rámci jednoduché lokální webové aplikace. Vývoj mobilní aplikace je postaven zejména na sledování prováděných změn a úprav

v reálném čase. V rámci jednoho běžícího procesu Expa jsou vývojáři schopni testovat aplikace na více zařízeních najednou. Tuto funkcionalitu zajišťuje Expo pomocí generování unikátních odkazů nebo QR kódu.

Testovat aplikace lze buď pomocí emulátoru, nebo přímo na fyzickém zařízení. Emulátor je jednoduchý program, nainstalovaný na vývojovém stroji, který simuluje prostředí daného mobilního operačního systému. Emulátor umožňuje velmi rychlé testování, které nicméně není natolik důkladné a komplexní jako při použití fyzického zařízení. Pro simulaci iOS je třeba vývojové prostředí XCode od společnosti Apple, u Androidu je zapotřebí Android Studio. U fyzického zařízení stačí mít nainstalovanou oficiální aplikaci Expa a za pomoci zmiňovaných odkazů dojde k instalaci vyvíjené aplikace. Služba navíc disponuje dalšími užitečnými funkcemi vhodnými při vývoji, jako je třeba výpis stavových chyb nebo hot reloading. [10]

Po dokončení vývoje přichází na řadu vydání aplikace. Expo v této chvíli představuje roli poskytovatele, kdy zdrojový kód je zkompilován a následně uložen na serverech Expa. Odtud je poté možné aplikaci distribuovat na jednotlivé mobilní platformy. Pro tyto účely je nutné aplikaci propojovat s vývojářskými účty, které jsou nezbytné pro publikování obsahu na iOS, respektive Android. Veškeré informace související s distribucí aplikace jsou nastavovány v rámci Expa pro každou platformu zvlášť.

Kromě výše uvedeného nabízí Expo ještě celou řadu podpůrných modulů a balíků. Skrze tuto službu lze například spravovat push notifikace, oprávnění nebo využívat další nativní aplikace jako fotoaparát nebo přístup k fotografiím či souborům.

4.2 Podpůrné knihovny

Volba podpůrných knihoven je do jisté míry volbou osobních preferencí. Níže sepsaný seznam reprezentuje soubor technologických řešení, který byl použit na vývoj mobilní aplikace Robeeta, s ohledem na převážně pozitivní zkušenosti s jejich použitím ve webovém prostředí. Díky tomu je zajištěna relativně velká přenositelnost zdrojového kódu napříč webovou a mobilní aplikací. Každé vybrané technologické řešení je popsáno na úrovni své funkcionality, účelu použití a jsou představeny také alternativní řešení a rozdílnost mezi nimi.

Prop Types

Jedná se o velmi jednoduchou knihovnu, která zajištuje kontrolu datových typů props v rámci React komponent. Jelikož JavaScript ve svém základu nedisponuje dostatečnými nástroji pro

validaci datových typů, Prop Types představují ideální nástroj. Jejich použitím se zvyšuje pravděpodobnost odhalení celé řady chyb, čímž se zvyšuje celková stabilita aplikace. [1]

Pro komplexní aplikace a kontrolu nejenom v rámci props je ideální využít například velmi populární TypeScript nebo Flow od tvůrců Reactu. Tyto nástroje umožnují pokrýt téměř každý vstupní bod v rámci aplikace, což Prop Types neumožnují. Složitost a celková doba správné implementace těchto knihoven je však znatelně vyšší. [11] [12]

React Navigation

Každá mobilní aplikace vyžaduje přítomnost routeru. React Navigation umožňuje využívat podobné rozhraní pro směrování mezi jednotlivými sekcemi aplikace, jako u webových aplikací, které jsou založené na Reactu. Knihovna také nabízí možnost definovat několik druhů navigací napříč celou aplikací. Lze si vybrat například mezi tzv. Stack nebo Drawer navigací. U každé stránky pak lze na míru upravovat chování navigačních prvků. Navigační prvky mají navíc design přizpůsobený jednotlivým platformám, je tedy zajištěn autentický vzhled zvlášť pro Android a iOS zařízení. [13]

Přestože samotná implementace této knihovny není u komplexnějších aplikací zcela jednoduchá, stále se jedná o nejpoužívanější knihovnu tohoto druhu pro React Native. Alternativou pak může být knihovna React Native Navigation, která však nenabízí zdaleka tolik možností. [14]

Redux

Redux je robustní knihovna, která zprostředkovává v rámci webové i mobilní aplikace práci s daty. I přesto, že původně byl Redux vyvinut jako rozšiřující knihovna k Reactu, lze ji použít i s využitím jiného frameworku. Ve svém základním konceptu existuje v aplikaci centrální zdroj dat. Ten může být rozdělen na dílčí části, tzv. reducery – například dle jednotlivých stránek nebo logických sekcí. Změnu dat lze docílit vyvoláním unikátních akcí, kdy každý reducer přesně ví, jak na danou akci reagovat. Redux poté jednotlivým komponentám distribuuje příslušná data ve formě props. Obecná logika práce s daty se tak přesouvá z komponent do jediného nadřazenějšího prvku. [15]

Redux je vhodné využít u aplikací s pokročilou složitostí a strukturou. Základní koncept vychází k původní knihovny Flux od Facebooku, která je však v praxi výrazně upozaděna daleko pozitivněji přijatým Reduxem. [16]

Reselect

V souvislosti s Reduxem je velmi přínosné využít funkcí balíčku Reselect. Ten skrze tzv. selektory umožnuje v rámci Redux procházet a následně seskupovat data, která jsou poté předávána do komponent. Díky tomu se tak v rámci komponent snižuje potřeba dopočítávat dodatečné údaje. Selektory jsou také výpočetně velmi efektivní – k přepočítání výpočtu dochází jen a tehdy, pokud došlo ke změně vstupujících hodnot. Jednotlivé dotazy lze také navzájem skládat a tvořit tak daleko komplexnější celky. [17]

Redux Form

Nezbytnou součástí aplikace je mimo jiné zpracování formulářových dat. K těmto účelům existuje celá řada knihoven. Pro webovou i mobilní aplikaci Robeeta byla zvolena knihovna Redux Form. Ta umožňuje sestavovat, upravovat a následně validovat data z formulářů. Vše je navíc napojeno do Reduxu, kdy samotný formulář a veškerá sekundární data k němu jsou reprezentována jako samostatný reducer. Díky použití jedné knihovny na obou platformách je tak implementace daleko rychlejší a kód je znovupoužitelný. [18]

Vhodnou alternativou se může být velmi populární knihovna Formik. Nabízí daleko jednodušší a kratší zápis, nicméně nedisponuje přímou integrací do Reduxu, což byl v případě této aplikace rozhodující faktor. [19]

Ramda

Při složitější struktuře dat je velmi vhodné využít utilizační nástroj Ramda – praktickou funkcionální knihovnu pro práci s objekty a poli v JavaScriptu. Nejčastějším problémem u aplikací s několika datovými úrovněmi bývá především mutace dat. Ramda poskytuje celou řadu intuitivních funkcí, které těmto problémům předchází. [20]

Facebook pro řešení těchto problémů dokonce přišel s vlastním řešením, které nese název Immutable. Tato knihovna nahrazuje tradiční pole a objekty řadou vlastních datových struktur se striktně definovanými pravidly. U webové aplikace Robeeto došlo k použití právě této knihovny. U aplikace mobilní však došlo k migraci na Ramdu, která nabízí daleko komplexnější a především výpočetně optimálnější řešení. [21]

Moment

Moment představuje jednu z obecně nejdéle používaných knihoven pro JavaScript. Využívá se pro správu datumových a časových hodnot. Poskytuje širokou nabídku funkcí pro manipulaci s časem, mnohajazyčné lokalizace časových údajů nebo převod do příslušných časových

pásem. I přes všechny zmíněné přínosy se jedná o knihovnu, která v posledních letech čelí masivnímu odlivu příznivců. Moment se potýká s celou řadou chyb a nedodělků. Největším neduhem je ale celková velikost samotné knihovny, která bez možnosti volby integruje všechny svoje dodatečné balíky. [22]

Ve světle těchto nedostatků tak v poslední době vznikají nové projekty s cílem nahradit tento kolos moderní a minimalistickou knihovnou. Jedním z takových projektů je modulární knihovna date-fns, která odstraňuje většinu problémů Momentu s použitím jiného přístupu a syntaxe. V rámci vývoje této aplikace však i nadále bude využit Moment a to zejména díky přetrvávající a perfektně zpracované jazykové lokalizaci, kterou žádný konkurent nenabízí. Do budoucna však lze předpokládat, že Moment bude vytlačen rychle rostoucími konkurenty. [23]

Axios

Axios je jednoduchý HTTP klient původně navržený pro webové prostředí. S příchodem dynamických aplikací, které jsou závislé na asynchronních přenosech dat, vznikla v rámci JavaScriptu konstrukce zvaná Promise, na které je tato knihovna postavena. Axios také automaticky transformuje příchozí i odchozí data do formátu JSON. [24]

Velmi zajímavá a užitečná je také možnost implementovat do projektu speciální middleware, skrze který Axios jednotlivé asynchronní požadavky integruje jako akce do Reduxu. Právě tyto drobné benefity však z Axiosu dělají velmi dobrý nástroj pro správu síťových požadavků jak na webu, tak i na mobilních zařízeních.

