Sem vložte zadání Vaší práce.

České vysoké učení technické v Praze
Fakulta informačních technologií
Katedra softwarového inženýrství

Diplomová práce

Letecké záznamy pro iOS pomocí moderních architektur a FRP

Bc. Martin Žid

Vedoucí práce: Ing. Dominik Veselý

5. září 2017

Poděkování

Doplňte, máte-li komu a za co děkovat. V opačném případě úplně odstraňte tento příkaz.

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. Každá osoba, která využije výše uvedenou licenci, se však zavazuje udělit ke každému dílu, které vznikne (byť jen zčásti) na základě Díla, úpravou Díla, spojením Díla s jiným dílem, zařazením Díla do díla souborného či zpracováním Díla (včetně překladu), licenci alespoň ve výše uvedeném rozsahu a zároveň zpřístupnit zdrojový kód takového díla alespoň srovnatelným způsobem a ve srovnatelném rozsahu, jako je zpřístupněn zdrojový kód Díla.

V Praze dne 5. září 2017 .....

Odkaz na tuto práci

Žid, Martin. *Letecké záznamy pro iOS pomocí moderních architektur a FRP.* Diplomová práce. Praha: České vysoké učení technické v Praze, Fakulta informačních technologií, 2017.

Abstrakt

Tato práce realizuje iOS aplikaci pro evidenci letů. V první části analyzuji obdobné aplikace a předpisy pro piloty České republiky, podle nichž probíhá návrh funkcionality vytvářené aplikace. Podle návrhu je následně zvolena vhodná architektura a vytvořeno uživatelského rozhraní v podobě wireframů.

Aplikace je implementována s použitím zvolené architektury a pomocí principů FRP. V průběhu implementace aplikace jsou realizovány jednotkové testy a na konci jsou provedeny uživatelské testy. Na základě výsledků testů je aplikace upravena do finální podoby.

V poslední části práce popisuji výhody a nevýhody, které přinesly postupy FRP. Také hodnotím časovou a implementační náročnost oproti standardním postupům a architektuře MVC.

V práci jsem vytvořil funkční iOS aplikaci s využitím moderní architektury a principů FRP. Aplikace bude sloužit pilotům České republiky pro elektronickou evidenci letů a bude jim také ulehčovat administrativu s evidencí spojenou.

V příloze této bakalářské práce je možné nalézt všechny zdrojové kódy jak aplikace, tak i testů společně s vytvořenými wireframy.

Klíčová slova mobilní aplikace pro evidenci letů, iOS, Swift, FRP, ReactiveCocoa, MVVM architektura

Abstract

Sem doplňte ekvivalent abstraktu Vaší práce v angličtině.

Keywords flight records mobile application, iOS, Swift, FRP, ReactiveCocoa, MVVM architecture

Obsah

Seznam obrázků

Úvod

V dnešní době, kdy existují mobilní aplikace na téměř vše, mě zarazil fakt, že u pilotů tomu tak nemusí být. Aplikace na evidenci letů samozřejmě existují, však je tu hned několik problémů. Ty aplikace jsou často velice drahé, nemusí odpovídat leteckým přepisům České republiky nebo nemají vyhovující funkcionalitu.

Z tohoto důvodu jsem se rozhodl vytvořit iOS aplikaci na evidenci letů. Tato aplikace bude pomáhat pilotům zaznamenávat elektronicky své lety, bude také kontrolovat předpisy a umožňovat export do formátu pro tisk.

Začínám analýzou podobných aplikací a to pro zařízení iOS i Android. Poté navrhuji vhodnou funkcionalitu a vytvářím návrh uživatelského rozhraní.

Dalším tématem, které ve své práci řeším jsou softwarové architektury při vývoji iOS aplikace. Zde analyzuji alternativy k architektuře MVC ve spojení s funkcionálně reaktivním programováním neboli FRP.

Tuto analýzu následně aplikuji v praxi, kdy se zvolenou architekturou a FRP implementuji společně s jednotkovými testy dříve zmíněnou aplikaci. Nakonec aplikaci podrobím uživatelským testům a podle jejich výsledků upravím aplikaci do finální podoby.

V poslední části své práce se snažím zhodnotit postupy FRP společně s mnou zvolenou moderní architekturou a jejich časovou a implementační náročnost oproti klasickému MVC.

