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

4. října 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 4. října 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 diplomové 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

Seznam tabulek

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. Tyto 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 se 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

Tvorba iOS aplikací

2.1 Možnosti vývoje

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. Přestože 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

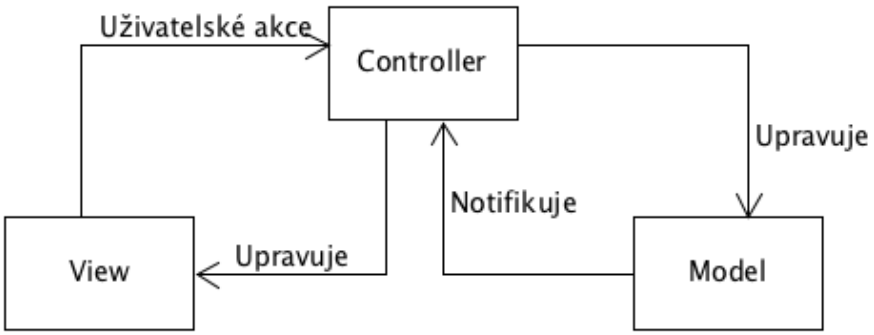
Architektura MVC je zkratka pro „Model View Controller“ neboli tři komponenty, ze kterých se architektura skládá. Jedná se o softwarovou architekturu, které se velice často používá při tvorbě aplikací s uživatelským rozhraním. [7]

- *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]

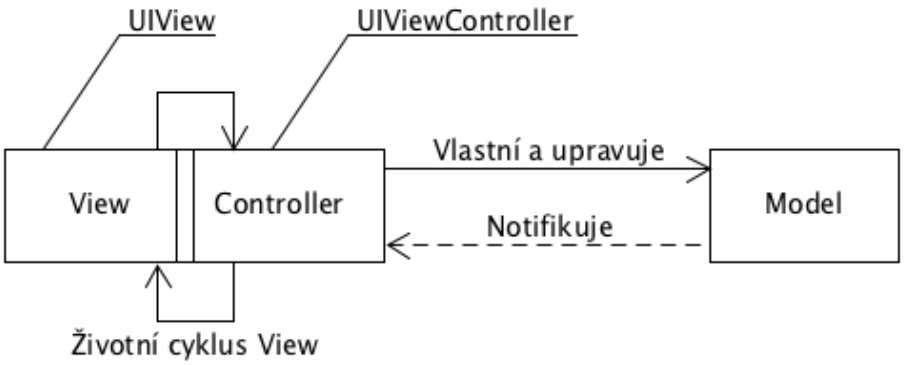
Však co se týče iOS vývoje, vrstvy *View* a *Controller* jsou téměř spojeny, protože *Controller* je příliš úzce zapojený do životního cyklu *View*, což následně způsobuje velký nárůst *Controller*.[9]

Základní myšlenku MVC a MVC při vývoji iOS aplikace ukazují obrázky 2.1 (převzato a přeloženo z originálu [10]) a 2.2 (převzato a přeloženo z originálu [11]).

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



Obrázek 2.1: Model-View-Controller diagram



Obrázek 2.2: Model-View-Controller při vývoji iOS aplikace

2.2.2 MVVM

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

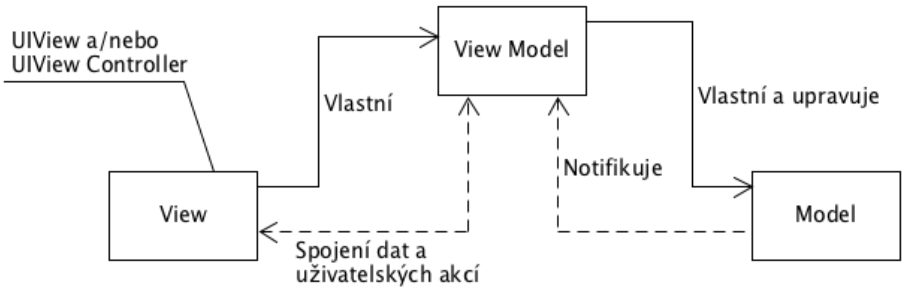
- *Model* je totožný s *Model* vrstvou architektury MVC, jedná se tedy o datovou část aplikace.
- *View* prezentuje aplikační data uživateli a monitoruje jeho akce. Však, jak již bylo zmíněno dříve, u iOS aplikací se jedná spíše o vrstvu *View/View controller*. Tato vrstva obsahuje pouze minimum logiky 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/View controller* a *Model*. Když potom uvažujeme architekturu MVVM ,*View/View controller* 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.

Architekturu MVVM zobrazuje obrázek 2.3 (převzato a přeloženo z originálu [14]).



Obrázek 2.3: Model-View-ViewModel architektura

2.2.3 VIPER

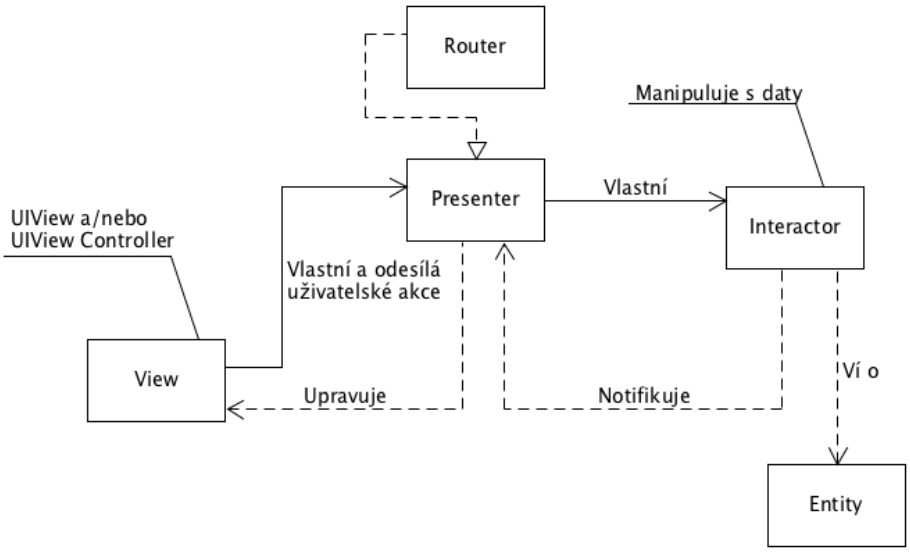
VIPER je poslední rozebíranou možností, co se týče architektur. I zde je název složen z prvních písmen jednotlivých vrstev architektury, tedy „View, Interactor, Presenter, Entity, Router“.

- *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 uživatelovy 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 zbytečně složitá pro menší aplikace. [9]
Architekturu VIPER zobrazuje obrázek 2.4 (převzato a přeloženo z originálu [16]).



Obrázek 2.4: VIPER architektura

2.3 Funkcionálně reaktivní programování

Funkcionálně reaktivní programování je kombinací funkcionálního a reaktivního programování, díky němuž dokáže aplikace dynamicky měnit stav a chování v závislosti na událostech přicházejících za daný čas. [17]

Pro vysvětlení, co je reaktivní programování cituji André Staltze: „reaktivní programování je programování s asynchronními datovými toky“. [18]

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*) např. 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 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]

2.4 Perzistence dat

Perzistence dat, neboli jejich uchování a uložení, je velice důležitou funkcionalitou většiny aplikací. V této kapitole jsou rozebírány tři možnosti zajišťující perzistenci dat iOS aplikací – Core Data, iCloud a Realm.