5. Implementace

Stěžejním bodem celého projektu je bezpochyby proces implementace dle stanoveného zadání za pomoci vybraných nástrojů. Tato kapitola se zaměřuje na koncepci projektové struktury a také prezentuje aplikační schéma, technologickou specifikaci jednotlivých komponent a obrazovek mobilní aplikace, včetně vydání do produkce.

5.1 Projektová struktura

Ideálně navržená struktura projektu zlepšuje přehlednost a zvyšuje efektivnost programátora. Při zvolení správného vzoru projektové struktury dochází k rozdělení aplikace na logické celky, které umožňují věnovat maximální úsilí samotnému zdrojovému kódu. Tento faktor hraje důležitou roli zejména s růstem objemu a náročnosti vyvíjené aplikace.

V praxi jsou aktuálně nejvíce zavedené dva přístupy. Function-first je přístup postavený na organizaci projektu na základě funkcí jednotlivých částí. To znamená, že jednotlivými částmi mohou být sekce jako actions, containers, reducers nebo screens. Feature-first naproti tomu preferuje členění dle jednotlivých sekcí aplikace nebo služby, jako například cart, listing, product nebo settings. [25]

Problém obou přístupů vyvstává při růstu projektu, kdy jednotlivé sekce nabobtnávají a orientace ve zdrojovém kódu se opět ztěžuje. V rámci této aplikace byl proto vybrán přístup, který kombinuje oba tyto přístupy dohromady, kdy top-level struktura projektu je koncipována podle function-first přístupu a na nižších úrovních je aplikována metodika feature-first. Konečné schéma projektové struktury pak má tuto podobu:

Api

Tato sekce v sobě obsahuje veškerá nezbytná nastavení k API. To v sobě zahrnuje konfiguraci síťových požadavků pomocí knihovny Axios nebo napojení na přihlašovací služby třetích stran – Facebook a Google.

Components

Veškeré dílčí komponenty aplikace náleží do této složky. Základním předpokladem pro umístění do této sekce je obecnost a znovupoužitelnost, která je nezávislá na místě využití.

Constants

Velmi důležitou, avšak často přehlíženou, oblastí je existence konstant. Standardem na poli vývoje v rámci JavaScriptu se stalo, že veškeré primitivní textové nebo číselné proměnné by

měly být implementovány jako konstanty. Díky tomu je vývojář schopný z jednoho místa aplikovat změny do všech částí aplikace, kde je daná hodnota použita. Dalším benefitem pak je eliminace opakovaného psaní stejných informací.

Ducks

Ducks je návrhový vzor pro škálovatelnou práci s daty v rámci Redux. Tento princip je založený na feature-first rozdělení aplikačních dat. V rámci každé dílčí sekce je třeba definovat tyto soubory:

- actions – jednotlivé akce volané v rámci komponent
- reducer – logika zpracování jednotlivých akcí
- selectors – agregace nebo očišťování dat pomocí knihovny Reselect
- types – jednoznačné identifikátory akcí

Aplikace těchto jednoduchých pravidel vedla ke zjednodušení a zpřehlednění práce s daty na úrovni Reduxu. Hlavním podkladem pro implementaci je článek Scaling your Redux App with ducks, který velmi poutavým způsobem prezentuje základní myšlenky a motivaci pro použití této metodiky. [25]

Styles

Již v předchozí kapitole bylo zmíněno, že při práci s React Native je tradiční práce s kaskádovými styly nahrazena vlastním modulem, který se nazývá StyleSheet. To nicméně nebrání členění stylování na příslušné části, které mají podobné charakteristiky jako komponenty.

Utils

Utils je velmi obecně pojatá sekce, která slouží zejména jako podpůrný nástroj pro práci s komponentami. Standardně se jedná o funkce s širokým spektrem použití. Utilizační funkce lze standardně využít na práci s textovými řetězci a číselnými hodnotami, například pro formátování jednotkových a časových hodnot nebo práci s desetinnými čísli. Velmi užitečné je také na této úrovni definovat funkce pro validaci formulářových dat a to včetně příslušných jednotkových testů.

5.2 Pravidla a společné principy

Pro zachování konzistence a maximalizaci uživatelské spokojenosti při používání je důležité, aby byly v rámci aplikace definovány a dodržovány jednotná pravidla chování a práce s obsahem. V rámci tohoto projektu byly vytyčeny tyto společné principy:

Seznamy

Výpis prvků stejného typu je realizován pomocí nekonečného seznamu. React Native pro tyto účely disponuje komponentou FlatList, která umí pracovat s tímto typem dat velmi efektně i efektivně. V případě práce s velkým množstvím prvků je reálně zobrazeno pouze několik málo prvků a pouze při pohybu uživatele k poslednímu prvku dojde k načtení nových položek. Tato metoda nese název infinite scrolling. Kromě toho implementuje komponenta FlatList také nativní gesto pull to refresh, které slouží pro aktualizaci stávajících prvků. Nekonečné seznamy nahrazují dnes již relativně zastaralý přístup stránkování obsahu.

Indikátory aktivity

V návaznosti na vykreslování seznamů je z uživatelského hlediska velmi důležitá přítomnost grafických indikátorů, které odlišují nové a dosud nezobrazené prvky od zbytku seznamu. Nezobrazené položky proto disponují sytějšími barvami a jsou doprovázeny červenou tečkou, která se pro tyto účely stala velmi rozšířenou metodou.

Formuláře

Zpracování uživatelského vstupu skrze formulář je také velmi důležitou součástí této aplikace. Na základě nastaveného trendu na poli mobilních aplikací bylo rozhodnuto, že každý formulář bude vyčleněn do samostatné obrazovky aplikace. Díky tomu dojde k jasnému oddělení aplikační logiky na ty části, které uživatelský vstup vyžadují a nevyžadují. Samotné formuláře pak také působí daleko přehledněji a umožnují uživateli soustředit se pouze na tuto úlohu.

Odkazy a zdroje akcí

Jednotlivé stránky aplikace jsou navzájem propojeny a mohou disponovat prvky, které vyvolávají určitý druh reakce. Nejznámějším příkladem jsou tlačítka nebo textové odkazy. Je nezbytně důležité, aby uživatel aplikace dokázal bezproblémově určit, zda daný prvek nějakou akci spouští. Právě špatné pochopení uživatelského rozhraní a jeho jednotlivých prvků je častým zdrojem problémů při používání mobilních aplikací. Pro zamezení těchto problémů, tlačítka a textové odkazy disponují dostatečně výraznými barvami a jasně čitelných textem. Z hlediska jejich umístění na obrazovce musí být zajištěno jejich dostatečné oddělení od

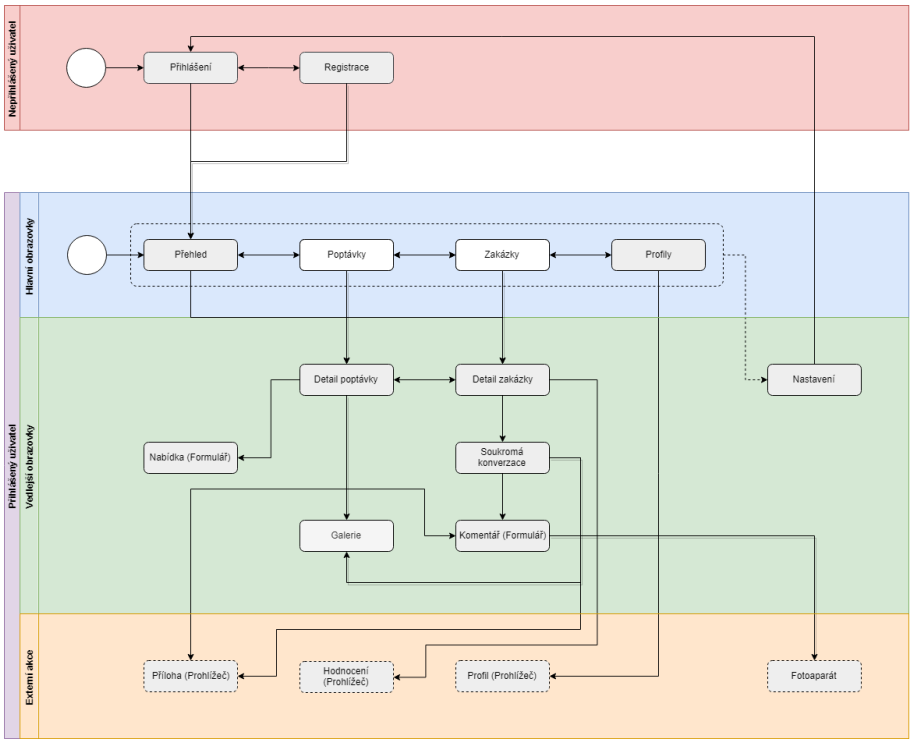
ostatního obsahu. Zavedeným trendem je také poskytnutí zpětné vazby uživateli v okamžik interakce a to například změnou pozadí daného prvku.

Zpracování chyb

Každá interakce uživatele s aplikací může z programátorského hlediska skončit úspěchem nebo neúspěchem. Z hlediska síťových požadavků, které probíhají na pozadí a pro uživatele jsou neviditelné, je úspěch následován zobrazením přijatých dat nebo například přesměrováním na jinou obrazovku. Klíčové je ovšem uživatele informovat také o případném neúspěšném zpracování jeho požadavku. V této aplikaci je tak stěžejní většina neúspěšných uživatelských požadavků doprovázena zobrazením nativní komponenty Alert, která disponuje nadpisem a detailním popisem chyby s případným řešením problému.

