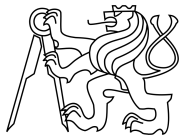
ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE
FAKULTA INFORMAČNÍCH TECHNOLOGIÍ



ZADÁNÍ DIPLOMOVÉ PRÁCE

Název:	Práce s iOS App Extensions v React Native
Student:	Bc. Matěj Kříž
Vedoucí:	Ing. Jan Václavík
Studijní program:	Informatika
Studijní obor:	Webové a softwarové inženýrství
Katedra:	Katedra softwarového inženýrství
Platnost zadání:	Do konce letního semestru 2017/18

Pokyny pro vypracování

Cílem práce je prozkoumat a zdokumentovat možnosti užití iOS aplikačních rozšíření (App Extensions - https://developer.apple.com/app-extensions/) s frameworkem pro vývoj mobilních aplikací React Native a vybrané rozšíření pro tento framework implementovat.

- Analyzujte dostupné způsoby implementace App Extensions pro iOS aplikace.
- Zaměřte se na implementaci (případně integraci) App Extensions s React Native. Také prozkoumejte a popište vhodnost implementace jednotlivých App Extensions v kombinaci s React Native.
- Implementujte samostatnou knihovnu umožňující tvorbu Today Widget v React Native. Knihovnu zdokumentujte a zpřístupněte jako opensource na Githubu.
- Implementujte a otestujte jednoduchou ukázkovou aplikaci v React Native používající Today Widget. Tuto implementaci porovnejte s nativní. Ukázková aplikace bude zobrazovat data z React Native aplikace a pomocí Today Widgetu je bude vhodně zobrazovat v souladu s iOS Human Interface Guidelines.

Seznam odborné literatury

Dodá vedoucí práce.

Ing. Michal Valenta, Ph.D.
vedoucí katedry

prof. Ing. Pavel Tvrdík, CSc.
děkan

V Praze dne 17. února 2017

Poděkování

Rád bych zde poděkoval panu Ing. Janu Václavíkovi za jeho cenné rady a připomínky.

Prohlášení

Prohlašuji, že jsem předloženou práci vypracoval(a) samostatně a že jsem uvedl(a) veškeré použité informační zdroje v souladu s Metodickým pokynem o etické přípravě vysokoškolských závěrečných prací.

Beru na vědomí, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorského zákona, ve znění pozdějších předpisů. V souladu s ust. § 46 odst. 6 tohoto zákona tímto uděluji nevýhradní oprávnění (licenci) k užití této mojí práce, a to včetně všech počítačových programů, jež jsou její součástí či přílohou a veškeré jejich dokumentace (dále souhrnně jen „Dílo“), a to všem osobám, které si přejí Dílo užít. Tyto osoby jsou oprávněny Dílo užít jakýmkoli způsobem, který nesnižuje hodnotu Díla a za jakýmkoli účelem (včetně užití k výdělečným účelům). Toto oprávnění je časově, teritoriálně i množstevně neomezené.

V Praze dne 9. ledna 2018 .....

České vysoké učení technické v Praze

Fakulta informačních technologií

Odkaz na tuto práci

Kříž, Matěj. *Práce s iOS App Extensions v React Native.* Diplomová práce. Praha: České vysoké učení technické v Praze, Fakulta informačních technologií, 2018.

Abstrakt

Platforma iOS nabízí několik aplikačních rozšíření, která mohou zvýšit dosah aplikací zpřístupněním funkcí integrovaných do systému a interakcí s ostatními aplikacemi. Tato práce stručně popisuje jednotlivá aplikační rozšíření a jejich možnosti použití s frameworkem React Native. Dokumentuje postup vytváření open source knihovny *react-native-today-widget* pro snadné přidání Today widgetu do React Native aplikace. Použití knihovny demonstruje na ukázkovém widgetu zobrazujícím auta v okolí, která jsou dostupná k zapůjčení prostřednictvím mobilní aplikace HoppyGo.

Klíčová slova React Native, Aplikační rozšíření, Today widget, JavaScript, open source, iOS

Abstract

The iOS platform supports several app extensions. They could increase the impact on users by enabling access to more system features and interactions with other apps. This thesis briefly describes individual app extensions and

their possibilities to be used with framework React Native. It documents a process of creating a new open-source library *react-native-today-widget* for easy embedding Today widgets to React Native apps. Use of the library is shown on widget presenting nearby cars for rent by HoppyGo mobile application.

Keywords React Native, App Extensions, Today Widget, JavaScript, open-source, iOS

Obsah

Seznam obrázků

Seznam tabulek

Úvod

React Native (RN)[1] je framework umožňující psát v jazyce JavaScript mobilní aplikace, které mohou být pro uživatele nerozpoznatelné od těch implementovaných v nativních jazycích. React Native má zcela jinou filozofii než dřívější hybridní frameworky, které pouze spouštěli webové aplikace ve WebView[2]. Umožňuje využívat přímo nativní prvky uživatelského rozhraní (UI). Nativní prvky jsou velice důležité pro uživatelský prožitek (UX), neboť jsou zpravidla dobře výkonově optimalizovány a také vypadají přesně podle zvyklostí dané platformy.

Dokud si v aplikaci vystačíme se základními UI prvky, můžeme psát sdílený kód pro více platforem. Pokud ale chceme využít potenciál platformy naplno, bez nativního kódu se zatím neobejdeme. Příkladem systémových funkcí, které v současné době RN nepodporuje (nebo je podporuje jen částečně), jsou jednotlivá aplikační rozšíření.

Kód RN lze bez problémů integrovat s nativním kódem. Větším týmům se vyplatí mít kromě RN vývojářů i nativního vývojáře pro každou požadovanou platformu. Já jsem zatím pracoval v týmech, kde jsme nativní vývojáře neměli a tak jsem začal do iOS a Androidu pronikat postupně sám.

Mojí hlavní motivací v rámci této práce bylo podpořit rozvoj open source projektu RN. Nejprve jsem uvažoval o implementaci RN pro některou novou platformu. To by ovšem zdaleka přesahovalo rozsah diplomové práce a samotnému by mi to trvalo příliš dlouho. Raději jsem tedy zvolil implementaci aplikačního rozšíření Today widget. Výhodou oproti implementaci RN pro zcela novou platformu je možnost použít většinu z již implementovaných komponent pro iOS.

[1] <https://facebook.github.io/react-native/>
[2] <https://developer.telerik.com/featured/what-is-a-webview/>

Další motivací bylo hlouběji proniknout do nativního vývoje na platformě iOS a poskytnout náhled na nativní vývoj i dalším vývojářům RN, kteří přichází tak jako já ze světa webových technologií. Z toho důvodu v této práci naleznete mnoho odkazů na užitečné zdroje informací.

Věřím, že tato práce by mohla být zajímavá i pro iOS vývojáře, kteří zde naleznou přehled dostupných aplikačních rozšíření. Také mohou nahlédnout do frameworku RN, který může být velmi užitečným doplňkem nativního vývoje.

Kapitola 1

Cíl práce

Cílem této práce je poskytnout širší přehled o specifických možnostech aplikací na platformě iOS vývojářům používajícím framework React Native (viz 2.1) nebo i jen uvažujícím o jeho použití. Je určena především pro ty vývojáře, kteří přicházejí ze světa webových aplikací. Je třeba vědět o možnostech jednotlivých platforem a umět je využívat. Jinak by byl React Native pro uživatele našich aplikací krok zpět oproti nativním aplikacím.

Konkrétně se práce věnuje jednotlivým dostupným aplikačním rozšířením (*App Extensions*)[3]. V kapitole 2.2 můžete nalézt jak základní přehled, tak odkazy na klíčové pojmy a souvislosti. U každého rozšíření je analyzována možnost a vhodnost implementace s použitím React Native.

Zásadním výstupem této práce je open source knihovna usnadňující tvorbu Today widgetů s pomocí React Native v jazyce JavaScript (viz 2.2.3). Cílem této knihovny je umožnit přidání widgetu do aplikace aniž by bylo nutné otevírat program XCode nebo psát nativní kód.

Použití implementované knihovny je demonstrováno v ukázkové aplikaci (viz 3.2) a porovnáno s čistě nativní implementací Today widgetu.

[3] <https://developer.apple.com/app-extensions/>

Kapitola 2

Analýza a návrh

Na začátku kapitoly objasním co je framework React Native a proč je zajímavý, poté se budu věnovat jednotlivým aplikačním rozšířením, návrhu knihovny pro tvorbu Today widgetů a návrhu ukázkové aplikace.

2.1 React Native

Framework pro tvorbu mobilních aplikací React Native[4] (RN) od společnosti Facebook byl představen v lednu 2015 na konferenci React.js Conf [1]. Hned 26. března téhož roku byl vydán jako open-source na GitHubu.

RN je založen na Reactu — JavaScriptové knihovně pro tvorbu uživatelských rozhraní (také od Facebooku). Jen místo internetového prohlížeče cílí na mobilní platformy [2]. Weboví vývojáři díky němu mohou psát mobilní aplikace a využívat přitom oblíbené JS knihovny, které již znají. Navíc RN umožňuje sdílet většinu kódu mezi platformami a je tedy možné zároveň vytvářet aplikaci pro iOS i Android.

Z Reactu je také v RN dostupná syntaxe JSX[5]. RN poskytuje komponenty, které se vykreslují jako nativní elementy podle aktuální platformy (např. RN komponenta `View` se vykreslí jako `UIView` na iOS, `android.view` pro Android, případně `<div>` v prohlížeči[6]). RN dále implementuje API pro přístup k funkcím mobilní platformy jako je fotoaparát, vibrace, nebo polohové služby. Aktuální seznam všech dostupných komponent a API můžeme najít v dokumentaci RN[7].

[4] <https://facebook.github.io/react-native/>
[5] <https://reactjs.org/docs/jsx-in-depth.html>
[6] RN pro pro prohlížeče je dostupný díky projektu:
<https://github.com/necolas/react-native-web>
[7] <https://facebook.github.io/react-native/docs/>

Kromě komponent implementovaných přímo v RN je možné vybírat z mnoha dalších. Open source komponenty se dají dohledat například v registru *NPM*[8] nebo v katalogu *JS.COACH*[9]. Při volbě komponenty je důležité sledovat nejen popularitu viditelnou v počtu stažení a stars[10], ale především aktivitu správců knihovny (řešení issues[11], pull requesty a kvalitu dokumentace).

2.1.1 Komunita

React Native se od svého vydání coby open-source velmi intenzivně vyvíjí. Kromě mnoha jednotlivých vývojářů[12] se na vývoji RN aktivně podílí i týmy ze společností Microsoft, Airbnb[13] nebo Callstack[14,15]. Microsoft RN používá pro aplikaci Skype a v souvislosti s jejím vývojem vytvořil 70 začleněných pull requestů [3]. Dalším velkým počinem Microsoftu v této oblasti je knihovna CodePush[16,17], která umožňuje aktualizovat JS kód uživatelům v jejich zařízení okamžitě a vyhnout se tak v mnoha případech schvalovacímu procesu pro aktualizace na App Store, které zdržují opravy o několik dní. Mike Grabowski — spoluzakladatel a CTO společnosti Callstack — se aktuálně stará o vydávání nových verzí RN[18]. Také je jedním ze dvou autorů knihovny RNPM[19] pro snadnou instalaci balíčků obsahujících nativní moduly, která je nyní začleněna přímo do projektu RN.

Společnost Callstack také uspořádala konferenci *React Native EU*[20], která se konala 6. - 7. 9. 2017 ve Vratislavi. Šlo o celosvětově první konferenci zaměřenou čistě na témata související s React Native.

2.1.2 Naučte se jednou, napište kdekoli

„Learn once, write anywhere“, tak zní heslo frameworku React Native pro tvorbu mobilních aplikací. Jasně definuje cíl tohoto frameworku, aby mohl jeden vývojář (jeden tým) psát aplikace pro různé platformy. Jak je popsáno v první kapitole [4], nativní vývoj pro jednotlivé platformy se zásadně liší (programovacím jazykem, nativním API i vývojovým prostředím). Na druhou

[8] <https://www.npmjs.com>
[9] <https://js.coach>
[10] <https://help.github.com/articles/about-stars/>
[11] <https://guides.github.com/features/issues/>
[12] 21. 10. 2017 měl RN na GitHubu 1506 přispěvatelů:
<https://github.com/facebook/react-native/graphs/contributors>
[13] <https://github.com/airbnb>
[14] <https://callstack.com>
[15] Tyto tři společnosti se nejviditelněji podílí na vývoji RN. Obsáhlejší seznam společností, které RN používají: <https://facebook.github.io/react-native/showcase.html>
[16] <https://microsoft.github.io/code-push/>
[17] CodePush je nyní součástí App Center <https://appcenter.ms>
[18] <https://github.com/facebook/react-native/releases>
[19] <https://github.com/rnpm/rnpm>
[20] <https://react-native.eu>

stranu není cílem přístup „Write once, run anywhere“[21] jako je tomu například u hybridních webových aplikací. RN respektuje, že jsou konvence a očekávání uživatelů na jednotlivých platformách různé.

Náklady na vývoj nativní mobilní aplikace pro iOS i Android jsou velmi vysoké, jak tvrdí Nader Dabit v [6]. Podle něj RN může tyto náklady zásadně snížit. Největší úsporu nákladů vidí ve sdílení kódu pro obě platformy. Dalších úspor je možné dosáhnout díky nižším platům JavaScript vývojářů ve srovnání s platy nativních vývojářů[22]. A jako třetí výhodu RN vidí ušetření času nutného pro kompilaci nativních aplikací, která může u složitějších aplikací narůst i na 15 minut.

Hybridní webový vývoj pomocí Apache Cordova[23] navíc nevyužívá potenciál mobilního zařízení naplno kvůli zobrazení webové aplikace ve WebView [2]. Prvky uživatelského rozhraní pak zpravidla výkonově zaostávají za těmi nativními. Zvláště na starších zařízeních uživatelé pocítí zhoršenou plynulost ovládání aplikace.

Proto RN nepoužívá pro běh aplikace WebView (samozřejmě je možné element WebView[24] použít tam, kde by dával smysl i v nativních aplikacích). Nepoužívá ani HTML a CSS [7]. V RN jsou aplikace psané v JavaScriptu (JS). Syntaxe JSX byla navržena tak, aby byla snadno použitelná pro každého, kdo zná HTML. Ale jde skutečně o JS funkce. Stejně tak styly[25] odpovídají CSS[26] [8] (jen jména používá v camelCase místo dash-case. Např. `backgroundColor` místo `background-color`), ale jde o JS objekty.

Samotný framework React Native je napsán jen z jedné třetiny v JavaScriptu, jak můžeme vidět na obrázku 2.1. Necelou druhou třetinu tvoří kód v jazyce Java — nativní kód pro platformu Android. A zhruba ze stejné části je RN napsán v jazycích nativních na platformě iOS: Objective-C/C++/C[27].