Kapitola 1

Cíl práce

Cílem této práce je navrhnout a implementovat aplikaci k evidenci letů pro platformu iOS. A to pomocí postupů FRP (funkcionálně reaktivního programování) a s využitím moderní softwarové architektury jako např. MVVM nebo VIPER. Tato aplikace bude sloužit pilotům České republiky k elektronické evidenci letů. Tento cíl je rozdělen do několika podúkolů.

V první části analyzuji podobné aplikace pro evidenci letů a to jak pro platformu iOS, tak i pro Android. Na základě této analýzy navrhnu vhodnou funkcionalitu pro vytvářenou aplikaci. Podle navržených funkcionalit si zvolím architekturu a navrhnu uživatelské rozhraní v podobě wireframů.

V dalším kroku aplikaci implementuji pomocí postupů FRP a se zvolenou architekturou. V průběhu realizace aplikace budou vytvářeny také testy a dokumentace aplikace.

Dále bude aplikace podrobena uživatelským testům, podle kterých bude vhodně upravena.

V poslední části budu popisovat výhody a nevýhody, které přinesly postupy FRP. Budu také hodnotit časovou a implementační náročnost oproti standardním postupům a architektuře MVC.

Kapitola 2

Analýza a návrh

2.1 Tvorba iOS aplikací

Vývoj iOS aplikace je možný hned několika způsoby, každý má své výhody a nevýhody, právě ty bych rád v této kapitole rozebral. Mezi možné způsoby vývoje bych rád zmínil nativní aplikace, hybridní aplikace a mobilní webové aplikace.

2.1.1 Nativní aplikace

Nativní aplikace jsou vyvíjeny specificky pro jednu platformu. Díky tomu mají přístup ke všem funkcím daného zařízení jako např. GPS. kamera nebo kontakty. Mohou fungovat i pouze offline, tedy bez nutnosti internetového připojení. [1]

Však pokud bychom chtěli aplikaci distribuovat na více platforem, tak s tímto přístup by bylo nutné vytvořit pro každou platformu vlastní aplikaci. To by prodloužilo vývoj a znesnadnilo následnou údržbu aplikací.

Co se týče iOS vývoje, je možné si zvolit z dvou programovacích jazyků – Objective-C nebo Swift. [2] [3]

2.1.2 Hybridní aplikace

Hybridní aplikace jsou aplikace tvořené nejčastěji pomocí HTML5 a JavaScriptu, následně jsou spuštěné v nativním kontejneru. [4] Jako příklad je možné uvést např. Apache Cordova. Tento kontejner umožňuje přístup k funkcím daného přístroje, podporuje použití aplikace offline a dává možnost publikace vytvořené aplikace do obchodu tzv. app store. [5]

Však výhodou nativních aplikacím proti hybridním je to, že jsou vytvářeny přesně pro danou platformu a tudíž jejich vzhled a výkon bude vždy lepší. [6]

2.1.3 Mobilní webové aplikace

Poslední možností jsou mobilní webové aplikace. Tyto aplikace jsou pouze upravené webové stránky do podoby a chování nativních aplikací. Přesto, že běží pouze v prohlížeči mohou mít i tyto aplikace přístup k určitým (ne však ke všem) nativním funkcím. [1]

Tím, že jsou mobilní webové aplikace spouštěny v prohlížeči a nejsou stahovány přes obchody, je ulehčena údržba a vývoj, protože si uživatel nemusí vždy stahovat novou verzi aplikace.

Však tento postup má i své nevýhody. Jak již bylo zmíněno dříve, aplikace nemá přístup ke všem nativním funkcím daného přístroje. S dalším problémem se můžeme setkat u offline ukládání dat a to se zabezpečením, které nemusí být tak dokonalé nebo uživatelsky přívětivé, jako u nativních aplikací. [4]

2.1.4 Zvolené řešení

Pro svou práci jsem si zvolil možnost nativní mobilní aplikace z důvodu zaměření pouze na platformu iOS. Bude se tedy jednat pouze o jednu aplikaci,která bude moci využít všech nativních funkcionalit, výkonu i vzhledu.

2.2 Architektury při tvorbě iOS aplikací

Při tvorbě iOS aplikace je možné si vybrat z několika architektur. V této kapitole budu rozebírat pouze MVC, MVVM a VIPER.