5.3 Aplikační schéma

Níže uvedený obrázek aplikačního schématu představuje průchod uživatele napříč vyvíjenou aplikací. Jednotlivé obrazovky aplikace jsou rozděleny na dvě dílčí skupiny. První skupina se týká obrazovek nepřihlášeného uživatele, jedná se tedy o přihlašovací a registrační obrazovky. Skupina obrazovek přihlášeného uživatele je pak dále členěna na hlavní a vedlejší obrazovky. Speciální podskupinou jsou pak externí akce, které na základě události uvnitř aplikace uživatele přesměrovávají do jiných aplikací nebo služeb.

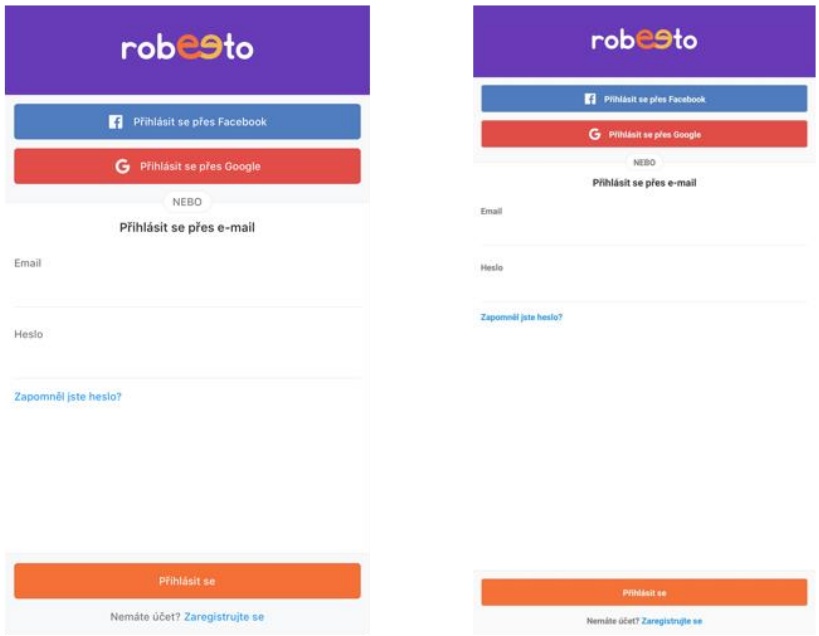


Obrázek 1: Aplikační schéma, zdroj: (autor 2019)

5.4 Přihlášení a registrace

První obrazovkou, která se zobrazí uživateli po instalaci aplikace, je přihlašovací resp. registrační sekce. Tato obrazovka umožňuje kromě klasického přihlášení pomocí emailu a hesla také přihlášení pomocí sociálních sítí Facebook a Google. Značná část této funkce je přímo implementována v rámci API Expa, kdy dochází k otevření nativního vyskakovacího okna přímo dané společnosti a programátorovi je pouze vrácen stav vyřízení požadavku přihlášení a to včetně požadovaných informací o uživateli.

Pro již dříve registrované uživatele zde také existuje možnost zaslání zapomenutého hesla. Uživatel poté obdrží informační email s dalšími pokyny. Jakmile dojde k úspěšnému ověření uživatele, je do paměti aplikace tato informace uložena. Při příštím otevření aplikace, tak uživatel nemusí znovu zadávat přihlašovací údaje. Práce s úložištěm aplikace je realizována pomocí nástroje nazvaném AsyncStorage, který je poskytován přímo v rámci React Native. Jedná se o ekvivalent LocalStorage z prostředí webových prohlížečů.



Obrázek 2: Přihlášení, zdroj: (autor 2019)

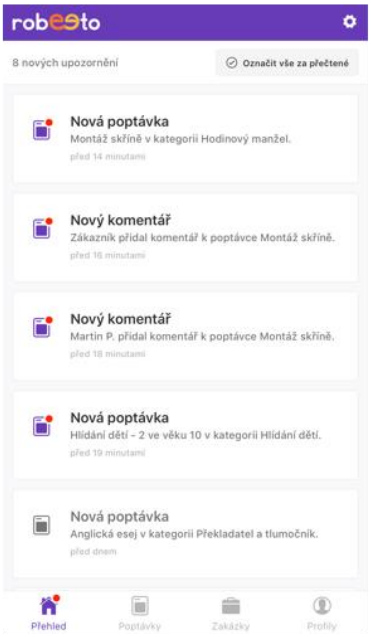
5.5 Hlavní obrazovky

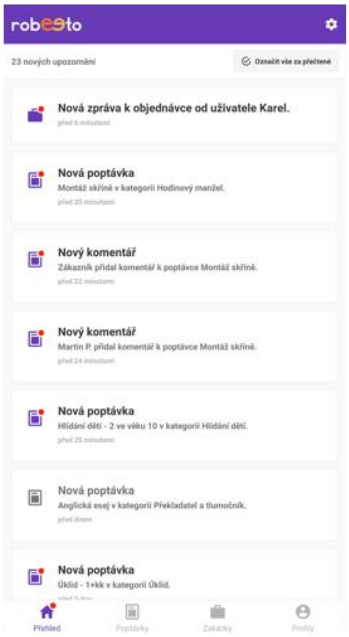
Jádro aplikace je tvořeno čtyřmi základními obrazovkami. Dohromady pak tyto obrazovky tvoří tab navigator – lištu ve spodu aplikace, která umožnuje jednoduchou navigaci mezi právě těmito základními částmi aplikace.

5.5.1 Přehled

Přehled je výchozí obrazovkou aplikace přihlášeného uživatele. Zobrazuje pracantovi výpis nejnovějších upozornění souvisejících s jeho sledovanými poptávkami nebo uzavřenými objednávkami. Každá položka zároveň funguje jako odkaz na příslušný detail poptávky nebo objednávky. Seznam a jeho jednotlivé položky splňují pravidla definovaná v předchozí podkapitole. Hlavička obrazovky informuje o počtu nepřečtených upozornění a zároveň dovoluje uživateli označit všechna tato upozornění jako přečtená.

Na pozadí této obrazovky běží velmi důležitý proces, který zajišťuje fungování push notifikací. Veškerá funkcionalita je realizována pomocí služby Firebase Cloud Messaging. Při prvotním přihlášení do aplikace je uživatel dotázán na povolení push notifikací. Po udělení souhlasu je pro jeho zařízení vygenerován přístupový token. Díky tomuto tokenu pak Firebase přesně ví, na která zařízená má upozornění zasílat. Důležité je zmínit, že realizace push notifikací pomocí FCM je oficiálně podporována přímo v rámci Expa a samotná implementace vyžaduje pouze minimální délku zdrojového kódu. [26]



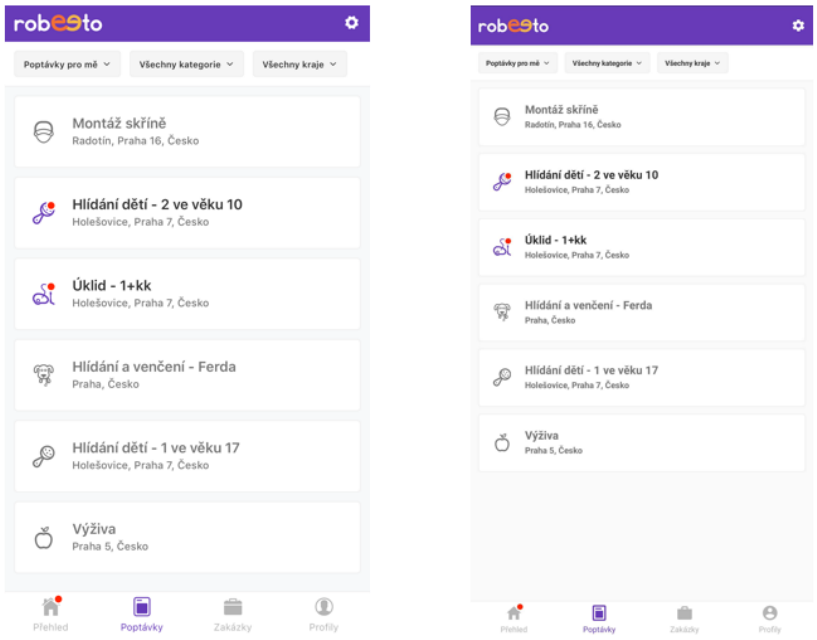


Obrázek 3: Přehled, zdroj: (autor 2019)

5.5.2 Poptávky

Výpis poptávek je stejně jako zbytek základních obrazovek realizován pomocí nekonečného seznamu. Jednotlivé poptávky jsou prezentovány ikonou kategorie, názvem poptávky práce a místem realizace. V případě, že u dané poptávky již zákazník učinil nabídku, je zde přítomen také dodatečný řádek s informací o stavu dané nabídky. Nabídka může být aktivní, neaktivní (v případě změny informací poptávky je vyžadováno opětovné potvrzení nabídky ze strany pracanta) nebo objednaná ze strany zákazníka. Každý stav je vyjádřen jinou barvou textu, které reflektují stylizační pravidla webové verze Robeeto.