Při psaní aplikace pomocí frameworku RN je taktéž možné kombinovat JavaScript s nativními jazyky (včetně Swift na iOS, viz 2.4.2). Ovšem pouze kód psaný v JavaScriptu je sdílen napříč platformami. Nativní kód je zapouzdřen do nativních modulů[28]. Když se zaměříme na platformu iOS, volání metod z nativních modulů v JavaScriptu lze umožnit pomocí makra

[21] Slogan společnosti Sun Microsystems pro jazyk Java. [5]
[22] <`https://www.payscale.com/research/US/Skill=JavaScript/Salary`>
[23] <`https://cordova.apache.org`>
[24] <`https://facebook.github.io/react-native/docs/webview.html`>
[25] <`https://facebook.github.io/react-native/docs/style.html`>
[26] Ne všechny vlastnosti jsou dostupné pro všechny komponenty. Text, Image a View mají dostupné různé vlastnosti. Například vlastnosti pro Text jsou zde: <`https://facebook.github.io/react-native/docs/text-style-props.html`>
[27] Jazyk Swift byl představen jen půl roku před RN [9].
[28] <`https://facebook.github.io/react-native/docs/native-modules-ios.html`>

Obrázek 2.1: Přehled použití jednotlivých programovacích jazyků v repozitáři React Native. Zdroj: [10]

`RCT_EXPORT_METHOD()`. Do modulů je možné předat JS funkce jako callback. Nativní modul může také posílat JS události nebo mu při inicializaci předat konstanty.

2.1.3 React Native Bridge

React Native aplikace běží na dvou doménách [11]:

- *Nativní doména*: V této doméně se vykresluje uživatelské rozhraní (všechna `View`). S uživatelským rozhraním se manipuluje výhradně v hlavním vlákně, ale jsou dostupná i další vlákna pro výpočty na pozadí. Na iOS je použit jazyk Objective-C nebo Swift, na Androidu Java či Kotlin.
- *JS doména*: JavaScript se vykonává ve vlastním vlákně pomocí JS enginu JavaScriptCore[29] (viz 2.1.4).

Proměnné definované v jedné doméně nejsou přímo přístupné v druhé doméně. To znamená, že veškerá komunikace mezi těmito dvěma doménami musí probíhat přes takzvaný *bridge*. Vykonávání kódu v obou doménách je rychlé. Největší slabinou je tak komunikace mezi těmito dvěma doménami, a proto bychom ji měli omezovat na minimum[30]. Jak zmiňuje [12], komunikace přes bridge je dávkovaná, asynchronní a posílané zprávy jsou serializované.

Podrobněji architekturu RN vysvětluje Radeu Zagallo v [7] a [13]. Hovoří ještě kromě hlavního vlákna (nativní doména) a JS vlákna o frontách *GCD queues*[31] pro každý nativní modul a jedné (*„shadow queue“*), která obsluhuje rozmístění prvků. Na iOS každý nativní modul ve výchozím stavu používá vlastní fond vláken (thread pool), na Android sdílí nativní moduly jedno vlákno.

[29] To platí jak pro simulátory tak reálné zařízení s iOS i Android, pouze pro debuggování v Chrome se použije V8

[30] Koncept virtuálního DOMu v Reactu velmi dobře optimalizuje přenos dat pro aktualizaci rozhraní.

[31] <`https://developer.apple.com/library/content/documentation/General/Conceptual/ConcurrencyProgrammingGuide/OperationQueues/OperationQueues.html`>

Popisuje co se vlastně děje po spuštění RN aplikace. Nejprve se načte JS bundle, pak nativní moduly a spustí je JS virtuální stroj (VM). VM zkopíruje zdrojový kód, parsuje ho a vygeneruje abstraktní syntaktický strom (AST). Z něj následně generuje bytecode a ten vykonává.

2.1.4 JavaScriptCore

JS engine JavaScriptCore[32] pochází z WebKitu[33]. RN ho používá pro vykonávání JS kódu nejen pro iOS, ale také pro Android a to jak pro simulátory, tak reálná zařízení. Jen když spustíme debuggování přes Chrome, tak se JS vykonává přímo v něm a je tedy použit engine *V8*[34] [8].

JavaScriptCore implementuje ECMAScript podle specifikace ECMA-262[35] neboli ES2017. Obsahuje také JIT kompilátor, který bohužel není na iOS použitý, protože iOS neposkytuje zapisovatelnou a současně vykonavatelnou paměť [8].

2.2 Aplikační rozšíření pro iOS

Aplikační rozšíření (App Extensions, dále také jen rozšíření) jsou malé kousky funkcionality, které rozšiřují možnosti provázání mobilních aplikací se systémem iOS a nebo s ostatními aplikacemi. Umožňují využití některých systémových funkcí, které by jinak samotným aplikacím nebyly přístupné. Oblasti systému, které umožňují rozšíření se nazývají body rozšíření (Extension points).

Aplikační rozšíření jsou dostupná také pro platformu OS X a obdoby existují i na dalších platformách.

Jak je vysvětleno v [14], každé aplikační rozšíření odpovídá právě jednomu bodu rozšíření. Nevytváříme obecná rozšíření, která by využívala více bodů rozšíření. Body rozšíření jsou vždy zabalené v systémových frameworcích a není možné vytvářet nové body rozšíření třetích stran.

Představení aplikačních rozšíření proběhlo v roce 2014 na vývojářské konferenci Apple WWDC[36]. Jsou dostupná v iOS od verze 8. Zahrnutí rozšíření v aplikaci nijak neomezí její distribuci pro starší systémy, ale samotná rozšíření samozřejmě na starších verzích systému nebudou dostupná. Platforma iOS má však velmi vysoké procento přijetí nových verzí systému a tak jsou aplikační rozšíření dostupná pro více než 90 % uživatelů. Oficiální Apple dokumentace uvádí v [15], že 5. 6. 2017 mělo přes 97 % zařízení nainstalováno

[32] <https://trac.webkit.org/wiki/JavaScriptCore>
[33] <https://webkit.org>
[34] <https://developers.google.com/v8/>
[35] <http://www.ecma-international.org/publications/standards/Ecma-262.htm>
[36] <https://developer.apple.com/videos/wwdc2014/>

iOS 9 nebo novější. Alternativní údaj od Davida Smithe z 10. 6. 2017 [16] uvádí 92,4 %, ale ten vychází pouze z údajů o uživatelích jedné aplikace[37], i když velmi rozšířené[38].

Ian Baird vysvětluje ve své prezentaci na Apple WWDC 2014 [17], že rozšíření nejsou samostatné aplikace ani kousky našich aplikací, ani zvláštní mód naší aplikace — jsou to účelově sestavené binárky (special purpose binary) s vlastním podpisem (code signature), s vlastní množinou oprávnění a vlastním kontejnerem. Nelze je tedy chápat ani jako implementaci mechanizmu meziprocesové komunikace (IPC[39]), ale umožňují otevření hlavní aplikace nebo s dodatečným schválením při distribuci do App Store i otevírání aplikací třetích stran implementujících URL schéma.

K rozšířením se přistupuje přes kód Apple frameworků (Apple frameworks code), nikdy je nespouštíme přímo. Například Share Extension se spouští přes Social Framework, Today widget pomocí Notification Center. Definují také některé zásady (Policy) jako charakteristiky spouštění (Launch characteristics)[40]. Pro implementaci rozšíření se využívá třída `UIViewController`[41].

Rozšíření obohacují aplikace. Přicházejí a odcházejí s aplikací. Uživatelé je nemohou instalovat samostatně [18]. Systémová oprávnění jsou sdílená. Aplikaci, která zahrnuje jedno nebo více aplikačních rozšíření nazývám *hlavní aplikace* (v angličtině bývá označována jako containing app). Rozšíření běží samostatně, zcela nezávisle na hlavní aplikaci (mají izolovaný adresní prostor, vykonávají se samostatně). Ale i tak jsou neoddělitelně spjaty s hlavní aplikací a musí mít její Bundle ID jako prefix svého Bundle ID (např. hlavní aplikace `org.example` a Today widget `org.example.Today`).

Přídání rozšíření k hlavní aplikaci je možné v XCode vytvořením nového *target*[42]. XCode target definuje nastavení a soubory, které jsou použity pro sestavení produktu v rámci XCode projektu. Hlavní aplikace může tedy mít více rozšíření, když přidáme více targetů.

Když si uživatel nainstaluje hlavní aplikaci z App Store, nainstaluje tím automaticky i její rozšíření. Pro aktivaci jednotlivých rozšíření ale musí uživatel provést nějakou další akci. Někdy jde o povolení v nastavení iOS zařízení. Často je možné rozšíření aktivovat v místě jeho použití. Například na obra-

[37] Audiobooks <`https://itunes.apple.com/us/app/audiobooks/id311507490`>
[38] Přes 50 tis. uživatelských hodnocení na App Store.
[39] <`https://www.techopedia.com/definition/3818/inter-process-communication-ipc`>
[40] Například Share Extension (2.2.1) vždy spouští novou nezávislou instanci a není tedy možné z jedné instance rozbít jinou.
[41] <`https://developer.apple.com/documentation/uikit/uiviewcontroller`>
[42] <`https://developer.apple.com/library/content/featuredarticles/XcodeConcepts/Concept-Targets.html#//apple_ref/doc/uid/TP40009328-CH4`>

zovce Dnes[43] je možné přidávat dostupné Today widgety.

V rozšířeních je možné znovupoužívat kód z hlavní aplikace (Model a Controller) pomocí Frameworků. K tomu je nutné nastavit minimum deployment target na iOS 8 (pro rozšíření, ne nutně pro aplikaci), starší iOS neví jak frameworky dešifrovat.

Pro sdílení dat je možné vytvořit sdílený kontejner (Shared Container). Výchozí stav jsou zcela oddělené kontejnery, a tedy žádná sdílená data. Při implementaci sdíleného kontejneru je nutné ohlídat kolize. K tomu je možné použít CoreData[44] nebo pro více specifické použití NSFileCoordination[45]. Pro jednodušší databáze typu klíč-hodnota je možné použít UserDefaults[46].

Damien Sorresso z týmu iOS prezentoval na WWDC14 [18] tyto tři zásady pro programátory aplikačních rozšíření:

- *Be Lean*: Nepoužívat mnoho systémových zdrojů, aplikace je stále to nejdůležitější.
- *Be Stateless*: Rozšíření může být kdykoliv agresivně přerušeno (killed) nebo pozastaveno systémem. Rozšíření nemají dostupný žádný multitasking. Podporují rychlé dokončení úkolu (short task-completion) pro uložení stavu na disk.
- *Be Awesome*: Dělejte rozšíření hladce běžící, užitečné, překvapivé a uspokojující!

Dostupné body rozšíření pro platformu iOS jsou v přehledové tabulce 2.1. Dále se budeme věnovat jednotlivým rozšířením zvlášť.

2.2.1 Sdílení (Share extension)

Rozšíření Sdílení umožňuje sdílet obsah do hlavní aplikace. Pro svoji aplikaci tedy programátor může implementovat toto rozšíření a pomocí pravidel aktivace definovaných ve slovníku `NSExtensionActivationRule` určit, pro jaký obsah má být toto rozšíření dostupné v ostatních aplikacích. Konkrétně se podpora pro jednotlivé formáty definuje pomocí klíčů, které jsou ve formátu `NSExtensionActivationSupports{dataType}`, kde se za `{dataType}` dosadí konkrétní sémantický typ, který má být možné sdílet (např. `Text`, `ImageWithMaxCount`, `WebURL`, `WebPage`...) [14].

[43] <https://help.apple.com/iphone/11/?lang=cs#/iphb8f1bf206>
[44] <https://developer.apple.com/library/content/documentation/Cocoa/Conceptual/CoreDaindex.html>
[45] <https://developer.apple.com/documentation/foundation/nsfilecoordinator>
[46] <https://developer.apple.com/documentation/foundation/userdefaults>

Po kliknutí na ikonu sdílení v konkrétní aplikaci se zobrazí uživateli okno (`View`) umožňující sdílení obsahu. Je možné použít výchozí systémové okno `SLComposeServiceViewController` poskytující i obrázkový náhled a počítání znaků nebo může programátor definovat zcela vlastní rozhraní pomocí třídy `NSViewController`. Před samotným sdílením je ještě možné obsah validovat či editovat.

Obrázek 2.2: Ukázka voleb sdílení a voleb akcí poskytovaných třídou *UIActivityViewController*.

Toto rozšíření vždy používá ikonu hlavní aplikace. Aplikace může nabídnout nanejvýš jednu volbu sdílení.

Použití cizích voleb sdílení je možné pomocí třídy `NSItemProvider`[47]. Tato třída reprezentuje dokument v různých formátech (např. TXT, PDF, HTML). V React Native jsou pro sdílení obsahu z naší aplikace dostupná hned dvě API:

[47] <`https://developer.apple.com/documentation/foundation/nsitemprovider`>

- `Share`[48] funguje i pro platformu Android.
- `ActionSheetIOS`[49] je dostupné pouze pro iOS.

Obě nám umožňuje sdílet z naší aplikace jak prostý text, tak soubor (obrázek, video, PDF…).

Pro implementaci sdílení do vlastní aplikace je v RN možné použít jednoho z NPM[50] balíčků:

- *react-native-share-extension*[51] od Ali Najafizadeh,
- *react-native-share-menu*[52] od Caio Almeida.

Knihovna react-native-share-extension je o něco populárnější (měřeno počtem stars na GitHubu[53]), přestože je o rok mladší.

2.2.2 Akce (Action Extension)

Rozšíření Akce umožňuje manipulaci s obsahem, který se oproti rozšíření Sdílení (2.2.1) následně vrací do hostující aplikace. Jeho účelem je tedy pomoci uživateli vidět současný dokument jiným způsobem. Uživatelem vybraný obsah získává pouze pokud je hostující aplikací explicitně poskytnutý. Jako příklad si můžeme představit překlad webu přímo v prohlížeči.

Některé akce nám nabízí iOS rovnou — například kopírování do schránky, tisk, uložení souboru…

Stejně jako Sdílení je i rozšíření Akce obsluhováno pomocí rozhraní třídy `UIActivityViewController`, nabízejících použití některých standardních služeb (viz obrázek 2.2). Datové typy se také definují třídou `NSItemProvider`. Akce může zpracovávat jednotlivé datové typy různě. Je dobré uživatelům poskytnout obsluhu co nejvíce datových typů. Toto rozšíření může, ale nemusí mít `View`. Jedna aplikace může poskytovat i více akcí.

V prohlížeči Safari získává akce navíc přístup k DOM[54] a možnost použít JavaScript. Při zpracování pomocí JS je potřeba do globální proměnné

[48] <https://facebook.github.io/react-native/docs/share.html>
[49] <https://facebook.github.io/react-native/docs/actionsheetios.html>
[50] NPM - Node package manager <https://www.npmjs.com>
[51] <https://github.com/alinz/react-native-share-extension>
[52] <https://github.com/meedan/react-native-share-menu>
[53] 251 stars pro react-native-share-extension oproti 64 pro react-native-share-menu dne 29. 10. 2017
[54] <https://developer.mozilla.org/en-US/docs/Web/API/Document_Object_Model/Introduction>

`ExtensionPreprocessingJS` přiřadit objekt s metodami `run(arguments)` a `finalize(arguments)`.

Ikona akce se vytváří z černobílé šablony, nemůže obsahovat barvy.

Pro implementaci vlastní akce v RN se mi nepodařilo nalézt žádnou dostupnou knihovnu (4. 11. 2017). Zůstává tedy pouze možnost nativní implementace. Postup začlenění nativní implementace rozšíření Akce do RN aplikace je popsán v [19].

2.2.3 Today widget

Today widgety poskytují uživateli rychlý přístup k aktuálním informacím. Například nadcházející události v kalendáři, cena akcií, zbývající data a volné minuty v rámci tarifu telefonního operátora, adresa pro příjem bitcoinů nebo nejbližší zajímavost pozorovatelná na noční obloze. Zároveň mohou umožnit provést jednoduchou akci jako označení položky v seznamu úkolů za splněnou nebo i spuštění definovaného workflow[55] (například sdílení odhadovaného času příjezdu domu nebo spočítání adekvátní výše spropitného).