2.4.1 Core Data

Core Data je Apple framework, který má na starosti model vrstvu aplikace. Stará se o životní cyklus objektů, jejich vztahy (objektový graf) i perzistenci. [26]

Core Data má více možností na způsob uložení např. SQLite a XML. Jedná se tedy o uložení dat na disku daného zařízení. [27]

Výhodou tohoto frameworku je to, že je zcela zdarma a má podporu přímo v Xcode. [28]

2.4.2 iCloud

iCloud je cloudové úložiště od společnosti Apple. Umožňuje ukládat aplikační data i dokumenty a přistupovat k nim na všech Apple zařízeních a na webu. [29]

Při vývoji iOS aplikací se pro využití iCloud používá framework CloudKit. CloudKit zajišťuje rozhraní pro komunikaci dané aplikace a iCloud. [30] Poskytuje ověření uživatele, tři druhy databáze – soukromou, veřejnou a sdílenou, dále také analytický nástroj CloudKit Dashboard, který umožňuje prozkoumání dat, měření aktivity uživatelů a další. [31]

CloudKit je dostupný pro členy Apple Developer programu. [32] Tento program stojí ročně v přepočtu 2150 Kč. [33]

2.4.3 Realm

Realm, s oficiální stránkou `https://realm.io`, je multiplatformní mobilní databáze, která je připravená pro jazyky Java (Android), Swift, Objective-C, JavaScript a Xamarin. Hlavní myšlenkou je kontejner objektů tzv. Realm. V těchto kontejnerech jsou uložena data, na které je možné se dotazovat, tyto data filtrovat a podobně. Na rozdíl od klasických např. SQL databází, zde pracujeme přímo s „živými“ objekty, tedy pokud provedeme změnu není nutné ukládat změněný objekt do databáze, ale změna se provede automaticky.

Realm je rozdělený na dvě části – mobilní databáze Realm (Realm Mobile Database) a objektový server Realm (Realm Object Server). Jak je již z názvů možné usuzovat mobilní databáze je pouze na mobilním zařízení, jedná se tedy o offline uložení dat. Pokud však chceme data např. sdílet na více zařízení, je možné se připojit na objektový server Realm a s tím se synchronizovat. Realm se řídí strategii „nejprve offine“ – čtení a zápis probíhá nejprve lokálně a až poté probíhá synchronizace se serverem.

Jedna aplikace může využívat hned několik Realm kontejnerů, a to jak lokální, tak i vzdálené, kde každý z nich může mít různá oprávnění pro různé uživatele.

Realm Mobile Database je open source, tedy zdarma. [34]

Kapitola 3

Analýza

3.1 EASA

EASA, neboli European Aviation Safety Agency, je agentura spadající pod Evropskou Unii, která má na starosti technické přepisy, bezpečnost, regulace a certifikace v oboru letectví. [35]

Pro tuto diplomovou práci je EASA důležitá, protože vydává i pokyny např. pro evidenci letů nebo limity odlétaných hodin. [36]

3.1.1 Pokyny pro evidenci letů

Pokyny pro evidenci letů udává předpis FCL.050. Tento předpis specifikuje povinné položky každého leteckého záznamu. [37]

„ Každý záznam letů by měl obsahovat minimálně tyto informace:

1. osobní informace: jméno a adresu pilota;
2. každý záznam letu by měl obsahovat:
   - jméno velícího pilota (PIC – Pilot-in-command),
   - datum letu,
   - čas a místo odletu a příletu,
   - typ, značku, model, variantu a registraci letadla,
   - označení zda je letadlo jednomotorové (SE – single engine) nebo vícemotorové (ME – multi engine),
   - čas letu,
   - celkový čas letu.
3. každý záznam z výcvikového zařízení pro simulaci letu (FSTD – flight simulation training devices) by měl obsahovat:

- typ a kvalifikační číslo výcvikového zařízení,
- instrukce výcvikového zařízení pro simulaci letu,
- datum,
- čas,
- celkový čas.

4. funkce pilota – velící pilot (včetně sólového, velícím pilotem student (Student PIC) nebo velící pilot pod dohledem (PICUS – pilot-in-command under supervision)), druhý pilot, dvojí pilot (dual), instruktor (FI – Flight Instructor) nebo zkoušející (FE – Flight Examiner);

5. provozní podmínky – pokud se let uskutečnil v noci nebo pokud byl prováděn podle pravidel pro let podle přístrojů.

“ [37] (překlad vlastní)

3.1.2 Limity

Limity letového času a času ve službě obsahuje předpis ORO.FTL.210.

„ Celková doba služby, na kterou může být člen posádky přidělen, nesmí překročit:

1. 60 hodin služby za 7 po sobě jdoucích dnů;

2. 110 hodin služby za 14 po sobě jdoucích dnů; a

3. 190 hodin služby za 28 po sobě jdoucích dnů, rozdělených co nejrovnoměrněji během tohoto období.

Celkový čas, na který je jedinec přidělen jako člen provozní posádky, nesmí překročit:

1. 100 hodin letu za 28 po sobě jdoucích dnů;

2. 900 hodin letu v kalendářním roce; a

3. 1000 hodin letu během 12 po sobě jdoucích kalendářních měsících.

Poletová služba se počítá do doby služby. “ [38] (překlad vlastní)

3.1.3 Zdravotní certifikáty

Informace o zdravotních certifikátech obsahuje předpis Part-MED. Certifikáty jsou tří druhů – zdravotní certifikát třídy 1 (Class 1 medical certificate), zdravotní certifikát třídy 2 (Class 2 medical certificate) a zdravotní certifikát pro licence na lehká letadla (LAPL – Light Aircraft Pilot Licence). Každý z těchto

certifikátů má jinak nastavenou dobu platnosti a je pro jiné typy pilotních licencí.

LAPL certifikát je pouze pro pilotní licence na lehká letadla. Platnost je 60 měsíců u pilotů do věku 40 let, poté je platnost pouze 24 měsíců.

Zdravotní certifikát třídy 2 je pro pilotní licence PPL (Private Pilot Licence), SPL (Sailplane Pilot Licence) a BPL (Balloon Pilot Licence), tedy pro piloty soukromých letadel, kluzáků a balónů. Platnost licence se znovu odvíjí od věku pilota – 60 měsíců u pilotů do věku 40 let, následně 24 měsíců do věku 50 let a nakonec platnost licence klesá na 12 měsíců.

Zdravotní certifikát třídy 1 je certifikát nejvyšší úrovně. Je pro pilotní licence CPL (Commercial Pilot Licence), MPL (Multi-crew Pilot Licence) a ATPL (Airline Transport Pilot Licence), tedy pro piloty komerčních, vícečlenných a dopravních letadel. Platnost licence je 12 měsíců. To neplatí u pilotů starších 40 let, létajících jednopilotní komerční lety s cestujícími nebo u pilotů starších 60 let, zde se platnost licence snižuje na 6 měsíců. [39]