V hlavičce obrazovky se vyskytují tři druhy filtrů – třídění z hlediska vztahu k poptávce, kategorie poptávky nebo místa realizace. První z uvedených filtrů umožňuje uživateli zobrazovat například pouze poptávky, na které reálně může podávat nabídky (dle existence profilu v dané kategorii), nebo dříve sledované a odmítnuté poptávky. Filtrace podle kategorie zužuje výběr pouze na danou množinu požadovaných prací a v poslední řadě filtrace dle místa realizace umožňuje hledat pracantovi práci na základě jednoho z krajů České republiky.

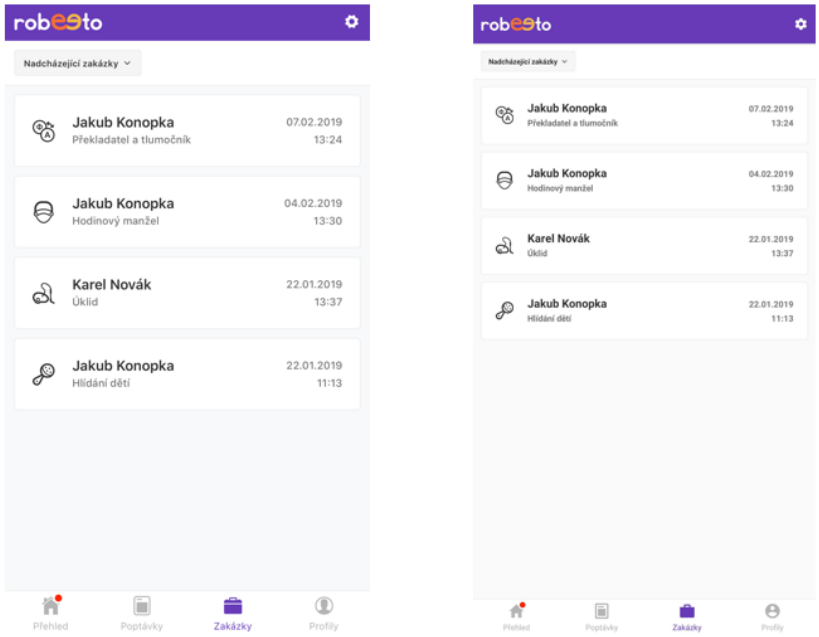


Obrázek 4: Poptávky, zdroj: (autor 2019)

5.5.3 Zakázky

Výpis zakázek je v mnoha ohledech velmi podobný předchozí obrazovce. Jednotlivé zakázky jsou opět reprezentovány ikonou kategorie, dále se zde nachází jméno zákazníka, název kategorie realizovaných služeb a datum vystavení objednávky. Zakázky jsou řazeny chronologicky.

Hlavička obrazovky nabízí filtraci mezi nadcházejícími a již uskutečněnými zakázkami. Ve výchozím stavu je jsou zobrazovány zakázky nadcházející – pracant tak okamžitě ví, jaké zakázky ho v nejbližší době čekají. Uskutečněné zakázky jsou pro pracanty důležité zejména kvůli hodnocení, které mohou od zákazníka obdržet po realizaci sjednané služby.

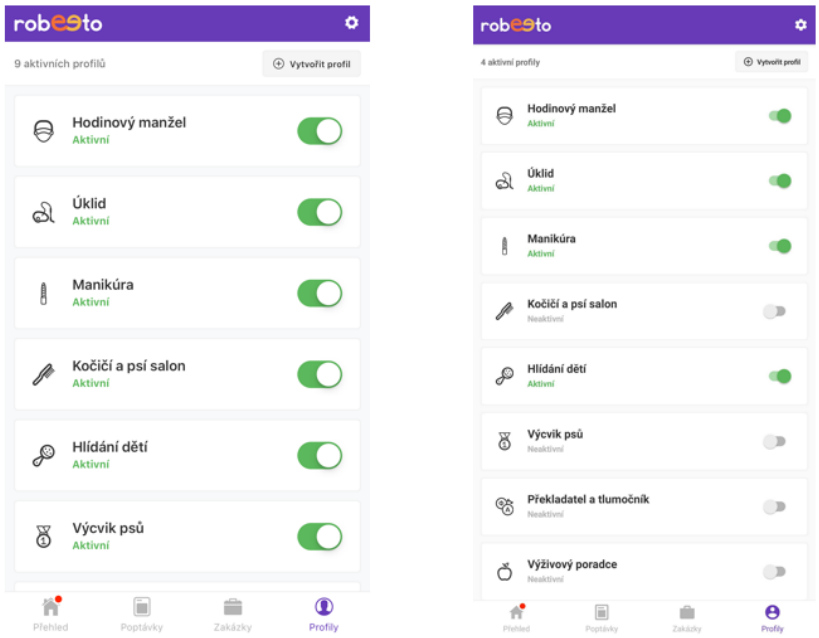


Obrázek 5: Zakázky, zdroj: (autor 2019)

5.5.4 Profily

Poslední obrazovkou je sekce uživatelských profilů. Zde pracant přehledně zjistí, v jakých kategoriích nabízí své služby. Za tímto účelem existuje i možnost již existující profil deaktivovat nebo znovu aktivovat. Tato funkcionalita je zajištěna pomocí nativní komponenty Switch, která zajišťuje snadnou pochopitelnost nezávislou na platformě. Jako pojistka je zde i textově vyjádřená informace o stavu daného profilu. Na základě stavu spolu přepínač i text sdílí stejnou barvu. V případě aktivního profilu se jedná o zelené zabarvení, u neaktivního profilu pak šedé.

Posledním důležitým prvkem na obrazovce je tlačítko pro vytvoření nového profilu. Samotná funkcionalita tvorby profilu se ale jevila natolik složitě a časově náročně, že od ní nakonec bylo odstoupeno a je realizována pouze skrze příslušný odkaz do webové aplikace. V závislosti na časových možnostech bude se zvýšenou pravděpodobností realizována v nadcházejících verzích. Absence této funkce není natolik závažná, jelikož mobilní aplikace bude zpočátku nabízena nejprve dlouhodobým pracantům, kteří tvorbu profilu využijí již jen minimálně.



Obrázek 6: Profily, zdroj: (autor 2019)

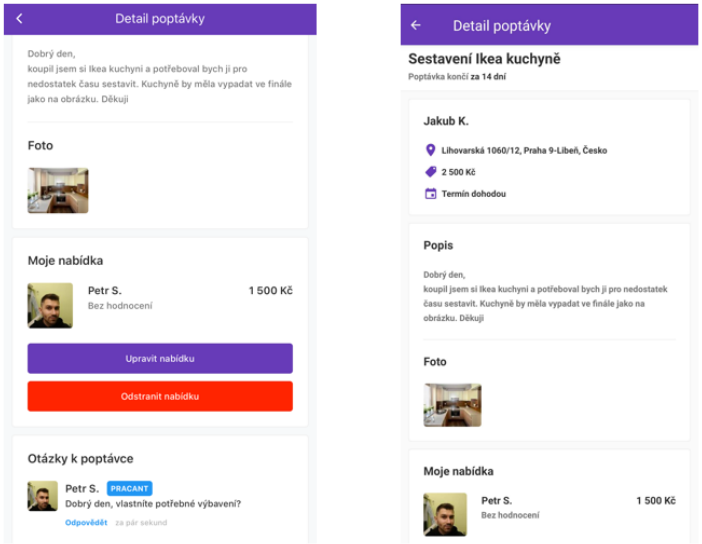
5.6 Vedlejší obrazovky

V dolní části vedlejších obrazovek se tab navigator nevyskytuje. Díky tomu zde vzniká další prostor, který umožňuje zobrazit jiné dodatečné informace. V rámci hlavičkové navigace zde však existuje možnost přesměrování zpět na hlavní obrazovky a to pomocí šipky zpět.

5.6.1 Detail poptávky

Detail poptávky je realizován pomocí oddělených boxů, které zahrnují obecné informace (místo, cena, termín), popis, specifické informace, seznam podaných nabídek a otázek k poptávce. U seznamu podaných nabídek je na první místo přednostně umístěna nabídka přihlášeného pracanta. V této části má pracant možnost nabídku také upravit nebo zrušit. Tlačítko přidání nabídky pak uživatele přesměruje do samostatného formuláře.

Vzhled sekce s otázkami byl inspirován pravidly používaných u blogových příspěvků. Jednotlivá vlákna jsou vizuálně oddělena a je tak velmi jednoduché například oddělit novou otázku od reakce na otázku stávající. Na základě této implementace byl následně sjednocen vzhled této sekce i v rámci webové verze. Pro zjednodušení orientace v otázkách a následných odpovědích byl u každého příspěvku zaveden štítek, který udává, zda je autorem příspěvku zákazník nebo pracant. Podle poskytnuté zpětné vazby testovacích uživatelů tato drobná funkcionalita výrazně zvyšuje čitelnost a orientaci zejména u poptávek s mnoha položenými otázkami. Stejně jako v případě podání nabídky je i přidání komentáře realizováno na samostatné obrazovce.