Widgety se zobrazují na obrazovce Dnes[56], jak je vidět na obrázku 2.3 vlevo. Zobrazení dnešního dne je možné přejetím doprava od levého okraje obrazovky plochy nebo uzamčené obrazovky[57]. Další možnost zobrazení widgetu je nad rychlými akcemi po přitlačení na ikonu aplikace na zařízeních podporujících 3D Touch[58] — na obrázku 2.3 vpravo.

Pokud aplikace zahrnuje více widgetů, je možné zvolit ten, který se má zobrazovat spolu s rychlými akcemi pomocí klíče `UIApplicationShortcutWidget`[59] v konfiguračním souboru *Info.plist*.

S widgety není možné interagovat pomocí klávesnice a tak konfigurace jejich obsahu a chování obvykle probíhá v hlavní aplikaci.

Pro vytváření a správu Today widgetů slouží framework *NotificationCenter*[60]. Pomocí jeho třídy `NCWidgetController` může rozšíření i hlavní aplikace definovat zda má widget nějaký obsah k zobrazení.

[55] S pomocí zdarma dostupné aplikace Workflow: <`https://itunes.apple.com/us/app/workflow/id915249334`>

[56] <`https://help.apple.com/iphone/11/?lang=cs#/iphb8f1bf206`>

[57] Povolit přístup k zobrazení Dnes z uzamčené obrazovky je možné v nastavení zařízení.

[58] <`https://developer.apple.com/ios/3d-touch/`>

[59] <`https://developer.apple.com/library/content/documentation/General/Reference/InfoPlistKeyReference/Articles/iPhoneOSKeys.html#//apple_ref/doc/uid/TP40009252-SW32`>

[60] <`https://developer.apple.com/documentation/notificationcenter`>

Obrázek 2.3: Ukázka Today widgetů. Vlevo na obrazovce Dnes, zdroj: [20]. Vpravo po přitlačení na ikonu aplikace, zdroj: [21].

Systém příležitostně obnovuje obsah widgetů na pozadí, pomocí protokolu `NCWidgetProviding`. Konkrétně se volá metoda:

```
(void)widgetPerformUpdateWithCompletionHandler:
  (void (^)(NCUpdateResult result))completionHandler;
```

Kód 2.1: Metoda pro aktualizaci obsahu widgetu.

která volá po aktualizaci obsahu widgetu blok `completionHandler`. Ten přijímá parametr definující, jestli aktualizace přinesla nějaká nová data:

- `NCUpdateResultNewData`: jsou dostupná nová data,
- `NCUpdateResultNoData`: od poslední aktualizace nejsou dostupná nová data a widget tedy není třeba překreslovat.

U Today widgetu je potřeba velmi dbát na optimalizaci výkonu. Limit na použitou paměť je jen 16 MB (viz 3.1.8), jak je vysvětleno v přednášce

[22] Conrada Kramera. Widgetů může běžet několik zároveň, proto nemohou spotřebovávat tolik systémových prostředků jako běžná aplikace. Náročnější operace je potřeba pustit na pozadí a co nejdříve. Důležité je také cachovat výsledky.

Widgety mohou být ze strany systému kdykoliv přerušeny a v případě nedostatku paměti jim nemusí být ani poskytnut čas na dokončení probíhajících operací. Když uživatel zavře notifikační centrum, všechny widgety se v tu chvíli přeruší. Ian Baird popisuje [23] jak využít techniku nazývanou *Task assertions*, abychom řekli systému, že potřebujeme nějakou operaci dokončit. Využívá se k tomu API `NSProcessInfo`[61].

Pro sdílení dat mezi widgetem a hlavní aplikací je potřeba je začlenit do sdílené skupiny (App group)[62]. Pak mohou jak widget, tak hlavní aplikace přistupovat ke společnému API `NSUserDefaults`[63].

Widget může umožnit otevření hlavní aplikace, která pro tento účel musí registrovat URL schémata. Také je možné používat systémová schémata URL (odkazy na web, do zpráv, mapy...). Odkaz se ve widgetu realizuje pomocí metody `openURL`[64] z třídy `NSExtensionContext`:

```
- (void)openURL:(NSURL *)URL
  completionHandler:(void (^)(BOOL success))completionHandler;
```

Kód 2.2: Metoda pro odkaz z widgetu.

První zveřejněné pokusy o implementaci widgetu pomocí RN jsou od Richarda Laie již z května 2016 v repozitáři *React-Native-Today-Widget*[65]. (Až na velikost písmen je název shodný s knihovnou *react-native-today-widget* 3.1, ale tento repozitář Lai nepublikoval jako knihovnu. Navíc na mnou vytvořenou knihovnu hned v úvodu README nyní odkazuje). Repozitář obsahuje kromě zdrojového kódu výčet potřebných kroků. Uvádí však, že se podařilo ukázku spustit pouze v simulátoru a ne na reálném zařízení.

V říjnu 2017 vytvořil Jesús Darío *react-native-android-widget-poc*[66] – pokus implementovat widget v RN pro platformu Android.

K aplikaci implementované v RN není problém přidat widget implemen-

[61] <https://developer.apple.com/documentation/foundation/nsprocessinfo>
[62] <https://developer.apple.com/library/content/documentation/IDEs/Conceptual/AppDistributionGuide/AddingCapabilities/AddingCapabilities.html#//apple_ref/doc/uid/TP40012582-CH26-SW61>
[63] <https://developer.apple.com/documentation/foundation/nsuserdefaults>
[64] <https://developer.apple.com/documentation/foundation/nsextensioncontext/1416791-openurl?language=objc>
[65] <https://github.com/rclai/React-Native-Today-Widget>
[66] <https://github.com/netbeast/react-native-android-widget-poc>

tovaný nativně. Pouze se tím připravíme o možnost sdílení JS kódu a další výhody RN (jako HMR[67] a dostupné JS knihovny). Na druhou stranu tím ušetříme systémové prostředky nutné pro běh JavaScriptu – především paměť, což je velmi zásadní kvůli jejímu nízkému limitu pro widget (viz 3.1.8).

2.2.4 iMessage

Aplikace iMessage jsou dostupné od iOS 10 díky frameworku Messages[68]. Jednak uživateli poskytují možnost doplnit konverzaci v aplikaci Zprávy o Nálepky (Stickers)[69] a Animoji[70] a jednak umožňují zprávy doplnit o interakci (například instantní přeložení do jiného jazyka, rozmazání obrázku dokud se uživatel neprokáže Touch ID, naskenování dokumentu...).

Distribuce aplikací iMessage je možná prostřednictvím zvláštního App Store[71] dostupného přímo v aplikaci Zprávy.

Pro vytvoření vlastní kolekce Nálepek není třeba nic programovat, stačí nahrát mediální soubory[72] pomocí XCode[73]. Přínos implementace pomocí RN by byl tedy pro tvorbu statických nálepek relativně malý.

Postup tvorby interaktivní Messages aplikace pomocí RN dobře popsal Hugo Dozois-C. v [24]. Jde v zásadě o stejný postup, který jsem zautomatizoval pro widgety v knihovně *react-native-today-widget* (viz 3.1).

Messages aplikace mohou využívat to co běžné aplikace — například nákupy v aplikacích, apple pay, fotoaparát. Mohou generovat nálepky nebo i ostatní obsah, který aplikace Zprávy podporuje (fotografie, videa, text, odkazy...). Mohou odkázat do hlavní aplikace, pokud ji má příjemce nainstalovanou nebo na ni zobrazit odkaz do App Store (tzv. Inline App Attribution).

Messages aplikace mohou běžet jen na iOS 10 nebo novějším, ale obsah z nich je možné přijmout i na watchOS a macOS. Z watchOS 3 je možné posílat Nálepky naposledy použité v iOS.

Na rozdíl od všech ostatních rozšíření může fungovat Messages aplikace i bez hlavní aplikace. Je ji možné samostatně instalovat z Messages App Store a zobrazí se v aplikaci Zprávy v liště (App Drawer), nezobrazuje se na ploše.

[67] <https://facebook.github.io/react-native/blog/2016/03/24/introducing-hot-reloading.html>
[68] <https://developer.apple.com/documentation/messages>
[69] <https://developer.apple.com/stickers/>
[70] <https://support.apple.com/en-us/HT208190> dostupné pouze s iPhone X
[71] <https://developer.apple.com/imessage/>
[72] Nejlépe ve formátu PNG kvůli průhlednosti, GIF může způsobit nežádoucí artefakty.
[73] Postup je popsaný zde: <https://developer.apple.com/stickers/>

Podrobnější informace o možnostech a nativní implementaci iMessage aplikací můžete nalézt v prezentacích z WWDC 2016 [25] a [26].

2.2.5 Siri

Pro integraci hlasové asistentky Siri poskytuje Apple dva body rozšíření [27]:

- `Intents` pro přijmutí uživatelova požadavku a provedení akce v aplikaci,
- `Intents UI` pro zobrazení odpovědi poté, co je akce dokončena (toto není povinné).

K Siri přistupujeme přes *SiriKit*[74]. Použití SiriKit je představeno v [28].

Proces použití Siri vypadá zjednodušeně takto:

1. Siri extrahuje z řeči uživatele záměr neboli `Intent`.
2. Naše rozšíření zpracuje záměr a provede v rámci aplikace požadovanou akci.
3. Siri zobrazí uživateli případnou zpracovanou odpověď.

V aplikaci definujeme dva druhy slovníku, které pomohou Siri porozumět specifickým slovům a frázím:

- Slovník aplikace (`App vocabulary`) obsahuje slova a fráze, které jsou součástí hlavní aplikace (např. název pokrmu pro aplikaci s kuchařskými recepty).
- Uživatelův slovník (`User vocabulary`) slova a fráze důležité pouze pro konkrétního uživatele (např. seznam kontaktů).

Slovník aplikace je definován v `plist` konfiguračním souboru. Je lokalizovatelný a obsahuje i příklady použití, fráze a výslovnost.

Uživatelův slovník je poskytován aplikací za běhu. Je uložen do `OrderedSet`, seřazený podle důležitosti. Seznam kontaktů je dostupný automaticky všem aplikacím, kterým uživatel schválí příslušné oprávnění. Aktualizace tohoto slovníku je třeba posílat Siri okamžitě při změně, jinak by to mohlo vést k velmi špatnému uživatelskému zážitku (UX). Když by například uživatel

[74] <https://developer.apple.com/sirikit/>

smazal kontakt a zeptal se na něj Siri a ona by ho ještě znala, uživatel by jistě nebyl spokojený.

O veškerou komunikaci s uživatelem se stará Siri (které jako programátor pouze pomáhám poskytnutím slovníků). Programátor aplikace se stará jen o vlastní funkcionalitu. Případné gafické rozhraní odpovědi je možné vytvořit vlastní nebo použít výchozí systémové.

`Intent` (záměr, akce) může mít 0 až mnoho parametrů a je zařazený do nějaké domény. Siri rozlišuje domény například pro zprávy, VoIP hovory, platby, cvičení, jízdu autem, hledání fotografie... Životní cyklus Intentu se skládá ze tří fází, které implementuje rozšíření Intents:

- *resolve*: získání vstupů, vyžaduje někdy několik doplňujících dotazů od Siri směrem k uživateli,
- *confirm*: potvrzení uživatelem, že ho Siri správně pochopila,
- *handle*: zavolání akce.

Výsledkem akce je `IntentResponse`, který obsahuje výsledek, kód odpovědi (úspěch/neúspěch) a instanci třídy `NSUserActivity` pro možnost spuštění hlavní aplikace.

Vývojář může povolit některé Intenty obsloužit jen při odemknutém zařízení. To se definuje v *Info.plist*. Pro dodatečné přihlášení slouží framework *LocalAuthentication*[75]. Dalším bezpečnostním prvkem je, že uživatel ke spuštění naší aplikace musí vyslovit její celé jméno (`CFBundleDisplayName`).

Další zajímavou možností SiriKit je použití `NSUserActivity`[76] pro poskytnutí kontextu systému, který může nabízet data z mé aplikace ve vhodném místě (např. v nápovědě nebo mapách).

Postup začlenění Siri do vlastní aplikace nativní cestou naleznete v [29].

Pro implementaci s RN existuje knihovna *react-native-share-actions*[77], kde je ovšem pouze 5 inicializačních revizí z října 2016 a od té doby žádný vývoj. Nepodařilo se mi dohledat žádné další pokusy o implementaci Siri do RN aplikace. Nic by tomu však nemělo principiálně bránit.

[75] <https://developer.apple.com/documentation/localauthentication/>
[76] <https://developer.apple.com/documentation/foundation/nsuseractivity>
[77] <https://github.com/moorinteractive/react-native-share-actions>

2.2.6 Photo Editing

Rozšíření Photo Editing poskytuje možnost přidat vlastní možnosti editace fotek a videí z aplikace Fotky. Spouští se přímo z této aplikace, nikoliv z hlavní aplikace. Příklad podoby rozhraní je na obrázku 2.4.

Obrázek 2.4: Ukázka možné podoby rozšíření Photo Editing, zdroj: [21].

Základní popis tohoto rozšíření naleznete v [14]. Aplikace Fotky ukládá současnou i originální podobu fotografie a informace o provedených změnách z originálu do současné podoby. Pokud je rozšíření schopno číst data o změnách[78], dostane originální podobu i tyto údaje o postupných změnách a může tak umožnit uživateli zasahovat do již provedených změn. Pokud rozšíření řekne, že data o změnách číst neumí, dostane od aplikace Fotky jen současnou verzi. Jiná je situace u videí. Aplikace Fotky neukládá současnou verzi videa, a tak pokud rozšíření neumí přečíst data o změnách, může jedině přepsat již provedené změny.

K vytvoření Photo Editing rozšíření je třeba implementovat metody defi-

[78] a deklaruje to metodou *canHandleAdjustmentData:*

nované protokolem `PHContentEditingController`[79]. Také je důležité pamatovat na paměťový limit 120 MB (tabulka 3.1).

Nenalezl jsem zatím žádné pokusy o implementaci rozšíření Photo Editing pomocí RN. Zůstává tedy opět možnost přidat k RN aplikaci rozšíření implementované nativně.

2.3 Tvorba Today widgetu

V seznamu požadavků na funkcionalitu React Native[80] se implementace rozšíření Today widget objevilo 4. května 2016, od té doby zatím nasbíral tento návrh 24 hlasů (10. 12. 2017). Mohlo by se zdát, že to není mnoho, ale aktuálně nejvyšší počet hlasů pro jeden návrh je 514 a aplikační rozšíření je jistě z hlediska RN spíše okrajové téma.

Zároveň se zadáním tohoto požadavku zveřejnil Richard Lai repozitář *React-Native-Today-Widget*[81] s pokusem o implementaci widgetu pomocí RN. Popisoval v devíti krocích jak přidat widget k RN aplikaci. Nepodařilo se mu však widget zprovoznit i na zařízení, fungoval pouze v simulátoru. Řešení tohoto problému popsal až 31. března 2017 Garrett Bjerkhoel[82].

2.3.1 Současný stav

V současné chvíli je nejschůdnější variantou přidání Today widgetu do RN aplikace jeho nativní implementace. Tím se však připravíme o možnost sdílení kódu z hlavní aplikace[83]. A ještě větší komplikací může být, pokud s nativní implementací nemáme zkušenosti a neovládáme Objective-C či Swift.

Pro tvorbu nativních widgetů existuje několik kvalitních tutorialů jak pro jazyk Swift: [30], [31], tak Objective-C: [32], [33].