2.2.1 MVC

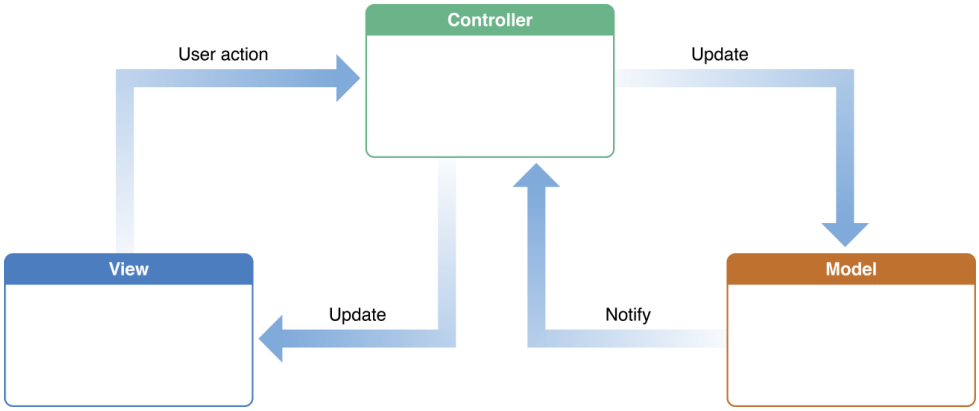
Architektura MVC je zkratka pro „Model View Controller“ neboli tři komponenty, ze kterých se architektura skládá. Jedná se o softwarovou architekturu, které se velice často používá při tvorbě aplikací s uživatelským rozhraním. [7]

- *Model* definuje jaká data aplikace obsahuje a pokud dojde k jakékoliv změně, tak informuje buď *Controller* nebo *View* (tzv. své observery). [8]
- *View* vrstva je prezentována samotnému uživateli. Tedy jsou zde zobrazena aplikační data a je zachycována uživatelova práce s aplikací.[7]
- *Controller* je vrstva mezi *View* a *Model* zabezpečující logiku aplikace. Stará se o promítnutí změn do *View* pokud se změní *Model*. Zároveň provádí úpravy v *Model* při uživatelově manipulaci s *View*. [8]

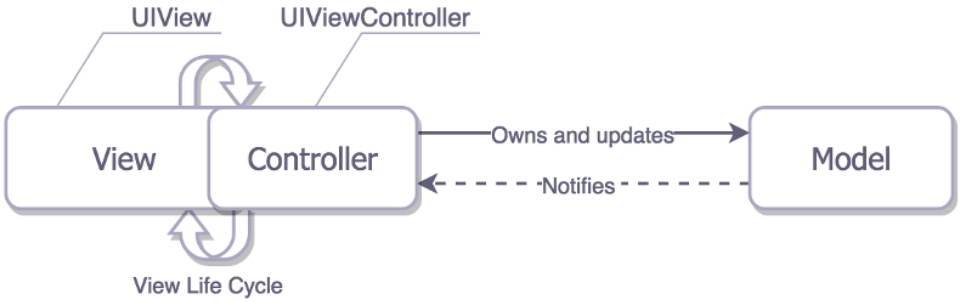
Však co se týče iOS vývoje, vrstvy *View* a *Controller* jsou téměř spojeny, protože *Controller* je příliš úzce zapojený do životního cyklu *View*. Což následně způsobuje velký nárůst *Controller*.[9]

Základní myšlenku MVC a MVC při vývoji iOS aplikace ukazují obrázky 2.1 a 2.2.

MVC je základní architekturou pro tvorbu iOS aplikací. Není však jedinou možností.



Obrázek 2.1: Model-View-Controller diagram [10]



Obrázek 2.2: Model-View-Controller při vývoji iOS aplikace [11]

2.2.2 MVVM

Architektura MVVM má obdobné koncepce jako MVC. Jedná se také o zkratku, tentokrát „Model-View-ViewModel“. [12]

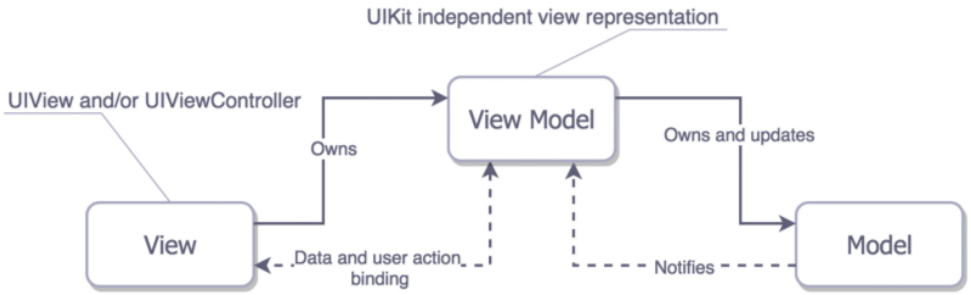
- *Model* je totožný s *Model* vrstvou architektury MVC, jedná se tedy o datovou část aplikace.
- *View* prezentuje aplikační data uživateli a monitoruje jeho akce. Však, jak již bylo zmíněno dříve, u iOS aplikací se jedná spíše o vrstvu *View/Viewcontroller*. Tato vrstva obsahuje pouze minimum logiky aplikace a reaguje hlavně na *ViewModel*. [13]