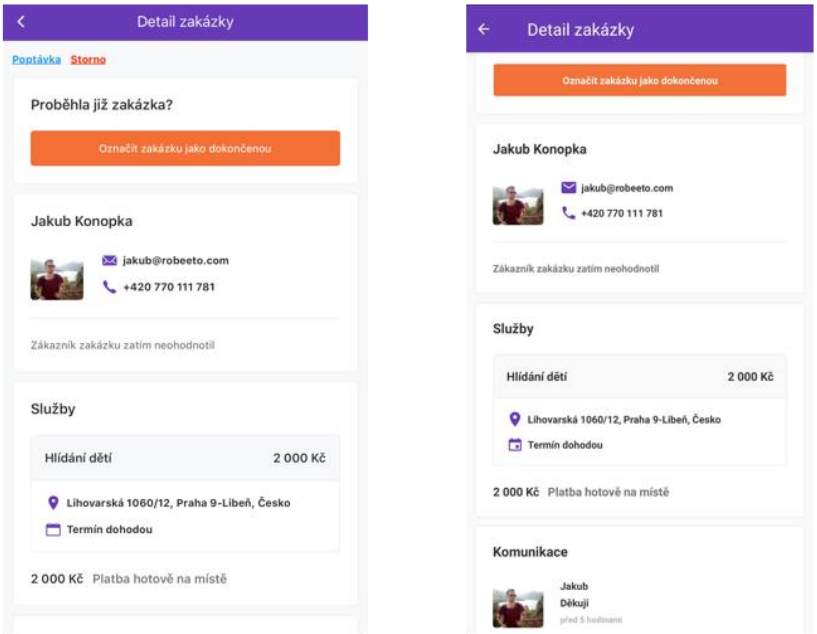


Obrázek 7: Detail poptávky, zdroj: (autor 2019)

5.6.2 Detail objednávky

Po vzoru výpisu poptávek a objednávek je dodržováno stejné rozložení prvků i v rámci detailu poptávky o objednávky. V horní části obrazovky má pracant možnost objednávku stornovat nebo naopak potvrdit její dokončení. Obě akce ještě vyžadují dodatečné potvrzení, aby nedocházelo k nechtěnému jednání ze strany uživatelů. V kontaktní sekci jsou zobrazeny kontaktní údaje na zákazníka – emailová adresa a telefonní číslo. Při interakci dochází k vyvolání nativních událostí – otevření emailového klienta nebo vytočení daného čísla. Jedná se o velmi důležitou část, která zprostředkovává přímý kontakt mezi zákazníkem a poskytovatelem.

Dále jsou k dispozici informace o jednotlivých objednaných službách a to včetně množství, ceny a časové náročnosti. Poslední box této obrazovky je věnován soukromé konverzaci. Oproti otázkám k poptávce, zde nejsou vypsány jednotlivé zprávy mezi oběma stranami, ale pouze poslední zpráva, která je navíc graficky odlišena v případě, že zpráva doposud nebyla přečtena druhou stranou. Tato metoda zobrazení pouhého náhledu konverzace na úkor celé historie vychází z nejrozšířenější komunikační platformy současnosti – Facebook Messenger. Kompletní výpis soukromých zpráv je pak realizován na samostatné obrazovce. Díky tomuto oddělení působí detail objednávky velmi minimalisticky. Dlouhý výpis několika desítek soukromých zpráv by tento stav mohl narušit, a proto bylo vybráno právě toto řešení.



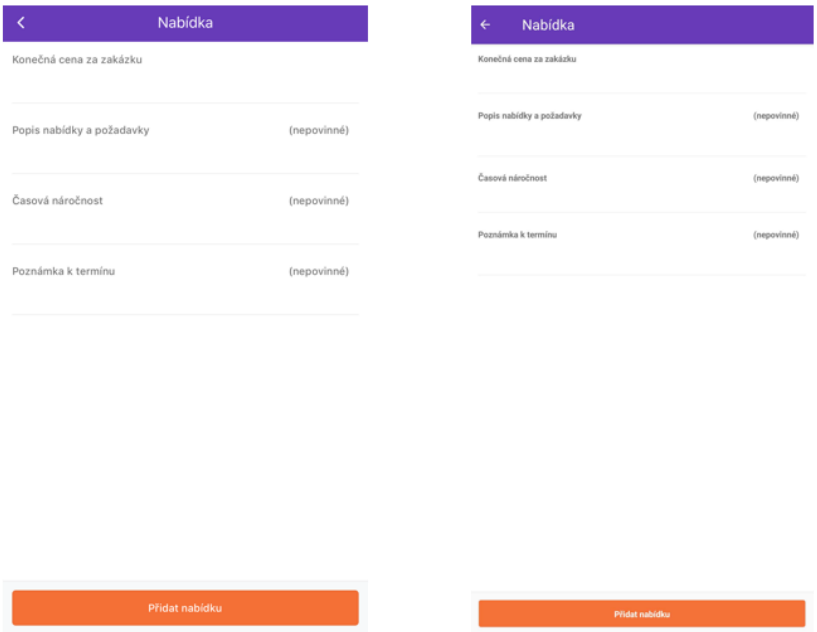
Obrázek 8: Detail zakázky, zdroj: (autor 2019)

5.6.3 Formuláře

Na veškerých formulářových obrazovkách byla aplikována stejná pravidla, jejich implementace se tak ve své podstatě liší pouze počtem a typy inputů a jejich následné validace, která je realizována knihovnou Redux Form. Pro účely mobilní aplikace Robeeto byly vytvořeny následující formulářové obrazovky:

- Přidání a úprava nabídky
- Přidání otázky nebo odpovědi k poptávce
- Přidání soukromé zprávy

Kromě samotných inputů, které jsou postaveny na komponentě Input, existuje u některých formulářů možnost nahrát fotografie. Toho je docíleno pomocí komponenty ImagePicker, která umožňuje vybrat a následně nahrát vybrané fotografie na operačních systémech iOS a Android. Pro přístup k fotografiím je nicméně vyžadováno systémové povolení.

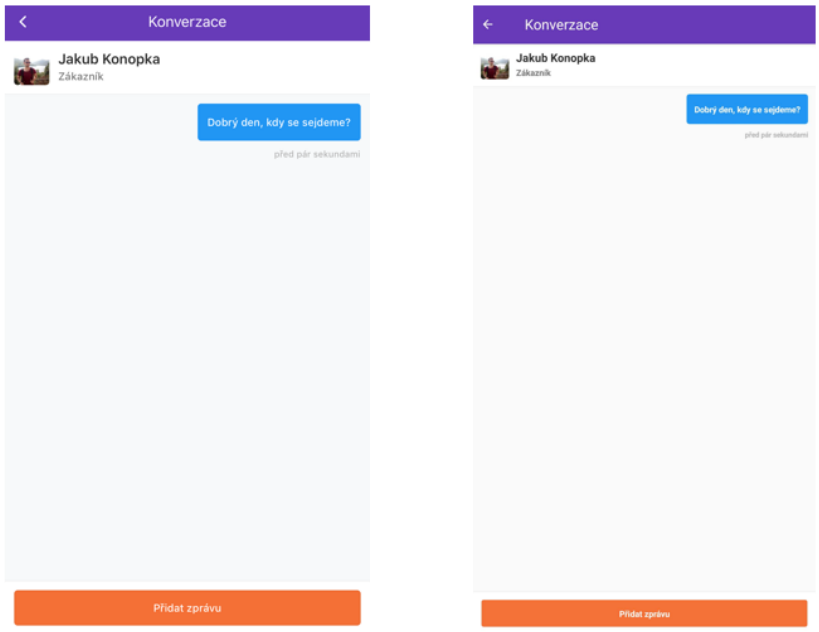


Obrázek 9: Přidání nabídky, zdroj: (autor 2019)

5.6.4 Soukromá konverzace

Obrazovka soukromá konverzace se graficky drží stylu jiných zavedených chatovacích aplikací, jako jsou Facebook Messenger nebo iMessage. Základem je výpis jednotlivých zpráv, které jsou barevně rozlišeny podle strany odesílatele. Zprávy přijaté disponují bílou barvou, naproti tomu zprávy odeslané jsou modré a systémové zprávy fialové.

Liší se také zarovnání zpráv. Zprávy odeslané jsou zarovnávány doprava, systémové a přijaté zase vlevo. Z technologického hlediska se opět jedná o použití komponenty FlatList, tentokrát však s opačným pořadím výpisů jednotlivých prvků. Nejnovější zprávy jsou vypisovány jako poslední a pro zobrazení starších zpráv musí uživatel scrollovat nahoru. Dodržením těchto masivně zavedených principů je zajištěno jednoduché osvojení ze strany uživatelů.