Po zkombinování instrukcí pro inicializaci widgetu s postupem pro integraci RN do existujících aplikací [8][84] je možné do widgetu přidat `View` vykreslující RN komponentu. Přijdeme tím ovšem o Dev Menu, ze kterého je běžně možné v RN aplikacích spouštět například obnovení aplikace při změně kódu[85]. Postup přidání widgetu do RN aplikace je popsaný v [34].

[79] <`https://developer.apple.com/documentation/photosui/phcontenteditingcontroller?language=objc`>
[80] <`https://react-native.canny.io/feature-requests/p/ios-today-widget-extension`>
[81] <`https://github.com/rclai/React-Native-Today-Widget`>
[82] <`https://github.com/rclai/React-Native-Today-Widget/issues/1`>
[83] Pokud je celá aplikace v RN, nativní části je možné sdílet pomocí Shared Containers.
[84] <`https://facebook.github.io/react-native/docs/integration-with-existing-apps.html`>
[85] Live Reload a Hot Reload.

Protože je princip integrace RN pro některá aplikační rozšíření v zásadě stejný, stojí za to se podívat i na pokusy o integraci jiných rozšíření než pouze Today widget. Dobře popsaný postup pro Share Extension (2.2.1) můžete nalézt v [19]. Pro Share Extension existuje i npm knihovna *react-native-share-extension*[86], která ovšem vyžaduje manuální linkování v XCode.

Existuje také projekt, který se pokoušel implementovat pomocí RN widget na platformě Android. Jedná se o *react-native-android-widget-poc*[87].

2.3.2 Varianty řešení

Nabízí se více způsobů jak tvorbu widgetů pomocí RN usnadnit. Zvažoval jsem následující možnosti:

- *Integrace do React Native*
  Bylo by možné vytvořit fork[88] RN a přidat do něj podporu pro aplikační rozšíření podobným způsobem, jako je už teď v RN dostupná implementace aplikací pro Apple TV. Toto řešení by bylo implementačně nejsnazší, stačí přidat Today widget podle [34].

  Velkou nevýhodou však je náročnost údržby při velmi rychlém tempu vývoje RN. Programátoři by pravděpodobně nebyli příliš ochotní používat místo oficiální distribuce RN něčí fork, zvlášť pokud by na něm nebyly aplikovány všechny změny z hlavního repozitáře.

- *Knihovna*
  Další variantou je vytvoření NPM knihovny. Výhodou je možnost použít automatického linkování pomocí knihovny RNPM[89], která je integrována do RN. NPM knihovny také umožňují zavolání skriptu po instalaci a případné akce potřebné pro počáteční konfiguraci widgetu je tak možné automaticky spouštět.

  Za největší výhodu považuji, že jsou vývojáři zvyklí hledat potřebné knihovny v registru NPM[90].

- *Nativní modul*
  Pouhé implementování nativního modulu není možné, je potřeba do nativní části aplikace přidat nový target[91] a ten řádně nakonfigurovat. To nelze provést přímo z aplikace a nativní modul by tedy pro přidání widgetu nestačil.

[86] <https://github.com/alinz/react-native-share-extension>
[87] <https://github.com/netbeast/react-native-android-widget-poc>
[88] Fork je kopie repozitáře využitelná jako počátek vlastního projektu [35].
[89] <https://github.com/rnpm/rnpm>
[90] <https://www.npmjs.com>
[91] <https://developer.apple.com/library/content/featuredarticles/XcodeConcepts/Concept-Targets.html>

- *Skript*
  Pro přidání widgetu do aplikace je potřeba provést několik kroků, které lze zautomatizovat. Bylo by tedy možné napsat skript, jehož spuštění by do RN projektu přidalo nový widget. Toto řešení je možné zkombinovat s variantou 2.3.2, která má tu výhodu, že usnadní počáteční připojení widgetu do hlavní aplikace pomocí RNPM.

2.3.3 Zásady pro uživatelské rozhraní

Ať už k implementaci používáme JavaScript nebo nativní kód, je potřeba dodržovat doporučení pro uživatelské rozhraní na dané platformě. Důležité je to pro to, aby naše aplikace odpovídala tomu, na co jsou uživatelé zvyklí, a aby tak neměli problém se v aplikace orientovat a používat ji.

Nejzákladnější doporučení pro uživatelské rozhraní na iOS nám poskytují *Human Interface Guidelies*[92] přímo od společnosti Apple. Tyto zásady je nezbytné pročíst a porozumět jim. Krátce shrnuto [21]:

- *Přehlednost*: Informace musí být stručné a případná interakce velmi jednoduchá — pokud je to možné, tak proveditelná jedním dotekem. Panning a scrolling[93] není ve widgetech možný.
- *Rychlost*: Obsah se musí zobrazit velmi rychle, protože uživatelé nevěnují pohledu na widgety mnoho času.
- *Odsazování*: Obsah by neměl zasahovat až ke stranám widgetu. Ideálně by mě být odsazen tak, aby byl vlevo zarovnán se středem ikony. Maximální počet ikon nebo tlačítek na řádek by měl být 4.
- *Přizpůsobivost*: Šířka widgetu závisí na zařízení a aktuální orientaci. Widget může být roztažitelný na výšku. I v takovém případě by ale v základním zmenšeném stavu měl zobrazovat ucelené informace. Po roztažení může zobrazit informace obohacující ty základní. Neměl by přesáhnout výšku obrazovky.
- *Pozadí*: Ponechte výchozí průhledné pozadí. Nikdy nepoužívejte na pozadí widgetu fotografii, mohla by kolidovat s pozadím na zamčené obrazovce nebo na ploše.
- *Písmo*: Použijte systémový font a černou nebo tmavě šedou barvu.

[92] <`https://developer.apple.com/ios/human-interface-guidelines/extensions/widgets/`>

[93] Panning a scrolling jsou dotyková gesta sloužící pro posouvání obsahu na obrazovce. Rozdíl je vysvětlen zde: <`https://stackoverflow.com/a/9899299/7414186`>

- *Odkaz do hlavní aplikace*: Umožněte otevření hlavní aplikace z widgetu. Jako odkaz by měl sloužit obsah widgetu — nepřidávejte tlačítko navíc, které by zabíralo zbytečně místo.
- *Jméno*: Použijte jméno vaší hlavní aplikace.
- *Autentifikace*: Pokud Váš widget poskytuje něco navíc v případě, že je uživatel přihlášený, dejte o tom odhlášeným uživatelům vědět například zobrazením zprávy.
- *Rychlá akce*: Pokud Vaše aplikace poskytuje více widgetů, vyberte jeden pro zobrazení po přitlačení[94] na ikonu aplikace.

2.4 Knihovna s nativním kódem

Pro tvorbu knihovny pro RN s nativním kódem není dostupná žádná oficiální dokumentace. Existuje ale několik blogů, ze kterých je možné se v postupu inspirovat. Jedním takovým je třídílný tutorial pro tvorbu knihovny pro RN od Carlose Lópeze [36].

2.4.1 9-project-layout

Ben Wixen nenašel žádný dobrý návod na strukturování nativního kódu v knihovně pro RN a tak přišel s vlastním návrhem, který nazval *9-project-layout*[95] [37].

Devět vrstev značí devět složek. Hlavní složky komponent, složky pro testy a složky s ukázkami použití, rozděluje podle tří jazyků. Jde o JavaScript pro sdílený kód, Objective-C pro iOS a Javu pro Android platformu.

2.4.2 Swift

Pro implementaci RN nepoužil Facebook jazyk Swift (jak je patrné z obrázku 2.1). Důvod je podle mě zřejmý — Swift byl představen jen půl roku před RN [9]. Swift je ale lze s jazykem Objective-C kombinovat i v jednom projektu [38]. Díky tomu je možné psát nové nativní moduly pro RN i s použitím Swiftu [8]. Vyžaduje to jen trochu konfigurace navíc, protože Swift nepodporuje makra, která RN využívá pro komunikaci s JS[96].

[94] Zařízení musí podporovat 3D Touch: <https://developer.apple.com/ios/3d-touch/>
[95] <https://github.com/benwixen/9-project-layout>
[96] <https://facebook.github.io/react-native/docs/native-modules-ios.html#exporting-swift>

2.5 Návrh ukázkové aplikace

Pro ukázkový widget jsem potřeboval najít dostatečně jednoduché zadání, ale aby bylo zároveň možné demonstrovat co nejvíce možností knihovny *react-native-today-widget*. Domluvil jsem se s *HoppyGo*[97] na doplnění Today widgetu do jejich aplikace, která je napsána v RN a je již přes půl roku dostupná na App Store[98] i Google Play[99]. HoppyGo je platforma pro carsharing[100], kterou vyvíjí *ŠKODA AUTO DigiLab*[101] a funguje především ve velkých městech v ČR.

2.5.1 Hlavní aplikace – *HoppyGo*

HoppyGo je služba pro sdílení aut, která propojuje vlastníky aut a řidiče, kteří by si rádi auto půjčili [39]. Aplikace umožňuje majitelům aut zadat nabídku svého vozu, automaticky sjednat pojištění při půjčení cizí osobě a spravovat výpůjčky. Řidičům, kteří si chtějí auto půjčit, umožňuje vyhledání vhodného vozu v jejich okolí, zadání objednávky a vyřízení předávání vozu. Ukázku rozhraní aplikace můžete vidět na obrázku 2.5.

2.5.2 Widget – *HoppyGo: Auta v mém okolí*

Protože jde především o demonstraci použitelnosti knihovny *react-native-today-widget*, nebyla provedena žádná prvotní studie s uživateli. Jediné zadání bylo umožnit uživateli vidět auta dostupná k zapůjčení v jeho okolí.

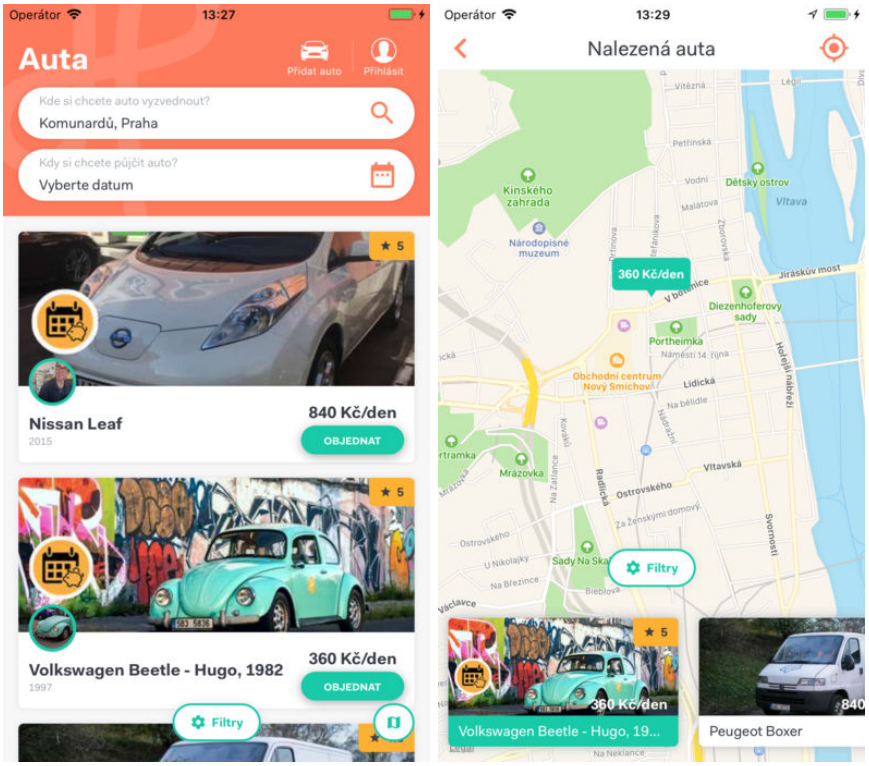
Systém má dvě hlavní role uživatelů:

- *Majitel*: Nabízí své auto k zapůjčení.
- *Řidič*: Chce si vypůjčit auto.

Na základě zadání jsem formuloval dvě User stories[102]:

- *US1*: Jako řidič chci mít rychlý přehled o tom, jaká auta jsou k zapůjčení v mém okolí, abych získal představu o dostupné nabídce a cenách.
- *US2*: Jako řidič chci mít možnost vidět podrobné informace o autech dostupných v mém okolí, abych zjistil, jestli mám o nabízené auto zájem.

[97] <https://hoppygo.cz>
[98] <https://itunes.apple.com/cz/app/HoppyGo/id1195045971>
[99] <https://play.google.com/store/apps/details?id=com.creativedock.hoppycar>
[100] <https://cs.wikipedia.org/wiki/Sdílení_aut>
[101] <https://skodaautodigilab.com>
[102] <https://soch.cz/blog/management/agile/proc-piseme-user-story/>



Obrázek 2.5: Ukázka aplikace HoppyGo. Vlevo je vidět obrazovka pro vyhledávání aut dostupných podle navolených filtrů, vpravo přehled nalezených aut na mapě. Zdroj: App Store

Dále jsem dostal návrh podoby uživatelského rozhraní, viz obrázek 2.6. Návrh počítá s tím, že by bylo možné kliknutím na název vozu rozbalit podrobnější informace o vozidle. Toto rozbalování však nebylo nutně vyžadováno kvůli hrozící nutnosti šetřit paměťové nároky widgetu.

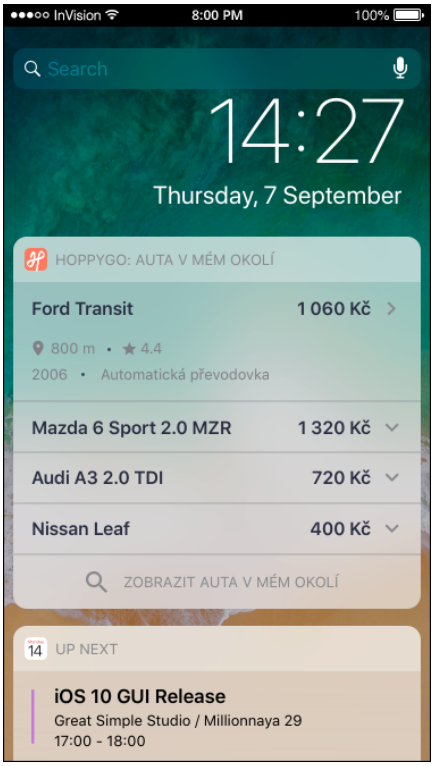
2.5.3 API

HoppyGo poskytuje otevřené REST API přístupné bez autorizace.

URL: `/api/car/browser/`

Parametry použité v URL (query params):

+ `pick_up_distance (nepovinný) [decimal,decimal,decimal]`
  zeměpisná šířka, zeměpisná výška a maximální vzdálenost v metrech



Obrázek 2.6: Poskytnutý návrh uživatelského rozhraní.

+ `available_between [Datetime(ISO-8601),Datetime(ISO-8601)]`
  časové rozpětí dostupnosti

Parametry použité v hlavičce (request header params):

+ `X-Base`
  maximální požadovaný počet výsledků
+ `X-Fields`
  výběr polí v odpovědi

Všechny použité parametry by měly být snadno konfigurovatelné, aby je bylo možné vyladit později až na základě uživatelského testování.