3.2 Analýza existujících aplikací pro evidenci letů

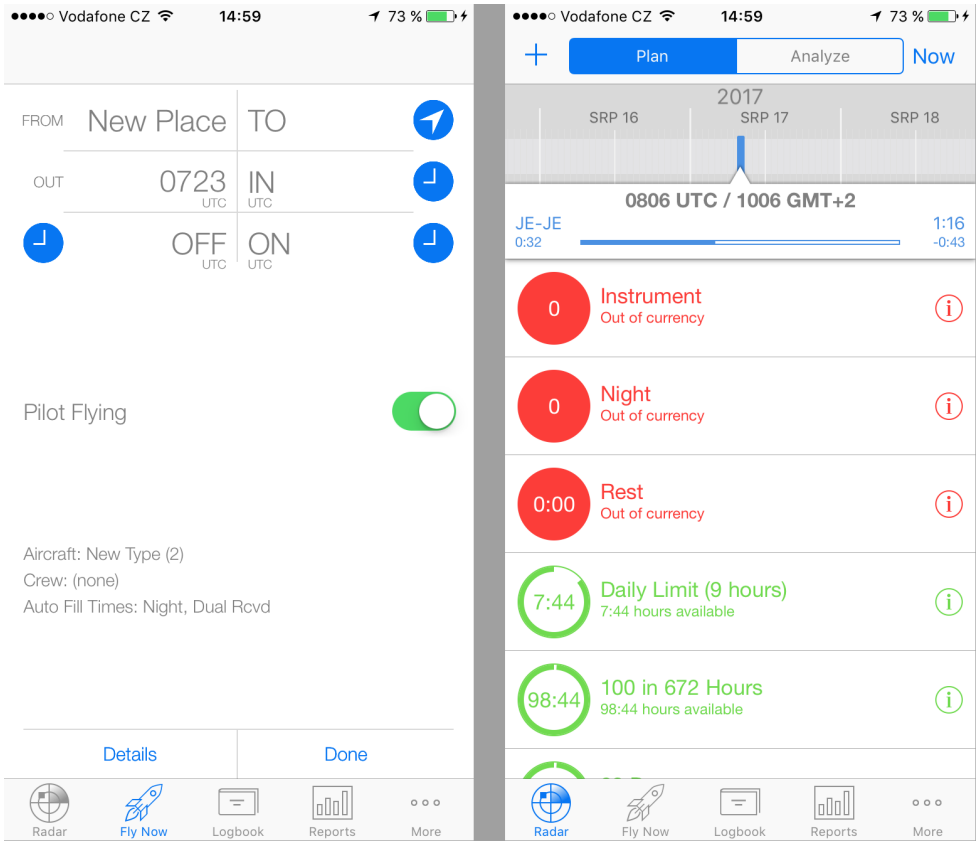
Tato kapitola se zabývá analýzou již existujících aplikací pro evidenci letů. Pro analýzu bylo vybráno pět aplikací – tři pro iOS a dvě pro platformu Android.

1. LogTen Pro X – iOS aplikace v angličtině vyvíjená společností Coradine Aviation. [40]
2. Logbook Pro Aviation Flight Log for Pilots – druhá iOS aplikace, také v angličtině, vytvořená NC Software, Inc. [41]
3. Safelog Pilot Logbook – poslední z analyzovaných iOS aplikací. I tato aplikace je v anglickém jazyce. Publikována Dauntless Software. [42]
4. FlyLogio - Pilot Logbook – česká aplikace vyvinutá pro platformu Android společností FlyLogio.com. [43]
5. Smart Logbook – anglická Andriod aplikace vydána firmou Kviation, Inc. [44]

3.2.1 Kritéria hodnocení

Kritéria hodnocení byla rozdělena do několika kategorií – přihlášení, platby, napojení externích databází, doplňkové položky při vkládání záznamu, limity a certifikáty, reporty a zálohování.

Při vkládání záznamu jsou brány v potaz pouze doplňkové položky, protože aplikace bude tvořena podle předpisu EASA FCL.050, který udává povinné údaje při evidování letu. Také položka v hodnocení – reporty podle EASA, je analyzována z pohledu předpisu FCL.050.



Obrázek 3.1: LogTen Pro X

Všechny položky hodnocení jsou uváděny z pohledu mobilní/tablet aplikace.

3.2.2 Srovnávací tabulka

Tabulka 3.1: Srovnávací tabulka

Kritéria	Log Ten Pro X	Logbook Pro	Safelog	FlyLogio	Smart Logbook	Výsledek
Přihlášení						
Aplikace funkční bez přihlášení	✓	✗	✗	✗	✗	1/5
Možnost přihlášení	✓	✓	✓	✓	✓	5/5
Platby						

Tabulka 3.1: Srovnávací tabulka

Kritéria	Log Ten Pro X	Logbook Pro	Safelog	FlyLogio	Smart Logbook	Výsledek
Platby jednorázové	✗	✗	✗	✗	✓	1/5
Opakované platby	✓	✓	✓	✓	✓	5/5
Napojení externích databází						
Napojení na databázi letišť	✗	✓[1]	✓	✓	✓	4/5
Napojení na databázi letadel	✗	✗	✓	✓	✗	2/5
Doplňkové položky při vkládání záznamu						
Možnost přidání fotky	✓	✗	✓	✗	✗	2/5
Možnost přidání dokumentu	✓	✗	✓	✗	✗	2/5
Limity a certifikáty						
Kontrola limitů	✓	✓	✓[2]	✗	✓	4/5
Certifikáty	✓	✓	✓[3]	✗	✓	4/5
Reporty						
Generování reportů	✓	✗	✓[4]	✗	✓	3/5
Reporty podle EASA	✗	✗	✓	✗	✓	2/5
Jiné reporty	✓	✓	✓	✗	✓	2/5
Perzistence dat						
iCloud/Google	✓	✗	✗	✓	✓	3/5
Vlastní řešení	✗	✓	✓	✗	✗	2/5
Synchronizace více zařízení	✓	✓	✓	✓	✓	5/5

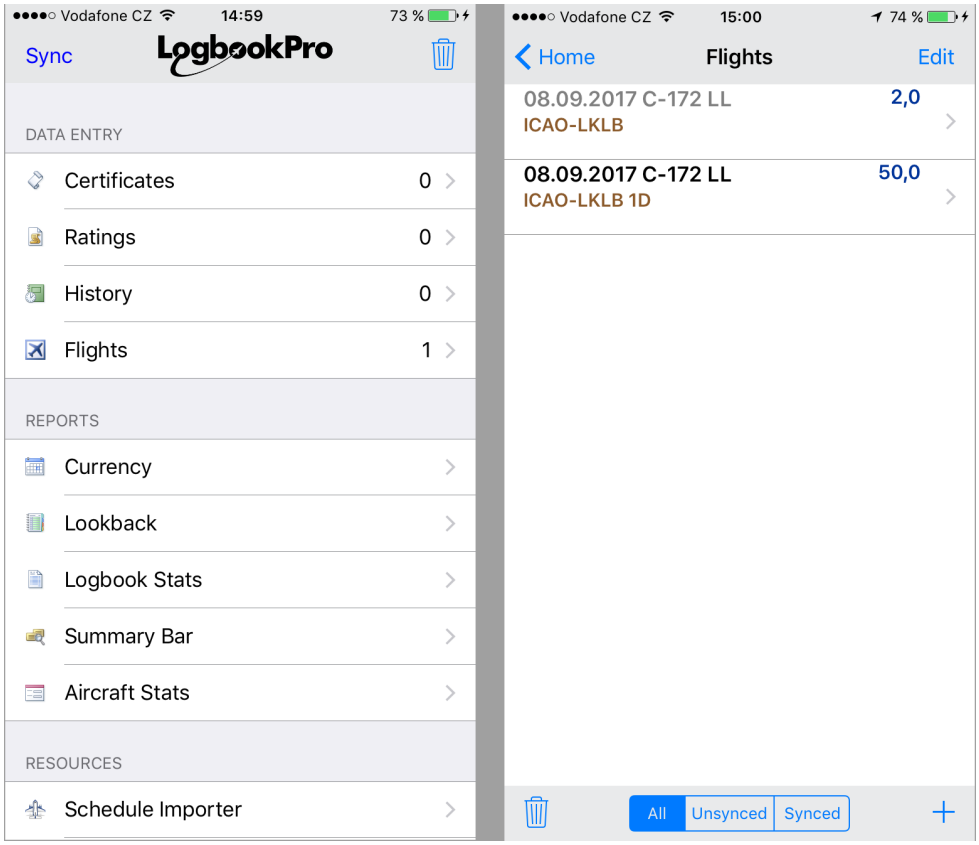
3.2.3 Výsledky a vlastní zhodnocení

3.2.3.1 LogTen Pro X

LogTen Pro X je z analyzovaných iOS aplikací nejvíce uživatelsky přívětivá. Zobrazuje přehledně limity a certifikáty, i vkládání je intuitivní. Má však i několik nedostatků:

- není napojená na databázi letišť, tudíž uživatel musí vyplnit všechny informace o daném letišti sám, bez automatického doplnění nebo našeptávání;

[1] Logbook Pro umí nalézt pouze nejbližší letiště.
[2] Safelog zobrazuje limity ve webové verzi.
[3] Safelog zobrazuje certifikáty ve webové verzi.
[4] Safelog zobrazuje reporty ve webové verzi.



Obrázek 3.2: Logbook Pro Aviation Flight Log for Pilots

- neumožňuje generování reportů podle předpisu EASA FCL.050;
- aplikace je placená ročně –
  - iPhone + iPad + Mac – 3550 Kč,
  - Mac – 3550 Kč,
  - iPhone + iPad – 2150 Kč.

Data o aplikaci a cenách jsou získány přímo z aplikace LogTen Pro X.

3.2.3.2 Logbook Pro Aviation Flight Log for Pilots

Aplikace Logbook Pro vyžaduje pro přihlášení stažení PC aplikace (pouze pro Windows). S touto aplikací je následně synchronizován. Některá funkcionalita, např. generování reportů, je dostupná pouze v PC verzi.

PC verze aplikace je zadarmo pouze ve zkušební verzi, poté základní verze stojí v přepočtu 1800 Kč. iOS verze aplikace se platí ročně v přepočtu za 1045 Kč, je nutné si zaplatit i zálohování a další funkcionality. [45]

Obrázek 3.3: Safelog Pilot Logbook

3.2.3.3 Safelog Pilot Logbook

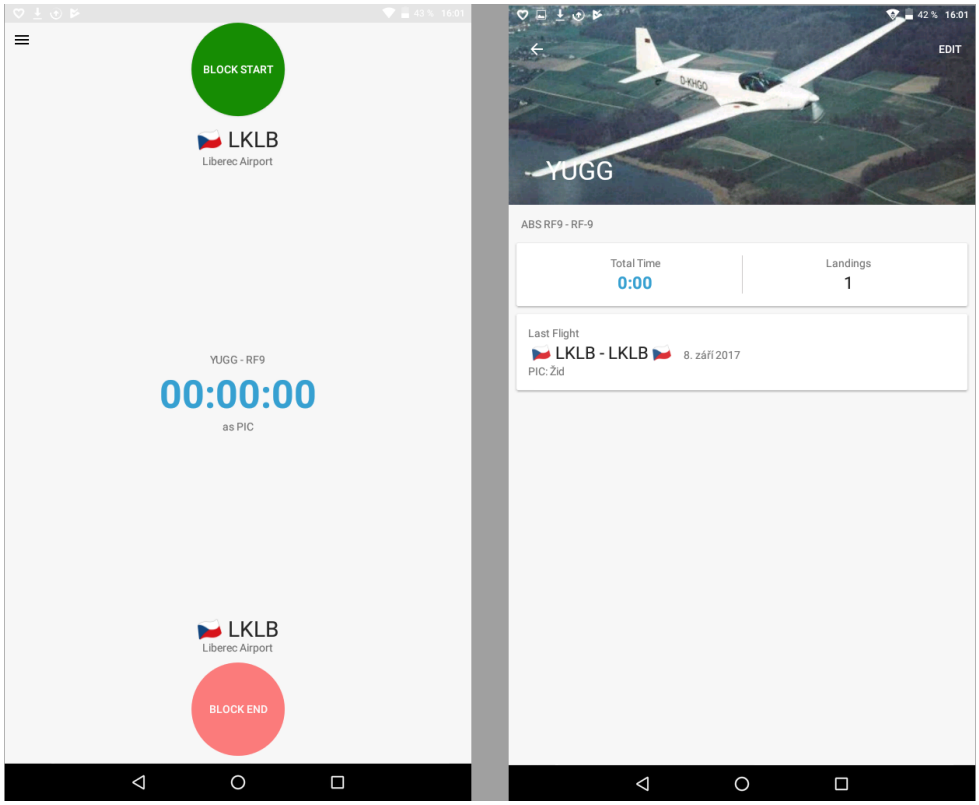
Safelog Pilot Logbook obsahuje pouze některé funkce přímo v aplikaci, u ostatních je uživatel odkázán do webového rozhraní (SafelogWeb Cloud) viz. 3.3. Toto webové rozhraní zobrazené v aplikaci však není přizpůsobené pro mobilní zařízení, často je zobrazena pouze část stránky a není možné např. vyplnit všechna pole formuláře.

Tato aplikace však, pokud budeme brát v potaz i funkce ve webovém rozhraní, obsahuje nejširší spektrum funkcionalit.

Samotná aplikace je zadarmo, ale pro plnou verzi aplikace je nutné předplatné. To se pohybuje od 1320 Kč za jeden rok až po 8990 Kč za deset let. Data o aplikaci a cenách jsou získány z aplikace Safelog Pilot Logbook.

3.2.3.4 FlyLogio - Pilot Logbook

Android aplikace FlyLogio - Pilot Logbook je jedinou aplikací kompletně zadarmo. Z uživatelského pohledu se jedná o přehlednou a jednoduchou aplikaci.



Obrázek 3.4: FlyLogio

Však neobsahuje takovou funkcionalitu jako ostatní placené aplikace, např. chybí generování jakýchkoliv reportů nebo kontrolování limitů.