Obrázek 10: Soukromá konverzace, zdroj: (autor 2019)

5.6.5 Galerie

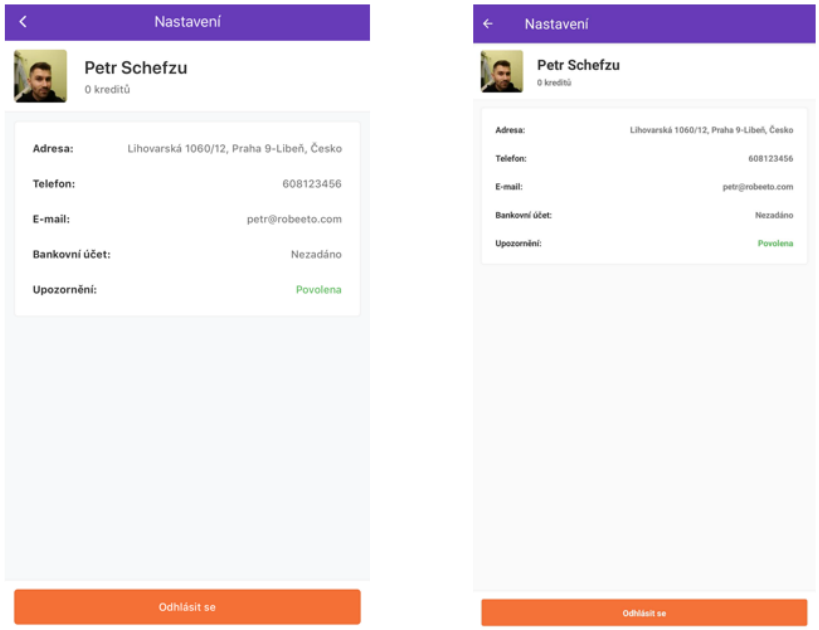
Jak již bylo zmíněno, v některých formulářích existuje možnost nahrát fotografii jako přílohu k otázce nebo zprávě. Pro následné zobrazení těchto příloh proto byla vytvořena samostatná obrazovka. Ta je postavena na externí knihovně ImageViewer. Ta se zaměřuje na práce si fotografiemi přímo na platformě React Native a implementuje širokou škálu užitečných funkcí a gest. Knihovna například umožnuje gesty fotografii přibližovat a zmenšovat, nebo navigaci mezi jednotlivými snímky. [27]

Obrázek 11: Galerie, zdroj: (autor 2019)

5.6.6 Nastavení

Nastavení je poslední obrazovkou implementovanou v prvotní verzi aplikace Robeeto. Obrazovka zobrazuje informace o stávajícím přihlášeném uživateli pomocí profilové fotografie, jména a aktuálního stavu kreditů. Kromě dodatečných informací jako jsou adresa, telefonní číslo, emailová adresa nebo číslo účtu je zde také uveden stav zasílání upozornění.

Uživatel má tak možnost dodatečně povolit zasílání push notifikací v případě, že tak neučinil při prvotním startu aplikace. V dalších verzích bude tato obrazovka rozšířena o jednoduchou sekci pro zaslání zpětné vazby uživatelů a jejich celkové spokojenosti s aplikací. Dolní části aplikace dominuje tlačítko pro odhlášení, které uživatele navrátí na přihlašovací obrazovku.



Obrázek 12: Nastavení, zdroj: (autor 2019)

5.7 Komponenty

Komponenty představují nedílnou součást vývoje v rámci React Native. Tyto znovupoužitelné bloky uživatelského rozhraní zásadně zjednodušují práci vývojářům a zvyšují jednotnost aplikace. V rámci této podkapitoly budou představeny vybrané komponenty, které sehrály stěžejní roli při implementaci mobilní aplikace Robeeto.

Button

Téměř žádná mobilní aplikace se v dnešní době neobejde, z pohledu svého uživatelského rozhraní, bez tlačítek. React Native disponuje vlastní komponentou Button, která se však vizuálně velmi liší mezi platformami iOS a Android. Zejména v rámci operačního systému iOS působí tlačítka velmi nevýrazně a proto bylo implementováno vlastní řešení této komponenty. To se inspiruje vzhledem tlačítek ve webových aplikacích. Díky vlastní implementaci tak mají tlačítka takřka stejný vizuál nehledě na operační systém.

Komponenta samotná přijímá atributy jako typ, ikona, popis a reakce na událost. Typ definuje, jaká kombinace barev bude na tlačítko aplikována. Jedná se o výčtový datový typ, který vychází z předem definovaného seznamu barevných kombinací. Díky tomu je zachována vizuální integrita jednotlivých tlačítek napříč aplikací. Reakce na událost je ve své podstatě jednoduchá funkce, která definuje, jaká bude reakce po interakci s tlačítkem.

File

Na některých obrazovkách aplikace existuje možnost nahrát nebo zobrazit přílohy. Ty jsou rozděleny do dvou skupin: obrázky (JPG, PNG, GIF) a ostatní (např. PDF). Pro tyto potřeby je nutné mít k dispozici jednoduchou komponentu, která rozhodne, jak bude každá jednotlivá příloha, na základě svého typu, zobrazena.

Přílohy typu obrázek díky implementaci této komponenty zavedou uživatele aplikace na obrazovku Galerie. Složitější je situace u druhé skupiny příloh. Obecně platí, že operační systém Android je daleko benevolentnější ohledně otevírání souborů s různými koncovkami, zatímco iOS umožňuje otevírat daleko menší množství souborových typů. Optimálním řešením tohoto multiplatformního problému se stala komponenta WebBrowser. Ta danou přílohu otevře v rámci výchozího webového prohlížeče systému, který poté sám uživateli nabídne, co s daným souborem je schopný udělat. Zodpovědnost za manipulaci s přílohami různých typů je tak elegantně přenesena na samotný operační systém.

Icon

Ikony, stejně jako tlačítka, jsou považovány za naprosto elementární grafické indikátory uživatelského rozhraní. U webových aplikací neexistují žádná striktní pravidla, která by předepisovala použití konkrétních setů ikon. V rámci vývoje mobilních aplikací se však samotní výrobci operačních systémů snaží na svých platformách prosadit použití jimi předdefinovaných ikon. Apple definuje sadu ikon na základě pravidel Flat Designu, Google zase vychází ze svého Material Designu. Rozdílnost ikon napříč systémy je tedy značná. Ani zde však neexistují přímá nařízení, která vývojáře limitují k použití některého z těchto balíčků ikon, nicméně jejich použití je velmi doporučováno a to zejména pro zachování bezproblémového pochopení uživatelského rozhraní samotnými uživateli aplikací.

Při práci s React Native, kdy je aplikace vyvíjena pro obě platformy zároveň, bylo proto potřeba najít řešení, které vyhoví oběma stranám. Řešením je jednoduchá komponenta, která jako argument přijímá název, popřípadě rozměry ikony. Název ikony by sám o sobě tento problém nevyřešil, nicméně Expo poskytuje vývojářům celou řadu různých setů ikon. Balíček Ionicons pak představuje řešení výše zmíněného problému. Ionicons totiž disponují desítkami ikon, kdy každá ikona je přítomna ve dvou variantách – pro iOS i Android. Pro rozlišení, která varianta má být použita, slouží krátký prefix. Poté, co komponenta Icon obdrží název požadované ikony, zkontroluje, na které platformě je aplikace spuštěna a poté vrátí příslušnou variantu.

Input

Pro vytváření formulářových obrazovek byla navržena univerzální komponenta Input, která umožňuje definovat inputy různých druhů při automatickém zachování stejného vzhledu a vzorce chování. Komponenta přijímá argumenty typ inputu, popis, meta data a doplňková volitelná nastavení. Typ rozhoduje o tom, zda bude zobrazen textový, číselníkový nebo jiný input. Bez ohledu na typ bude vždy navrácena stejná vizuální struktura, která zahrnuje pozici, stylizaci popisu a vizualizaci inputu samotného.

Vzhled inputu je sjednocen napříč operačními systémy, kdy jsou použity principy z obou světů. Důležitým benefitem je fakt, že jakýkoliv další input není třeba stylovat nebo upravovat jeho chování, vše je defaultně nastaveno touto komponentou. Meta data jsou poskytována knihovnou Redux Form a sdělují informace o tom, zda je například hodnota daného inputu validní. Za situace, kdy hodnota validní není, dochází automaticky k zobrazení chybové hlášky, která navádí uživatele k vyplnění správné hodnoty. Implementací této komponenty došlo ke

značnému zjednodušení procesu tvorby formulářových sekcí, zejména díky centralizovanému řízení všech typů inputů na jednom místě.

Picker

Pro filtraci dat na výpisových obrazovkách (Poptávky, Zakázky) byla použita vlastní komponenta Picker, která přijímá seznam možných hodnot a aktuálně vybranou hodnotu. Po vyvolání filtračního dialogu zobrazuje seznam variant ve formě zaškrtávacích možností. Jedná se o zavedenou a uznávanou metodu využívanou v rámci formulářů na webu i na mobilních zařízeních. React Native ve svém základu disponuje podobnou komponentou, nicméně její API je do značné míry omezené, kdy nelze dost dobře přizpůsobit vzhled jednotlivých prvků a především nelze tvořit vnořené výběry, které jsou v rámci aplikace nezbytné například při výběru kategorie služeb a potažmo subkategorie služeb.

Spinner

Velmi jednoduchou komponentou na závěr je Spinner. Jedná se o velmi jednoduchý grafický prvek, založený na rotujícím kolečku, který bývá využíván jako indikátor při vyřizování různých druhů požadavků. Použitím Spinneru je uživateli dáno najevo, že právě dochází k výměně dat. Uživatel tak není udržován v nevědomosti, zda jeho akce vyvolala požadovanou reakci. Tato komponenta ve své velmi jednoduché podstatě využívá nativní komponentu stejného jména, které lze dodatečně nastavit velikost nebo barevné provedení. Hlavním přidaným benefitem pak je, že díky jednoduchým stylizačním pravidlům je možné tuto komponentu využít téměř kdekoliv a komponenta se vždy proporcionálně přizpůsobí danému prostoru. Vzniká tak prostý grafický indikátor se širokým spektrem použití napříč aplikací.

5.8 Deployment

Po úspěšné implementaci přichází na řadu vydání aplikace neboli deployment. Rozsah technologií, se kterými se pracuje v této bakalářské práci, přináší do této fáze vývoje softwaru svá specifika. Centrálním bodem práce s React Native je již několikrát zmíněné technologické řešení Expo. To drží celou aplikaci pohromadě a má neodmyslitelnou roli i v rámci vydávání aplikace do produkce.

