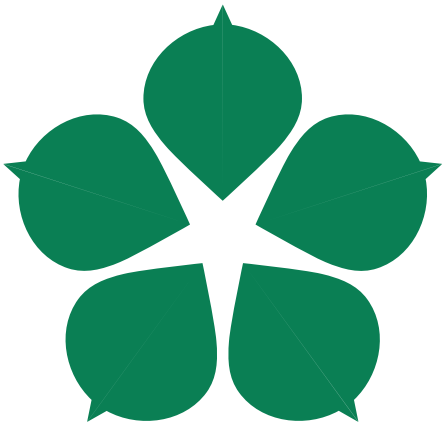
Jihočeská univerzita v Českých Budějovicích
Přírodovědecká fakulta



Multiplatformní aplikace v REACT NATIVE pro evidenci leteckých záznamů bezpilotních prostředků

Bakalářská práce

Zdeněk Pašek

Vedoucí práce: PhDr. Milan Novák, Ph.D.

České Budějovice 2020

Jihočeská univerzita v Českých Budějovicích
Přírodovědecká fakulta

ZADÁVACÍ PROTOKOL BAKALÁŘSKÉ PRÁCE

Student: Zdeněk Pašek
(jméno, příjmení, tituly)

Obor – zaměření studia: Aplikovaná informatika

Katedra: Ústav aplikované informatiky

Školitel: PhDr. Milan Novák, Ph.D.
(jméno, příjmení, tituly, u externího š. název a adresa pracoviště, telefon, fax, e-mail)

Garant z PřF:
(jméno, příjmení, tituly, katedra – jen v případě externího školitele)

Školitel – specialista, konzultant:
(jméno, příjmení, tituly, u externího š. název a adresa pracoviště, telefon, fax, e-mail)

Téma bakalářské práce: Multiplatformní aplikace v REACT NATIVE pro evidenci leteckých záznamů bezpilotních prostředků.

Cíle práce:

Základním cílem práce je navrhnout a implementovat uživatelsky příjemnou a intuitivní multiplatformní aplikaci ve frameworku React Native pro systémy iOS a Android, která bude svým uživatelům umožňovat vyplňovat data o leteckých pracích prostřednictvím bezpilotních prostředků. Konkrétní vyplňovaná data budou v souladu s legislativou ČR pro bezpilotní létání. Aplikace bude v maximální možné míře přebírat data z mobilního zařízení, popř. dalších služeb a bude umožňovat export záznamů do souborů ve formátu PDF, Excel apod.

Základní doporučená literatura:

EISENMAN, Bonnie. *Learning react native: building native mobile apps with JavaScript*. Second edition. Boston: O'Reilly, [2018]. ISBN 1491989149.

Frank Zammetti. *Practical React Native: Build Two Full Projects and One Full Game using React Native*. First edition. Boston: Apress, [2018].

WARD, Dan. *React Native Cookbook: Recipes for solving common React Native development problems*. Second edition. Boston: Packt Publishing, [2019].

Financování práce :.....

Vedoucí práce: PhDr. Milan Novák, Ph. D. .....podpis : .....

U externích vedoucích fakultní garant práce.....podpis : .....

Garant oboru bak. studia (nepožaduje se u zaměření „příprava na mag. studium biologie)

..... podpis : .....

Vedoucí oddělení: PhDr. Milan Novák, Ph. D. ..... podpis .....

Případný souhlas vedoucího ústavu AV .....podpis : .....

V Českých Budějovicích dne 16.10.2019

Převzal/a dne 16.10.2019 podpis : .....

Bibliografické údaje

Pašek Zdeněk, 2020: Multiplatformní aplikace v REACT NATIVE pro evidenci leteckých záznamů bezpilotních prostředků. [Multiplatform application for storing records of UAV in REACT NATIVE. Bc. Thesis, in Czech] – 59 p., Faculty of Science, University of South Bohemia, České Budějovice, Czech Republic.

Anotace

Tato bakalářská práce se zabývá tvorbou multiplatformní mobilní aplikace, která uživateli umožní evidovat a následně zpracovávat své letecké záznamy bezpilotních prostředků. Tyto záznamy se primárně zabývají provedenými misemi, informacemi o pilotech a bezpilotních prostředcích nebo různými statistikami konkrétního pilota. Součástí práce je analýza, návrh, samotná implementace, testování aplikace a plánovaný rozvoj.

Annotation

This bachelor's thesis deals with the creation of a multiplatform mobile application that allows users to record and subsequently process their aerial records of unmanned aerial vehicles. These records primarily deal with missions, information about pilots and UAVs or various statistics of a specific pilot. Part of this work is also analysis, design, implementation, application testing and planned development.

Prohlášení

Prohlašuji, že svoji bakalářskou práci jsem vypracoval samostatně pouze s použitím pramenů a literatury uvedených v seznamu citované literatury.

Prohlašuji, že v souladu s § 47b zákona č. 111/1998 Sb. v platném znění souhlasím se zveřejněním své bakalářské práce, a to – v nezkrácené podobě – elektronickou cestou ve veřejně přístupné části databáze STAG provozované Jihočeskou univerzitou v Českých Budějovicích na jejich internetových stránkách, a to se zachováním mého autorského práva k odevzdanému textu této kvalifikační práce. Souhlasím dále s tím, aby toutéž elektronickou cestou byly v souladu s uvedeným ustanovením zákona č. 111/1998 Sb. zveřejněny posudky školitele a oponentů práce i záznam o průběhu a výsledku obhajoby kvalifikační práce. Rovněž souhlasím s porovnáním textu mé kvalifikační práce s databází kvalifikačních prací Theses.cz provozovanou Národním registrem vysokoškolských kvalifikačních prací a systémem na odhalování plagiátů.

V Českých Budějovicích dne 7.12. 2020 ……..

Zdeněk Pašek

Poděkování

Tímto bych rád poděkoval vedoucímu práce PhDr. Milanu Novákovi, Ph.D. za podnětné rady a čas věnovaný vedení této práce. Poděkování patří také těm, kteří mě v průběhu psaní práce podporovali, zejména rodině a přátelům.

Obsah

1 Úvod

V dnešní době jsou velkým tématem bezpilotní prostředky (drony), které se hojně využívají například v armádě, průmyslu, zemědělství nebo v civilní a komerční sféře. V armádě se klade důraz na průzkumné nebo útočné lety, v zemědělství například tyto prostředky sbírají data o kvalitě, množství nebo zdraví rostlin na polích a následně jsou zpracovávány a vyhodnocovány [1]. V těchto sférách jsou bezpilotní prostředky velmi nákladné na pořízení, ovšem v poslední době dochází v komerční sféře k rapidnímu snížení cen, čímž se zvedla dostupnost pro „běžné" uživatele. Tito uživatelé využívají bezpilotní prostředky převážně pro pořizování obsahu, jako je fotografování nebo natáčení videa.

Tématem bakalářské práce je tvorba multiplatformní mobilní aplikace pro evidenci leteckých záznamů bezpilotních prostředků. Primárně je aplikace vytvořena pro ústav Aplikované informatiky Jihočeské univerzity. Ústav vlastní velké množství bezpilotních prostředků, a je proto potřeba evidovat záznamy o letech a následně je zpracovávat. Aplikace je samozřejmě dostupná i pro ostatní uživatele dronů (širokou veřejnost).

Aplikace je rozdělena do tří částí (evidence pilotů, evidence strojů a evidence letů) a všechny tyto části jsou vzájemně propojené. Uživatel má tedy možnost založit si vlastní pilotní profil, ve kterém uvede, jaké vlastní bezpilotní prostředky a následně s nimi může vykonává lety (mise). Tyto mise sebou nesou velké množství dat (název, počasí, datum, stroj atd.) a tyto data jsou primárně získávána jako vstup od uživatele a sekundárně pomocí API. Výsledkem je možnost exportovat tyto data do formátu PDF.

První část práce slouží jako dokumentace a obsahuje logický rámec, ve kterém jsou údaje o cíli, účelu a výstupu práce. Následně jsou zde funkční a nefunkční požadavky, zvolené technologie, uživatelské scénáře a v neposlední řadě celková architektura aplikace. Druhá část se zabývá samotnou implementací mobilní aplikace a popisem postupu této implementace. Dále je zde rozebíráno testování aplikace a plánovaný rozvoj.

1.1. Cíle práce

Hlavním cílem této bakalářské práce je navrhnout a implementovat uživatelsky příjemnou a intuitivní mobilní aplikaci ve frameworku React Native pro systémy iOS a Android. Aplikace bude svým uživatelům umožňovat vyplňovat data o leteckých pracích prostřednictvím bezpilotních prostředků. Konkrétní vyplňovaná data budou v souladu s legislativou ČR pro bezpilotní létání. Aplikace bude v maximální možné míře přebírat data z mobilního zařízení, popř. dalších služeb a bude umožňovat export záznamů do souborů ve formátu PDF, Excel apod. Dále se předpokládá splnění těchto následujících úkolů:

- Analýza podobných typů aplikací
- Zvolení vhodných technologií (front-end/back-end)
- Navržení uživatelských scénářů a architektury aplikace
- Určení funkčních a nefunkčních požadavků
- Implementace
- Testování aplikace
- Určení plánovaného rozvoje aplikace

2 Analýza

2.1. Scénář realizace

Před každým vývojem aplikace, je potřeba si definovat určitý proces, kterým se bude vývojář/designer aplikace řídit. Tento proces se skládá z průzkumu, myšlenky, designu a testování. Klade si základní otázky: pro koho je aplikace určena – jaká je proto persona, s jakými problémy se potencionální uživatel potýká a jaká je základní funkcionalita aplikace. Po zodpovězení na tyto klíčové otázky se vytvoří pojmová mapa, ze které se vytvoří již samotný design a provede testování.

2.1.1. Proces

Obrázek 1: Proces designu

Průzkum

- Pro koho je aplikace určena (jaký je typický uživatel – dočasná proto-persona)
- Jaké jsou problémy / složitosti se kterými se uživatel potýká (např. veškeré záznamy píše ručně a dlouhosáhle do nečitelných spisů)
- Jak může aplikace pomoci uživateli v tom, aby mu ulehčila práci
- Jaká je základní funkcionalita aplikace
- Generovat a seskupit nápady
- Prozkoumat podobné typy aplikací (neexistuje už taková aplikace?), inspirace z jiných projektů

Myšlenka

- Vzít nápady a vytvořit z nich návrhy (sketche, wireframy, myšlenkové mapy) a navrhnout základní strukturu pomocí hlavních elementů aplikace

Design

- Vytvořit 2 nebo 3 designy z předešlých návrhů a nápadů

Test

- Vytvořit prototyp aplikace k testování
- Získat feedback od uživatelů a podle výsledků udělat změny

Průzkum

Proto-persona

- 16 - ∞ let
- Uživatel dronu

Problémy

- Veškeré záznamy o letech si zapisuje ručně a repetetivně
- Polohu, počasí, čas letu musí hledat zvlášť z různých zdrojů (trvá to dlouho)
- Záznamy nemá pohromadě a nemá žádný způsob jak s nimi pracovat elektronicky (v tabulkách, dokumentech)

Jak může aplikace pomoci

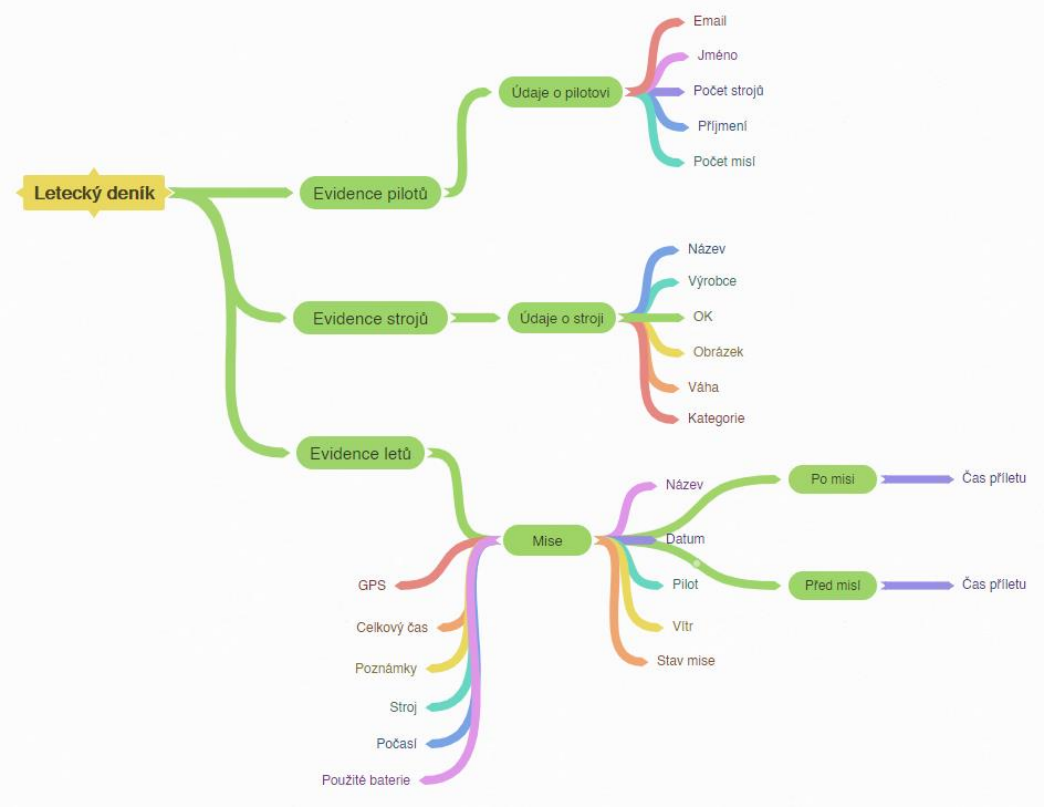
- Data potřebná pro záznam letu maximálně přebírat z informací které už jsou známé (pilot, dron, OK atd.)
- Data jako počasí, poloha získávat automaticky z externích zdrojů pomocí API
- Možnost záznamy jednoduše exportovat a následně s nimi pracovat

Základní funkcionalita

- Vytvořit uživatelský účet
- Vlastní profil se základními informacemi (statistiky, drony, atd.)
- Akce s dronem (přidat, odebrat, editovat, přiřadit k misi)
- Akce s misí (přidat, odebrat, editovat, exportovat)
- Nastavení aplikace (jazyk, jednotky, atd.)

Myšlenka

Tato myšlenková mapa primárně posloužila jako startovní bod pro ujasnění, jaké části a informace budou v aplikaci potřebné. Prvotní myšlenka byla rozdělení na 3 hlavní části – evidence pilotů, evidence strojů a evidence letů. Každá z těchto části v sobě uchovává nezbytné informace pro evidenci, přičemž v konečné fázi je nejdůležitější evidování samotných misí. V těch se sjednotí veškeré potřebné informace a exportují se. Díky tomuto návrhu se následně začaly konstruovat wireframy a samotný design.



Obrázek 2: Myšlenková mapa

2.2. Podobné typy aplikací

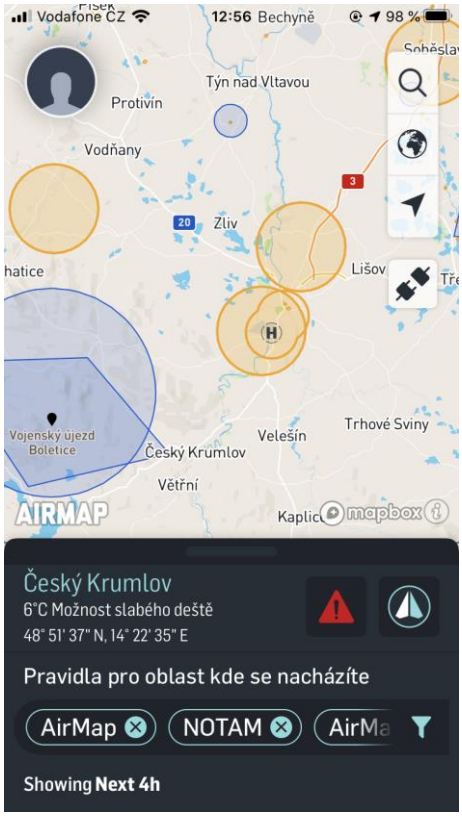
Aplikací, které usnadňují létání s drony je celá řada, nicméně většina aplikací se soustředí na velké množství funkcí, které ovšem nefungují tak, jak by uživatel očekával nebo nefungují vůbec. Na trhu chybí jednoduchá aplikace, která slouží k jednoznačnému účelu, a to evidování záznamu o letech s možností pohodlného exportu. Proto se tedy nabízí vytvoření této aplikace.