- *ViewModel* spojuje *View* a *Model* a zajišťuje hlavní logiku aplikace. *ViewModel* tedy komunikuje s *Model* a jeho metodami a následně připravuje data pro *View*. Obsahuje také implementaci funkcí, které reagují a zpracovávají akce uživatele, např.: kliknutí na tlačítko. [12]

Tedy pro shrnutí rozdílů MVC a MVVM u iOS bych zmínil to, že iOS MVC má ve výsledku téměř jen dvě vrstvy *View/Viewcontroller* a *Model*. Když potom uvažujeme architekturu MVVM *View/Viewcontroller* je opravdu pouze jednou vrstvou a mezi ní a *Model* je vložena nová vrstva *ViewModel*, která je spojuje a do které je přesunuta i většina aplikační logiky.

Mezi výhody architektury MVVM oproti MVC patří např.:

- poskytuje návrhový princip tzv. separation of concerns, neboli oddělení zájmů;
- zlepšuje možnost testovatelnosti aplikace.



Obrázek 2.3: Model-View-ViewModel architektura [14]

2.2.3 VIPER

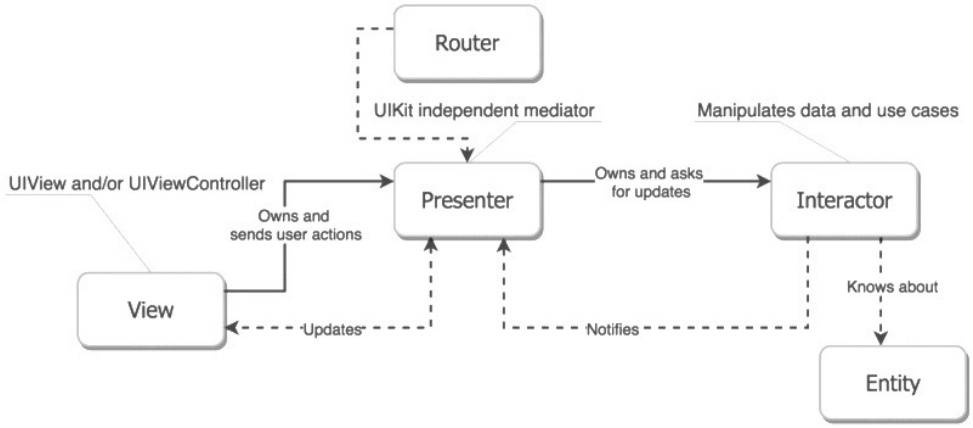
VIPER je poslední rozebíranou možností, co se týče architektur. I zde je název složen z prvních písmen jednotlivých vrstev architektury, tedy „View, Interactor, Presenter, Entity, Router“.

- *View* zobrazuje data uživateli a předává uživatelovi vstupy vrstvě *Presenter*.
- *Interactor* obsahuje logiku aplikace spojenou s daty (*Entity*).
- *Presenter* vrstva má na starosti *View* logiku. Reaguje tedy na uživatelovi akce a komunikuje s vrstvou *Interactor*, od ní také přijímá nová data. [9]
- *Entity* jsou datové objekty aplikace přístupné pouze části *Interactor*.
- *Routing* obsahuje navigační logiku. [15]

Mezi výhody architektury VIPER znovu patří např.:

- dobře rozděluje odpovědnosti;
- zlepšuje možnost testovatelnosti aplikace. [9]

Tato architektura však může být příliš náročná a přehnaná pro menší aplikace. [9]



Obrázek 2.4: VIPER architektura [16]

2.3 Funkcionálně reaktivní programování

Podle [17] je funkcionálně reaktivní programování kombinací funkcionálního a reaktivního programování. Díky němuž dokáže aplikace dynamicky měnit stav a chování v závislosti na událostech přicházejících za nějaký čas.