Build

Jakmile je zdrojový kód kompletní a aplikace plně funkční, přichází na řadu vytvoření tzv. obrazu aplikace neboli build. Jedná se o již přeložený zdrojový kód do nativního jazyka dané mobilní platformy. Výsledkem je jediný soubor s příponou dle cílového operačního systému.

Build pro operační systém android nese příponu APK, zatímco u iOS aplikace je to IPA. Tento soubor je pak možné publikovat v příslušném obchodě aplikací v rámci poslední fáze vydání.

Každý build je jednoznačně identifikován skrze název aplikace, verzi aplikace a číslo příslušného buildu v rámci verze. Vytvoření obrazu aplikace je možné vyvolat s pomocí příkazu v příkazové řádce dle API Expa. Důležité je však upozornit, že v této fázi vývoje je již nezbytné, aby vývojářský tým disponoval příslušným developerským účtem na platformě, kam plánuje aplikaci vydávat. Po registraci do systému je třeba uhradit roční poplatek, který vývojářům umožňuje po celou dobu vydávat neomezené množství aplikací, které se samozřejmě ale musí řídit definovanými pravidly a směrnicemi. Při vytváření buildu pro danou platformu je vývojář tedy vyzván k zadání přihlašovacích údajů a poté je tento build trvale spjat s tímto účtem. Tento proces je nicméně nutné podstoupit pouze při první tvorbě obrazu aplikace. V jakékoliv fázi vývoje však lze kdykoliv tento účet vyměnit, nicméně změny se projeví pouze u nadcházejících buildů.

Store

Store neboli obchod je místo, kde mají uživatelé mobilní platformy možnost aplikace volně stahovat. Bez ohledu na cílovou platformu je proces nahrání buildu do obchodu totožný. Pro založení aplikace pro operační systém iOS je třeba využít služby iTunes Connect, zatímco pro Android existuje služba Google Play Console. [28] [29]

Nejprve je nutné vytvořit profil nové aplikace, zde dochází k vyplnění základních údajů o aplikaci, včetně autora nebo věkového omezení. Běžnou praxí je také nahrání několika vzorových obrazovek jako forma náhledu pro návštěvníky profilu aplikace. To umožňuje potencionálním uživatelům vidět vzhled a funkcionality aplikace dříve než samotnou aplikaci nainstalují. Po vyplnění všech potřebných informací o aplikaci je třeba nahrát již konkrétní verzi buildu, se kterým bude aplikace vydána. U již vydaných aplikací představuje nahrání nového obrazu formu aktualizace. S nahráním buildu se automaticky do aplikace propisuje i jeho číslo verze. Systém správy verzí hraje u mobilních aplikací důležitou roli. Uživatelé mají možnost u každé nové verze aplikace sledovat, jaké novinky tato aktualizace přináší.

Posledním krokem je pak schválení aplikace kontrolory ze strany Applu či Googlu. Tento proces trvá zpravidla 48 hodin, během kterých je zjišťováno, jestli aplikace neporušuje některá pravidla a zda obsah odpovídá popisu. Možnými výsledky tohoto procesu je buď přijetí a následné oficiální vydání aplikace do obchodu nebo zamítnutí, kdy je potřeba odstranit nalezené nedostatky a poté znovu zažádat o schválení.

Testing

Proces testování softwaru vyvíjeného za pomocí knihoven React Native a Expo probíhá v drtivé většině času v rámci emulátoru. Toto téma již bylo představeno v kapitole, která se zabývá knihovnou Expo a jeho funkcionalitami. Jakmile je aplikace dokončena a připravena k vydání, je více než nezbytné ji otestovat na fyzických zařízeních. Podobně jako proces nahrávání aplikace do obchodu jsou také možnosti testování pro iOS i Android velmi podobné. Testování samotné probíhá skrze služby iTunes Connect, potažmo Google Play Console.

Ve speciálních záložkách zde existuje možnost přizvat konkrétní uživatele do testování. Po přijetí pozvánky k testování je poté testerům stažena příslušná verze buildu aplikace. Při testování na mobilní platformě od Applu je nicméně pro testery nutností stáhnout speciální aplikaci Testflight. Vývojáři mají poté možnost sledovat v reálném čase výsledky testování a to včetně zpětné vazby, poskytnuté testery či automatickým odchytáváním pádů aplikace. [30]

Publish

Expo disponuje poměrně nově ještě jednou drobnou funkcionalitou, která se týká aktualizací vydané aplikace. Jak již bylo zmíněno, pro záměry oficiálního vydání nové verze aplikace je třeba nahrát do obchodu novou verzi buildu. Jedná se o standardizovaný proces, jehož výhody již byly zmíněny. Nicméně v konkrétních situacích se nejedná o to správné řešení.

Publish je jednoduchý příkaz v příkazové řádce Expa, který umožňuje aktualizovat konkrétní build aplikace, aniž by bylo potřeba jít oficiální cestou přes obchod aplikací dané mobilní platformy. Namísto toho je při spuštění staré verze aplikace uživatelům na pozadí stažena verze nová a při příštím spuštění již aplikace běží na nové verzi. Jedná se o decentní a velmi užitečný nástroj, který se hodí například při opravě nalezených chyb, kdy není dostatek prostoru vyčkat na schválení nové verze oficiální cestou. Při vývoji mobilní verze Robeeto byl tento nástroj několik použit a jeho fungování je naprosto bezproblémové. Vydání aktualizace aplikace touto cestou by nicméně mělo být využíváno zejména pro odstraňování chyb.

Distribuce aplikace

Mobilní aplikace Robeeto byla vydána v lednu roku 2019 a pro zájemce je volně dostupná ke stažení zde:

- iOS verze: https://itunes.apple.com/cz/app/robeeto-pro-pracanty/id1449985556
- Android verze: https://play.google.com/store/apps/details?id=com.myps.robeeto

6. Zhodnocení technologie

Po úspěšně dotažené implementaci a následném vydání aplikace je třeba zpětně zhodnotit klady a zápory použitého technologického řešení. Tato kapitola vychází z vlastních zkušeností nabytým v rámci práce na projektu mobilní aplikace Robeeto.

6.1 Kladné stránky technologie

Odborná náročnost

Vývoj mobilní aplikace, stejně jako téma této bakalářské práce, byl postaven na knihovně React Native. Stále se jedná o relativně mladou technologii, která se však velmi rychle vyvíjí. Pro absolutního nováčka s nulovými zkušenostmi s vývojem webových aplikací může jít o velké sousto, nicméně pro vývojáře, který je obeznámen minimálně se základními principy Reactu, půjde o známé prostředí, které je rozšířené o řadu nových funkcí.

Právě provázanost s původním frameworkem od Facebooku je klíčová pro osvojení principů vývoje v React Native. Přestože se jedná o práci na nové platformě, spousta principů z prostředí webu je zachována, což obecnou složitost značně snižuje. Díky tomu bylo možné některé bloky kódu kompletně převzít z webové verze aplikace. Možnost znovupoužitelného zdrojového kódu je velmi užitečným benefitem.

Dokumentace a komunita

Nesmírně důležitým faktorem při vývoji jakéhokoliv softwaru je stav dokumentace u používaných technologií. V rámci této práce bylo zpracování dokumentace příslušných technologií a knihoven shledáno jako více než dostatečné. Na veškeré problémy, které vznikly v průběhu vývoje, byly nalezeny řešení buď na oficiálních stránkách vývojářů anebo v rámci populárních diskuzních fór. Popularita Reactu a potažmo také React Native stále roste strmě vzhůru, díky čemuž si vývojové týmy dávají na zpracování dokumentace a komunikaci s komunitou velmi záležet.

Komplexita a přizpůsobitelnost

Právě díky dobré práci s komunitou, která každým měsícem rychle roste, má dnes vývojář k dispozici celou řadu knihoven a rozšiřujících funkcí, které lze v kombinaci s React Native použít. React Native spolu s Expem v základu disponují širokou řadou komponent a funkcí, které umožňují využívat těch nejpopulárnějších stavebních bloků mobilních aplikací, které zároveň také respektují poslední trendy v rámci tohoto odvětví. Je na rozhodnutí samotného

vývojáře, zda využije výchozí nativní knihovny, knihovny třetích stran, či dokonce implementuje vlastní řešení.

Rychlost a stabilita

Velké otazníky před začátkem vývoje vzbuzovala celková rychlost aplikace. React Native přišel na trh s tvrzením, že výsledný kód je, co do rychlosti, v podstatě totožný s nativním. Po absolvování celé etapy vývoje s tímto tvrzením lze téměř bez výjimky souhlasit. V průběhu testování se nevyskytl žádný zásadní problém a aplikace běží velmi svižně. S přihlédnutím k ostatním hybridním řešením lze říci, že React Native konkurenty o třídu převyšuje. Velmi dobře se technologie jeví i na relativně starších mobilních zařízeních, kde existují drobná omezení, která však nebrání běžnému použití aplikace.

6.2 Záporné stránky technologie

Cross-platform podpora

Stěžejním benefitem bývá u React Native uváděna schopnost distribuovat aplikace na obě nejrozšířenější mobilní platformy na světě. To s sebou nicméně přináší řadu kompromisů. Během vývoje byla objevena řada problémů a překážek, kdy se dílčí komponenty zkrátka na každé platformě zobrazovaly nebo chovaly jinak. Některé funkcionality nativních komponent jsou také striktně vázány pouze k jedné z platforem (ve většině případů iOS). To znemožňuje vývojářům konstruovat komponenty naprosto nezávislé na platformě. V některých případech se dokonce určité druhy komponent nechovaly dle uváděné dokumentace. Tento druh problémů se projevoval zejména v rámci operačního systému Android.