2.5.4 Odkaz do aplikace (deep link)

Aplikace HoppyGo má veřejné URL pro odkaz do aplikace, umožňující přistoupit přímo k detailu jednotlivého auta:

- `hoppygo://car/{carID}`

2.5.5 Nativní implementace

Today widget by bylo možné k aplikaci HoppyGo doplnit i nativně implementovaný v jazyce Objective-C nebo Swift. Widget má totiž vlastní *target*[103], a integrace s hlavní aplikací v RN je tak bezproblémová. Největší komplikací může být nutnost ovládat nativní vývoj.

Dále není možné při kombinaci hlavní aplikace v RN a widgetu v Objective-C/Swift sdílet části kódu, ale to nebylo v tomto případě vhodné ani při implementaci widgetu v RN. Ze smluvních důvodů mi nebyl zpřístupněn kód hlavní aplikace a widget jsem vyvíjel zcela zvlášť s výchozí hlavní aplikací z inicializace RN projektu.

Nativní vývoj by ušetřil paměťový prostor, který spotřebuje RN na běh JSCore a RN Bridge. Omezení dostupné paměti pro widget považuji za největší nevýhodu implementace v RN.

Největší výhodou oproti nativní implementaci je používání sjednoceného vývojového prostředí. Sjednocené vývojové prostředí znamená nejen použití jazyku JavaScript, ale také dostupnost knihoven, nástrojů pro statickou analýzu (např. ESLint[104], Flow[105]) a dalších nástrojů pro usnadnění vývoje jako Webpack[106] nebo gulp[107] a další.

Bohužel zatím v *react-native-today-widget* neimplementuji widgety i pro Android[108] a výhoda multiplatformnosti tak zatím odpadá. Zůstává ale možnost sdílení částí kódu mezi widgetem a hlavní aplikací. To může být užitečné zejména pro různé pomocné funkce, případně aplikační logiku a nebo správu stavu (např. Redux[109]).

Další velkou výhodou jsou možnosti instantního obnovování při změnách v kódu. I ve widgetu je možné použít jak *Live Reload*[110], tak *Hot reloading*[111]. Live Reload při každé změně v kódu restartuje aplikaci, abychom viděli novou

[103] <https://developer.apple.com/library/content/featuredarticles/XcodeConcepts/Concept-Targets.html#//apple_ref/doc/uid/TP40009328-CH4>
[104] <https://eslint.org>
[105] <https://flowtype.org>
[106] <https://webpack.github.io>
[107] <https://gulpjs.com>
[108] Widgety na platformě Android se zobrazují přímo na domácí obrazovce a jejich použití je tak trochu odlišné od těch na iOS [40]
[109] <https://redux.js.org>
[110] <https://facebook.github.io/react-native/docs/debugging.html#reloading-javascript>
[111] <https://facebook.github.io/react-native/blog/2016/03/24/introducing-hot-reloading.html>

verzi. Hot reloading nerestartuje celou aplikaci, ale přímo aktualizuje pouze změněnou komponentu, pokud je to možné. Díky tomu neztratíme aktuální stav. To je velmi užitečné především při úpravách uživatelského rozhraní.

Bod rozšíření	**Typická funkcionalita rozšíření**
Action	Zobrazení nebo manipulace s obsahem pocházejícím z hostující aplikace.
Audio Unit	Generování audio streamu odesílaného hostující aplikaci nebo modifikování audio streamu z hostující aplikace a jeho opětovné zaslání zpět hostující aplikaci.
Call Directory	Identifikace a blokování příchozích volajících podle jejich telefonního čísla.
Content Blocker	Indikace pro WebKit, že vaše aplikace blokující obsah aktualizovala svá pravidla. (Toto rozšíření nemá žádné uživatelské rozhraní.)
Custom Keyboard	Náhrada systémové klávesnice vlastní klávesnicí pro použití ve všech aplikacích.
Document Provider	Poskytuje přístup a spravuje repozitář souborů.
iMessage	Interakce s aplikací Zprávy.
Intents	Obsluhování úkonů souvisejících s podporou integrace Siri do vaší aplikace.
Intents UI	Přizpůsobení rozhraní Siri nebo map po obsloužení úkonu souvisejícího s podporou integrace Siri do vaší aplikace.
Photo Editing	Editace fotky nebo videa prostřednictvím aplikace Fotky.
Share	Odeslání na webovou stránku pro sdílení obsahu nebo sdílení obsahu s ostatními.
Spotlight Index	Indexace obsahu vaší aplikace zatímco není v provozu.
Sticker Pack	Poskytnutí souboru nálepek, které uživatel může použít v aplikaci Zprávy.
Today	Získání rychlé aktualizace nebo vykonání krátkého úkonu v notifikačním centru.

Tabulka 2.1: Přehled dostupných bodů rozšíření pro platformu iOS. Zdroj: [14]

Kapitola 3

Realizace

3.1 Knihovna *react-native-today-widget*

Jedním z cílů této práce, jak je popsáno v kapitole 1, je vytvoření open source knihovny, která by programátorům znalým RN usnadnila přidání Today widgetu do jejich RN aplikací.

V této kapitole popíšu jednotlivé kroky k realizaci nové open source knihovny pro RN. Všechny kroky, které se nakonec promítly v kódu knihovny, jsou zdokumentovány v Git repozitáři. Přehled všech jednotlivých revizí je dostupný v GitHubu[112].

3.1.1 Volba jména

Pokud vytváříte knihovnu pro použití s RN, je dobré, aby její název začínal *react-native-*. Zvlášť pokud nejde o čistě JS knihovnu, ale je potřeba knihovnu linkovat. Nástroj RNPM totiž zkouší automaticky linkovat knihovny s touto předponou v názvu při zavolání příkazu `react-native link`, pokud jsou uvedené v závislostech vašeho projektu v souboru `package.json`.

Pro knihovnu určenou k přidání Today widgetu se nabízel jako nejjednodušší název *react-native-today-widget* (RNTW), který byl naštěstí v registru NPM ještě volný.

3.1.2 Inicializace

Protože jsem vytvářel v podstatě šablonu pro novou RN aplikaci[113], použil jsem inicializaci RN projektu pomocí příkazu `react-native init`. Tuto inicializaci jsem provedl 20. 5. 2017, kdy byl RN ve verzi `0.44.0`.

[112] <`https://github.com/matejkriz/react-native-today-widget/commits/master`>
[113] Today widget je samostatnou aplikací, jak je popsáno v 2.2.3.

Pro tuto inicializaci by se nehodil projekt Create React Native App[114] (CRNA). CRNA, uváděný v dokumentaci RN [8] jako nejsnazší varianta vývoje RN aplikace, neumožňuje přímo editovat nativní část, což pro tvorbu widgetu potřebujeme.

Do projektu jsem rovnou přidal ukázkovou aplikaci pro demonstraci použití knihovny. Základní ukázkovou aplikaci jsem taktéž vygeneroval pomocí `react-native init`.

Po základní inicializaci RN aplikace jsou knihovny `react` a `react-native` uvedené jako závislosti v souboru `package.json`. Protože RNTW má sloužit jako knihovna, přesunul jsem tyto dvě závislosti do takzvaných *peer dependencies*[115]. To znamená, že RNTW nepřidává do `node_modules` vlastní verze těchto závislostí, ale očekává, že již budou dostupné. Pro peer dependencies je zároveň možné definovat rozsah verzí, se kterými by měla být knihovna kompatibilní.

Dále jsem smazal celou složku pro platformu Android, protože zatím knihovnu implementuji jen pro platformu iOS. Do budoucna by bylo možné přidat i implementaci pro Android. Smazal jsem také další přebytečné části jako XCode targets pro tvOS.

3.1.3 Vývojové prostředí

Pro usnadnění vývoje kvalitního kódu jsem přidal Flow[116] pro typovou kontrolu a ESLint[117] pro statickou kontrolu kvality kódu. Pro ESLint jsem použil poměrně velmi striktní konfiguraci od Airbnb[118]. Dále jsem přidal nástroj Prettier[119] pro automatické konzistentní formátování kódu.

3.1.4 Bundle ID – postinstall skript

Today widget se musí distribuovat spolu s hlavní aplikaci a podmínkou je, aby jeho Bundle ID uložené v souboru `Info.plist` u klíče `CFBundleIdentifier` začínalo celým ID hlavní aplikace.

Aby to nemusel programátor nastavovat pokaždé manuálně, vytvořil jsem skript volaný během instalace knihovny RNTW do hlavní aplikace. Volání během instalace je zajištěno pomocí NPM skriptu `postinstall`[120]. Díky tomu

[114] <https://github.com/react-community/create-react-native-app>
[115] <https://blog.domenic.me/peer-dependencies/>
[116] <https://flow.org>
[117] <https://eslint.org>
[118] <https://www.npmjs.com/package/eslint-config-airbnb>
[119] <https://prettier.io>
[120] <https://docs.npmjs.com/misc/scripts>

se Bundle ID nastaví kdykoliv programátor zavolá kterýkoliv z příkazů uvedených v 3.1:

```
$ npm install --save react-native-today-widget

$ yarn add react-native-today-widget

$ react-native install react-native-today-widget

# nebo pokud už máte závislost uvedenou v souboru package.json:
$ npm install

$ yarn
```

Kód 3.1: Možné varianty příkazů pro instalaci knihovny RNTW.

Bundle ID hlavní aplikace jsem nejprve četl pomocí Bash skriptu ne příliš robustním způsobem:

```
1  #!/bin/bash
2
3  # ...
4
5  PACKAGE_NAME=$(sed -nE 's/^\s*"name": "(.*?)",$/\1/p'
       $PROJECT_PATH'package.json')
6  PLIST_PATH_PARENT=$PROJECT_PATH'ios/'${PACKAGE_NAME}'/Info.plist'
7
8  BUNDLE_IDENTIFIER=$(defaults read $PLIST_PATH_PARENT
       CFBundleIdentifier)
9  BUNDLE_ID=${BUNDLE_IDENTIFIER/\$\(PRODUCT_NAME:rfc1034identifier
       \)/$PACKAGE_NAME}
10
11 # ...
```

Kód 3.2: Ukázka části první verze skriptu pro automatické nastavení Bundle ID.

Předpokládal jsem výchozí cestu k souboru `Info.plist`, shodnou s názvem balíčku v souboru `package.json` viz kód 3.2, řádek 5. Výchozí Bundle ID obsahuje proměnnou `PRODUCT_NAME` převedenou do formátu platného pro identifikátor pomocí makra `rfc1034identifier`. Tato proměnná je nahrazena názvem balíčku na řádku 9. Slabinou tohoto řešení byla především závislost na názvu balíčku v souboru `package.json` a nutnost, aby tento název byl ve validním formátu.

Další verzi tohoto skriptu jsem napsal v Ruby. Prvním vylepšením je, že již nepředpokládám `PRODUCT_NAME` shodný s názvem balíčku v `package.json`, ale čtu jeho hodnotu přímo z XCode projektu. Navíc ho převádím do formátu shodného s `rfc1034identifier`. Na čtení a zápis hodnot do nativního projektu v Ruby jsem použil knihovnu Xcodeproj[121], která je využívaná mimo

[121] <https://github.com/CocoaPods/Xcodeproj>

jiné v projektu CocoaPods[122].

Současná podoba skriptu je dostupná v GitHub repozitáři[123]. Skript je možné spustit i manuálně, díky následující konfiguraci v `package.json`:

```
"bin": {
  "bundle-id": "./bin/setBundleId.rb",
  "embed-extension": "./bin/embedExtension.rb"
}
```

Kód 3.3: Nastavení přidání skriptů do *node_modules/.bin* složky v souboru *package.json*.

3.1.5 Začlenění do hlavního projektu – postlink skript

Začlenění nativního projektu rozšíření do hlavní aplikace je částečně automatické pomocí již zmiňovaného RNPM (viz 2.3.2). Díky tomuto nástroji, který je již součástí RN je projekt rozšíření (`RNTodayWidgetExtension.xcodeproj`) příkazem `react-native link` automaticky přidán jako knihovna do hlavní aplikace. To je pro některé nativní knihovny dostatečné k plnému využití, ale v případě aplikačních rozšíření potřebujeme ještě provést několik dalších kroků. Tyto kroky v RNTW provádí skript, který se automaticky volá po ukončení příkazu `react-native link`. Volání je zajištěno následující konfigurací v souboru `package.json`:

```
"rnpm": {
  "commands": {
    "postlink": "./node_modules/.bin/embed-extension"
  }
}
```

Kód 3.4: Nastavení automatického volání skriptu v souboru *package.json*.

Skript `embedExtension.rb`[124] opět využívá XCodeproj, stejně jako skript pro nastavení Bundle ID (3.1.4).

Na jednotlivé kroky, které musí skript provést pro zcela automatické začlenění do projektu jsem přicházel analyticky. Kroky jsem prováděl manuálně pomocí XCode a sledoval změny v generovaných souborech. Následně jsem hledal způsob jak tyto změny provést pomocí skriptu automaticky. Jak postupovat manuálně jsem čerpal především z [8][125] a [41].

[122] <https://cocoapods.org>
[123] <https://github.com/matejkriz/react-native-today-widget/blob/master/bin/setBundleId.rb>
[124] <https://github.com/matejkriz/react-native-today-widget/blob/master/bin/embedExtension.rb>
[125] <http://facebook.github.io/react-native/docs/linking-libraries-ios.html>

Skript nejprve načte jak hlavní XCode projekt, tak projekt rozšíření. Z rozšíření načte též soubor `TodayWidgetExtension.appex`, což je binární soubor představující produkt rozšíření. Ten se začlení do hlavního projektu přidáním `PBXContainerItemProxy`[126], dále přidáním reference `PBXReferenceProxy`[127] a konečně přidáním této reference do skupiny `PBXGroup`[128]. Uvedené změny provádí část skriptu v následující ukázce kódu:

```
#!/usr/bin/env ruby
require 'rubygems'
require 'xcodeproj'

# Přidání PBXContainerItemProxy pro appex objekt
container_item = project.new(
  Xcodeproj::Project::Object::PBXContainerItemProxy
)
container_item.container_portal = extension_ref.uuid
container_item.proxy_type = "2"
container_item.remote_info = "TodayWidgetExtension"
container_item.remote_global_id_string = appex.uuid

# Přidání proxy PBXReferenceProxy do hlavního projektu
reference_proxy = project.new(
  Xcodeproj::Project::Object::PBXReferenceProxy
)
reference_proxy.file_type = appex.explicit_file_type
reference_proxy.path = appex.path
reference_proxy.remote_ref = container_item
reference_proxy.source_tree = appex.source_tree

# Přidání proxy do PBXGroup
products_group = project.new(
  Xcodeproj::Project::Object::PBXGroup
)
products_group.name = 'Products'
products_group.source_tree = "<group>"
products_group<<reference_proxy

project_reference = Xcodeproj::Project::ObjectDictionary.new(
  project.root_object.references_by_keys_attributes.first,
      project.root_object
)
project_reference['ProjectRef'] = extension_ref
project_reference['ProductGroup'] = products_group
project.root_object.project_references << project_reference
```

Kód 3.5: Ukázka části skriptu *embedExtension.rb* pro začlenění rozšíření do hlavního projektu.

[126] <http://www.rubydoc.info/github/CocoaPods/Xcodeproj/Xcodeproj/Project/Object/PBXContainerItemProxy>
[127] <http://www.rubydoc.info/gems/xcodeproj/Xcodeproj/Project/Object/PBXReferenceProxy>
[128] <http://www.rubydoc.info/gems/xcodeproj/Xcodeproj/Project/Object/PBXGroup>

Další krok skriptu přidává cíl rozšíření (target)[129] jako závislost pro cíl hlavní aplikace. To je důležité, aby se při sestavování hlavní aplikace sestavilo i rozšíření.