2.2.1. AirMap

AirMap je aplikace určena zejména k plánování letů, kdy si uživatel zvolí, do jaké oblasti plánuje letět a podle toho se vyžádá specifická digitální autorizace letu. Aplikace nabízí celou řadu funkcí jako interaktivní mapu, kde jsou vyznačené zakázané prostory pro létání, výpis pravidel létání pro danou oblast, možnost nastavení vlastního profilu pilota nebo upozornění o tom, co se nachází v okolí (elektrárny, letiště atd.) [2].

2.2.2. Hover

Hover je ve srovnání s AirMap jednodušší a je na první pohled znatelné, že nejde o profesionální řešení. Nicméně má velký počet stažení jak na AppStore, tak na Google Play. Poskytuje funkci záznam letu (Flight Log), který se převážně zadává ručně a nejde žádným způsobem exportovat. Dále nabízí informace o aktuálním počasí v dané lokalitě, funkci bezpečného vzletu, kdy zkontroluje, zda je ideální počasí nebo zda se uživatel nenachází v zakázané oblasti [3].



Obrázek 3: Rozhraní aplikace AirMap

2.3. Metodika vývoje

Zvolení metodiky vývoje softwaru spolu nese spousta úskalí, která jsou potřeba rozhodnout před zahájením vývoje projektu. Každý projekt je unikátní a má své specifické požadavky, a proto nelze aplikovat jednotnou metodiku vývoje. Je třeba například rozhodnout, kolik vývojářů se na projektu bude podílet, jak časté je potřeba dělat iterace nebo kolik času máme na dokončení projektu. Dále je důležité si ujasnit, jestli zvolená metodika bude vhodná pro náš tým, klienty nebo jestli máme finanční prostředky na pokrytí dané metodiky [4].

2.3.1. Getting Real

Pro vývoj této aplikace byla zvolena agilní metodika Getting Real. Jedná se o metodiku vytvořenou společností Basecamp (dříve 37signals), která se zabývá vývojem webových aplikací a je tvůrcem populárního frameworku Ruby on Rails [5].

Cílem této metodiky je neklást důraz na věci, které nejsou pro tvorbu aplikace tak důležité (grafy, schémata atd.) ale soustředit se na budování reálných věcí, které uvidí a pocítí koncový uživatel (UI/UX) a na řešení skutečných problémů [6]. Dále doporučuje budování aplikací, které se soustředí na menší množství funkcionalit, kdy fungují bezchybně a spolehlivě. To přináší výhodu v následných aktualizacích nebo integraci nových technologií. Méně toho, co není pro aplikaci esenciální. Getting Real je o častých iteracích, konstantním překrucování, vylepšování a produkování, což je největší výhodou ve světě, kde se software a technologie rychle vyvíjí a mění.

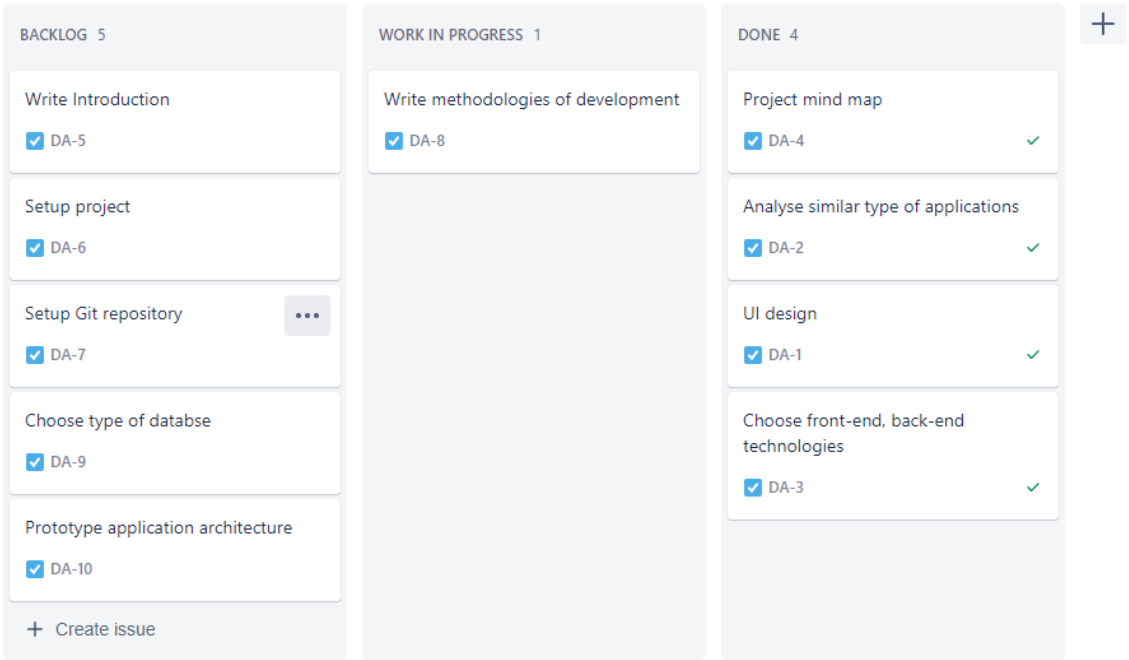
Hlavní body, kterými se Getting Real řídí:

- Budování skutečně reálných věcí
- Méně funkcionalit
- Méně možností/preferencí
- Malý vývojářský tým
- Velmi krátké iterace
- Jednoduchost
- Bezchybnost, kvalita a spolehlivost komponent
- Velký důraz na UI/UX
- Časté aktualizace, vylepšování

2.3.2. Kanban

Kanban vychází z japonského slova, které v překladu znamená „tabule“. Jde o metodu managmentu práce, která je velmi populární v oblasti vývoje softwaru, a to díky jednoduché implementaci, vizualizaci a efektivnímu managmentu rozpracovaných úkolů [7].

Hlavním cílem je vizualizovat jednotlivé části projektu do tzv. **Kanban boardu** (tabule). Jedná se o nástroj pro vizualizaci workflow, který zlepší chápání celého procesu vývoje. Tato tabule je realizována několika sloupci, kde každý vyjadřuje stav jednotlivých úkolů. Zpravidla se rozděluje na části jako: **Backlog** – fronta úkolů, které budou zpracovávány, **Work in progress** – rozpracované úkoly, **Done** – dokončené úkoly. Backlog je následně naplněn množstvím úkolů a ty se přetahují do sloupců podle jejich stavu. Po dokončení úkolu a přetáhnutí do sloupce Done, cyklus úkolu končí. Tento systém doporučuje pracovat vždy na jednom úkolu a po jeho vykonání je možné proces opakovat s novým zadáním.



Obrázek 4: Příklad kanban boardu (Jira Software)

Shrnutí hlavních výhod využívání Kanban metody [7]:

- Zlepšení chápání celého procesu vývoje
- Jednoduchá implementace
- Lepší pozornost na zákazníka a jeho potřeby
- Větší motivace developerů
- Zvýšená komunikace a koordinace v týmu
- Jednoduchý managment úkolů

Pro tuto bakalářskou práci bylo zvoleno agilní řešení Jira Software od společnosti Attlasian [8]. Jedná se o nástroj pro agilní vývoj softwaru, který nabízí například Scrumovou metodiku nebo již zmíněný Kanban board. Hlavním důvodem byla jednoduchá implementace a absence týmu. Dále byl použit Git k verzování kódu.

2.4. Logický rámec

Cíl projektu	**Objektivně ověřitelné ukazatele**	**Prostředky ověření**	**Předpoklady**
Navržení a implementace multiplatformní mobilní aplikace (iOS, Android).	Plně funkční mobilní aplikace, která je uživatelsky přívětivá a intuitivní.	Zpětná vazba od uživatelů ve formě dotazníků.	Zájem UAI a ostatních uživatelů bezpilotních prostředků o mobilní aplikaci.

Účel projektu	**Objektivně ověřitelné ukazatele**	**Prostředky ověření**	**Předpoklady**
Poskytnout uživatelům příjemné prostředí pro zaznamenávání a následné zpracovávání leteckých záznamů bezpilotních prostředků ve formě mobilní aplikace.	Zájem o aplikaci a spokojenost u uživatelů, na které byla cílena.	Výstupy z dotazníků a statistiky stažení aplikace.	Uživatelé bezpilotních prostředků, kteří chtějí evidovat a zpracovávat data o svých letech.

Výstupy projektu	**Objektivně ověřitelné ukazatele**	**Prostředky ověření**	**Předpoklady**
1) Dokumentace požadavků 2) Dokumentace architektury/designu 3) Technická dokumentace 4) Mobilní aplikace 5) Uživatelská dokumentace	Dostupná dokumentace Přístupná aplikace uživatelům Úspěšné testování aplikace	Data z dokumentace a výsledků testování.	Aplikace bude splňovat funkční a nefunkční požadavky. Získání dostatku uživatelů a dat při testování aplikace.

Aktivity projektu	**Objektivně ověřitelné ukazatele**	**Prostředky ověření**	**Předpoklady**
1. Analýza a návrh řešení *1.1. Analýza požadavků* *1.2. Návrh architektury* 1.3. *Návrh GUI* 1.4. *Návrh databáze* 1.5. *Návrh REST API* 2. Implementace *2.1. Klientská část* *2.2. Serverová část* *2.3. REST API* *2.4. Testování aplikace*	Splnění testování aplikace a vyplnění dotazníků.	Výsledky testování a dotazníků.	Vhodně zvolená metodika, technologie a postupy. Validní zdrojový kód, dostupné mobilní zařízení a dostatečný počet uživatelů.

2.5. Specifikace softwarových požadavků

Specifikace softwarových požadavků je velmi důležitou součástí vývoje softwaru. Jde o stanovení základních požadavků, funkcionalit a chování aplikace. Také popisuje uživatelské interakce a pomáhá vývojáři k celkovému pochopení problematiky. Rozděluje se na funkční a nefunkční požadavky. Stanovení těchto požadavků proběhlo z velké části v procesu průzkumu aplikace (viz Proces/Průzkum).

2.5.1. Funkční požadavky

1. Uživatel si bude moci vytvořit uživatelský účet a spravovat ho
2. Uživatel bude moci spravovat vlastní profil se základními informacemi
3. Uživatel bude moci provádět akce s drony (přidat, smazat, upravit, přiřadit ke konkrétní misi)
4. Uživatel bude moci provádět akce s misemi (přidat, smazat, upravit, exportovat)
5. Uživatel bude moci exportovat data o misích a následně s nimi pracovat
6. Uživatel si bude moci nastavit aplikaci podle jeho požadavků
7. Systém bude získávat GPS polohu uživatele
8. Systém bude získávat údaje o počasí z externích zdrojů
9. Systém bude ukládat a zpracovávat zadané informace od uživatele
10. Systém bude zobrazovat veškeré drony uživatele a jejich detail
11. Systém bude zobrazovat veškeré mise uživatele a jejich detail
12. Systém bude zobrazovat statistiky uživatele
13. Systém bude udržovat uživatelský účet chráněn

2.5.2. Nefunkční požadavky

1. Aplikace bude mít příjemné a intuitivní prostředí pro uživatele
2. Aplikace bude pracovat online
3. Aplikace bude dostupná jak pro iOS zařízení tak pro Android
4. Aplikace bude pracovat s veškerými daty pomocí REST API rozhraní
5. Kód aplikace bude psaný podle konvence a jeho komponenty budou v maximální možné míře znovupoužitelné
6. Aplikaci bude možno rozšířit o další funkcionality, popřípadě upravit podle potřeb
7. Aplikace bude v maximální možné míře responzivní a vhodná pro jakékoliv zařízení

2.6. Uživatelské scénáře

Název: Registrace

Popis:

Uživatel si chce založit vlastní uživatelský účet. V hlavním okně klikne na tlačítko s titulkem „Sign up“ nebo „Registrovat“. Aplikace ho přesměruje na obrazovku registrace, kde ve formuláři vyplní veškeré potřebné informace pro založení účtu. Pokud budou veškerá povinná pole vyplněna validně, pošle se požadavek na server a uživatel bude přesměrován přímo do aplikace na domovskou stránku.

Funkční požadavky:

1. Uživatel si bude moci vytvořit uživatelský účet a spravovat ho
9. Systém bude ukládat a zpracovávat zadané informace od uživatele
13. Systém bude udržovat uživatelský účet chráněn

Předpoklady:

- Uživatel má přístupný internet
- Uživatel validně vyplní formulář

Název: Přidání dronu

Popis:

Uživatel si chce přidat dron do svého seznamu. Nejdříve se úspěšně přihlásí nebo registruje a kliknutím na spodní navigační panel, konkrétně na ikonu „letadla“ se přesune na seznam všech jeho bezpilotních prostředků. Pokud žádné ještě nevlastní, seznam bude prázdný. V tomto okně se v pravém horním rohu aplikace nachází „plus“ ikona. Na tuto ikonu uživatel klikne a dostane se na obrazovku pro přidání UAV. Zde validně vyplní formulář a pokud server schválí toto UAV, uživatel se přesune zpět na seznam všech jeho dronů. V tomto seznamu už bude již právě vytvořený dron.

Funkční požadavky:

1. Uživatel si bude moci vytvořit uživatelský účet a spravovat ho
3. Uživatel bude moci provádět akce s drony (přidat, smazat, upravit, přiřadit ke konkrétní misi)
9. Systém bude ukládat a zpracovávat zadané informace od uživatele
10. Systém bude zobrazovat veškeré drony uživatele a jejich detail
13. Systém bude udržovat uživatelský účet chráněn

Předpoklady:

- Uživatel má přístupný internet
- Uživatel se úspěšně přihlásí nebo registruje
- Uživatel validně vyplní formulář

Název: Přidání/Realizace mise

Popis:

Uživatel chce realizovat misi se svým bezpilotním prostředkem. Pro jeho realizaci se musí úspěšně registrovat/přihlásit do aplikace a mít již vytvořený bezpilotní prostředek. Následně ve spodním navigačním panelu klikne na ikonu „mise“ a bude přesunut na obrazovku se seznamem veškerých misí. Pokud ještě uživatel žádnou nerealizoval, seznam bude prázdný. V pravém horním rohu tohoto okna se nachází „plus“ ikona. Po kliknutí na tuto ikonu se uživatel přesune na obrazovku pro vytvoření/realizaci mise. Zde validně vyplní formulář, vybere si s jakým bezpilotním prostředkem chce misi realizovat, s jakou baterií a systém mu automaticky předvyplní údaje jako GPS, datum, čas, počasí a vítr. Pokud bude vše validní, aplikace uživateli podá zprávu o úspěšném přidání a přesune ho na seznam realizovaných misí, už s již nově vytvořenou misí.