Debugging

Debugging neboli ladění programu patří spíše mezi záporné stránky vývoje v React Native. Zejména chybové hlášky představovaly během implementace častá zdržení. Některé zprávy o chybách nemají žádnou vypovídající hodnotu, je proto velmi těžké odhalit vzniklý problém. Některé verze React Native jsou zase nepříliš dobře podporovány knihovnou Expo, která se kompletně stará o kompilaci. Při manuálním zvolení používané verze React Native může Expo vracet chyby a výjimky, které by u oficiálně podporované verze nenastaly. Při plánování vývoje je toto nutné brát v potaz, aby se zabránilo případným problémům v budoucnu.

6.3 Shrnutí

React Native bezesporu představuje velice zajímavý způsob vývoje mobilních aplikací. Základní myšlenku, kdy jeden zdrojový kód obsluhuje aplikaci pro obě platformy, shledává autor této práce za stěžejní benefit celé technologie. V případě použití React Native nemusí zaměstnavatel využívat služeb a znalostí odborníků z řad iOS a Android vývojářů, ale postačí si s vývojáři, kteří mají znalost webového vývoje a Javascriptu. Tato skutečnost do značné míry ovlivní i finanční stránku věci. Web developeři mají v současné situaci na trhu stále znatelně nižší finanční nároky, než vývojáři ryze mobilních aplikací. React Native je tak velice výhodným řešením pro společnosti s menším počátečním kapitálem. Samotná technologie pak disponuje širokou škálou funkcí, jakožto i rostoucím počtem členů komunity, která React Native nadále posouvá dále.

Na druhou stranu s sebou React Native stále nese jistá omezení. Některé funkce nejsou zdaleka optimalizovány pro bezproblémový běh na obou platformách a s rostoucím nabobtnáváním aplikace se přehlednost a orientace v kódu dramaticky zhoršuje. Na základě vlastních zkušeností nabytých v rámci vývoje mobilní aplikace Robeeto, autor práce shledává React Native optimální volbou pro menší až středně velké aplikace, kde výhody technologie stále znatelně převyšují zápory.

7. Závěr

Nyní nastává čas sumarizace obsahu práce, včetně naplnění cílů práce, přínosu bakalářské práce a také zhodnocení využitelnosti dosažených výsledků práce.

Shrnutí práce a míra naplnění cílů práce

V rámci této bakalářské práce byly definovány tři dílčí cíle:

1. *Identifikace současných trendů na poli mobilních aplikací, rozdílnosti mezi mobilní a webovou platformou a představení nejpoužívanějších nástrojů pro vývoj mobilních aplikací.*

Veškeré výše uvedené body byly detailně popsány v úvodních kapitolách této práce. V bakalářské práci byly nejprve analyzovány rozdílnosti mezi webovou a mobilní platformou a poté specifika jednotlivých platforem, včetně aktuálních trendů a principů, které je nutné při vývoji dodržovat. Následně byly představeny stěžejní nástroje pro vývoj mobilních aplikací pomocí React Native, kde byl kladen důraz na vysvětlení základních principů a možností, které tyto nástroje přináší.

2. *Výběr konkrétního technologického řešení pro vývoj, odůvodnění jeho použití s ohledem na výkon, rozšířenost nebo zpracování dokumentace a to včetně představení možných alternativ.*

V další části práce bylo představeno kompletní technologické řešení, které bylo použito pro vývoj mobilní aplikace Robeeto. Dle zadaných cílů je u každého řešení vysvětlen jeho přínos a využití v procesu vývoje a také jsou představeny alternativní řešení, která mohou pro podobné účely být také využity.

3. *Samotná implementace mobilní aplikace podle stanoveného zadání, která také zahrnuje volbu počáteční projektové struktury, popis funkčnosti jednotlivých sekcí aplikace a celkové shrnutí kladných či záporných stránek React Native.*

Poslední a zároveň nejobsáhlejší část se zabývá realizací mobilní aplikace. Autor zde postupně představuje aplikační schéma, zahrnuta a objasněna je také volba projektové struktury a poté následuje představení jednotlivých částí aplikace. Každá sekce zachycuje strukturu jednotlivých obrazovek, které jsou doprovázeny komentářem autora o dílčích problémech, které byly v průběhu vývoje objeveny. Práce také kompletně popisuje průběh vydání aplikace do produkce. Závěrem je zhodnocena technologie React Native při použití v komerčním prostředí.

Přínosy bakalářské práce

Autor se ve své práci primárně zaobírá poznatky získanými v praxi. Díky tomu práce poskytuje komplexní přehled v problematice moderního vývoje mobilních aplikací. Práce sleduje aktuální trendy, a proto může sloužit jako doporučení či návod pro zájemce o vývoj v React Native. Některé poznatky, které byly získány v rámci vývoje mobilní aplikace Robeeto, mohou začínajícím i pokročilým vývojářům poskytnout potřebná řešení problémů, se kterými se během procesu vývoje velmi pravděpodobně setkají.

Zhodnocení využitelnosti dosažených výsledků práce

Práce je využitelná jednak pro vývojáře se zájmem o mobilní aplikace, tak i pro vedoucí vývojových týmů, kteří hledají vhodné technologické řešení. Práce sleduje proces vývoje od úplných začátků (kdy je vůbec definována přínosnost mobilní aplikace pro danou společnost), přes kompletní výběr použitých technologií, až po samotnou implementaci a vydání aplikace. Zájemci o práci s React Native tak mají k dispozici vše, co k vývoji potřebují.

8. Zdroje informací

1. React: A JavaScript library for building user interfaces [online]. [cit. 2018-10-07]. Dostupné z: https://reactjs.org
2. Angular [online]. [cit. 2018-10-07]. Dostupné z: https://angular.io
3. Ionic Framework [online]. [cit. 2018-11-18]. Dostupné z: https://ionicframework.com
4. PhoneGap [online]. [cit. 2018-11-18]. Dostupné z: https://phonegap.com
5. React Native: A framework for building native apps using React [online]. [cit. 2018-10-07]. Dostupné z: https://facebook.github.io/react-native
6. Robeeto: Máte radost z práce [online]. [cit. 2018-11-18]. Dostupné z: https://www.robeeto.com
7. Zásilkovna: Síť výdejních míst [online]. [cit. 2018-11-18]. Dostupné z: https://www.zasilkovna.cz
8. Material Design [online]. [cit. 2018-11-18]. Dostupné z: https://material.io/design
9. Human Interface Guidelines [online]. [cit. 2018-11-18]. Dostupné z: https://developer.apple.com/design/human-interface-guidelines
10. Expo: The fastest way to build an app [online]. [cit. 2018-10-07]. Dostupné z: https://expo.io
11. TypeScript [online]. [cit. 2018-12-26]. Dostupné z: https://www.typescriptlang.org
12. Flow: A Static Type Checker for JavaScript [online]. [cit. 2018-12-26]. Dostupné z: https://flow.org
13. React Navigation [online]. [cit. 2018-12-26]. Dostupné z: https://reactnavigation.org
14. React Native Navigation [online]. [cit. 2018-12-26]. Dostupné z: https://wix.github.io/react-native-navigation
15. Redux: A Predictable State Container for JavaScript Apps [online]. [cit. 2018-12-26]. Dostupné z: https://redux.js.org
16. Flux [online]. [cit. 2018-12-26]. Dostupné z: https://facebook.github.io/flux
17. Reselect [online]. [cit. 2018-12-26]. Dostupné z: https://github.com/reduxjs/reselect
18. Redux Form [online]. [cit. 2018-12-26]. Dostupné z: https://redux-form.com

19. Formik [online]. [cit. 2018-12-26]. Dostupné z: https://jaredpalmer.com/formik

20. Ramda [online]. [cit. 2018-12-26]. Dostupné z: https://ramdajs.com

21. Immutable.js [online]. [cit. 2018-12-26]. Dostupné z: https://facebook.github.io/immutable-js

22. Moment.js [online]. [cit. 2018-12-26]. Dostupné z: https://momentjs.com

23. date-fns: modern JavaScript date utility library [online]. [cit. 2018-12-26]. Dostupné z: https://github.com/date-fns/date-fns

24. Axios [online]. [cit. 2018-12-26]. Dostupné z: https://github.com/axios/axios

25. Scaling your Redux App with ducks [online]. [cit. 2019-01-23]. Dostupné z: https://medium.freecodecamp.org/scaling-your-redux-app-with-ducks-6115955638be

26. Firebase Cloud Messaging [online]. [cit. 2019-02-17]. Dostupné z: https://firebase.google.com/docs/cloud-messaging

27. ImageViewer [online]. [cit. 2019-02-17]. Dostupné z: https://github.com/magicwing/react-native-image-zoom-viewer

28. ITunes Connect [online]. [cit. 2019-03-03]. Dostupné z: https://itunesconnect.apple.com

29. Google Play Console [online]. [cit. 2019-03-03]. Dostupné z: https://developer.android.com/distribute/console

30. Testflight [online]. [cit. 2019-03-03]. Dostupné z: https://developer.apple.com/testflight