Předposledním krokem je přidání kopírovací fáze sestavování, která zkopíruje soubor `TodayWidgetExtension.appex` a posledním uložení provedených změn v projektu.

3.1.6 Publikace

Zdrojové kódy knihovny jsou veřejně dostupné na serveru GitHub[130]. Zde mohou ostatní programátoři nahlašovat nalezené problémy jako *GitHub issues*[131]. Protože se ukázalo jako velmi důležité pro řešení těchto problémů znát konfiguraci nahlašovatele, vytvořil jsem jednoduchou šablonu, která se předvyplní při vytváření nového GitHub issue:

```
### Description
Please describe expected and actual behavior.

### Steps to reproduce

### System configuration
Please add result of ‘react-native info‘ command
and ‘react-native-today-widget‘ version used.
```

Kód 3.6: Šablona pro GitHub issue *.github/issue_template.md*.

Příkaz `react-native info` zobrazí nejen verze balíčků *react* a *react-native*, ale také verzi systému, Node, balíčkovacího systému (npm/Yarn) a Xcode.

Protože zatím na projektu pracuji samostatně, přidávám většinu změn rovnou do hlavní větve repozitáře (`master`), ze které vydávám nové verze.

Knihovnu publikuji pomocí Node Package Manageru (NPM)[132]. Příkazem `npm version` napřed navýším verzi knihovny zaznamenanou v souboru `package.json` a značkou (tag). Přitom zachovávám sémantické verzování[133]. Novou verzi pak publikuji příkazem `npm publish`.

3.1.7 Editace XCode projektu

Jedním z cílů práce bylo přidání Today widgetu co nejvíce usnadnit programátorům, kteří zatím nemají zkušenosti s nativními iOS projekty. Proto bylo

[129] <https://developer.apple.com/library/content/featuredarticles/XcodeConcepts/Concept-Targets.html>
[130] <https://github.com/matejkriz/react-native-today-widget>
[131] 3. 11. 2017 přibylo 6. issue nahlášené někým jiným než mnou samotným.
[132] <https://www.npmjs.com/package/react-native-today-widget>
[133] <https://semver.org>

potřeba pro všechny nutné změny v XCode projektu připravit automatické skripty (viz 3.1.4 a 3.1.5).

Abych mohl automatické skripty napsat, potřeboval jsem napřed dobře porozumět struktuře XCode projektu a prozkoumat dostupné nástroje. K tomu mi nejlépe posloužil blog [42] spolu s [43].

Mezi nástroji pro editaci XCode projektu jsem hledal v první řadě nějaký s rozhraním v JS, aby byl snadno pochopitelný pro většinu RN vývojářů. Nalezl jsem jen *node-xcode*[134], který byl převzat do Apache Cordova a je nadále udržovaný jako *cordova-node-xcode*[135]. Tento nástroj neposkytuje prakticky žádnou dokumentaci API a vše je tak potřeba vyčíst z kódu testů.

Další kandidát byl *XcodeEditor*[136], který je napsán v Objective-C. Oproti *cordova-node-xcode* je výrazně populárnější (dne 22. 12. 2017 měl XcodeEditor 776 Stars na GitHub a cordova-node-xcode jen 24).

Nakonec jsem zvolil *Xcodeproj*[137], který je sice napsán v Ruby a zavádí tak do již multiplatformního projektu další jazyk, ale je využíván v projektu CocoaPods[138] a má kvalitní dokumentaci[139]. Jeho jedinou nevýhodou je přidání závislosti, která není distribuována přes NPM a znamená to pro uživatele RNTW jeden krok navíc – příkaz `gem install xcodeproj`. Ruby mají uživatelé MacOS předinstalován.

3.1.8 Paměťové limity

Nejzásadnější problém, se kterým jsem se v průběhu implementace RNTW potýkal, je omezení maximální dostupné paměti pro Today widget. Je o tom jediná zmínka v oficiální dokumentaci [14], kde se uvádí, že je paměťový limit pro rozšíření výrazně menší než pro aplikace běžící samostatně v popředí. Zvláště nízký limit má Today widget, protože jich může mít uživatel spuštěných několik najednou.

Dobře to vysvětluje Conrad Kramer ve videu [22]. IOS zařízení mají nulový swap[140] a když by došla zařízení paměť, dojde k restartu. Aby k tomu nedocházelo, má iOS implementován Jetsam[141], který monitoruje použití paměti všemi procesy. Jetsam ukončuje aplikace, které překročí limit. Při nedostatku paměti v zařízení ukončuje aplikace podle nejmenší priority. Tato pravidla

[134] <https://github.com/alunny/node-xcode>
[135] <https://github.com/apache/cordova-node-xcode>
[136] <https://github.com/appsquickly/XcodeEditor>
[137] <https://github.com/CocoaPods/Xcodeproj>
[138] <https://cocoapods.org>
[139] <http://www.rubydoc.info/gems/xcodeproj/>
[140] <https://serverfault.com/questions/48486/what-is-swap-memory>
[141] <http://newosxbook.com/articles/MemoryPressure.html>

jsou definována v `com.apple.jetsamproperties.J81.plist`. Jejich přehled pro některá rozšíření je v tabulce 3.1.

Proces	**Priorita**	**Paměťový limit (MB)**
Aplikace	10	650
Share Extension	9	120
Action Extension	9	120
Photo Editor	9	120
Document Picker	9	84
Keyboard	8	48
WatchKit Extension	9	24
Today widget	9	16
Document Provider	9	10

Tabulka 3.1: Přehled paměťových limitů a priorit některých aplikačních rozšíření. Vyšší číslo značí vyšší prioritu. Zdroj: [22]

Conrad Kramer popisuje tři varianty paměti:

- *dirty*: Data potřebná pro chod aplikace jsou přímo v paměti zařízení. Jde o nejběžnější paměť, která obsahuje objekty, které naše aplikace používá (`viewController`, `model`).
- *purgable*: Cache, která obsahuje data, která mohou být zahozena, jen optimalizuje výkon aplikace.
- *clean*: Data jsou uložená v souborech na disku a do cache jsou tyto soubory namapovány.

Ve chvíli, kdy dochází paměť v zařízení, Jetsam nejprve pošle varování `[UIApplicationDelegate applicationDidReceiveMemory warning]` aplikaci s nejnižší prioritou. *Clean* paměť je uvolněna, data zůstávají pouze na disku. Pokud i poté vyžaduje aplikace příliš mnoho paměti, je okamžitě ukončena.

Z toho plyne, že je třeba minimalizovat *dirty* paměť a maximalizovat použití *purgable* a *clean* paměti. Jak minimalizovat *dirty* paměť najdeme například v [44]. Důležité je nenačítat do paměti žádné velké soubory. Ty je možné sta-

hovat nebo posílat přímo z disku. Operace s obrázky provádět též přímo na disku. Pro inspekci použití paměti slouží nástroj Instruments[142].

Pro maximalizaci *purgable* paměti je třeba používat cache. Nativně jde o `NSPurgableData` místo `NSData` a používání `NSCache`. Pro využití *clean* paměti slouží přepínač `NSDataReadingMappedAlways`[143]. Tato technika je vhodná nejen pro optimalizaci paměti, ale také výkonu. Je možné mít například zdrojová obrazová data v souborech namapovaných do paměti a umožnit uživateli listovat ve stovkách obrázků. Více o optimalizaci v [45] a [46].

Pro RN vývojáře je optimalizace paměti zkomplikována tím, že je často potřeba prohlížet přímo zdrojové soubory použitých komponent a sledovat jestli používají optimální nativní třídy. Navíc samotný běh JSCore potřebný pro RN spotřebuje část paměti.

Limit 16 MB pro Today widget jsem též ověřil experimentálně umístěním do widgetu komponenty `Image`[144] se zdrojem na webu s velkým obrázkem. K pádu došlo při stažení takové části obrázku, kdy bylo vyčerpáno právě 16 MB. Údaje o použité paměti jsem sledoval v debuggovacím nástroji přímo v XCode[145].

Nejjednodušší widget implementovaný pomocí RNTW s pouze jedním textovým polem potřebuje zhruba 11 MB paměti. To nedává příliš velký prostor pro její další využití.

Paměťové limity jsou pro uživatele větší problém než pomalejší vykreslování, protože mohou zcela zabránit spuštění widgetu.

3.1.9 Kompatibilita s verzemi RN

Další velkou výzvou bylo udržení kompatibility s frameworkem RN, který má každé dva týdny novou verzi. Ač se většinou nejedná o zpětně nekompatibilní změny v API, několikrát jsem musel RNTW upravovat kvůli změně struktury RN. Například ve verzi RN 0.46 byl přesunut soubor `react-native-xcode.sh` ze složky `packager` do složky `scripts`.

Pokud bych chtěl zachovat kompatibilitu i pro starší verze RN, musel bych tyto verze mít možnost detekovat. Tato možnost již v RN je[146], ale je dostupná

[142] <https://developer.apple.com/library/content/documentation/DeveloperTools/Conceptual/InstrumentsUserGuide/index.html>
[143] <https://developer.apple.com/documentation/foundation/nsdatareadingoptions/nsdatareadingmappedalways>
[144] <https://facebook.github.io/react-native/docs/image.html>
[145] <https://developer.apple.com/library/content/documentation/DeveloperTools/Conceptual/debugging_with_xcode/chapters/debugging_tools.html>
[146] <https://github.com/facebook/react-native/blob/master/Libraries/Core/

pouze v JS kódu a není zdokumentována. Její rozhraní se taktéž změnilo od verze RN 0.49.

Teoreticky by tedy bylo možné kompatibilitu rozšířit i na starší verze RN, ale zatím RNTW udržuji a testuji pouze pro aktuální verzi.

3.1.10 Developer menu

Výchozí developer menu[147] (DM) není ve widgetu přístupné, protože Today widget nemá dostupnou komponentu Action Sheet[148], ve které je vykresleno v hlavní aplikaci. Až v průběhu implementace ukázkové aplikace jsem si uvědomil jak moc mi DM chybí. Nabízí totiž spuštění automatického obnovování při změně v kódu a debuggování pomocí Chrome.

Naštěstí jsem narazil v [24] na popis možnosti zavolání některých funkcí DM přímo z JS kódu a tak jsem DM do RNTW doplnil. Výsledek je vidět na obrázku 3.1 vpravo.

Používám v něm funkce:

- `NativeModules.DevSettings.setIsDebuggingRemotely(isActive);` Spustí nebo přeruší debuggování pomocí Chrome.
- `NativeModules.DevSettings.setLiveReloadEnabled(true);` Spustí automatický restart aplikace po změně v kódu.
- `NativeModules.DevSettings.setHotLoadingEnabled(true);` Spustí automatickou obnovu změněné komponenty se zachováním stavu aplikace.

Pro detekci zda už je debuggování aktivní ověřuji jestli je dostupná funkce `importScripts`:

```
const isDebuggingRemotelyActive =
    () => typeof importScripts === 'function';
```

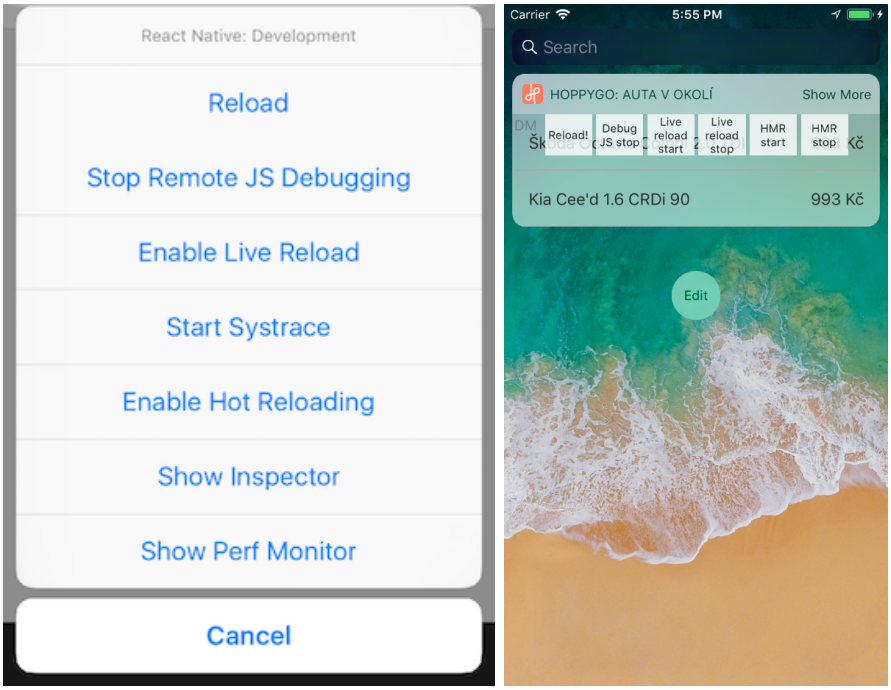
Kód 3.7: Ověření, zda je spuštěno vzdálené debuggování v Chrome. Soubor */lib/DevMenu/index.js*

Komponentu `DevMenu` pro zobrazení DM je možné importovat do vlastního widgetu z knihovny RNTW, jak je vidět v kódu 3.8. DM ve widgetu není možné vyvolat gestem zatřesení, jako je tomu v hlavní aplikaci. Výchozí způsob

ReactNativeVersion.js>

[147] <https://facebook.github.io/react-native/docs/debugging.html>

[148] <https://developer.apple.com/ios/human-interface-guidelines/views/action-sheets/>



Obrázek 3.1: Developer menu. Vlevo výchozí v RN. Zdroj: [8]. Vpravo vlastní ve widgetu.

zobrazení DM je klikatelný, částečně průhledný text „DM“ v levém horním rohu. Komponenta `DevMenu` má několik volitelných parametrů umožňujících přizpůsobení podoby odkazu na DM:

- `title?: string`: Text zobrazený jako odkaz pro otevření DM.
- `style?: StyleObj`: Styly textového odkazu pro otevření DM.
- `children?: React.Node`: Vlastní komponenta odkazu pro otevření DM.

```
import { DevMenu } from 'react-native-today-widget';

const TodayWidget = () => (
  <View>
    <YourMainComponent />
    {__DEV__ && <DevMenu />
  </View>
);
```

Kód 3.8: Ukázka nejjednoduššího použití developer menu.

3.1.11 Odkaz do aplikace

Ve widgetu bohužel nefunguje API Linking[149], díky kterému je jinak v hlavní aplikaci možné otevírat odkazy v jiných aplikacích. Ve widgetu má třída `NSExtensionContext` metodu

```
- (void)openURL:(NSURL *)URL
   completionHandler:(void (^)(BOOL success))completionHandler;
```

Kód 3.9: Metoda openURL třídy NSExtensionContext

Implementoval jsem tedy do RNTW nativní modul Linking, díky kterému je nyní možné otevřít hlavní aplikaci přímo z widgetu.

```
RCT_EXPORT_METHOD(openURL:(NSURL *)url )
{
  [widgetContext openURL:url completionHandler:nil];
}
```

Kód 3.10: Ukázka exportování metody pro otevření odkazu z widgetu. Soubor */ios/TodayWidgetExtension/Linking.m*

Příklad použití nativního modulu Linking ve vlastním widgetu je vidět v kódu 3.11.

```
import { openURL } from 'react-native-today-widget';

openURL('http://maps.apple.com/?saddr=Praha&daddr=Brno');
```

Kód 3.11: Ukázka použití nativního modulu Linking pro odkaz do aplikace Mapy.