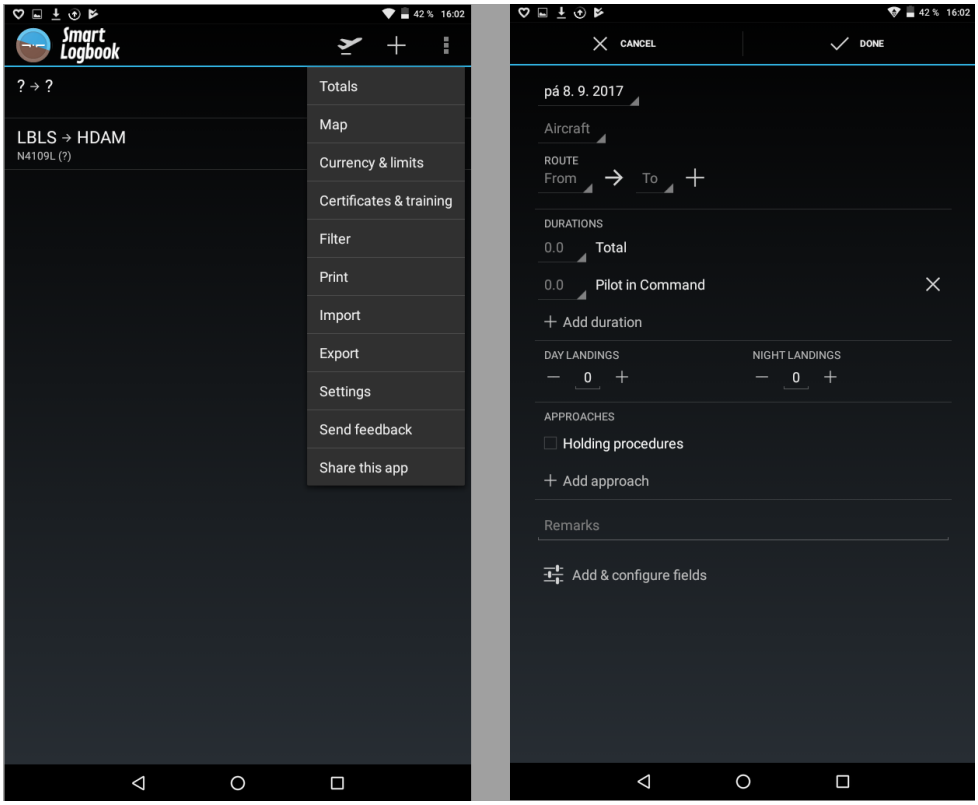
3.2.3.5 Smart Logbook

Smart Logbook je poslední analyzovanou aplikací, jedná se o Android aplikaci. Obsahuje mnoho funkcionalit – generování reportů podle mnoha norem, zobrazení mapy se zaznamenáním jednotlivých letů, hlídání limitů i expirace certifikátů.

Tato aplikace je zadarmo pouze ve zkušební verzi, následně je nutné aplikaci zakoupit za 300 Kč. Je nutné také platit za zálohování a synchronizaci dat, a to buď 20 Kč za měsíc, nebo 120 Kč za rok. Informace o cenách jsou získány z aplikace Smart Logbook.

3.2.3.6 Závěr analýzy

Tato analýza posloužila při návrhu funkcionalit iOS aplikace ve formě případů užití, při návrhu uživatelského rozhraní a také při výběru vhodného nástroje na perzistenci dat.



Obrázek 3.5: Smart Logbook

3.3 Analýza požadavků

3.3.1 Funkční požadavky

Funkční požadavky jsou uvedeny pouze jmenovitě. Podrobnější popis je uveden v podobě případů užití.

1. Evidování leteckých záznamů.
2. Vyhledávání v leteckých záznamech.
3. Kontrola limitů.
4. Evidence zdravotních certifikátů.
5. Generování reportů do formátu PDF.

3.3.2 Nefunkční požadavky

1. Funkční pro iOS 11 – aplikace bude dostupná pro verzi operačního systému iOS 11.

2. Aplikace pro iPhone a iPad – uživatelské rozhraní bude přizpůsobeno jak telefonům, tak tabletům.

3. Stejná data na více uživatelových zařízeních – všechna data budou zálohována online a uživatel k nim bude mít přístup na všech zařízeních s iOS 11 pod svým účtem.

4. Dostupnost dat offline – uživatel bude mít přistup ke všem dříve staženým/vytvořeným záznamům. Aplikaci bude možné používat i bez internetového připojení, k synchronizaci dojde až ve chvíli, kdy bude internetové připojení k dispozici.

5. Aplikace podle EASA – aplikace se bude řídit předpisy EASA a to přesně: FCL.050 u evidence letů, ORO.FTL.210 v případě limitů a Part-MED při evidenci zdravotních certifikátů.

Kapitola 4

Návrh

4.1 Navržená funkcionalita v podobě případů užití

Případy užití (use cases) byly využity při návrhu funkcionalit iOS aplikace. Jsou obsaženy přímo v práci z důvodu přehledného zobrazení navržených funkcionalit a možnosti dovysvětlením u některých z nich.

Diagram případů užití zobrazuje obrázek 4.1

4.1.1 Vytvořit záznam letu

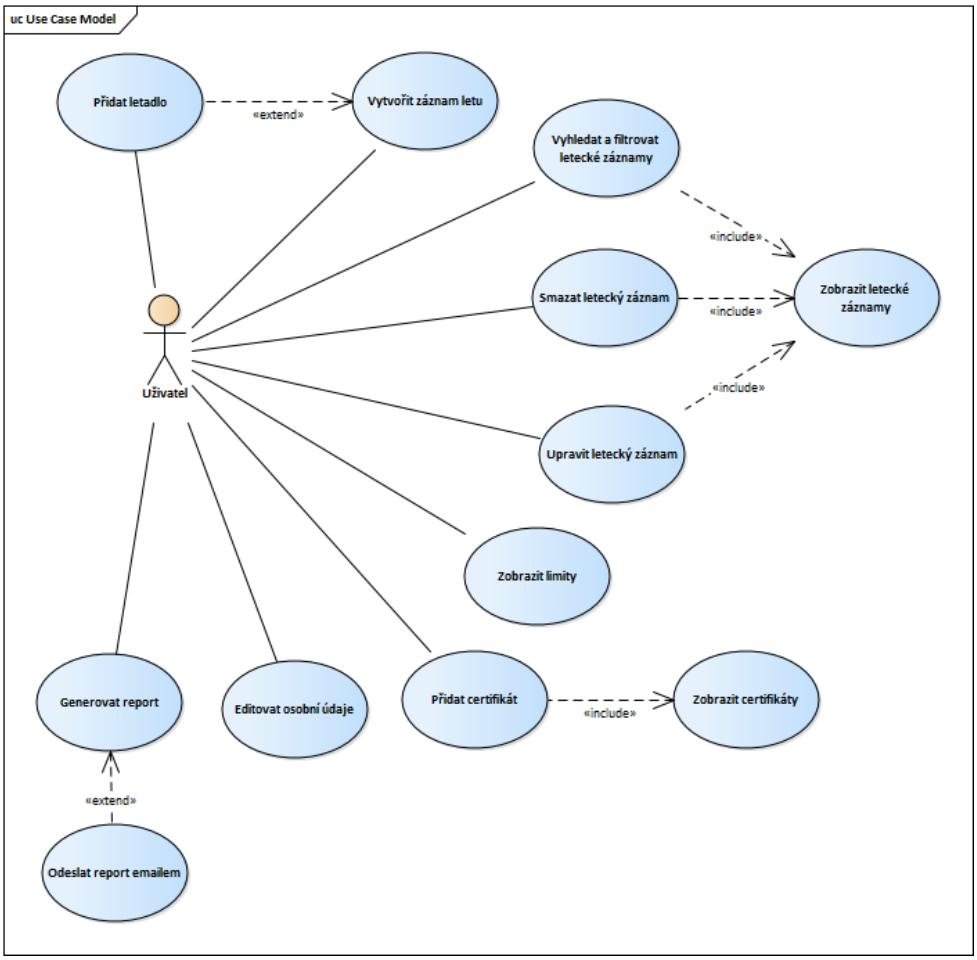
Vytvoření leteckého záznam umožňuje uživateli vložit nový záznam letu do aplikace.

Hlavní scénář –

- Případ užití začíná, když chce uživatel evidovat svůj let.
- Systém zobrazí formulář umožňující zadat: jméno velícího pilota, datum letu, čas a místo odletu a příletu, letadlo, čas letu, celkový čas letu, počet vzletů a přistání, pilotovu funkci při letu a provozní podmínky.
- Uživatel vyplní všechny povinné položky.
- Aplikace uloží informace o letu.