Funkční požadavky:

1. Uživatel si bude moci vytvořit uživatelský účet a spravovat ho
3. Uživatel bude moci provádět akce s drony (přidat, smazat, upravit, přiřadit ke konkrétní misi)
4. Uživatel bude moci provádět akce s misemi (přidat, smazat, upravit, exportovat)
7. Systém bude získávat GPS polohu uživatele
8. Systém bude získávat údaje o počasí z externích zdrojů
9. Systém bude ukládat a zpracovávat zadané informace od uživatele
10. Systém bude zobrazovat veškeré drony uživatele a jejich detail
11. Systém bude zobrazovat veškeré mise uživatele a jejich detail
13. Systém bude udržovat uživatelský účet chráněn

Předpoklady:

- Uživatel má přístupný internet
- Uživatel se úspěšně přihlásí nebo registruje
- Uživatel si vytvoří svůj bezpilotní prostředek
- Uživatel validně vyplní formulář

Název: Exportování misí

Popis:

Uživatel chce své realizované mise exportovat do formátu, aby s daty mohl pracovat. Je potřeba aby se uživatel úspěšně přihlásil/registroval. Po tomto kroku uživatel musí přidat svůj vlastní bezpilotní prostředek a realizovat s ním misi. Jakmile uživatel realizoval misi, po kliknutí na spodní navigační panel na ikonu „mise" se uživatel přesune na seznam realizovaných misí. V tomto okně najde v horním pravém rohu ikonku pro „export", hned vedle ikony „plus". Jakmile na tuto ikonu uživatel klikne, otevře se modální okno s potvrzením, zda chce uživatel jeho mise exportovat. Po udělení souhlasu se na serveru vytvoří soubor, ten se nahraje do cloudu a uživateli se pošle ve formě URL. Tento odkaz se otevře v nativně závislém webovém prohlížeči zařízení, kde je možné s ním jednoduše pracovat.

Funkční požadavky:

1. Uživatel si bude moci vytvořit uživatelský účet a spravovat ho
3. Uživatel bude moci provádět akce s drony (přidat, smazat, upravit, přiřadit ke konkrétní misi)
4. Uživatel bude moci provádět akce s misemi (přidat, smazat, upravit, exportovat)
5. Uživatel bude moci exportovat data o misích a následně s nimi pracovat
7. Systém bude získávat GPS polohu uživatele
8. Systém bude získávat údaje o počasí z externích zdrojů
9. Systém bude ukládat a zpracovávat zadané informace od uživatele
10. Systém bude zobrazovat veškeré drony uživatele a jejich detail
11. Systém bude zobrazovat veškeré mise uživatele a jejich detail
13. Systém bude udržovat uživatelský účet chráněn

Předpoklady:

- Uživatel má přístupný internet
- Uživatel se úspěšně přihlásí nebo registruje
- Uživatel si vytvoří svůj bezpilotní prostředek
- Uživatel realizuje misi se svým bezpilotním prostředkem
- Uživatel validně vyplní formulář

2.7. Zvolení mobilní technologie

Před vývojem mobilní aplikace je velmi důležité zvážit současné technologie a samotný vývoj. Dnes se vývoj primárně rozděluje na tři části – Nativní, Multiplatformní a Hybridní [9]. Každý má své pro a proti. Zde je stručný popis různých přístupů vývoje:

	Nativní vývoj	**Multiplatformní vývoj**	**Hybridní vývoj**
Programovací jazyk	Android: Java, Kotlin iOS: Objective-C, Swift	Podle technologie: JavaScript, Dart, C#	Standardní webové technologie: JavaScript, HTML5, CSS
Technologie	-	React Native, Flutter, Xamarin	PhoneGap, Cordova, Ionic
Prostředí	Android: Android Studio iOS: XCode	Podle preference: Microsoft Visual Studio/Code, Xamarin studio, Atom	Podle preference: Visual Studio Code, Atom, WebStorm
Aplikace	Uber, Lyft, Trello	Instagram, Skype, Alibaba, UPS	Diesel, McLaren Automotive, Sanvello
Výhody	+ Rychlost aplikace + Jednoduchá integrace s celým ekosystémem (google/apple pay, siri) + Bezpečnost a stabilita + Dobré UX	+ Jeden kód pro obě platformy (iOS, Android) + Rychlejší a levnější vývoj + Velká komunita + Použití Reactu jak pro web, tak pro mobilní vývoj	+ Jeden kód pro obě platformy (iOS, Android) + Rychlejší a levnější vývoj + Znovupoužitelnost kódu jak pro web tak mobilní vývoj + Konzistentní UX na všech platformách
Nevýhody	- Vývoj aplikace pro každou platformu zvlášť (iOS, Android) - Dražší vývoj - Potřeba více vývojářů	- Omezená podpora nástrojů operačního systému - Menší rychlost aplikace (převážně u graficky náročných aplikací) - Limitované UI/UX	- Omezená podpora nástrojů operačního systému - Menší rychlost aplikace

2.7.1. Multiplatformní vývoj

Pro tuto aplikaci bylo zohledněno více přístupu vývoje. Z mnoha důvodů byl zvolen multiplatformní vývoj, konkrétně s technologií React Native. Tato aplikace ve své podstatě nedisponuje funkcionalitami, které by byly potřebné pro zvolení nativního vývoje. Aplikace má jednoduchý design a nezakládá si na animacích – tudíž není třeba využívat výhody rychlosti aplikace u nativního vývoje (aplikace bude srovnatelně rychlá). Dále aplikace využívá GPS polohu uživatele, kterou lze získat pomocí platformově závislých API – proto není třeba využívat celkové integrace ekosystému (Google/Apple pay, siri), kterým opět disponuje nativní vývoj. Naopak bude benefitovat jednotným kódem pro všechny platformy, rychlostí vývoje a samozřejmě velkou komunitou, která stojí za technologií React Native. Dále jsou výhodou autorovo předchozí znalosti jazyka JavaScript a knihovny ReactJS.

3 Design

3.1. Wireframes

Wireframy definují základní kostru aplikace/webové stránky a vizuálně určují, kde se přibližně budou na obrazovkách nacházet různé elementy (tlačítka, obrázky, texty). Nesoustředí se přímo na detaily jako jsou barvy, odsazení, velikost písma nebo obsah, ale snaží se popsat základní rozložení obrazovky aplikace a její funkce. Tato aplikace se skládá z dvanácti obrazovek, kde každá je určena ke specifické funkci aplikace a obsahuje statické spodní menu pro navigaci.

MainScreen

Tato obrazovka se zobrazí uživateli ihned po spuštění aplikace a nutí uživatele k přihlášení nebo k vytvoření účtu. Slouží jako uvítací obrazovka a rozcestník mezi přihlášením a registrací.

RegisterScreen

Obrazovka určena k registrování uživatele do aplikace. Uživatel je požádán o jméno, email a heslo.

SignInScreen

SignInScreen umožňuje přihlášení uživatele do aplikace a po úspěšném přihlášení přepnutí na domovskou obrazovku s konkrétními informacemi.

HomeScreen

Hlavní obrazovka celé aplikace, která slouží zároveň jako domovská. Zobrazí se po úspěšném přihlášení/registraci a je přístupná z navigace. Hlavním cílem je zobrazení základních informací o uživateli (jméno, email) a dále přístup k základním statistikám o uživateli a jeho letech – počet splněných misí, počet vlastněných dronů nebo celkový čas strávený ve vzduchu.

UAVScreen (list, add, selected)

UAVScreen je přístupná ze spodní navigace a je rozdělena na tři obrazovky. Slouží k zobrazení všech bezpilotních prostředků, které uživatel vlastní, přidání nového prostředku a zobrazení specifického prostředku s podrobnými informacemi. Přidání nového prostředku vyžaduje zadání informací jako jméno, typ, OK, váha nebo kategorie. Dále je možné při zobrazení konkrétního prostředku upravit zadané informace nebo prostředek smazat.

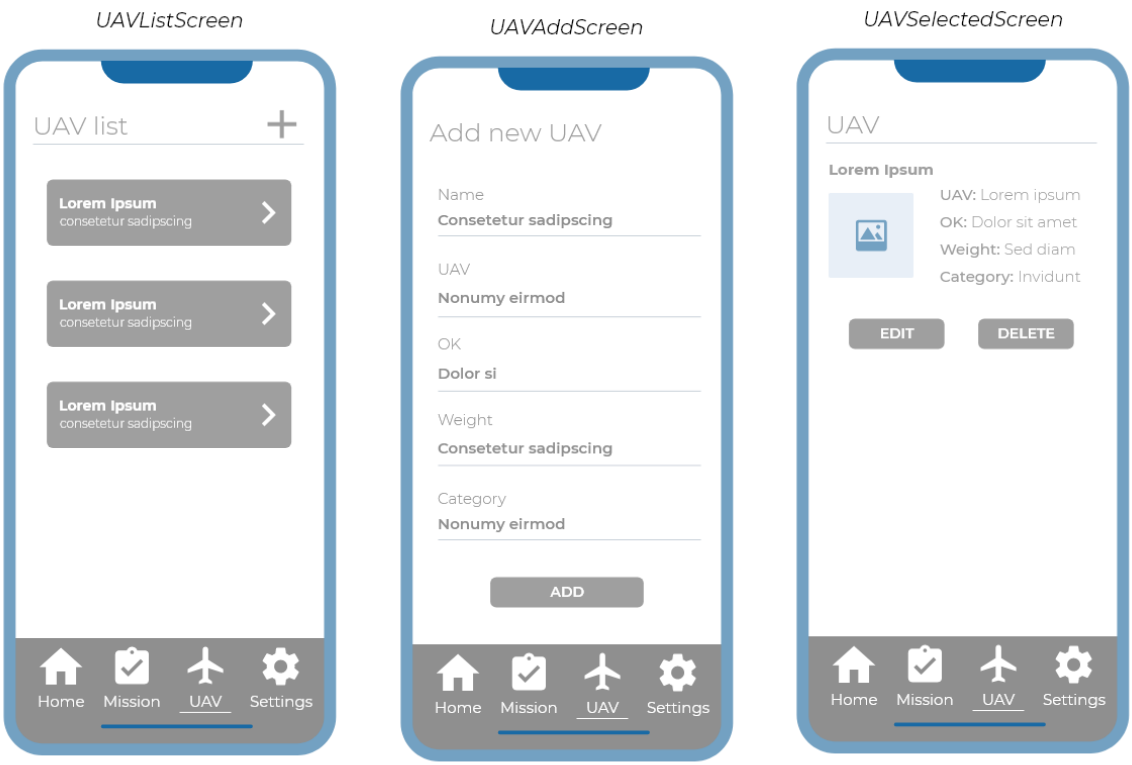
MissionScreen (list, add, selected)

Další obrazovka přístupná z navigace, která je opět rozdělená na tři obrazovky, kdy poskytuje vypsání veškerých realizovaných misí uživatele, přidání nové mise se specifickými údaji jako pilot, UAV, GPS lokace, datum nebo současné počasí. Dále zobrazení konkrétní mise s podrobnými informacemi, úprava údajů mise nebo její smazání. Na hlavní obrazovce se také nachází tlačítko pro export misí.

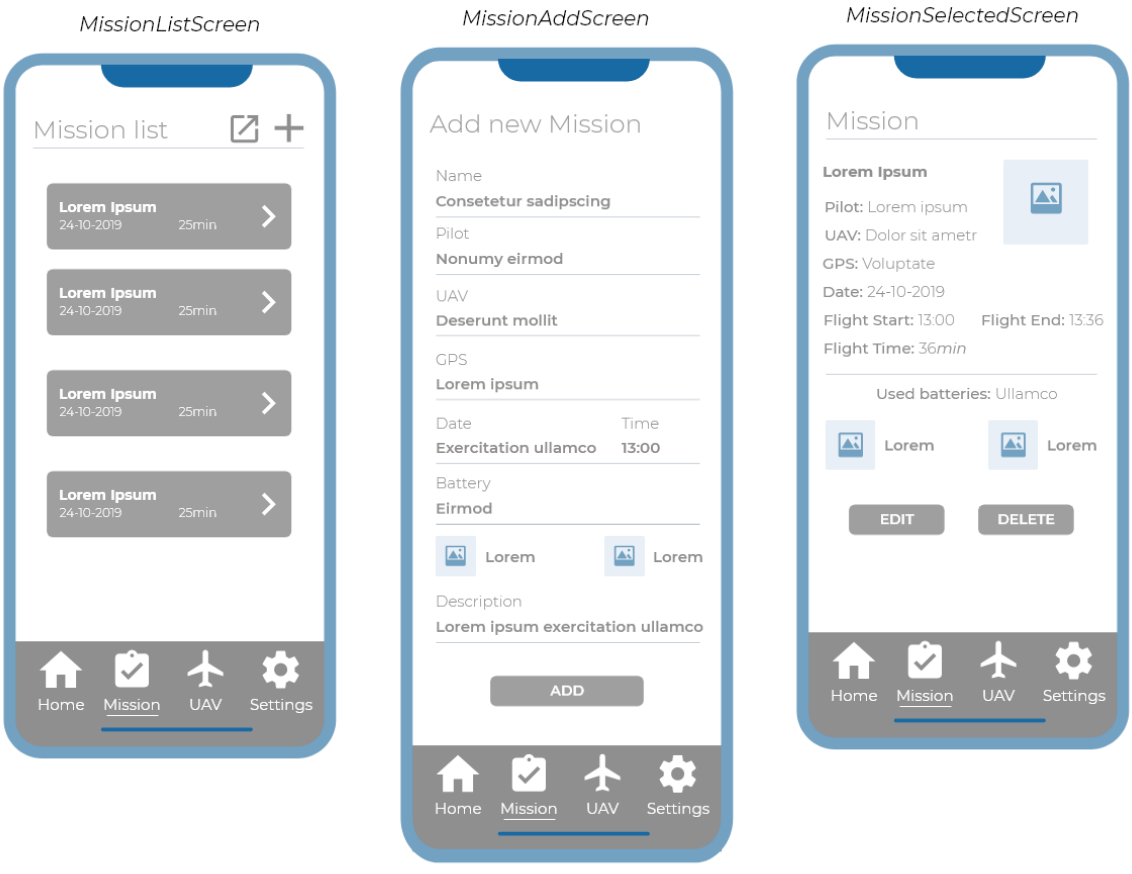
SettingsScreen

Tato obrazovka je určena pro změnu specifických nastavení aplikace (jazyk) nebo pro odhlášení uživatele z aplikace. Je dostupná ze spodní navigace.