Pro vysvětlení, co je reaktivní programování cituji [18] „reaktivní programování je programování s asynchronními datovými toky".

Na spojení funkcionální a reaktivního programování může dívat i jako na návrhový vzor *observer*. [19] Pozorujeme tedy např. určité vstupní pole, tlačítko, nebo i dotaz na server a jsme informování o každé změně v podobě asynchronního datového toku. Na tyto datové toky je možné aplikovat funkcionální programování. Je tedy možné toky spojovat ($merge$), filtrovat ($filter$) pouze události, které nás zajímají, mapovat (map) jeden tok na nový a další. [18]

2.3.1 FRP frameworky pro iOS

V této kapitole jsou pouze rozebrány základy jednotlivých frameworků, podrobnější vysvětlení (zvoleného frameworku) společně s ukázkami jsou v k nalezení v kapitole: Realizace.

2.3.1.1 ReactiveSwift

ReactiveSwift je prvním frameworkem pro iOS podporující FRP. Obsahuje řadu základních prvků (*Signal*, *SignalProducer*, *Property*, *Action*, ...) a operátorů podporujících myšlenku „tok hodnot za čas“. [20]

2.3.1.2 ReactiveCocoa

ReactiveCocoa je další z FRP frameworků pro iOS. ReactiveCocoa rozšiřuje různé aspekty Apple Cocoa frameworku základními prvky frameworku ReactiveSwift. Umožňuje vazbu na prvky uživatelského rozhraní, u interaktivních prvků napojuje *Signal* a *Action* pro kontrolu událostí a změn. Dále také umožňuje vytvářet signály na volání metod (např. i pro UIKit třídy). [21]

2.3.1.3 RxSwift

RxSwift je Swift verzí knihovny Reactive Extensions (Rx). [22] Tato knihovna umožňuje vytvářet aplikace založené na událostech a asynchronních datových tocích pomocí tzv. Observables. [23] I přesto, že RxSwift není striktně FRP frameworkem [24], je zde uváděn a to z důvodu velkého využití knihovny Reactive Extensions i na jiných platformách např.: JavaScript, C#, Python. [25]

Kapitola 3

Realizace

Závěr

Literatura

[1] Mobile: Native Apps, Web Apps, and Hybrid Apps. *Nielsen Norman Group* [online]. United States of America: Nielsen Norman Group, © 1998-2017, [cit. 2017-09-05]. Dostupné z: https://www.nngroup.com/articles/mobile-native-apps/

[2] About Objective-C. *Apple Developer* [online]. California, U.S.: Apple Inc., © 2014, [cit. 2017-09-05]. Dostupné z: https://developer.apple.com/library/content/documentation/Cocoa/Conceptual/ProgrammingWithObjectiveC/Introduction/Introduction.html

[3] Swift. *Apple Developer* [online]. California, U.S.: Apple Inc., © 2017, [cit. 2017-09-05]. Dostupné z: https://developer.apple.com/swift/

[4] Native, HTML5, or Hybrid: Understanding Your Mobile Application Development Options. *Saleforce Developers* [online]. Suite 300, San Francisco, CA 94105, United States: Salesforce.com, inc., © 2000-2017, [cit. 2017-09-05]. Dostupné z: https://developer.salesforce.com/page/Native,_HTML5,_or_Hybrid:_Understanding_Your_Mobile_Application_Development_Options

[5] Apache Cordova. *Apache Cordova* [online]. Forest Hill, Maryland, United States: The Apache Software Foundation, © 2015, [cit. 2017-09-05]. Dostupné z: https://cordova.apache.org/

[6] Should You Build a Hybrid Mobile App? *UpWork* [online]. Mountain View, CA, US.: Upwork Global Inc., © 2015 - 2017, [cit. 2017-09-05]. Dostupné z: https://www.upwork.com/hiring/mobile/should-you-build-a-hybrid-mobile-app/

[7] MVC Architecture. *MDN web docs* [online]. Mountain View, California, United States: Mozilla and individual contributors, © 2005-2017, [cit. 2017-08-29]. Dostupné z: https://developer.mozilla.org/cs/

[8] MVC Architecture. *Developer Chrome* [online]. Silicon Valley: Google, © 2017, [cit. 2017-08-29]. Dostupné z: https://developer.mozilla.org/en-US/Apps/Fundamentals/Modern_web_app_architecture/MVC_architecture