3.1.12 Zobrazit více/méně

Pro možnost roztažení výšky widgetu a jeho opětovné zmenšení jsem implementoval nativní modul `DisplayMode`:

```
#import "DisplayMode.h"

#import <NotificationCenter/NotificationCenter.h>
#import <Foundation/Foundation.h>

@implementation DisplayMode

static float maxHeight = 110;

NSExtensionContext* extensionContext;

// To export a module named DisplayMode
RCT_EXPORT_MODULE();
```

[149] <https://facebook.github.io/react-native/docs/linking.html#openurl>

```
- (id)initWithContext:(NSExtensionContext*)context {
  self = [super init];
  extensionContext = context;
  return self;
}

+ (float)getMaxHeight {
  return maxHeight;
}

RCT_EXPORT_METHOD(setExpandable:(BOOL)expandable height:(float)
    height )
{
  maxHeight = height;
  if (expandable) {
    [extensionContext setWidgetLargestAvailableDisplayMode:
        NCWidgetDisplayModeExpanded];
  } else {
    [extensionContext setWidgetLargestAvailableDisplayMode:
        NCWidgetDisplayModeCompact];
  }
}

@end
```

Kód 3.12: Implementace nativního modulu DisplayMode v souboru */ios/TodayWidgetExtension/DisplayMode.m*

V kódu 3.13 je ukázka použití nativního modulu DisplayMode, včetně odchycení události změny výšky widgetu. Tento příklad je publikován v knihovně RNTW ve složce *Examples/Expandable/*[150].

```
import React, { Component } from 'react';
import { Text, View } from 'react-native';
import { setExpandable } from 'react-native-today-widget';

class ExpandableTodayWidget extends Component {
  constructor() {
    super();
    const isExpandable = true;
    const maxHeight = 500;
    setExpandable(isExpandable, maxHeight);
    this.onLayout = (event) => {
      const height = event.nativeEvent.layout.height;
      if (height <= 110) {
        console.log('widget is in compact mode');
      } else if (height > 110) {
        console.log('widget is in expanded mode');
      }
```

[150] <https://github.com/matejkriz/react-native-today-widget/blob/master/Examples/Expandable/index.ios.js>

```
    };
  }

  render() {
    return (
      <View
        onLayout={this.onLayout}
      >
        <Text>
          Hello expandable world!
        </Text>
      </View>
    );
  }
}
```

Kód 3.13: Ukázka použití nativního modulu DisplayMode. Zdroj: RNTW

3.1.13 Návod k použití RNTW

Pro přidání widgetu k RN aplikaci nejprve nainstalujte knihovnu *react-native-today-widget* (alternativní příkazy pro instalaci nativní závislosti naleznete v kódu 3.1):

```
$ npm install --save react-native-today-widget
```

Následně zavolejte automatické začlenění nativní závislosti do hlavního projektu[151]:

```
$ react-native link
```

Posledním krokem je registrace vlastní libovolné komponenty widgetu do modulu `TodayWidgetExtension` ve vstupním souboru projektu:

```
import React from 'react';
import { AppRegistry, Text, View } from 'react-native';

const TodayWidget = () => (
  <View>
    <Text>Hello Today Widget!</Text>
  </View>
);

AppRegistry.registerComponent('TodayWidgetExtension', () =>
    TodayWidget);
```

Kód 3.14: Ukázka přidání jednoduchého widgetu v souboru *index.js*.

[151] Pro tento krok je nutné, aby již byla knihovna react-native-today-widget uvedená v seznamu závislostí v souboru *package.json*.

Všimněte si, že pro přidání widgetu není vůbec nutné importovat do kódu *react-native-today-widget*. To je potřeba jen pro použití nativních modulů (viz 3.1.10, 3.1.12 a 3.1.11).

3.2 Realizace ukázkové aplikace *HoppyGo: Auta v mém okolí*

Ukázková aplikace je veřejně dostupná v repozitáři knihovny *react-native-today-widget* jako příklad použití ve složce `/Examples/Complex/`[152].

3.2.1 Dosažený stav

Widget jsem realizoval v souladu se zjednodušeným návrhem bez rozbalování detailu auta. Zobrazení detailu vozu (US2, 2.5.2) jsem realizoval odkazem do hlavní aplikace po kliknutí na libovolné z nabízených aut. Na obrázku 3.2 je vidět výsledná podoba widgetu poskytujícího rychlý přehled o nabízených vozech (US1, 2.5.2).

3.2.2 Použité technologie a nástroje

Pro realizaci ukázkové aplikace jsem samozřejmě použil React Native, knihovnu *react-native-today-widget* a jazyk JavaScript.

Pro stylování aplikace jsem použil výchozí styly[153] implementované v RN.

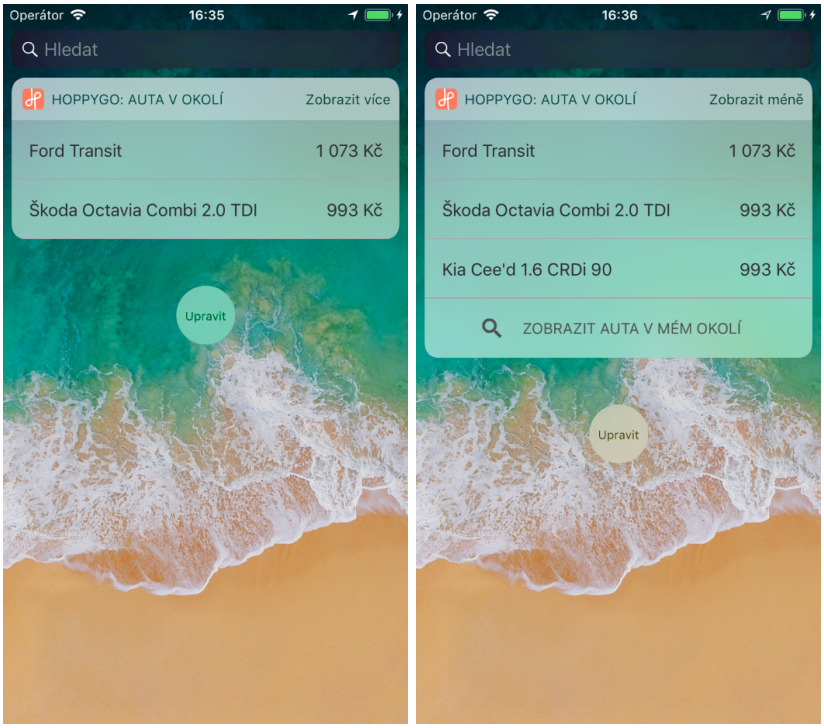
3.2.2.1 JavaScript

React Native používá JS engine *JavaScriptCore* (viz 2.1.4). Zároveň obsahuje JS kompilátor *Babel*[154] a tak jsem využil novou syntaxi z verzí ES5, ES6 i ES7. Podrobně jsou dostupné transformace rozebrány v [8][155].

Uvedu několik příkladů nové syntaxe, kterou jsem použil:

- *Arrow functions*
  ```
  <TouchableOpacity onPress={() => openInApp({ id })}>
  ```
- *Classes*
  ```
  class CarsNearby extends
    Component<CarsNearbyProps, CarsNearbyState>
  ```

[152] <https://github.com/matejkriz/react-native-today-widget/tree/master/Examples/Complex>
[153] <https://facebook.github.io/react-native/docs/style.html>
[154] <https://babeljs.io>
[155] <https://facebook.github.io/react-native/docs/javascript-environment.html>



Obrázek 3.2: Výsledná podoba ukázkové aplikace. Vlevo widget v základním stavu, vpravo po kliknutí na „Zobrazit více“.

- *Destructuring*
  ```
  const { latitude, longitude } = this.props;
  ```
- *Modules*
  ```
  import { openURL } from 'react-native-today-widget';
  ```
- *Object Short Notation*
  ```
  const paddingHorizontal = 16;
  StyleSheet.create({container: { paddingHorizontal }})
  ```
- *Rest Params*
  ```
  <ComposedComponent {...this.props} date={new Date()}/>
  ```
- *Template Literals*
  ```
  `${manufacturer} ${model}`
  ```
- *Async Functions*
  ```
  const carsNearbyBlob = await loadFromFile(
    { path: carsNearbyPath }
  );
  ```

3.2.2.2 Statická analýza

Stejně jako pro realizaci samotné knihovny (viz 3.1.3) jsem použil Flow pro typovou kontrolu, ESLint s konfigurací od Airbnb a Prettier na automatické formátování kódu.

Do ESLint konfigurace jsem navíc přidal plugin *eslint-plugin-fp*[156] s pravidly pro funkcionální programování. Snažil jsem se tedy vyhnout třídám a mutacím objektů.

3.2.3 Struktura projektu

```
components ..... veškeré React komponenty
├── __tests__ ..... definice snapshot testů
│   └── __snapshots__ ..... uložené snapshoty
├── Car.js ..... komponenta vykreslující jeden řádek s autem
├── CrsNearby.js ..... kontejner načítající data z API
├── withDate.js ..... HOC přidává parametr s aktuálním datem
└── withLocation.js ..... HOC přidává parametry s aktuální polohou
config
├── api.js ..... parametry pro API
├── geolocation.js ..... parametry pro sledováí aktuální pozice
└── index.js ..... obecná konfigurace a cesty
lib ..... znovupoužitelné pomocné funkce
├── date.js ..... formátování data
├── date.spec.js ..... unit test
├── fetchBlob.js ..... stahování dat přímo do souboru na disku
├── price.js ..... formátování ceny
└── price.spec.js ..... unit test
index.js ..... vstupní soubor, kde je widget registrován
App.js ..... výchozí vstupní soubor hlavní aplikace
TodayWidget.js ..... kontejner celého widgetu
```

Obrázek 3.3: Přehled struktury aplikace s vynecháním výchozích inicializačních souborů.

Díky tomu, že se jedná pouze o widget, je struktura projektu poměrně jednoduchá. Výchozí vstupní soubor hlavní aplikace bude nahrazen vstupním souborem aplikace HoppyGo. Snažil jsem se vyhnout přidávání závislostí kvůli omezením paměti. Jedinou přidanou knihovnou je tak *react-native-fetch-blob*[157], který jsem přidal kvůli dalšímu snížení paměťových nároků (zajišťuje stahování dat z API přímo do souboru na disku místo do paměti aplikace).

[156] <https://www.npmjs.com/package/eslint-plugin-fp>
[157] <https://github.com/wkh237/react-native-fetch-blob>

3.2.4 Automatické testy

Pro automatické testy jsem použil *Jest*[158], který je stejně jako RN od Facebooku a v RN je rovnou po inicializaci nového projektu připraven k použití.

Unit testy jsem napsal pro pomocné funkce na formátování datumů a ceny. Uvádím ukázku testu pro formátování datumu. Datum je potřeba formátovat pro dotaz do API na auta dostupná následujícího dne:

```
it('get the day after', () => {
  expect(getTomorrowDate(new Date(1510909200000))).toEqual(
    new Date(1510995600000),
  );
});

it('convert date to YYYY-MM-DDTHH:MM formate', () => {
  expect(formateDate(new Date(1510909200000))).toBe('2017-11-17
     T09:00');
});

it('get date range', () => {
  expect(getTomorrowDateRange(new Date(1510909200000))).toBe(
    '2017-11-18T09:00,2017-11-19T09:00',
  );
});
```

Kód 3.15: Ukázka unit testu */lib/date.spec.js*

Pro React komponenty jsem využil snapshot testy opět pomocí Jestu. Uvádím příklad toho nejjednoduššího testu. Komplexnější ukázku můžete nalézt v souboru */components/__tests__/CarsNearby.spec.js*.

```
import 'react-native';
import React from 'react';
import Car from '../Car';

import renderer from 'react-test-renderer';

test('renders correctly', () => {
  const tree = renderer
    .create(
      <Car id={1} manufacturer="manufacturer" model="model" price
         ="1 042 Kč" />,
    )
    .toJSON();
  expect(tree).toMatchSnapshot();
});
```

Kód 3.16: Ukázka snapshot testu */components/__tests__/Car.spec.js*

[158] <https://facebook.github.io/jest/>

3.2.5 Stahování dat

Data o autech dostupných v okolí jsem potřeboval uchovat někde trvale, aby nebylo potřeba po restartu widgetu čekat na stažení dat i v případě, že se nezměnila zeměpisná poloha uživatele. Využil jsem knihovnu *react-native-fetch-blob*[159], která umožňuje stahování dat přímo do souboru na disku. Knihovna react-native-fetch-blob umožňuje ušetřit mnoho paměti především v případě použití mediálních souborů, protože lze používat v aplikaci data rovnou ze souborů na disku a z aplikace je pouze namapovat. Přímé mapování souboru z disku v tomto případě nebylo možné, protože potřebuji pracovat s obsahem JSON objektu přijatého v odpovědi z API.

Stahování dat do souboru i načítání obsahu souboru je implementováno v souboru *lib/fetchBlob.js*:

```
// @flow
import RNFetchBlob from 'react-native-fetch-blob';

type FetchToFileProps = {
  headers: mixed,
  path: string,
  requestPath: string,
};

type LoadFromFileProps = {
  path: string,
};

export const { dirs } = RNFetchBlob.fs;

export const fetchToFile = ({
  headers,
  path,
  requestPath,
}: FetchToFileProps): Promise<string> =>
  new Promise((resolve, reject) => {
    RNFetchBlob.config({
      path,
    })
      .fetch('GET', requestPath, {
        ...headers,
      })
      .then(response => resolve(response))
      .catch(error => reject(error));
  });

export const loadFromFile = ({ path }: LoadFromFileProps):
    Promise<string> =>
  new Promise((resolve, reject) => {
    RNFetchBlob.fs
```

[159] <https://github.com/wkh237/react-native-fetch-blob>

```
      .readStream(path)
      .then(stream => {
        let data = '';
        stream.open();
        stream.onData(chunk => {
          data += chunk;
        });
        stream.onEnd(() => resolve(data));
      })
      .catch(error => reject(error));
  });
```

Kód 3.17: Stahování dat do souboru a načtení souboru z disku. Soubor *lib/fetchBlob.js*

3.2.6 Načítání aktuální pozice

Pro načtení aktuální pozice jsem použil API Geolocation[160], což je polyfill[161] `navigator.geolocation`[162]. Konkrétně jsem použil pro průběžné sledování `navigator.geolocation.watchPosition(success, error?, options?)`[163]. Parametry jsem umístil do konfiguračního souboru:

```
const watchPositionOptions = {
  // o~kolik metrů se musí nejméně změnit pozice, aby byl zavolán
       callback:
  distanceFilter: 100,
  // toleruje i méně přesnou polohu, ale šetří baterii:
  enableHighAccuracy: false,
  // maximální přijatelné stáří cachovaného údaje o~pozici, při
      starším údaji je vyžádána aktuální hodnota:
  maximumAge: 3600000,
  // maxilální doba čekání na vrácení současné hodnoty:
  timeout: 30000,
};

export default watchPositionOptions;
```

Kód 3.18: Konfigurace sledování pozice v souboru *config/geolocation.js*

Pozice se načítá v rámci takzvané Higher-Order komponenty (HOC)[164]. Jde o způsob kompozice komponent užívaný v Reactu. HOC *components/withLocation.js* obohacuje komponentu o parametry `latitude` a `longitude`:

```
// @flow
import React, { Component } from 'react';
```

[160] <https://facebook.github.io/react-native/docs/geolocation.html>
[161] <https://en.wikipedia.org/wiki/Polyfill_(programming)>
[162] <https://developer.mozilla.org/en-US/docs/Web/API/Geolocation>
[163] <https://facebook.github.io/react-native/docs/geolocation.html#watchposition>
[164] <https://reactjs.org/docs/higher-order-components.html>

```
import type { ComponentType } from 'react';
import watchPositionOptions from '../config/geolocation';

type GeolocationState = {
  error: ?string,
  latitude: ?number,
  longitude: ?number,
};

type GeolocationProps = any;

const Geolocation = (ComposedComponent: ComponentType<any>) =>
  class extends Component<GeolocationProps, GeolocationState> {
    constructor(props: GeolocationProps) {
      super(props);

      this.state = {
        error: null,
        latitude: null,
        longitude: null,
      };
    }

    componentDidMount() {
      if ('geolocation' in navigator) {
        this.geolocation = navigator.geolocation.watchPosition(
          position => {
            this.setState({
              error: null,
              latitude: position.coords.latitude,
              longitude: position.coords.longitude,
            });
          },
          error => this.setState({ error: error.message }),
          watchPositionOptions,
        );
      }
    }

    componentWillUnmount() {
      if ('geolocation' in navigator) {
        navigator.geolocation.clearWatch(this.geolocation);
      }
    }

    geolocation: any;

    render() {
      const { error, latitude, longitude } = this.state;

      if (error) {
        return null;
      }
```

```
      return (
        <ComposedComponent
          {...this.props}
          latitude={latitude}
          longitude={longitude}
        />
      );
    }
  };

export default Geolocation;
```

Kód 3.19: HOC *withLocation* přidávající komponentě parametry s aktuální pozicí. Soubor *components/withLocation.js*

Závěr

Knihovnu *react-native-today-widget* jsem zveřejnil 19. května 2017[165] jako open source na GitHubu a za první půlrok nasbírala již 65 stars[166] a má více než 2800 stažení[167]. Richard Lai, který jako první zveřejnil pokus o přidání Today widgetu do RN, odkázal 10. října 2017 ze svého repozitáře[168] na *react-native-today-widget*. Odkaz na moji knihovnu byl dokonce přidán i do oficiální dokumentace frameworku React Native[169].