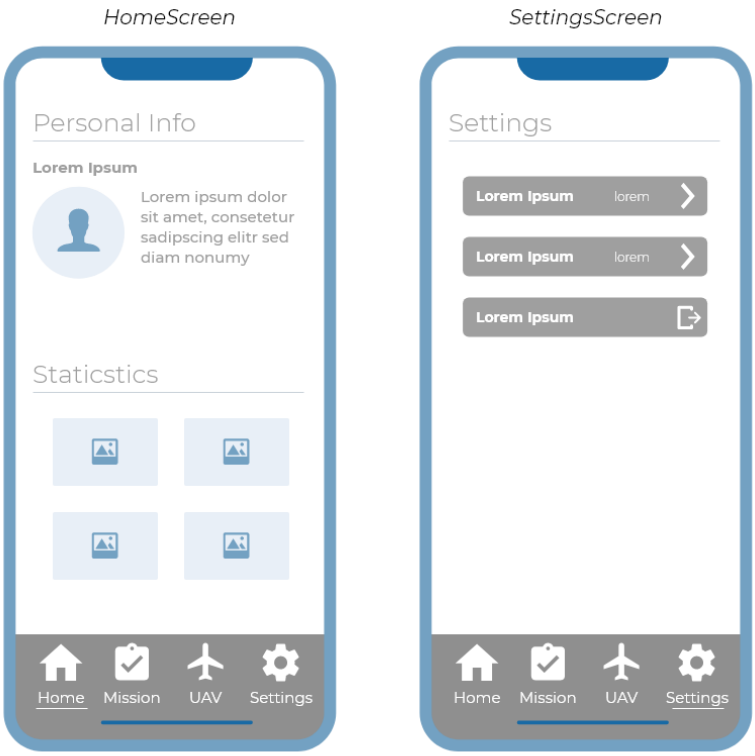
Obrázek 5: Základní obrazovky aplikace



Obrázek 6: UAV obrazovky



Obrázek 7: Mission obrazovky

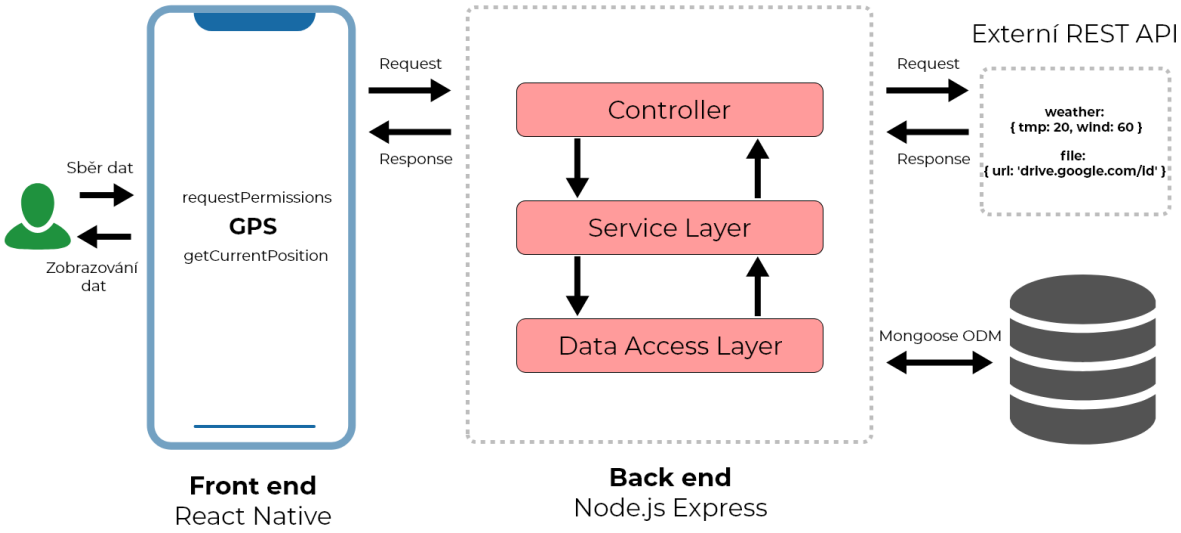


Obrázek 8: Domovská obrazovka a nastavení

3.2. Architektura systému

Před samotnou implementací je důležitou součástí designu návrh architektury systému. Správný návrh dokáže ušetřit spoustu času při následném rozšiřování sytému (škálování) nebo změnách v systému. Klíčové je navrhnout části tak, aby na sobě byly co nejméně závislé, a šlo tedy při požadovaných změnách jednoduše a rychle část vyměnit.

Architektura začíná u uživatele, který kumuluje data a následně je předává front endu – v tomto případě mobilnímu zařízení na kterém běží samotná aplikace. Tyto data jsou dále předávána pomocí REST API rozhraní back endu, jenž je navržen dle vícevrstvé architektury. Ta se skládá z Controlleru, Service vrstvy a Data Access vrstvy. Každá z těchto vrstev má svůj jasný úkol. Controller se stará o zacházení s uživatelskými interakcemi. Servisní vrstva v sobě zapouzdřuje veškerou business logiku a Data Access vrstva přistupuje k databázi. Back end také přistupuje k externím datovým zdrojům pomocí REST API a získává potřebné informace. Po úspěšném získání dat z databáze, back end vrací potřebná data uživateli a zobrazuje je na mobilním zařízení.



Obrázek 9: Architektura systému

3.3. Datový model

Pro datový model byla zvolena NoSQL databáze MongoDB. Jedná se o dokumentovou databázi, která ukládá strukturované objekty ve formátu JSON. Na rozdíl od SQL databází, které využívají ACID[1], kde jsou data vždy konzistentní, NoSQL databáze využívají BASE[2], čímž upřednostňují výkon před konzistencí – důležité je rychle vrátit nějaká data bez ohledu na to, jestli jsou to data konzistentní.

Mongo nabízí dva přístupy pro modelování dat – Embedded data models (zanořené) a Normalized data models (normalizované). Zanořené modely umožnují ukládat související data v jednom záznamu (dokumentu), a tím docílit větší rychlosti při čtení z databáze a menšího počtu potřebných dotazů. Naopak normalizované popisují vztahy mezi dokumenty pomocí referencí a je tedy potřeba provádět více dotazů. V ideálním případě se tyto dva přístupy kombinují podle vztahů mezi daty (1:1, 1:N, M:N) [10].

Datový model je rozdělen do dvou dokumentů (users a missions). Users obsahují veškeré informace o jednotlivých uživatelích. Jelikož uživatel může vlastnit více dronů (UAV) a zároveň se nepředpokládá, že jich bude vlastnit tisíce, využívá se zanořeného modelu. Dále může uživatel provádět více misí, které obsahují velké množství dat a s předpokladem, že misí může být velký počet. Proto je zde využit normalizovaný model, kde v dokumentu users jsou pouze reference na mise, které jsou v samostatném dokumentu.

[1] Atomicita, Konzistence, Izolovanost, Trvanlivost
[2] Basic Availability, Soft state, Eventual Consistency

Příklad datového modelu:

```
{
  _id: ObjectId('abcd'),
  name: 'zdenekpasek',
  password: '$2b$10$dbvCEx.dsdsdrruqo',
  uavs: [
    {
      _id: ObjectId('u1'),
      UAV: 'DJI Mavic Air',
      weight: '200g',
      category: 'Professional',
    },
    {
      _id: ObjectId('u2'),
      UAV: 'Custom drone',
      weight: '500g',
      category: 'Custom',
    },
  ],
  missions: [ObjectId('m1'), ObjectId('m2'), ObjectId('m3')
}

{
  _id: ObjectId('m1'),
  name: 'Mission Impossible',
  pilot: ObjectId('abcd'),
  uav: ObjectId('u2'),
  gps: [40.741895, -73.989308],
  date: '12-04-2020',
  battery: 'battery1',
  weather: { tmp: 20, wind: 60 },
  desc: 'description',
}
```

3.4. Použité technologie

V této kapitole jsou popsány zvolené technologie, které byli použity při vývoji klientské a serverové části aplikace. Také se zde nachází stručný popis, proč byli tyto technologie zvoleny včetně jejich výhod a nevýhod.

3.4.1. Front End

Stěžejní technologií na klientské straně aplikace je React-Native. Tento framework byl zvolen hlavně z možnosti vývoje multiplatformních aplikací pro iOS a Android. Oproti nativnímu vývoji má výhodu v tom, že není potřeba psát specifický kód pro určitou platformu, ale je použit pouze jeden programovací jazyk (JavaScript). Dále byl zvolen kvůli předchozím zkušenostem autora práce s JavaScript knihovnou ReactJS, která je tomuto frameworku velice podobná a pracuje na stejných principech [11].

React-Native

Jde o JavaScript framework pro tvorbu multiplatformních mobilních aplikací vytvořený společností Facebook [12]. Je založený na populární JavaScriptové knihovně ReactJS, která je hlavně oblíbená v oblasti vývoje webových aplikací. Podobně jako u zmíněné knihovny, jsou aplikace v React-Native psány pomocí jazyka JavaScript a JavaScript-XML (JSX). Na rozdíl od ReactJS, který renderuje webové komponenty pomocí Virutal-DOM, React-Native vytváří nativní UI komponenty pomocí „mostu“, který vyvolává renderování API v Objective-C/Swift (pro iOS) nebo v Javě (pro Android). Díky tomu mají aplikace přístup k platformově závislým API, a proto můžou přistupovat například k funkcím jako je současná poloha nebo fotoaparát [13].

Výhody React-Native:

- Multiplatformní vývoj
- Znovupoužitelnost komponent
- Použití jednoho programovacího jazyka
- Rychlost aplikací
- Přístup k platformovým API
- Podpora ECMAScript 6 (ES6)

Nevýhody React-Native:

- Relativně nová technologie
- Některé funkce iOS a Android nejsou podporované

Expo

Před vývojem mobilní aplikace pomocí frameworku React-Native je důležité se rozhodnout, zda budeme pracovat s „čistým“ React-Native nebo s dopomocí nějakého frameworku. Hlavní nevýhodou nativního vývoje je složité nastavování vývojového prostředí (XCode, Android Studio) nebo nutnost vlastnit zařízení s macOS pro vývoj iOS aplikace. Pro tuto práci byl zvolen framework Expo, který umožňuje rychlou a jednoduchou instalaci aplikace, bez nutnosti vlastnit specifická zařízení pro vývoj.

Expo je sada nástrojů a služeb pro tvorbu aplikací v React-Native, která poskytuje efektivnější způsoby pro instalaci, vývoj a testování aplikace [14]. Mezi základní nástroje patří Expo CLI a Expo client. Expo-cli je konzolová nebo webová aplikace, která slouží jako interface mezi developerem a Expo nástroji. Hlavní využití má při vytváření nových aplikací, vývoji, zobrazování logů nebo publikování aplikace. Expo client je mobilní aplikace, díky které je možné real-time testovat a zobrazovat aplikaci rovnou na fyzickém zařízení a není tedy nutné buildovat aplikaci v XCode nebo Android Studiu. Dále umožňuje jednoduché posílání aplikace ostatním k testování, bez nutnosti vytvářet apk nebo ipa soubory [15].

Výhody Expo:

- Rychlá a jednoduchá instalace (setup) nové aplikace
- Vývoj bez nutnosti vlastnit specifická zařízení
- Rychlé propojení pomocí QR kódu
- Efektivní vývoj a testování aplikace
- Zobrazování aplikace přímo na fyzickém zařízení
- Expo SDK – přístup k akcelerometru, kameře, notifikacím atd.

Nevýhody Expo:

- Není možné přidávat nativní moduly napsané v Objective-C, Swift, Java, Kotlin
- Aplikace mají větší velikost (HelloWorld např. 25Mb)
- Bez možnosti vybrat si verzi React-Native (závislé na Expo)
- Pomalejší zobrazování změn při ukládání kódu

3.4.2. Back End

Pro serverovou část aplikace bylo zvoleno JavaScript open-source běhové prostředí Node.js. Hlavním důvodem zvolení této technologie je použití programovacího jazyka JavaScript, jak na klientské, tak na serverové části aplikace. Dalším důvodem je vysoká rychlost aplikací, jelikož kód je na straně serveru kompilován pomocí V8 JavaScript enginu, který je hlavní součástí webových prohlížečů založených na Chromiu [16].

Node.js

Jedná se o cross-platformní běhové prostředí, které umožňuje využívat JavaScript i mimo webové prohlížeče. Toto například pomáhá programátorům, kteří se zabývají převážně klientskou části, pracovat i na straně serveru, bez nutnosti učit se nový programovací jazyk. Node.js využívá modelu událostí, kdy callback funkce signalizují o tom, zda byl úkol splněn [17]. Dále je založen na asynchronních I/O operacích, které minimalizují režii procesoru, maximalizují výkon a umožňují tisíce různých spojení v jeden moment, bez nutnosti řešení souběžnosti threadů. Jakmile se klient chce připojit k serveru, Node nevytváří pro každé spojení nový thread, ale místo toho obdrží veškeré žádosti v jednom threadu, a ty deleguje ke zpracování „workerům“ [18].

Velkou výhodou je předinstalovaný správce balíčků (Node Package Manager), který umožňuje velmi efektivně a jednoduše instalovat velké množství knihoven. Typickým a ideálním příkladem aplikace využívající Node.js je chat, který potřebuje v jeden moment velké množství spojení a zároveň rychlou odezvu. Naopak Node se nehodí například na aplikace, které provádějí složité matematické operace.

Výhody Node.js:

- Využití jednoho programovacího jazyka (klient/server)
- Rychlost aplikací
- Možnost velkého množství spojení najednou
- Správce balíčků (NPM)

Nevýhody Node.js:

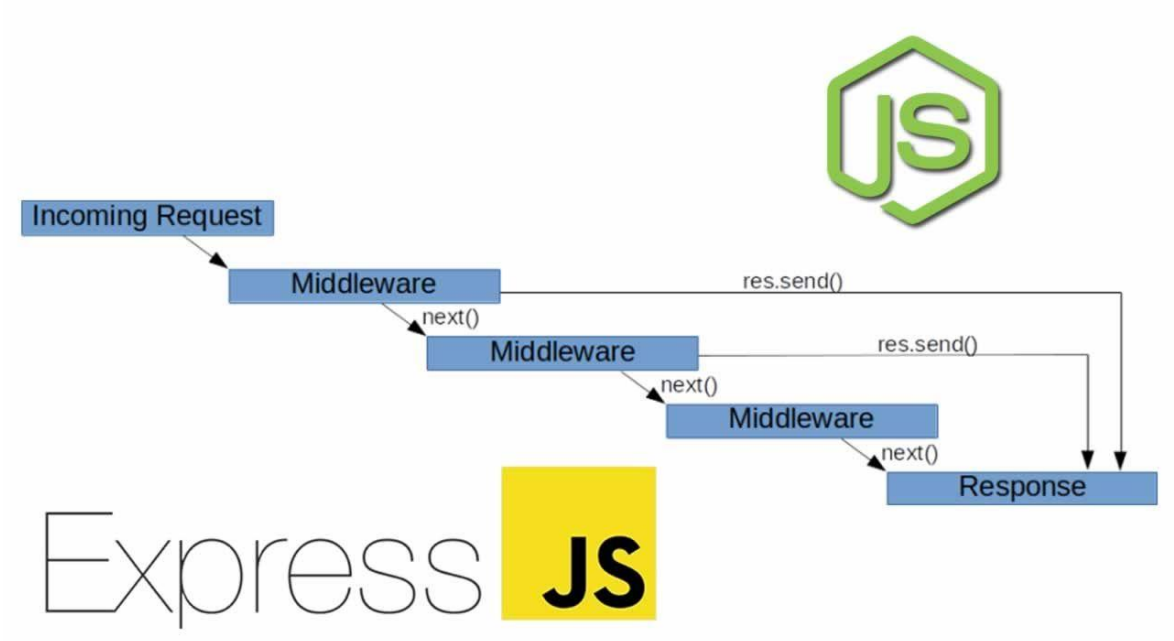
- Nevhodnost pro aplikace využívající náročné operace na procesor
- Horší práce s relačními databázemi
- Horší škálovatelnost

Express.js

Express je flexibilní a jednoduchý Node.js framework, který poskytuje robustní sadu funkcí pro webové a mobilní aplikace. Pomáhá k rychlejší a chytřejší implementaci server-side webových aplikací. Velkou výhodou je jednoduchost kódu, škálovatelnost, databázová integrace a možnosti používání middleware funkcí. Tyto funkce umožnují aplikovat akce u příchozích requestů (upravovat req a res objekty, ukončit cyklus, zavolat další middleware) a poslat zpět odpověď [19].