Alternativní scénář –

- Případ užití začíná, když chce uživatel evidovat záznam z výcvikového zařízení pro simulaci letu.
- Systém zobrazí formulář umožňující zadat: typ a kvalifikační číslo výcvikového zařízení, datum a čas.
- Případ užití pokračuje 3. krokem hlavního scénáře.



Obrázek 4.1: Případy užití

Aplikace bude napojena na databázi letišť, pro jednoduché evidování místa odletu a příletu. Nebude však napojena na databázi letadel, protože uživatel u každého letadla musí vyplnit minimálně registrační číslo. Pokud by uživatel musel letadlo najít a zeditovat, je výhodnější pokud si sám letadlo přidá. Dalším důvodem k tomu rozhodnutí je, že pilot létá velmi často pouze s jedním letadlem a proto funkcionalitu přidání letadla nebude používat příliš často.

4.1.2 Přidat letadlo

Přidání letadla dává uživateli možnost přidat letadlo, které pak může vkládat do záznamů o letu.

- Případ užití začíná, když chce uživatel přidat nové letadlo.

- Systém zobrazí formulář umožňující zadat: typ, značku model, variantu, registrační číslo letadla a zda je letadlo jednomotorové nebo vícemotorové.
- Uživatel vyplní všechna pole formuláře.
- Aplikace uloží letadlo.

4.1.3 Zobrazit letecké záznamy

Zobrazení leteckých záznamů zobrazuje jednotlivé záznamy v podobě tabulky, kde u každého záznamu jsou vidět základní informace. Mezi tyto informace patří: místo odletu a přílet, datum a čas letu.

4.1.4 Vyhledat a filtrovat letecké záznamy

Tato funkcionalita umožňuje uživateli vyhledávání a filtrování leteckých záznamů.

- Případ užití začíná, pokud chce uživatel vyhledat nebo vyfiltrovat letecké záznamy.
- Include (Zobrazit letecké záznamy).
- Aplikace zobrazí formulář, který umožňuje: zadat hledaný text, nastavit zda se jedná o záznam letu nebo o záznam z výcvikového zařízení, zvolit letadlo, nastavit délku letu, zvolit datum příletu a odletu, nastavit uživatelovu funkci pilota a zvolit provozní podmínky.
- Uživatel vyplní pole, podle kterých chce vyhledávat/filtrovat.
- Systém zobrazí pouze záznamy odpovídající zvoleným parametrům.

4.1.5 Smazat letecký záznam

Smazání leteckého záznamu umožňuje uživateli smazat letecký záznam, který předtím sám vytvořil.

- Případ užití začíná, když chce uživatel smazat jeden ze svých leteckých záznamů.
- Include (Zobrazit letecké záznamy).
- Uživatel si zvolí záznam, který chce smazat.
- Aplikace zobrazí potvrzovací dialog.
- Uživatel potvrdí smazání.
- Aplikace odstraní položku ze seznamu.

4.1.6 Upravit letecký záznam

Upravení leteckého záznamu umožňuje uživateli upravit všechny položky zvoleného letecké záznamu.

- Případ užití začíná, když chce uživatel upravit letecký záznam.
- Include (Zobrazit letecké záznamy).
- Uživatel si zvolí záznam, který chce upravit.
- Scénář pokračuje krokem 2 Vytvořit záznam letu.

4.1.7 Zobrazit limity

Tato funkcionalita slouží ke zobrazení limitů a kontrole zda jsou všechny limity v normě.

4.1.8 Zobrazit certifikáty

Tato funkcionalita umožňuje uživateli zobrazit všechny jeho certifikáty, společně s kontrolou platnosti a počtem dní do jejich expirace.

4.1.9 Přidat certifikát

Tato funkce umožňuje uživateli přidat certifikát a to buď dle šablony pro zdravotní certifikáty (LALP, třídy 1 a třídy 2), nebo vlastní.

- Případ užití začíná, když chce uživatel vytvořit nový certifikát.
- Include (Zobrazit certifikáty).
- Aplikace zobrazí formulář s možností vytvoření vlastního certifikátu nebo dle šablony. Ve formuláři je následně možné zadat: název certifikátu, datum vydání, datum expirace a popis.
- Uživatel povinně vyplní název a datum expirace.
- Aplikace uloží certifikát.

4.1.10 Editovat osobní údaje

Editace osobních údajů umožňuje uživateli upravit své osobní informace. Mezi tyto informace patří: jméno a příjmení, adresa a věk.

U osobních údajů je důležitý hlavně věk, který hraje roli u zdravotních certifikátů.

4.1.11 Generovat report

Generování reportu umožňuje uživateli vygenerovat report ve formátu PDF z nímž zvolených záznamů.

- Případ užití začíná, jestliže se uživatel rozhodne vygenerovat report.
- Aplikace zobrazí formulář s možným výběrem záznamů, které se v reportu objeví.
- Uživatel si zvolí záznamy.
- Aplikace vytvoří report ve formátu PDF se zvolenými záznamy letů.

4.1.12 Odeslat report emailem

Funkcionalita odeslání reportu emailem umožňuje uživateli, poté co vygeneroval report, odeslat tento report přes email.

4.2 Návrh uživatelského rozhraní

Návrh uživatelského rozhraní přímo navazuje na návrh funkcionalit. Pro tento návrh byl použit wireframe.

4.2.1 Wireframe

Wireframe je technika, která se používá v brzké fázi vývoje softwaru [46] pro rozvržení základních prvků obrazovek a navržení hlavních průchodů aplikací (navigace). Protože se jedná pouze o návrh, nemusí wireframe podporovat dynamicky měnící se obsah nebo např. nemusí zobrazovat chybové hlášky.

Wireframe může posloužit k první heuristické analýze a k uživatelským testům ještě před vývojem dané aplikace. [47]

Wireframe pro tuto práci byl vytvořen ve webovém nástroji NinjaMock (`https://ninjamock.com`). Z tohoto nástroje byl wireframe následně exportován do formátu pdf a ninjamock package. Tyto exporty jsou k nalezení v příloze této diplomové práce.

Exportovaný wireframe ve formátu pdf je uložený vždy ve dvou provedeních: se zobrazeným mobilním zařízením a bez něj, je tomu tak, protože ve verzi se zobrazeným mobilním zařízením u obrazovek s více položkami (např. přidání záznamu letu) nejsou všechny položky vidět. U provedení bez zobrazeného mobilního zařízení jsou vždy všechny položky vidět, ale wireframe působí neúplným dojmem. Pro úplnost je přidána verze ninjamock package, kterou je možné importovat do webového nástroje NinjaMock. V této verzi jsou funkční i odkazy mezi jednotlivými obrazovkami.