Jako ukázku použití knihovny jsem implementoval Today widget pro aplikaci HoppyGo. Jde o službu na sdílení aut. Today widget zobrazuje dostupná auta v blízkém okolí. V souladu se zásadami uživatelského rozhraní (2.3.3) jednoduše a přehledně zobrazuje v základním zobrazení dvě a v rozšířeném tři dostupná vozidla a odkaz do aplikace. Každé vozidlo odkazuje na svůj detail v hlavní aplikaci HoppyGo. Zobrazení widgetu je přizpůsobivé pro obrazovky všech iOS zařízení jak v zobrazení na šířku, tak na výšku.

V kapitole 2.2 je základní přehled dostupných rozšíření s popisem možností použití s RN. Zvláštní pozornost byla věnována implementaci Today widgetu, které se analyticky věnuje kapitola 2.3 a realizaci popisuje kapitola 3.1.

V průběhu implementace knihovny *react-native-today-widget* jsem narazil na mnoho problémů, z nichž nejzásadnější jsou paměťové limity, kterým se věnuji v kapitole 3.1.8. Omezení pouze na 16 MB dostupné paměti pro Today widget je v použití s RN velmi svazující. Pro rozšíření použitelnosti

[165] První commit:
<https://github.com/matejkriz/react-native-today-widget/commit/29804eae7dc5ffb57a0344d787bd32c064e92355>

[166] <https://help.github.com/articles/about-stars/>

[167] <http://npm-stats.com/~packages/react-native-today-widget>

[168] <https://github.com/rclai/React-Native-Today-Widget>

[169] <https://facebook.github.io/react-native/docs/app-extensions.html#today-widget>

této knihovny bude nutné se dále zaměřit na optimalizaci, aby se minimalizovala paměť zabraná samotným RN a maximalizovala paměť volná pro využití widgetem.

Plán dalšího rozvoje knihovny uvádím v `README`[170] projektu jako seznam „TODO“. Kromě vylepšování dokumentace plánuji především umožnit automatické nastavení zobrazovaného jména widgetu `CFBundleDisplayName`[171], nejlépe spolu s unikátním identifikátorem `CFBundleIdentifier`[172] podle hodnot uvedených v souboru `app.json`. Dále chci umožnit automatické přesunutí widgetu ze složky `node_modules`, která obvykle není verzována[173]. Jako poslední bod seznamu mám možnost podmínečného zobrazení widgetu.

[170] <https://github.com/matejkriz/react-native-today-widget/blob/master/README.md#todo>
[171] <https://developer.apple.com/library/content/documentation/General/Reference/InfoPlistKeyReference/Articles/CoreFoundationKeys.html#//apple_ref/doc/uid/20001431-110725>
[172] <https://developer.apple.com/library/content/documentation/General/Reference/InfoPlistKeyReference/Articles/CoreFoundationKeys.html#//apple_ref/doc/uid/20001431-110725>
[173] <https://cs.wikipedia.org/wiki/Verzování>

Literatura

[1] *React.js Conf 2015 Keynote* [online]. 2017. [cit. 18. 09. 2017]. Dostupné z: <https://www.youtube.com/watch?v=KVZ-P-ZI6W4>.

[2] Eisenman, B.: *Learning React Native*. O'Reilly Media, Inc., 2016, ISBN 978-1-491-92900-1.

[3] Grabowski, M. *React Native Monthly #3* [online]. 2017. [cit. 30. 08. 2017]. Dostupné z: <https://facebook.github.io/react-native/blog/2017/08/30/react-native-monthly-3.html>.

[4] Dabit, N.: *React Native in Action*. Manning Publications Co., 2016, ISBN 978-1-617-29405-1.

[5] Write once, run anywhere? *ComputerWeekly*, květen 2002, [cit. 19. 11. 2017]. Dostupné z: <http://www.computerweekly.com/feature/Write-once-run-anywhere>

[6] Dabit, N. *The Cost of Native Mobile App Development is Too Damn High!* [online]. 2016. [cit. 20. 06. 2017]. Dostupné z: <https://hackernoon.com/the-cost-of-native-mobile-\app-development-is-too-damn-high-4d258025033a>.

[7] Zagallo, T. *React Native Architecture Overview* [online]. 2016. [cit. 20. 10. 2017]. Dostupné z: <https://www.youtube.com/watch?v=Ah2qNbI40vE>.

[8] *React Native Docs* [online]. 2017. [cit. 21. 12. 2017]. Dostupné z: <https://facebook.github.io/react-native/docs/getting-started.html>.

[9] Swift. *Wikipedia*, 2017, [cit. 2. 1. 2018]. Dostupné z: <https://en.wikipedia.org/wiki/Swift_(programming_language)>

[10] *Repozitář React Native na GitHub* [online]. [cit. 22. 10. 2017]. Dostupné z: <https://github.com/facebook/react-native>.

[11] Kol, T. [online]. [cit. 1. 1. 2018].

[12] Abernathy, C. *React Native Tutorial: Integrating in an Existing App* [online]. 2016. [cit. 31. 12. 2017]. Dostupné z: <https://www.raywenderlich.com/136047/react-native-existing-app>.

[13] Zagallo, T. *Bridging in React Native* [online]. 2015. [cit. 29. 12. 2017]. Dostupné z: <https://tadeuzagallo.com/blog/react-native-bridge/>.

[14] *App Extension Programming Guide* [online]. 2017. [cit. 18. 07. 2017]. Dostupné z: <https://developer.apple.com/library/content/documentation/General/Conceptual/ExtensibilityPG>.

[15] *Apple Developer Support: App Store* [online]. 2017. [cit. 05. 06. 2017]. Dostupné z: <https://developer.apple.com/support/app-store/>.

[16] Smith, D. *iOS Version Stats* [online]. 2017. [cit. 10. 06. 2017]. Dostupné z: <https://david-smith.org/iosversionstats/>.

[17] Baird, I. *Creating Extensions for iOS and OS X, Part 1* [online]. 2014. [cit. 20. 06. 2017]. Dostupné z: <https://developer.apple.com/videos/play/wwdc2014/205/>.

[18] Sorresso, D. *Creating Extensions for iOS and OS X, Part 2* [online]. 2014. [cit. 20. 06. 2017]. Dostupné z: <https://developer.apple.com/videos/play/wwdc2014/217/>.

[19] Smith, P. *Building iOS App Extensions With React Native* [online]. 2017. [cit. 5. 3. 2017]. Dostupné z: <https://www.promptworks.com/blog/building-ios-app-extensions-with-react-native>.

[20] *Uživatelská příručka pro iPhone* [online]. 2017. [cit. 14. 11. 2017]. Dostupné z: <https://help.apple.com/iphone/11/?lang=cs>.

[21] *iOS Human Interface Guidelines* [online]. 2017. [cit. 1. 9. 2017]. Dostupné z: <https://developer.apple.com/ios/human-interface-guidelines>.

[22] Kramer, C. *Memory Use in Extensions* [online]. 2017. [cit. 20. 08. 2017]. Dostupné z: <https://cocoaheads.tv/memory-use-in-extensions-by-conrad-kramer/>.

[23] Sophia Teutschler, I. B. *App Extension Best Practices* [online]. 2015. [cit. 22. 06. 2017]. Dostupné z: <https://developer.apple.com/videos/play/wwdc2015/224/>.

[24] Dozois-C., H. *Messages App/App Extension (iOS10) with React Native* [online]. 2016. [cit. 2. 12. 2017]. Dostupné z: <https://medium.com/rendez-voo/messages-app-app\-extension-ios10-with-react-native-6d22ece64598>.

[25] Bhaskar Sarma, L. H. *iMessage Apps and Stickers, Part 1* [online]. 2016. [cit. 2. 12. 2017]. Dostupné z: <https://developer.apple.com/videos/play/wwdc2016/204/>.

[26] Alex Carter, S. L. *iMessage Apps and Stickers, Part 2* [online]. 2016. [cit. 2. 12. 2017]. Dostupné z: <https://developer.apple.com/videos/play/wwdc2016/224/>.

[27] *SiriKit Documentation* [online]. 2017. [cit. 31. 12. 2017]. Dostupné z: <https://developer.apple.com/documentation/sirikit>.

[28] Walker, R. *Introducing SiriKit* [online]. 2016. [cit. 5. 12. 2017]. Dostupné z: <https://developer.apple.com/videos/play/wwdc2016/217/>.

[29] Khosla., V. *Extending Your Apps with SiriKit* [online]. 2016. [cit. 5. 12. 2017]. Dostupné z: <https://developer.apple.com/videos/play/wwdc2016/225/>.

[30] Wagner, C. *Today Extensions Tutorial: Getting Started* [online]. 2014. [cit. 10. 10. 2017]. Dostupné z: <https://www.raywenderlich.com/83809/ios-8-today-extension-tutorial>.

[31] Techotopia. *An iOS 9 Today Extension Widget Tutorial* [online]. 2016. [cit. 10. 10. 2017]. Dostupné z: <http://www.techotopia.com/index.php/An_iOS_9_Today_Extension_Widget_Tutorial>.

[32] Tessarin, C. *iOS 8: Creating a Today Widget* [online]. 2014. [cit. 10. 10. 2017]. Dostupné z: <https://code.tutsplus.com/tutorials/ios-8-creating-a-today-widget--cms-22379>.

[33] Petr, L. *How to create iOS 8 Today extension and share data with containing app* [online]. 2014. [cit. 10. 10. 2017]. Dostupné z: <http://www.glimsoft.com/06/28/ios-8-today-extension-tutorial/>.

[34] Skaarup, D. *Add iOS Today Widget to your React Native App* [online]. 2017. [cit. 10. 12. 2017]. Dostupné z: <https://medium.com/@davidskaarup/add-ios-today-widget-to-your-react-native-app-ed9c9b601cc>.

[35] Fork A Repo. *GitHub Help*, 2018, [cit. 6. 1. 2018]. Dostupné z: <https://help.github.com/articles/fork-a-repo/>

[36] López, C. *Making libraries for React Native* [online]. 2016. [cit. 10. 12. 2017]. Dostupné z: <https://medium.com/gbox-crew-blog/making-libraries-for-react-native-eaca35b5b1d7>.

[37] Wixen, B. *Distributing React Native components with native code* [online]. 2016. [cit. 10. 12. 2017]. Dostupné z: <https://www.benwixen.com/articles/distributing-react-native-components-with-native-code/>.

[38] Swift and Objective-C in the Same Project. *Guides and Sample Code*, 2017, [cit. 7. 1. 2018]. Dostupné z: <https://developer.apple.com/library/content/documentation/Swift/Conceptual/BuildingCocoaApps/MixandMatch.html>

[39] *HoppyGo FAQ* [online]. 2017. [cit. 28. 12. 2017]. Dostupné z: <https://www.hoppygo.cz/faq>.

[40] Drhlík, K. *Android pro začátečníky #7 – Co je to widget a jak jej používat* [online]. 2015. [cit. 7. 1. 2018]. Dostupné z: <https://www.svetandroida.cz/android-pro-zacatecniky-widget-201508/>.

[41] Dozois-C, H. *Writing your first React Native module* [online]. 2016. [cit. 21. 8. 2017]. Dostupné z: <https://medium.com/rendez-voo/writing-your-first-react-native-module-50ede0840dae>.

[42] Demi, S. *Managing Xcode* [online]. 2015. [cit. 21. 9. 2017]. Dostupné z: <https://pewpewthespells.com/blog/managing_xcode.html>.

[43] Hurst, A. *How to set up multiple schemes & configurations in Xcode for your React Native iOS app* [online]. 2016. [cit. 21. 9. 2017]. Dostupné z: <https://zeemee.engineering/how-to-set-up-multiple-schemes-configurations-in-xcode-for/-your-react-native-ios-app-7da4b5237966>.

[44] Paine, B. *Optimizing Your App for Multitasking on iPad in iOS 9* [online]. 2015. [cit. 2. 08. 2017]. Dostupné z: <https://developer.apple.com/videos/play/wwdc2015/212/>.

[45] *Performance Tips* [online]. 2017. [cit. 14. 10. 2017]. Dostupné z: <https://developer.apple.com/library/content/documentation/iPhone/Conceptual/iPhoneOSProgrammingGuide/PerformanceTips/PerformanceTips.html>.

[46] *About the Virtual Memory System* [online]. 2013. [cit. 14. 10. 2017]. Dostupné z: <https://developer.apple.com/library/content/documentation/Performance/Conceptual/ManagingMemory/Articles/AboutMemory.html>.

Příloha **A**

Seznam použitých zkratek

API Application Programming Interface

AST Abstract Syntax Tree

CRNA Create React Native App

ČR Česká Republika

Developer menu Developer menu

GCD Grand Central Dispatch

HOC Higher-Order Component

IPC meziprocesová komunikace

JIT Just in time

JS JavaScript

NPM Node Package Manager

REST Representational State Transfer

RN React Native

RNPM React Native Package Manager

RNTW react-native-today-widget

UI User Interface – uživatelské rozhraní

URL Uniform Resource Locator

VM Virtual Machine

Příloha B

Obsah přiloženého CD

- `readme.txt` ..... stručný popis obsahu CD
- `src`
  - `react-native-today-widget` ..... zdrojové kódy implementace
  - `DP_Kriz_Matej_2018` ..... zdrojová forma práce ve formátu LaTeX
- `text` ..... text práce
  - `DP_Kriz_Matej_2018.pdf` ..... text práce ve formátu PDF