Výhody Express.js:

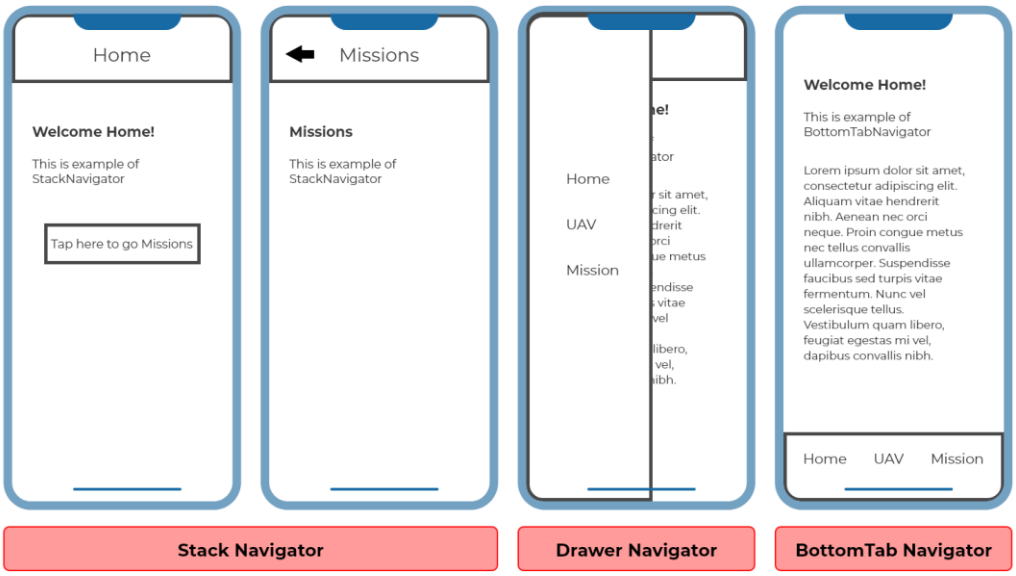
- Jednoduchost kódu
- Rychlá a chytrá implementace
- Škálovatelnost
- Databázová integrace
- Middleware funkce
- Velká komunita (standard node.js aplikace)



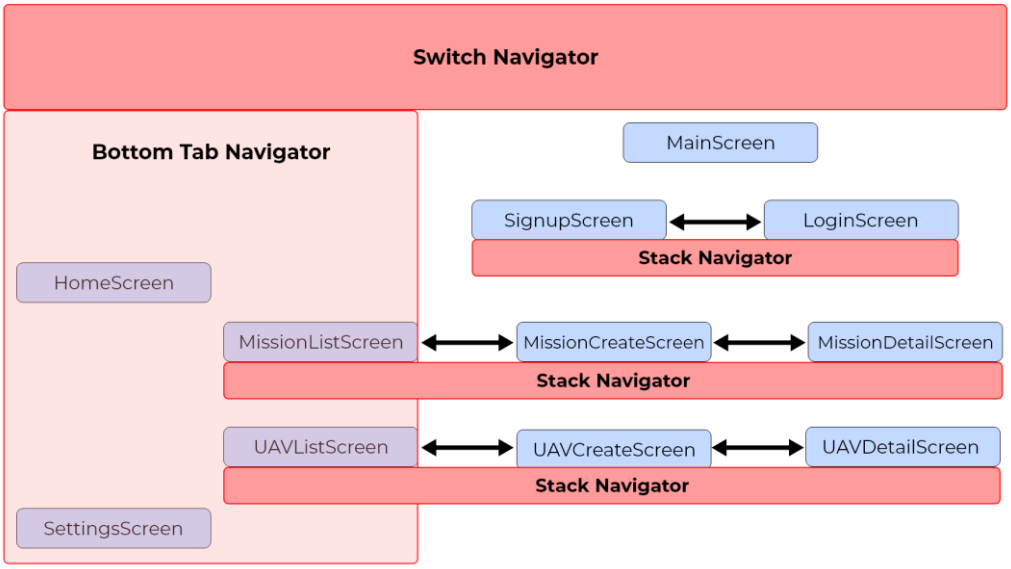
Obrázek 10: Express JS middleware

3.5. Navigace

Jelikož aplikace disponuje velkým množstvím obrazovek, je třeba si rozvrhnout, jakým způsobem budou jednotlivá okna mezi sebou interagovat. Je vhodné si navrhnout základní strukturu pro navigaci. React Native doporučuje pro implementaci navigace komunitní řešení, a to knihovnu React Navigation. Tato knihovna nabízí tři základní navigační řešení – Stack Navigator, Drawer Navigator a BottomTabNavigator. Dále doporučuje použití Switch Navigatoru, který se stará o zobrazení pouze jednoho okna v čase. Defaultně také při přepnutí na jiné okno resetuje stavy těchto oken na jejich defaultní a tím napomáhá Authentication Flow[3] [20].



Obrázek 11: Typy navigace v React Navigation



Obrázek 12: Navržená struktura navigace

[3] Autentizace v rámci aplikace (uživatel má přístup pouze k jeho přiděleným datům)

4 Implementace

V této části práce je popsána celková implementace aplikace, která se dělí na tři hlavní části – Front-end, Back-end a testování. V první části jsou probrány zásady vývoje, které jsou důležité pro čitelnost kódu, konzistenci a celkové pochopení sytému. Dále je v sekci Front-end popsáno založení projektu, struktura projektu a v neposlední řadě dependence, použité knihovny a příklady znovupoužitelných komponent. Back-end se zabývá samotným popisem logiky aplikace a jeho nasazení. V sekci testování jsou zejména testovány REST API endpointy, testování na fyzických zařízeních a uživatelské testování.

4.1. Zásady vývoje a nastavení prostředí

Cílem je dodržování veškerých programátorských konvencí, ať už obecných nebo specifických dle knihovny/frameworku, které vedou k celkové přehlednosti a srozumitelnosti kódu.

Jako vývojové prostředí je použit software Visual Studio Code [21], který poskytuje různé možnosti rozšíření právě pro dodržování těchto zásad. Jako hlavní rozšíření je použit ESLint – nástroj pro identifikování a reportování paternů kódu v reálném čase [22]. Napomáhá k celkové konzistenci a bug-free[4] kódu. Tento nástroj je zcela nastavitelný a záleží čistě na uživateli, která pravidla pro psaní kódu si zvolí. Autor aplikace používá konvenční pravidla od společnosti Airbnb [23], která pro svoji aplikaci používala dlouhá léta technologii React Native. Dále je použito rozšíření Prettier, které slouží pro specifické formátování kódu po uložení.

- Getting Real
- Dodržování konvencí
- Čistý a konzistentní kód
- Znovupoužitelnost komponent
- Verzování (Git)[5]

[4] Kód bez bugů (chyb)
[5] Distribuovaný systém pro verzování kódu (trackování změn)

4.2. Front-end

Tato kapitola popisuje implementaci klientské části aplikace. Klientská část se řídila z předešlých návrhů a designů. Finální design UI je dostupný v přílohách.

4.2.1. Setup a dependence

K vytvoření nového React Native projektu jsou aktuálně dvě možnosti – pomocí Expo-cli nebo React-Native-cli. Obě tyto metody spolu přinášejí určité výhody a nevýhody. Expo-cli defaultně nabízí přístup ke spoustě nativních API a také není potřeba separátně nastavovat buildování pro iOS a Android. Nevýhodou je, že je svázán nativní funkcionalitou, kterou nabízí React Native a Expo SDK a nelze tedy přidávat další nativní API. Na druhé straně React-Native-cli dává vývojáři větší kontrolu nad nastavením celé aplikace [24]. Pro vytvoření projektu byl zvolen Expo-cli (npx expo-cli init flightDiary).

Seznam dependencí pro běh aplikace a pro vývoj:

```
  "dependencies": {
    "@expo/vector-icons": "^10.2.1",
    "@react-native-community/checkbox": "^0.5.5",
    "@react-native-community/datetimepicker": "3.0.0",
    "@react-native-community/masked-view": "^0.1.10",
    "@react-native-community/picker": "^1.8.1",
    "axios": "^0.20.0",
    "expo": "~39.0.2",
    "expo-font": "~8.3.0",
    "expo-linear-gradient": "~8.3.0",
    "expo-localization": "~9.0.0",
    "expo-location": "~9.0.0",
    "expo-status-bar": "~1.0.2",
    "formik": "^2.2.1",
    "i18n-js": "^3.7.1",
    "lodash.memoize": "^4.1.2",
    "moment": "^2.29.1",
    "react": "16.13.1",
    "react-dom": "16.13.1",
    "react-native": "https://github.com/expo/react-native/archive/sdk-
39.0.2.tar.gz",
    "react-native-dotenv": "^2.4.2",
    "react-native-elements": "^2.3.2",
    "react-native-flash-message": "^0.1.16",
    "react-native-gesture-handler": "~1.7.0",
    "react-native-localize": "^1.4.2",
    "react-native-modal-datetime-picker": "^9.1.0",
    "react-native-reanimated": "~1.13.0",
    "react-native-responsive-screen": "^1.4.1",
    "react-native-safe-area-context": "3.1.4",
    "react-native-screens": "~2.10.1",
```

```
    "react-native-simple-dialogs": "^1.4.0",
    "react-native-touchable-scale": "^2.1.1",
    "react-native-web": "~0.13.12",
    "react-navigation": "^4.4.1",
    "react-navigation-stack": "^2.8.3",
    "react-navigation-tabs": "^2.9.1",
    "yup": "^0.29.3"
  },
  "devDependencies": {
    "@babel/core": "~7.9.0",
    "eslint": "^7.9.0",
    "eslint-config-airbnb": "^18.2.0",
    "eslint-config-prettier": "^6.11.0",
    "eslint-plugin-import": "^2.21.2",
    "eslint-plugin-jsx-a11y": "^6.3.0",
    "eslint-plugin-prettier": "^3.1.4",
    "eslint-plugin-react": "^7.21.1",
    "eslint-plugin-react-hooks": "^4.0.0",
    "prettier": "^2.1.2"
  },
```

4.2.2. Struktura

Struktura aplikace v React Native není nijakým způsobem dána a je tedy čistě na vývojáři, jaký si zvolí systém. Obecně se doporučuje držet se ověřeného stylu, který doporučuje komunita. Autor zvolil metodu, kdy každá funkcionalita má pohromadě všechno co potřebuje (styly, obrázky, akce nebo integrační testy). Lze tedy na funkcionalitu pohlížet jako nezávislý kus kódu. Ke správnému fungování je ale třeba dodržet pár pravidel [25]:

- Komponenta může definovat zanořenou komponentu nebo servicu. Nemůže definovat okno.
- Okno může definovat zanořené komponenty, okna a servicy,
- Servica může definovat zanořené servicy. Nemůže definovat komponenty a okna.
- Zanořené funkcionality může používat pouze parent (rodič).

Components

Komponenty jsou hlavním prvkem aplikace a slouží jako stavební bloky. Veškerá view aplikace se rozdělí na jednotlivé komponenty, které se v ideálním případě implementují jako znovupoužitelné. Znovupoužitelnost je docílena tím, že komponenta je napsána obecně a při volání této komponenty jsou předávány jiné props[6]. Rozdělují se na stateful a stateless – tzn. ty které uchovávají nějaký stav (chytré komponenty, sledují určitá data) a ty které jen prezentují (presentional/dumb component). Tyto komponenty jsou globální a můžou být tedy použity v jakékoliv části aplikace.

Příklad komponenty MyAppText pro vlastní text:

```
const MyAppText = ({ children, fontWeight, fontSize, customStyle }) => {
  const [loaded] = useFonts({
    Montserrat_light: require('../../assets/fonts/Montserrat-Light.ttf'),
    Montserrat_regular: require('../../assets/fonts/Montserrat-
Regular.ttf'),
    Montserrat_medium: require('../../assets/fonts/Montserrat-Medium.ttf'),
    Montserrat_semibold: require('../../assets/fonts/Montserrat-
SemiBold.ttf'),
    Montserrat_bold: require('../../assets/fonts/Montserrat-Bold.ttf'),
    Montserrat_extrabold: require('../../assets/fonts/Montserrat-
ExtraBold.ttf'),
  });

  if (!loaded) {
    return null;
  }

  return (
    <Text style={[styles(fontWeight, fontSize).text, customStyle]}>
      {children}
```

[6] Props – properties, předávaná data, nastavení komponenty

```
    </Text>
  );
};

const styles = (fontWeight, fontSize) =>
  StyleSheet.create({
    text: {
      fontFamily: !fontWeight
        ? 'Montserrat_regular'
        : `Montserrat_${fontWeight}`,
      fontSize,
    },
  });
```

Screens

Screens obsahují veškeré obrazovky/scény aplikace. Tyto obrazovky slouží jako kontejner, který zobrazuje různé komponenty a využívá funkcí a stavů z contextu. Konkrétní obrazovka může definovat své vlastní komponenty, servicy a obrazovky v případě, že se tyto prvky budou využívat jen na této obrazovce.

Příklad obrazovky pro registraci:

```
const SignupScreen = ({ navigation }) => {
  const { signup, errorMsg } = useContext(AuthContext);
  return (
    <View>
      <AuthForm
        title="Hello, Sign Up!"
        buttonText="Sign Up"
        isSignup={true}
        onSubmit={signup}
      />

      <Text
        style={{ color: 'blue' }}
        onPress={() => navigation.navigate('Login')}
      >
        Already have an account? Log in here.
      </Text>
    </View>
  );
};
```

Services

Services jsou znovupoužitelné nezávislé moduly, které můžou být použity komponentou nebo obrazovkou, ale nesmí definovat své vlastní komponenty a obrazovky. Hlavní funkcí je přístup k externím API.

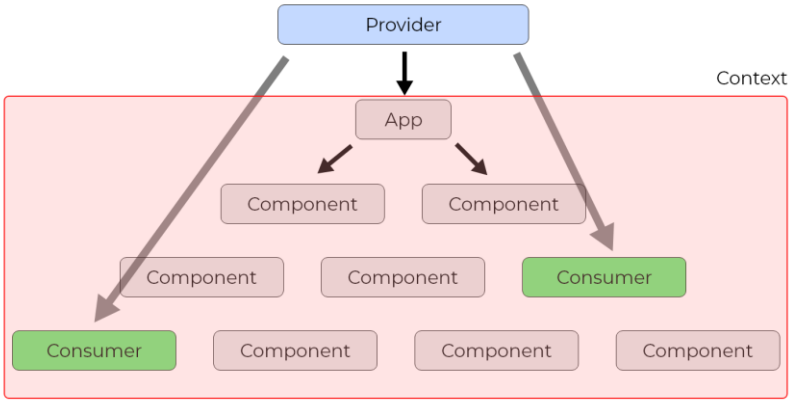
Příklad nastavení tokenu do hlavičky z localStorage:

```
const instance = axios.create({
  baseURL: 'http://51dab816177e.ngrok.io',
});

instance.interceptors.request.use(
  async (config) => {
    const token = await AsyncStorage.getItem('token');
    if (token) {
      config.headers.Authorization = `Bearer ${token}`;
    }
    return config;
  },
  (err) => {
    return Promise.reject(err);
  }
);
```

Context

V typické React aplikaci se data předávají ze shora-dolů (parent-child), ovšem pokud jsou tyto data potřebná ve více komponentách, je tento přístup neefektivní a nepřehledný. Context poskytuje cestu, jak předat data komponentám bez nutnosti manuálního předávání přes props. Data jsou tedy globálně dostupná v jakékoliv komponentě. V této složce se nachází veškeré soubory spojené s contextem.



Obrázek 13: React context-provider

Příklad znovupoužitelné komponenty createDataContext, která vytváří aplikační Provider a Context:

```
export default (reducer, actions, initialState) => {
  const Context = React.createContext();

  const Provider = ({ children }) => {
    const [state, dispatch] = useReducer(reducer, initialState);

    const boundActions = {};

    for (let key in actions) {
      boundActions[key] = actions[key](dispatch);
    }

    return (
      <Context.Provider value={{ state, ...boundActions }}>
        {children}
      </Context.Provider>
    );
  };

  return { Context, Provider };
};
```

Locales

Locales obsahují veškeré JSON soubory pro lokalizaci aplikace ve formátu klíč-hodnota.

Příklad funkce navigate a setNavigator:

```
import { NavigationActions } from 'react-navigation';

let navigator;
export const setNavigator = (nav) => {
  navigator = nav;
};

export const navigate = (routeName, params) => {
  navigator.dispatch(
    NavigationActions.navigate({
      routeName,
      params,
    })
  );
};
```

Hooks

Hooks slouží k extrakci logiky komponenty do znovupoužitelných funkcí. Pokud chceme sdílet stejnou logiku mezi komponenty, extrahujeme tuto logiku do samostatné funkce. Konvenčně tyto funkce začínají názvem „use" a můžou volat ostatní hooky.

Příklad znovupoužitelného hooku useWeather viz Geolokace a počasí

4.2.3. Jazyková lokalizace

Lokalizace je přizpůsobení aplikace pro práci s více jazyky. V této práci je lokalizace implementována za pomocí knihovny i18n-js [26]. Tato knihovna překládá prvky uživatelského rozhraní, kdy v příslušném JSON souboru najde definovaný klíč a vrátí hodnotu. Lokalizace se řídí podle proměnné locale, která definuje nativní nastavení jazyka zařízení. Hodnotu vrací ve standardním formátu (en, en-US, es-US). Pro tuto aplikaci jsou definovány dva hlavní jazyky – a to čeština a angličtina. Pokud je zařízení nastaveno na jiný jazyk, defaultně je aplikace překládána do angličtiny.