4.2.2 Heuristická analýza

Heuristická analýze je metoda, která se používá pro hledání problémů použitelnosti (usability problems) v uživatelském rozhraní. [48] Jakob Nielsen vytvořil deset základních principů pro uživatelské rozhraní –

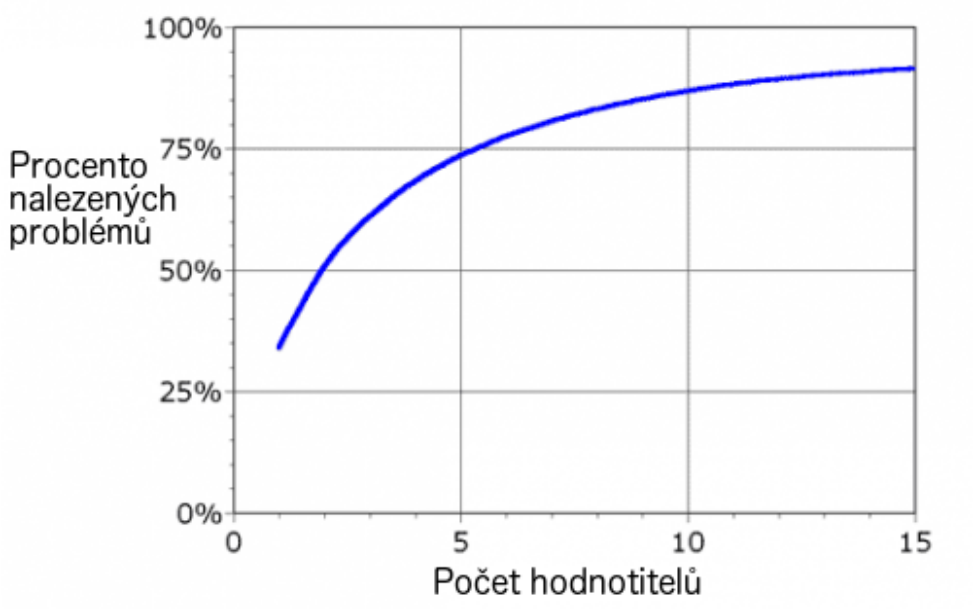
1. „viditelnost stavu systému,
2. spojení systému a reálného světa,
3. uživatelská kontrola a volnost,
4. konzistence a standardizace,
5. předcházení chyb,
6. rozpoznání místo vzpomínání,
7. flexibilita a efektivita použití,
8. estetika a minimalismus,
9. pomoci uživatelům rozpoznat, pochopit a vzpamatovat se z chyb,
10. nápověda a dokumentace“ [49] (překlad vlastní).

Tato pravidla byla použita při heuristické analýze vytvořeného wireframe. Však nebyla kontrolována pravidla:

- „předcházení chyb“ – z důvodu, že ve fázi návrhu není možné např. vyplnit formulář a nebo smazat záznam letu;
- „pomoci uživatelům rozpoznat, pochopit a vzpamatovat se z chyb“ – také není možné vyvolat chybu;
- „nápověda a dokumentace“.

Článek Jakoba Nielsena [48] doporučuje více hodnotitelů, kteří provedou nezávisle heuristickou analýzu nad daným návrhem uživatelského rozhraní. Počet nalezených problémů v závislosti na počtu hodnotitelů zobrazuje obrázek 4.2. Z grafu je možné usuzovat vhodný počet hodnotitelů pět až šest. Protože se však nyní jedná pouze o návrh, kde není možné testovat všechna pravidla, byli použiti pouze dva hodnotitelé.

Po provedení heuristické analýzy byla vytvořena tabulka 4.1 zobrazující nalezené problémy společně s jejich řešením a ohodnocením 1–5, kde číslo 1 identifikuje pouze drobný problém a číslo 5 velice závažný problém. Byla také vytvořena nová verze wireframe, která je také k nalezení v příloze této práce.



Obrázek 4.2: Počet nalezených problémů v závislosti na počtu hodnotitelů. [50] (překlad vlastní)

4.2.3 Uživatelské testy

Uživatelské testy byly provedeny po heuristické analýze (na upraveném wireframe) ještě před samotnou implementací. Testy také nebylo možné provést v plné míře a sloužili převážné k odhalení nejvíce problematických částí návrhu uživatelského rozhraní.

Jakob Nielsen [51] doporučuje pět uživatelů na uživatelské testování. Však ze stejných důvodů jako u heuristické analýzy, byly zvoleni nyní pouze dva uživatelé.

Rozsáhlejší testování a heuristická analýza budou provedeny až po dokončení prototypu aplikace.

Testovaní uživatelé nebyly děleni v této fázi do skupin z důvodu jejich malého počtu. Podmínky na uživatele však kladeny byly – oba uživatelé museli používat operační systém iOS minimálně po dobu jednoho rok a alespoň jeden z nich se musel pohybovat v leteckém průmyslu.

Pro oba uživatele byly vytvořeny stejné testovací scénáře. Uživateli, který neměl znalosti z letectví, byly nejprve vysvětleny základní pojmy nutné k absolvování scénářů a až poté bylo provedeno testování.

Testovaní uživatelé byli při jednotlivých testech pozorováni, aby bylo zjištěno co a proč dělají špatně. Z čehož byly následně vyvozeny nutné úpravy v uživatelském rozhraní.

Problém	Závažnost	Provedená úprava
1	3	Při přidávání nového záznamu se bude měnit nadpis (title) podle toho zda je přidáván záznam o letu nebo ze simulátoru.
2	2	Tlačítko „Zavřít“ bude přejmenováno na „Zrušit“. Tato změna se týká obrazovek – přidat záznam, přidat letadlo a přidat certifikát.
2	3	U nastavení hledání bude tlačítko „Zpět“ přejmenováno na „Zrušit“.
2	4	Přejmenování záložky „Nastavení“ na „Profil“ a změnění ikonky.
2	4	Přejmenování „Můj profil“ (v bývalé záložce „Nastavení“) na „Osobní informace“.
2	2	Doplnění „Certifikáty“ na „Zdravotní certifikáty“.
8	2	U zobrazení všech záznamů jsou zredukovány zobrazené informace na datum, čas, místo a letadlo.

Tabulka 4.1: Nalezené problémy použitelnosti

4.2.3.1 Scénáře

1. Právě jste dokončil svůj první let z Brna do Prahy. Z Brna jste odlétal v 7:00 a do Prahy jste přiletěl v 8:45, neměl jste žádné mezipřistání. Celou doby jste zastával funkci vedoucího pilota a letěl jste letadlem Boeing.

2. Zjistil jste, že jste u minulého záznamu chybně zaznamenal čas. Upravte tedy čas příletu do Prahy na 7:45.

3. Chcete si přidat další záznam, tentokrát z výcvikového zařízení. Zadejte dnešní datum, čas na zařízení dvě hodiny a znovu jste zastával funkci vedoucího pilota.

4. Aplikaci používáte již delší dobu a máte tedy i více záznamu. Zobrazte si pouze lety, které byly provedeny za poslední týden.

5. Jeden ze záznamů z minulého kroku smažte.

6. Zobrazte si svůj profil a zkontrolujte, zda jsou všechny položky vyplněny správně. Pokud ne, opravte chybně vyplněné nebo chybějící pole.

7. U letadla Boeing z prvního kroku chcete změnit model. Proveďte tuto úpravu.

8. Za poslední tři týdny jste absolvoval mnoho letů, zkontrolujte, zda jste nepřekročil jeden z limitů.

9. Dnes Vám vydali zdravotní certifikát třídy jedna (Class 1). Vložte ho do aplikace.
10. Vytvořte report ze všech záznamů letů i záznamů z výcvikového zařízení za posledních čtrnáct dní.

4.2.3.2 Výsledky testů a provedené úpravy

4.3 Zvolená řešení

Tato kapitola popisuje a zdůvodňuje zvolená řešení architektury, perzistence dat a FRP framework.

4.3.1 Architektura

Pro svou práci jsem si na základě navržených funkcionalit zvolil architekturu MVVM. Architektura MVVM eliminuje nevýhodu MVC – příliš mnoho logiky a kódu ve vrstvě *Controller*, a na druhou stranu není zbytečně složitá (pro navrhovanou funkcionalitu) jako VIPER.