[9] Orlov, B.: IOS Architecture Patterns. *Medium* [online], 2015, [cit. 2017-08-29]. Dostupné z: https://medium.com/ios-os-x-development/ios-architecture-patterns-ecba4c38de52

[10] Model-View-Controller. *Apple Developer* [online]. California, U.S.: Apple Inc., © 2015, [cit. 2017-08-29]. Dostupné z: https://developer.apple.com/library/content/documentation/General/Conceptual/DevPedia-CocoaCore/MVC.html

[11] Orlov, B.: Realistic Cocoa MVC. In: *Medium* [online], 2015, [cit. 2017-08-29]. Dostupné z: https://cdn-images-1.medium.com/max/800/1*PkWjDU0jqGJOB972cMsrnA.png

[12] The MVVM Pattern. *Microsoft Developer Network* [online]. Washington, U.S.: Microsoft, © 2017, [cit. 2017-08-29]. Dostupné z: https://msdn.microsoft.com/en-us/library/hh848246.aspx

[13] Morrison, J.; Schmidt, M.: IOS Design Patterns: MVC and MVVM. *CapTech*, 2014, [cit. 2017-08-29]. Dostupné z: https://www.captechconsulting.com/blogs/ios-design-patterns-mvc-and-mvvm

[14] Orlov, B.: MVVM. In: *Medium* [online], 2015, [cit. 2017-08-30]. Dostupné z: https://cdn-images-1.medium.com/max/800/1*uhPpTHYzTmHGrAZy8hiM7w.png

[15] Architecting iOS Apps with VIPER. *Objc* [online]. Berlin: Objc.io, 2013. Dostupné z: https://www.objc.io/issues/13-architecture/viper/

[16] Orlov, B.: VIPER. In: *Medium* [online], 2015, [cit. 2017-08-30]. Dostupné z: https://cdn-images-1.medium.com/max/800/1*0pN3BNTXfwKbf08lhwutag.png

[17] Functional Reactive Programming (FRP). *Technopedia* [online]. Techopedia Inc., © 2017, [cit. 2017-09-01]. Dostupné z: https://www.techopedia.com/definition/29571/functional-reactive-programming-frp

[18] Staltz, A.: The introduction to Reactive Programming you've been missing. In *GitHubGist*, California, US.: GitHub, 2014, [cit. 2017-09-01]. Dostupné z: https://gist.github.com/staltz/868e7e9bc2a7b8c1f754

[19] Blackheath, S.; Jones, A.: *Functional reactive programming*. United States: Manning Publications, 2016, ISBN 978-163-3430-105.

[20] ReactiveSwift. *GitHub* [online]. California, US.: GitHub Inc., © 2017, [cit. 2017-09-01]. Dostupné z: `https://github.com/ReactiveCocoa/ReactiveSwift`

[21] ReactiveCocoa. *GitHub* [online]. California, US.: GitHub Inc., © 2017, [cit. 2017-09-01]. Dostupné z: `https://github.com/ReactiveCocoa/ReactiveCocoa`

[22] RxSwift: ReactiveX for Swift. *GitHub* [online]. California, US.: GitHub Inc., © 2017, [cit. 2017-09-04]. Dostupné z: `https://github.com/ReactiveX/RxSwift`

[23] Reactive Extensions. *GitHub* [online]. California, US.: GitHub Inc., © 2017, [cit. 2017-09-04]. Dostupné z: `https://github.com/Reactive-Extensions/Rx.NET`

[24] ReactiveX. *ReactiveX* [online], [cit. 2017-09-04]. Dostupné z: `http://reactivex.io/intro.html`

[25] ReactiveX. *ReactiveX* [online], [cit. 2017-09-04]. Dostupné z: `http://reactivex.io/`

Příloha **A**

Seznam použitých zkratek

FRP funkcionálně reaktivní programování

MVC Model View Controller

MVVM Model View ViewModel

VIPER View Interactor Presenter Entity Router

Příloha **B**

Obsah přiloženého CD

- `readme.txt` ..... stručný popis obsahu CD
- `exe` ..... adresář se spustitelnou formou implementace
- `src`
  - `impl` ..... zdrojové kódy implementace
  - `thesis` ..... zdrojová forma práce ve formátu LaTeX
- `text` ..... text práce
  - `thesis.pdf` ..... text práce ve formátu PDF
  - `thesis.ps` ..... text práce ve formátu PS