Gettery pro získání JSON souborů ve formátu klíč-hodnota:

```
export const translationGetters = {
  'en-US': () => require('./locales/en/en.json'),
  'cs-CZ': () => require('./locales/cz/cz.json'),
};
```

Funkce pro překlad podle klíče:

- Využití knihovny memoize, jenž cachuje hodnoty a znatelně napomáhá rychlosti překladu

```
export const t = memoize(
  (key, config) =>
    i18n.t(key, config).includes('missing') ? key : i18n.t(key, config),
  (key, config) => (config ? key + JSON.stringify(config) : key)
);
```

Inicializační funkce pro nastavení lokalizačního tagu:

```
export const init = () => {
  let localeLanguageTag = Localization.locale;
  let isRTL = Localization.isRTL;

  t.cache.clear();
  I18nManager.forceRTL(isRTL);
  i18n.translations = {
    [localeLanguageTag]:
      localeLanguageTag === 'cs-CZ' || localeLanguageTag === 'en-US'
        ? translationGetters[localeLanguageTag]()
        : translationGetters['en-US'](),
  };
  i18n.locale = localeLanguageTag;
};
```

4.2.4. Validace vstupů

Validování uživatelských vstupů je řešeno zejména na klientské straně aplikace. Každý formulář disponuje svým specifickým schématem, jenž definuje, jaké se ve formuláři nachází vstupy a určuje pro ně typ a pravidla. Validace probíhá v reálném čase, tedy při každé změně stavu vstupu. To má velkou výhodu v tom, že pokud nejsou všechna pole validní, formulář nejde odeslat na server, čímž se šetří jeho zátěž. Konkrétně je využito knihovny Formik a Yup, které práci s formuláři velice usnadňují [27].

Příklad validačního schématu pro registraci uživatele včetně lokalizace:

```
import * as Yup from 'yup';
import { t, init } from '../../localization';

init();
export const authSchema = Yup.object().shape({
  name: Yup.string()
    .min(3, t('nameMinMessage'))
    .max(20, t('nameMaxMessage'))
    .required(t('nameReqMessage')),
  email: Yup.string()
    .email(t('emailValidMessage'))
    .required(t('emailReqMessage')),
  password: Yup.string()
    .min(8, t('passwordMinMessage'))
    .max(30, t('passwordMaxMessage'))
    .required(t('passwordReqMessage')),
  passwordConfirmation: Yup.string().oneOf(
    [Yup.ref('password'), null],
    t('passwordConfirmMessage')
  ),
});
```

4.2.5. Geolokace a počasí

Geolokace je potřebná pro zadávání nové mise a je získávána ze zařízení uživatele. Je tedy určena polohou uživatelovo fyzického zařízení za pomocí knihovny expo-location. Při vytváření nové mise, je uživatel vyzván k udělení práv pro jeho polohu. Jakmile má aplikace práva k získání polohy, nastaví se do vnitřního stavu aplikace a automaticky se předvyplní ve formuláři. Geolokace je získávána ve formátu zeměpisné šířky (latitude) a zeměpisné délky (longitude).

Příklad funkce pro získání geolokace z fyzického zařízení:

```
import { useState, useEffect } from 'react';
import * as Location from 'expo-location';

export default function useGeoLocation() {
  const [location, setLocation] = useState(null);
  const [errorMsg, setErrorMsg] = useState(null);

  useEffect(() => {
    (async () => {
      let { status } = await Location.requestPermissionsAsync();
      if (status !== 'granted') {
        setErrorMsg('Permission to access location was denied');
      }

      let location = await Location.getCurrentPositionAsync({});
      setLocation(location);
    })();
  }, []);

  return { location, errorMsg };
}
```

Jakmile dojde k získání polohy uživatele, je možné vyhledat konkrétní stav počasí v dané oblasti. K tomu je využito služby Open Weather API, které nabízí velkou škálu možností získání dat o počasí. V tomto případě je využito endpointu podle geografických souřadnic (latitude, longitude) - `api.openweathermap.org/data/2.5/weather?lat={lat}&lon={lon}&appid={API key}`

Defaultně jsou data posílána ve standardních jednotkách, kde je například teplota v Kelvinech. Je proto potřeba přidat parametr pro získání dat v metrických jednotkách - &units=metric [28].

Asynchronní hook pro získání počasí v lokalitě uživatele:

```
import { useState } from 'react';
import openWeatherApi from '../services/api/openWeatherApi';
import config from '../config';

export default () => {
  const [results, setResults] = useState([]);
  const [tmp, setTmp] = useState(null);
  const [wind, setWind] = useState(null);
  const [icon, setIcon] = useState(null);
  const [positionName, setPositionName] = useState(null);

  const useWeather = async (lat, long) => {
    try {
      const endpoint = `?lat=${lat}&lon=${long}&units=metric&appid=${config.
API_KEY_W}`;
      const response = await openWeatherApi.get(endpoint);
      const iconName = response.data.weather[0].icon;
      setResults(response.data);
      setTmp(response.data.main.temp);
      setWind(response.data.wind.speed);
      setIcon(iconName);
      setPositionName(response.data.name);
    } catch (err) {
      console.log(err);
    }
  };

  return [useWeather, results, icon, tmp, wind, positionName];
};
```

Použití ve formuláři pro vytvoření nové mise:

```
const [getWeather, results, icon, tmp, wind, positionName] = useWeather();
```

4.3. Back-end

V této kapitole je popsána serverová část aplikace, jenž vychází z návrhu architektury a datového modelu. Back-end primárně pracuje s daty získanými z uživatelského vstupu a tyto data následně zpracovává a ukládá do databáze. Komunikace mezi klientem a serverem je velice jednoduchá, klient pošle požadavek pomocí REST API rozhraní na server, server tento požadavek zpracuje a přiřadí příslušnému controlleru. Jako odpověď server pošle samotný stavový kód HTTP nebo stavový kód a požadovaná data ve formátu JSON.

4.3.1. Dependence

Seznam dependencí serverové části:

```
"dependencies": {
  "bcrypt": "^5.0.0",
  "dotenv": "^8.2.0",
  "express": "^4.17.1",
  "googleapis": "^65.0.0",
  "jsonwebtoken": "^8.5.1",
  "moment": "^2.29.1",
  "mongoose": "^5.10.9",
  "nodemon": "^2.0.5",
  "pdfkit": "^0.11.0"
}
```

4.3.2. Autorizace

Autorizace uživatele je důležitou součástí aplikace. Ověřuje jeho pravost a tím napomáhá k zabezpečení aplikace. K autorizaci je v tomto případě využit standard JSON Web Token. Tento standard definuje kompaktní řešení pro bezpečné předávání informací mezi dvěma stranami ve formě JSON objektu. Tato informace může být ověřena a považována za důvěryhodnou, protože je digitálně podepsána [29].

Token je generován na serverové části po úspěšném přihlášení nebo registraci uživatele a následně je token poslán klientovi. Klient tuto informaci zpracuje a uloží si ji do AsyncStorage, která slouží pro automatické vyplnění hlavičky Authorization tokenem a přihlášení uživatele po restartu aplikace. Klient je tedy nucen přikládat svůj vygenerovaný token při každém požadavku na server. Tento přístup umožňuje chránit veškeré routy, servicy a různé zdroje na serveru před neověřeným uživatelem.

Příklad middleware funkce pro autorizaci uživatele:

```
const mongoose = require('mongoose');
const jwt = require('jsonwebtoken');
const User = mongoose.model('User');

module.exports = (req, res, next) => {
  const { authorization } = req.headers;

  if (!authorization) {
    return res.status(401).send({ error: 'You must be logged in' });
  }

  const token = authorization.replace('Bearer ', '');
  jwt.verify(token, 'secretkey', async (err, payload) => {
    if (err) {
      return res.status(401).send({ error: 'You must be logged in' });
    }

    const { userId } = payload;

    const user = await User.findById(userId);
    req.user = user;
    next();
  });
};
```

Následně je před každým přístupem k routám využito této requireAuth funkce:

```
const express = require('express');
const requireAuth = require('../middlewares/requireAuth');
const UavController = require('../controllers/UavController');

const router = express.Router();

router.use(requireAuth);

router.get('/uav', UavController.getUavs);
router.post('/uav', UavController.create);
router.put('/uav/:id', UavController.update);
router.delete('/uav/:id', UavController.delete);

module.exports = router;
```

4.3.3. Hashování hesla

Zabezpečení uživatelských dat je důležitou součástí vývoje aplikace a je proto vhodné, udržovat citlivá data jako hesla v bezpečí. Do databáze by se nikdy neměla zadávat hesla v podobě uživatelského vstupu, ale použít pro zabezpečení hashovací funkci. V případě této práce byla zvolena hashovací funkce bcrypt, která interně generuje „salt“ hodnotu s počtem opakování 10. Konkrétně se tedy na serverové části před vložením hesla do databáze, zavolá pre-save hook, tento hook vygeneruje „salt“ hodnotu, zahashuje se heslo a následně vloží do databáze. Pro zajištění minimální délky hesla se na klientské straně validuje délka hesla 8 znaků.

Příklad pre-save hooku pro generování „salt“ hodnoty a hashování hesla:

```
userSchema.pre('save', function (next) {
  const user = this;
  if (!user.isModified('password')) {
    return next();
  }

  bcrypt.genSalt(10, (err, salt) => {
    if (err) {
      return next(err);
    }
    bcrypt.hash(user.password, salt, (err, hash) => {
      if (err) {
        return next();
      }
      user.password = hash;
      next();
    });
  });
});
```

4.3.4. Vytvoření a export PDF

Hlavním cílem aplikace bylo data o misích sjednotit a dostat do vhodného formátu. Důležité bylo, aby se s tímto formátem dalo jednoduše pracovat, byl platformově nezávislý a byla jistota, že se na veškerých zařízení zobrazí stejně. Veškeré tyto důvody vedli ke zvolení formátu PDF.

Veškerý export dat probíhá na serverové části aplikace, kdy uživatel požádá server o export svých dat, server tento požadavek zpracuje, vybere z databáze příslušná data, tyto data naplní do připravené šablony a vytvoří samotné PDF. Tento soubor má název podle uživatelovo identifikačního čísla, tzn. při opětovných exportech se pouze tento soubor přepíše a nevytváří se nový. Pro příjemnější práci s PDF a šablonami, je využito knihovny pdfkit [30].

Asynchronní servica pro vytvoření PDF:

- Funkce genereateInnerPdf plní připravenou šablonu vyselectovanými daty

```
const createPdf = async (req, res, missions) => {
  try {
    const { _id } = req.user;
    const pdfPath = path.join('data', 'pdf', _id + '.pdf');
    let doc = new PDFDocument({ margin: 50 });

    res.setHeader('Access-Control-Allow-Origin', '*');
    res.setHeader('Content-Type', 'application/pdf');

    doc.pipe(fs.createWriteStream(pdfPath));
    await doc.pipe(res);

    generateInnerPdf(doc, missions);

    doc.end();

    return { success: true };
  } catch (err) {
    return { success: false };
  }
};
```

Po vytvoření PDF souboru na serveru, bylo důležité zvážit zda poslat celý vytvořený soubor zpět klientovi nebo přijít na jiné řešení. Práce se soubory u multiplatformního vývoje s kombinací s Expem na klientské straně není úplně vhodná, jelikož se chová na každé platformě jinak a mohlo by dojít k nechtěným problémům. Proto bylo zvoleno jednodušší a spolehlivější řešení, kdy se na klientskou část pošle pouze URL souboru, který se otevře v příslušném prohlížeči mobilního zařízení.

Pro vytvoření URL bylo potřeba soubor nahrát na bezpečné místo, ke kterému bude mít přístup pouze uživatel. V tomto případě bylo využito aplikačního rozhraní od společnosti Google, a to konkrétně Google Drive API [31]. Pro aplikaci byl speciálně vytvořen servisní účet, který celé rozhraní spravuje.

Ověření podle JSON Web Tokenu:

```
const auth = new google.auth.JWT(
  credentials.client_email,
  null,
  credentials.private_key,
  scopes
);
```

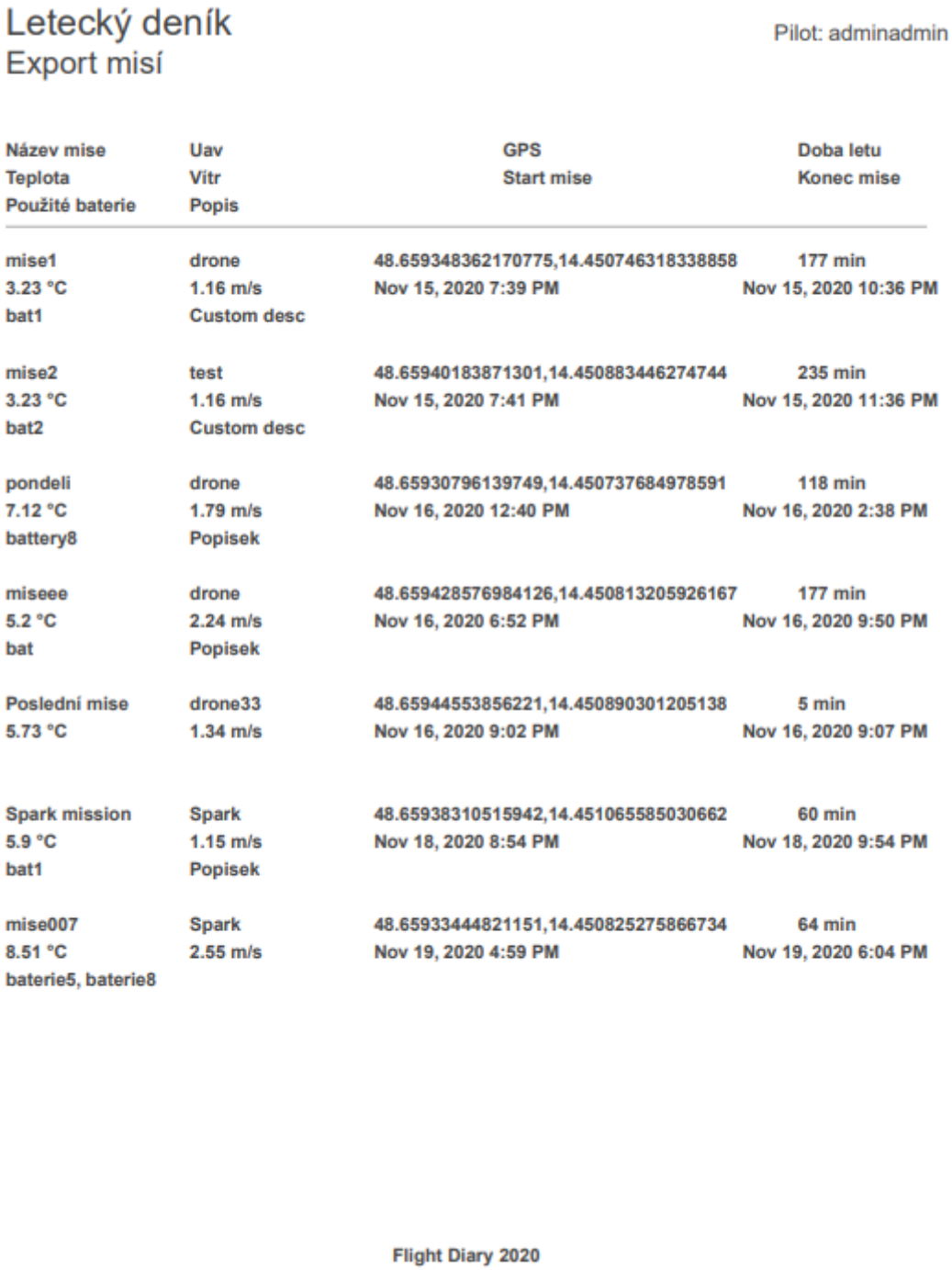
Funkce pro nahrání souboru na google drive podle _id uživatele:

```
exports.uploads = (_id) => {
  return new Promise((resolve) => {
    const auth = new google.auth.JWT(
      credentials.client_email,
      null,
      credentials.private_key,
      scopes
    );
    const drive = google.drive({ version: 'v3', auth });
    var folderId = '1UOHGugDc6r7nbUWROtR71Ql12XMaE3xg';

    var fileMetadata = {
      name: `${_id}.pdf`,
      parents: [folderId],
    };
    var media = {
      mimeType: 'application/pdf',
      body: fs.createReadStream(`${_id}.pdf`),
    };
    drive.files.create(
      {
        resource: fileMetadata,
        media: media,
        fields: 'id',
      },

      (error, result) => {
        resolve({
          fileUrl: `https://drive.google.com/file/d/${result.data.id}/view?u
sp=sharing`,
        });
      }
    );
```