4.3.2 Perzistence dat

Pro tvorbu aplikace na evidenci letů jsem si zvolil možnost perzistence dat pomocí Realm. Prvním z důvodů této volby je podpora online i offline uložení dat. To umožňuje uživateli mít stejná data na více zařízeních, však i společně s možností používat aplikaci offline. Dalším důvodem bylo to, že je Realm zcela zdarma, což mu dává výhodu oproti iCloud – Apple řešení.

4.3.3 FRP framework

Kapitola 5

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, [cit. 2017-08-30]. 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] Nayebi, F.: *Swift 3 Functional Programming*. Packt Publishing, 2016, ISBN 9781785881619, 200–201 s. Dostupné z: https://books.google.cz/books?id=LP9vDQAAQBAJ

[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/`

[26] What Is Core Data? *Apple Developer* [online]. California, U.S.: Apple Inc., © 2017, [cit. 2017-09-16]. Dostupné z: `https://developer.apple.com/library/content/documentation/Cocoa/Conceptual/CoreData/index.html`

[27] Persistent Store Types and Behaviors. *Apple Developer* [online]. California, U.S.: Apple Inc., © 2017, [cit. 2017-09-16]. Dostupné z: `https://developer.apple.com/library/content/documentation/Cocoa/Conceptual/CoreData/PersistentStoreFeatures.html`

[28] Creating a Managed Object Model. *Apple Developer* [online]. California, U.S.: Apple Inc., © 2017, [cit. 2017-09-16]. Dostupné z: `https://developer.apple.com/library/content/documentation/Cocoa/Conceptual/CoreData/KeyConcepts.html`

[29] iCloud. *Apple Developer* [online]. California, U.S.: Apple Inc., © 2017, [cit. 2017-09-16]. Dostupné z: `https://developer.apple.com/icloud/`

[30] CloudKit. *Apple Developer* [online]. California, U.S.: Apple Inc., © 2017, [cit. 2017-09-16]. Dostupné z: `https://developer.apple.com/documentation/cloudkit`

[31] CloudKit. *Apple Developer* [online]. California, U.S.: Apple Inc., © 2017, [cit. 2017-09-16]. Dostupné z: https://developer.apple.com/icloud/cloudkit/

[32] Capabilities Available to Developers. *Apple Developer* [online]. California, U.S.: Apple Inc., © 2017, [cit. 2017-09-16]. Dostupné z: https://developer.apple.com/support/app-capabilities/

[33] Purchase and Activation. *Apple Developer* [online]. California, U.S.: Apple Inc., © 2017, [cit. 2017-09-16]. Dostupné z: https://developer.apple.com/support/purchase-activation/

[34] The Realm Mobile Platform. *Realm* [online]. Realm, © 2014-2017, [cit. 2017-09-16]. Dostupné z: https://realm.io/docs/get-started/overview/

[35] European Aviation Safety Agency (EASA). *European Union* [online], [cit. 2017-09-14]. Dostupné z: https://europa.eu/european-union/about-eu/agencies/easa_en

[36] Regulations. *European Aviation Safety Agency* [online], © 2017, [cit. 2017-09-14]. Dostupné z: https://www.easa.europa.eu/regulations

[37] PART-FCL. London, United Kingdom: EASA, June 2016. Dostupné z: https://www.easa.europa.eu/system/files/dfu/Part-FCL.pdf

[38] Official Journal of the European Union. London, United Kingdom: EASA, 2014. Dostupné z: http://eur-lex.europa.eu/LexUriServ/LexUriServ.do?uri=OJ:L:2014:028:0017:0029:EN:PDF

[39] EU Part-MED medical certificates for EU licences. *Civil Aviation Authority* [online]. London, United Kingdom: Civil Aviation Authority, © 2015, [cit. 2017-09-14]. Dostupné z: https://www.caa.co.uk/General-aviation/Pilot-licences/EASA-requirements/Medical/EASA-Part-MED-requirements/

[40] LogTen Pro X. *ITunes* [online]. California, U.S.: Apple Inc., © 2017, [cit. 2017-09-08]. Dostupné z: https://itunes.apple.com/us/app/logten-pro-x/id837274884?mt=8

[41] Logbook Pro Aviation Flight Log for Pilots. *ITunes* [online]. California, U.S.: Apple Inc., © 2017, [cit. 2017-09-08]. Dostupné z: https://itunes.apple.com/us/app/logbook-pro-aviation-flight-log-for-pilots/id410773111?mt=8

[42] Safelog Pilot Logbook. *ITunes* [online]. California, U.S.: Apple Inc., © 2017, [cit. 2017-09-08]. Dostupné z: https://itunes.apple.com/us/app/safelog-pilot-logbook/id411409786

[43] FlyLogio - Pilot Logbook. *Google Play* [online]. Silicon Valley: Google, ©2017, [cit. 2017-09-08]. Dostupné z: https://play.google.com/store/apps/details?id=com.flylogio&hl=en

[44] Smart Logbook. *Google Play* [online]. Silicon Valley: Google, ©2017, [cit. 2017-09-08]. Dostupné z: https://play.google.com/store/apps/details?id=com.kviation.logbook&hl=en

[45] Logbook Pro Desktop. *NC Software* [online]. Richmond, Virginia: NC Software, Inc., © 2017, [cit. 2017-09-08]. Dostupné z: http://www.nc-software.com/Logbook-Pro-Flight-Log-Software-for-Pilots

[46] Wireframing and Prototyping. *Nielsen Norman Group* [online]. California, United States: Nielsen Norman Group, © 1998-2017, [cit. 2017-09-26]. Dostupné z: https://www.nngroup.com/courses/wireframing-and-prototyping/

[47] What is wireframing? *Experienceux* [online]. Bournemouth: Experience UX, © 2017, [cit. 2017-09-26]. Dostupné z: http://www.experienceux.co.uk/faqs/what-is-wireframing/

[48] Nielsen, J.: How to Conduct a Heuristic Evaluation. In: *Nielsen Norman Group* [online]. California, United States: Nielsen Norman Group, © 1998-2017, [cit. 2017-09-26]. Dostupné z: https://www.nngroup.com/articles/how-to-conduct-a-heuristic-evaluation/

[49] Nielsen, J.: 10 Usability Heuristics for User Interface Design. In: *Nielsen Norman Group* [online]. California, United States: Nielsen Norman Group, © 1998-2017, [cit. 2017-09-26]. Dostupné z: https://www.nngroup.com/articles/how-to-conduct-a-heuristic-evaluation/

[50] Nielsen, J.: Curve showing the proportion of usability problems in an interface found by heuristic evaluation using various numbers of evaluators. In: *Nielsen Norman Group* [online], © 1998-2017, [cit. 2017-09-26]. Dostupné z: https://media.nngroup.com/media/editor/2012/10/30/heur_eval_finding_curve.gif

[51] Nielsen, J.: How Many Test Users in a Usability Study? In: *Nielsen Norman Group* [online]. California, United States: Nielsen Norman Group, © 1998-2017, [cit. 2017-09-27]. Dostupné z: https://www.nngroup.com/articles/how-many-test-users/

Příloha **A**

Seznam použitých zkratek

FRP Funkcionálně reaktivní programování

MVC Model View Controller

MVVM Model View ViewModel

VIPER View Interactor Presenter Entity Router

PC Personal Computer

EASA European Aviation Safety Agency

PIC Pilot-in-command

SE Single engine

ME Multi engine

SPIC Student PIC

PICUS PIC under supervision

FSTD flight simulation training devices

FI Flight instructor

FE Flight examiner

LAPL Light Aircraft Pilot Licence

SPL Sailplane Pilot Licence

BPL Balloon Pilot Licence

PPL Private Pilot Licence

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