Finální podoba exportovaného PDF souboru:



Letecký deník
Export misí

Pilot: adminadmin

Název mise	Uav	GPS	Doba letu
Teplota	Vítr	Start mise	Konec mise
Použité baterie	Popis		
mise1	drone	48.659348362170775,14.450746318338858	177 min
3.23 °C	1.16 m/s	Nov 15, 2020 7:39 PM	Nov 15, 2020 10:36 PM
bat1	Custom desc		
mise2	test	48.65940183871301,14.450883446274744	235 min
3.23 °C	1.16 m/s	Nov 15, 2020 7:41 PM	Nov 15, 2020 11:36 PM
bat2	Custom desc		
pondeli	drone	48.65930796139749,14.450737684978591	118 min
7.12 °C	1.79 m/s	Nov 16, 2020 12:40 PM	Nov 16, 2020 2:38 PM
battery8	Popisek		
miseee	drone	48.659428576984126,14.450813205926167	177 min
5.2 °C	2.24 m/s	Nov 16, 2020 6:52 PM	Nov 16, 2020 9:50 PM
bat	Popisek		
Poslední mise	drone33	48.65944553856221,14.450890301205138	5 min
5.73 °C	1.34 m/s	Nov 16, 2020 9:02 PM	Nov 16, 2020 9:07 PM
Spark mission	Spark	48.65938310515942,14.451065585030662	60 min
5.9 °C	1.15 m/s	Nov 18, 2020 8:54 PM	Nov 18, 2020 9:54 PM
bat1	Popisek		
mise007	Spark	48.65933444821151,14.450825275866734	64 min
8.51 °C	2.55 m/s	Nov 19, 2020 4:59 PM	Nov 19, 2020 6:04 PM
baterie5, baterie8			

Flight Diary 2020

Obrázek 14: Příklad exportu misí v PDF

4.3.5. Nasazení

Serverová část aplikace je provozována na cloudové platformě Heroku. Tato platforma nabízí distribuční model *PaaS*[7] a je možné zde hostovat aplikace s různými jazyky (Java, Node.js, Go, PHP, Ruby). Aplikace běží na takzvaných *dynos*, což jsou virtuální počítače, které lze škálovat jak horizontálně, tak vertikálně. Heroku si následně měsíčně účtuje poplatek za počet *dynos* a jejich velikost. Ve výsledku je celá Heroku platforma a všechny její aplikace nasazena na Amazon Web Services (AWS). Samozřejmě je možné aplikaci nasadit přímo na AWS s distribučním modelem *IaaS*[8], jenže práce s touto službou je značně složitá a vyžaduje specifické znalosti. Pro představu AWS nabízí 8 certifikátů ověření znalostí této služby. Proto vznikla platforma Heroku, jenž vývojářům ulehčuje práci s nasazením, udržováním a škálováním aplikace.

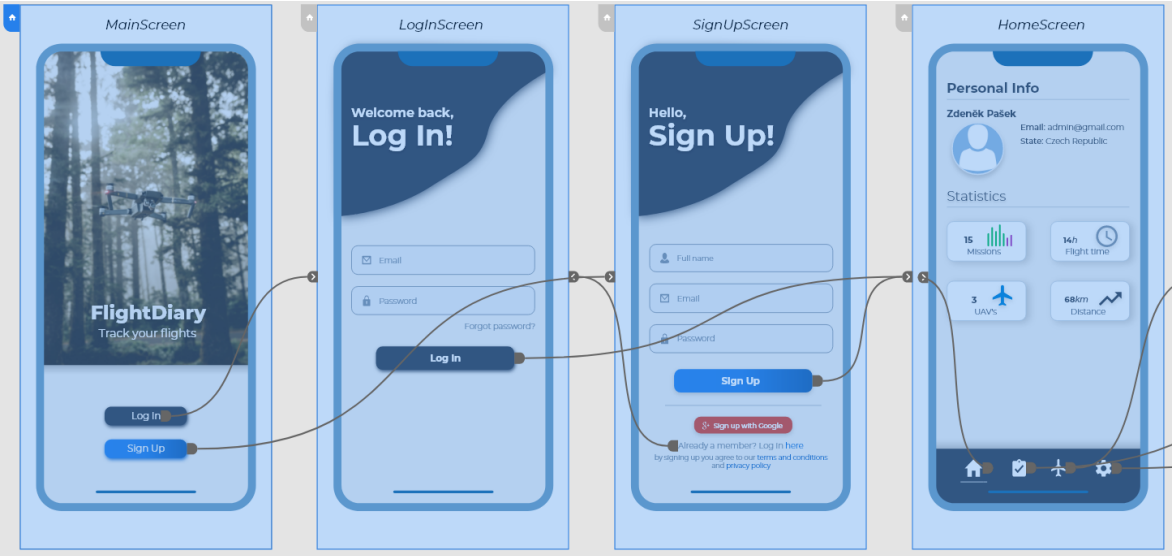
Pro tuto aplikace je prozatím zvolen plán zdarma s uložištěm 512MB a evropským regionem. Jediné omezení je běh serveru 550-1000 hodin měsíčně, což je pro prvotní testování dostačující. Server se po 30 minutové neaktivitě uloží do režimu spánku a čeká na dotaz.

Server je dostupný na adrese: https://limitless-tor-34551.herokuapp.com

[7] PaaS – Platforma jako služba, distribuční model cloud computingu
[8] IaaS – Infrastruktura jako služba, distribuční model cloud computingu

4.4. Testování

Testování aplikace probíhalo v rámci všech částí vývoje. Z počátku byl vytvořen interaktivní prototyp aplikace v programu Adobe XD, který nasimuloval přechody mezi obrazovkami, funkčnost veškerých tlačítek a vlastně určil základní user flow[9] a UX.



Obrázek 15: Základní user flow aplikace

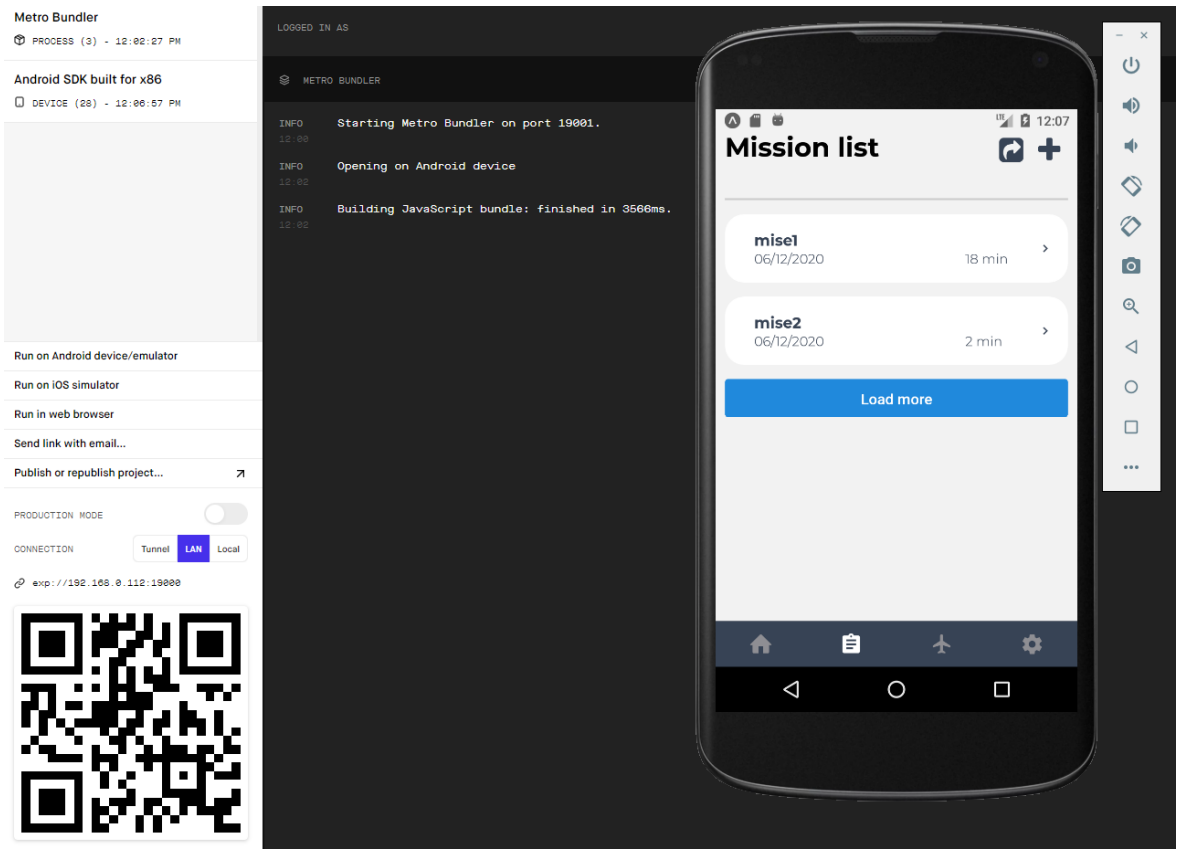
Dále probíhalo testování při samotné implementaci aplikace, kdy veškerá funkcionalita a responzivita byla testována na fyzických zařízeních pro iOS (iPhone 7 a iPhone 12). Pro Android bylo vytvořeno několik virtuálních zařízení s různými verzemi Androidu v aplikaci Android Studio.

Konkrétně šlo o zařízení:

- Nexus 4, 5 – API 23, Android 6.0 Marshmallow
- Pixel 2 – API 26, Android 8.0 Oreo
- Nexus 6 – API 26, Android 10.0 Q

U těchto zařízení bylo hlavně nutné řešit jiné zobrazování modálních oken, selectorů nebo dalších nativně závislých prvků. Bohužel se nepodařilo u některých verzí emulovaného Androidu zprovoznit získávání polohy uživatele pomocí Expo-location. Jedná se o známou chybu ze strany sady nástrojů Expo. Není tedy možné tuto chybu opravit dokud nevyjde aktualizace tohoto nástroje.

[9] User flow – základní cesta typického uživatele v rámci aplikace, cesta k dokončení nějakého úkolu/cíle



Obrázek 16: Testování aplikace na zařízení Nexus 4 API 23

Testování veškerých REST API endpointů bylo provedeno pomocí nástrojů Postman [32] a Swagger [33]. Jednotlivé endpointy byly provolávány těmito nástroji, kde se kladl důraz na vrácení správných dat, popřípadě pro otestování autorizace a chybových hlášek.

V poslední části bylo zavedeno samotné uživatelské testování, při kterém vybraní uživatelé používali aplikaci a podávali zpětnou vazbu.

5 Diskuze a plánovaný rozvoj

Aplikace splnila veškeré stanovené cíle a je velkým přínosem a ulehčením práce pro vlastníky a uživatele bezpilotních prostředků. Sjednocuje veškerá data na jedno místo, umožňuje rychlý přístup k těmto datům a práci s nimi. Uživatel má tedy přehled o veškerých jeho statistikách, vlastnictví a realizovaných misích, na jednom místě.

Limitem aplikace může být využívání sady nástrojů Expo. Aplikace je přímo závislá na této sadě a nelze přidávat vlastní nativní moduly. Pokud by byla vyžadována funkcionalita aplikace závislá na nativním modulu, jenž Expo nenabízí, funckionalita by nebyla možná implementovat.

Pro tuto aplikaci se nabízí rozsáhlý seznam nápadů a funkcionalit, který by podpořil celkové propojení s ostatními službami, uživatelskou autentizaci, samotnou integraci s jednotlivými drony, a tím zajistil příjemnější, rychlejší používání a využití aplikace.

Z počátku by bylo vhodné aplikaci poskytnout pro širokou veřejnost, tzn. nasadit na platformově závislé obchody s aplikacemi (Google Play, AppStore) a zjistit zpětnou vazbu od uživatelů.

Dalším bodem je propojení s ostatními službami. Při přihlašování a registraci uživatele se nabízí využití již existujících účtů. Konkrétní příklad je implementace automatického vytvoření účtu pomocí Google API a Facebook API. Základní potřebná data jako email a jméno by byla převzata z těchto služeb a došlo by k jednoduššímu a rychlejšímu přístupu při práci s uživatelskými účty.

Autentizace a bezpečnost uživatele je důležitou součástí aplikace. V případě mobilního vývoje by bylo velice přívětivé využití lokální autentizace pomocí biometrických vlastností (otisk prstu, snímaní obličeje). Samotná sada nástrojů Expo tuto funkcionalitu nabízí pro obě platformy pod názvem expo-local-authentication.

Jako další klíčovou funkcionalitou by mohla být integrace se samotnými drony. V aplikaci by bylo možné propojit se s konkrétním dronem pomocí USB, WiFi nebo Bluetooth a vzdáleně s ním pracovat přímo v aplikaci. Důležitá data o letu jako GPS pozice, barometr a délka letu by byla vyplňována automaticky přímo ze zařízení. Následně by mohla v detailu mise přibýt mapa, na které by byla vizuálně vyznačena trasa, kterou dron urazil. Veškerá tato funkcionalita by se dala realizovat v případě dronů od společnosti DJI. Tato společnost nabízí pro developery vlastní Mobile SDK, které tuto popsanou integritu podporuje [34].

6 Závěr

Cílem této práce bylo zejména navržení a implementace multiplatformní mobilní aplikace pro evidenci leteckých záznamů bezpilotních prostředků. Z počátku byla provedena analýza, která si kladla základní otázky – pro koho je aplikace určena, jaká je typická proto-persona, s jakými problémy se potencionální uživatel potýká a jak mu touto aplikací může autor usnadnit práci. Po zodpovězení na tyto otázky se vytvořila prvotní myšlenková mapa, která vedla zejména ke specifikaci softwarových požadavků a velkým způsobem pomohla při návrhu struktury navigace. Dalším krokem bylo určení uživatelských scénářů, které nastínili jakým způsobem se bude v aplikaci uživatel pohybovat a také přispěli k tvorbě wireframů. Následně byl stanoven datový model a architektura aplikace, ze které byly vhodně zvoleny použité technologie.

Aplikace byla na klientské straně vyvinuta pomocí frameworku React Native společně se sadou nástrojů Expo a dalších důležitých knihoven. Pro serverovou část bylo využito cross-platformní běhové prostředí pro Javascript s názvem Node.js a podpůrný framework s robustní sadou nástrojů Express. Server je implementován dle architektury REST a jako uložiště dat je použita NoSQL dokumentová databáze MongoDB.

Po celou dobu vývoje probíhalo testování, ať už při implementaci na fyzických zařízeních či emulátorech nebo při provolávání veškerých endpointů přes konkrétní nástroje. V konečné fázi proběhlo uživatelské testování a serverová část aplikace byla nasazena do produkčního prostředí.

Tato aplikace je vhodná zejména pro vlastníky a uživatele bezpilotních prostředků (dronů), kteří chtějí nebo potřebují evidovat záznamy o provedených letech/misích a s těmito záznamy následně pracovat. Aplikace má potenciál ke spoustě rozšíření a vylepšení, které by vedlo k propojení s více službami a ještě lepšímu pocitu při používání aplikace.

Celý projekt byl po celou dobu vývoje distribuován systémem správy verzí Git. Veškerý zdrojový kód je tedy veřejně dostupný jako open-source v oddělených repozitářích pro klientskou a serverovou část na stránce GitHub. V každém repozitáři je také podrobný návod (README.md) jak aplikaci zprovoznit a otestovat.

Klientská část: https://github.com/zdenekpasek/flightDiary
Serverová část: https://github.com/zdenekpasek/flightDiary-server

Seznam použité literatury

[1] *Agremo Crop and Field Analytics: What Agremo analytics is and how it works.* [online]. 2019 [cit. 2019-11-10]. Dostupné z: https://agremo.com/agremo-analytics/

[2] *AIRMAP FOR DRONES: Fly Safely with Instant Access to Airspace Services* [online]. 2019 [cit. 2019-11-10]. Dostupné z: https://www.airmap.com/operators/airmap-for-drones/

[3] *The must have app for Drone Pilots* [online]. 2015 [cit. 2019-11-10]. Dostupné z: https://www.hoverapp.io/

[4] *Úskalí zavádění metodiky řízení vývoje softwaru* [online]. 2013 [cit. 2019-11-10]. Dostupné z: https://www.systemonline.cz/sprava-it/uskali-zavadeni-metodiky-rizeni-vyvoje-softwaru.htm

[5] *Basecamp: About our company: Basecamp: Project Management & Team Communication Software* [online]. 2019 [cit. 2019-11-10]. Dostupné z: https://basecamp.com/about

[6] *Getting Real: The smarter, faster, easier way to build a successful web application* [online]. 2019 [cit. 2019-11-22]. Dostupné z: https://basecamp.com/gettingreal

[7] *The Use of Kanban to Alleviate Collaboration and Communication Challenges of Global Software Development* [online]. 2017 [cit. 2019-11-22]. Dostupné z: https://www.informingscience.org/Publications/3662

[8] *Jira | Issue & Project Tracking Software | Atlassian. Atlassian | Software Development and Collaboration Tools* [online]. [cit. 2019-11-23]. Dostupné z: https://www.atlassian.com/software/jira

[9] *How to Choose a Technology Stack for Mobile Application Development* [online]. 2019 [cit. 2020-10-02]. Dostupné z: https://www.dashdevs.com/blog/how-to-choose-a-technology-stack-for-mobile-app-development/

[10] *Get started with MongoDB* [online]. [cit. 2020-10-25]. Dostupné z: https://docs.mongodb.com/

[11] *React: A JavaScript library for building user interfaces* [online]. [cit. 2019-11-28]. Dostupné z: https://reactjs.org/

[12] *React Native: Learn once, write anywhere.* [online]. [cit. 2019-11-28]. Dostupné z: https://reactnative.dev/

[13] *What Is React Native?* [online]. [cit. 2019-11-28]. Dostupné z: https://www.oreilly.com/library/view/learning-react-native/9781491929049/ch01.html

[14] *What is Expo?* [online]. [cit. 2019-12-05]. Dostupné z: https://docs.expo.io/versions/v35.0.0/

[15] *Expo vs Vanilla React Native: What to Choose for Your Project* [online]. 2018 [cit. 2019-12-06]. Dostupné z: https://apiko.com/blog/expo-vs-vanilla-react-native/

[16] *The Positive and Negative Aspects of Node.js Web App Development* [online]. 2018 [cit. 2020-01-20]. Dostupné z: https://www.mindinventory.com/blog/pros-and-cons-of-node-js-web-app-development/

[17] *Introduction to Node.js* [online]. [cit. 2020-01-20]. Dostupné z: https://nodejs.dev/learn/introduction-to-nodejs

[18] *Node JS Architecture – Single Threaded Event Loop* [online]. [cit. 2020-02-14]. Dostupné z: https://www.journaldev.com/7462/node-js-architecture-single-threaded-event-loop

[19] *Express: Fast, unopinionated, minimalist web framework for Node.js* [online]. [cit. 2020-10-14]. Dostupné z: https://expressjs.com/

[20] *React Navigation: Routing and navigation for your React Native apps* [online]. [cit. 2020-10-02]. Dostupné z: https://reactnavigation.org/

[21] *Code editing. Redefined.: Free. Built on open source. Runs everywhere.* [online]. [cit. 2020-10-02]. Dostupné z: https://code.visualstudio.com/

[22] *ESLint: Find and fix problems in your JavaScript code* [online]. [cit. 2020-10-02]. Dostupné z: https://eslint.org/

[23] *Airbnb JavaScript Style Guide: A mostly reasonable approach to JavaScript* [online]. [cit. 2020-10-02]. Dostupné z: https://github.com/airbnb/javascript

[24] *Expo vs React-Native-CLI: The 2 methods to get a React Native project up and running* [online]. [cit. 2020-10-02]. Dostupné z: https://medium.com/@dinukapiyadigama/expo-vs-react-native-cli-7a3019b2760d

[25] *How to better organize your React applications?* [online]. [cit. 2020-10-02]. Dostupné z: https://medium.com/@alexmngn/how-to-better-organize-your-react-applications-2fd3ea1920f1

[26] *It's a small library to provide the I18n translations on the Javascript. It comes with Rails support.* [online]. [cit. 2020-11-24]. Dostupné z: https://github.com/fnando/i18n-js

[27] *Build forms in React, without the tears* [online]. [cit. 2020-11-24]. Dostupné z: https://formik.org/

[28] *OpenWeather global services* [online]. [cit. 2020-11-24]. Dostupné z: https://openweathermap.org/

[29] *Introduction to JSON Web Tokens* [online]. [cit. 2020-11-23]. Dostupné z: https://jwt.io/introduction/

[30] *A JavaScript PDF generation library for Node and the browser.* [online]. [cit. 2020-11-24]. Dostupné z: https://pdfkit.org/

[31] *Create apps that read, write, and sync files in Google Drive.* [online]. [cit. 2020-11-24]. Dostupné z: https://developers.google.com/drive

[32] *The Collaboration Platform for API Development* [online]. [cit. 2020-11-28]. Dostupné z: https://www.postman.com/

[33] *API Development for Everyone* [online]. [cit. 2020-11-28]. Dostupné z: https://swagger.io/

[34] *Mobile SDK Introduction* [online]. [cit. 2020-11-28]. Dostupné z: https://developer.dji.com/mobile-sdk/documentation/introduction/mobile_sdk_introduction.html#feature-overview

Seznam obrázků

Slovník pojmů

Framework – struktura, která se snaží vyřešit typické problémy při programování, sada knihoven, podpora pro návrhové vzory a postupy

Wireframe - „drátěný model“, návrh aplikace, definuje základní rozvržení prvků aplikace a její funkce

UI/UX – uživatelské rozhraní / návrh produktu, služby aby splnil uživatelovy požadavky

Workflow – pracovní postup

Scrum – způsob řízení vývoje softwaru, definuje strategii vývoje

REST API – architektura pro webové aplikační rozhraní

Back End – Serverová část aplikace

Front End – Klientská část aplikace

JSON – formát pro přenos dat

Virtual-DOM – specifický pojem pro knihovnu/framework React, virtuální objektový model dokumentu vytváření v paměti

Open-source – otevřený software pro další modifikaci a distribuci

Cross-platformní – software, který je implementován na více platformách

Callback – funkce, která je volána po nějaké události

Thread – vlákno

Server-side – na straně serveru

Endpoint – koncový bod

Geolokace – metoda, která získává geografickou polohu objektu

Hook – specifický pojem pro knihovnu/framework React, jiný přístup pro psaní komponent, bez nutnosti psát třídy, nový a moderní přístup

Asynchronní – nesoudobé, nesoučasné

HTTP – internetový protokol, komunikace s WWW servery

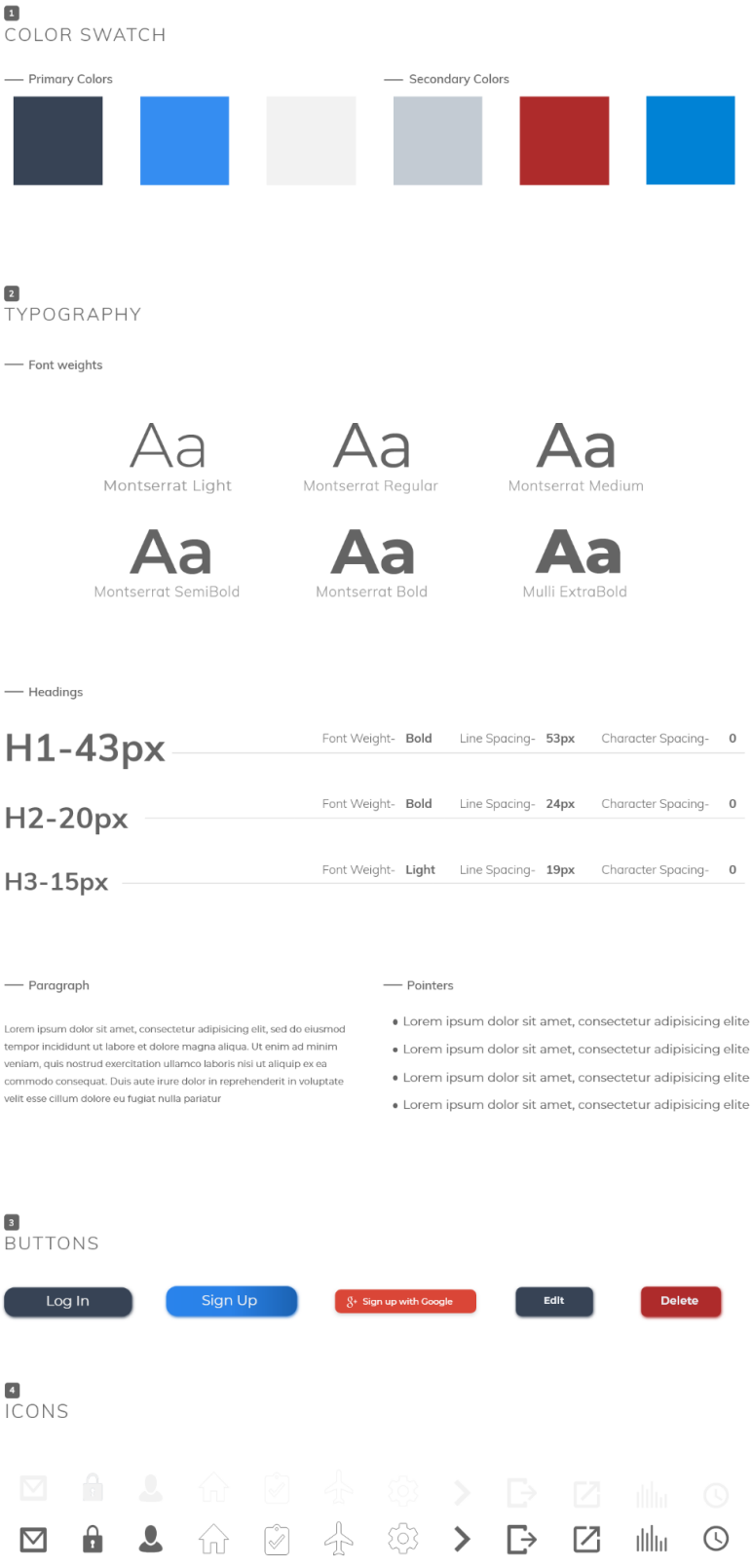
AsyncStorage – je v React Native globální asynchronní uložiště, princip klíč-hodnota

URL – jednotná adresa zdroje

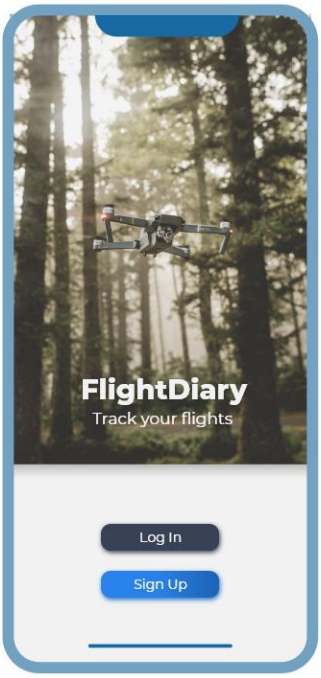
UAV – Unmanned Aerial Vehicle, bezpilotní letadlo, dron

Přílohy

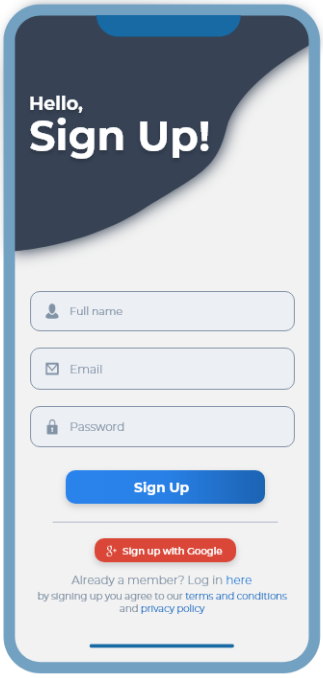
Design



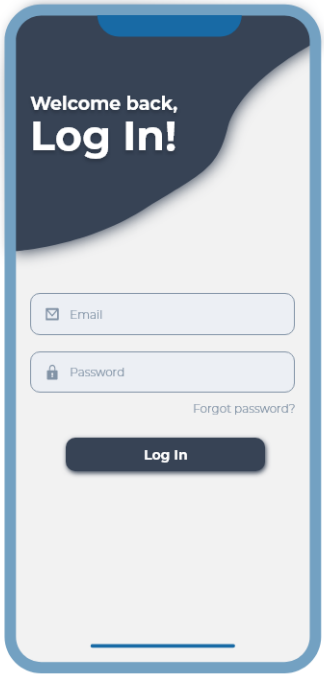
MainScreen



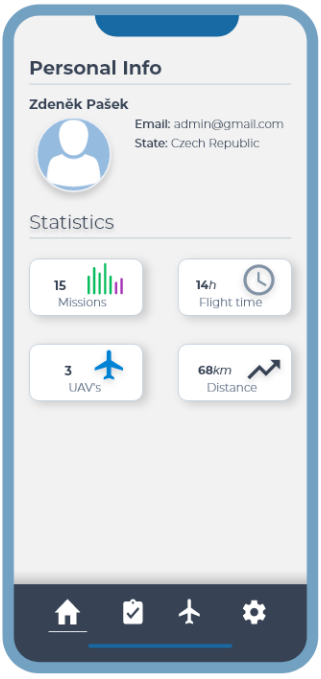
SignUpScreen



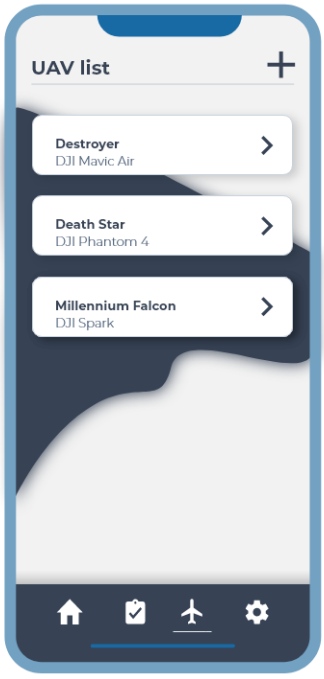
LogInScreen



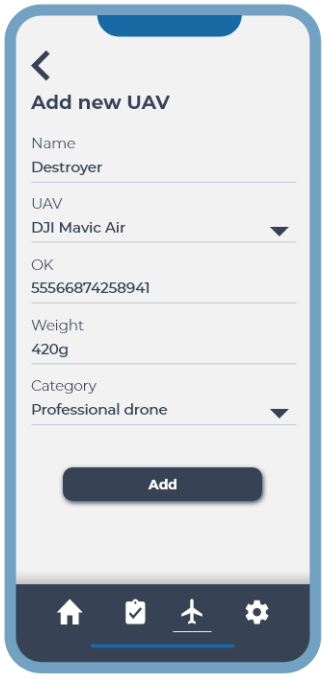
HomeScreen



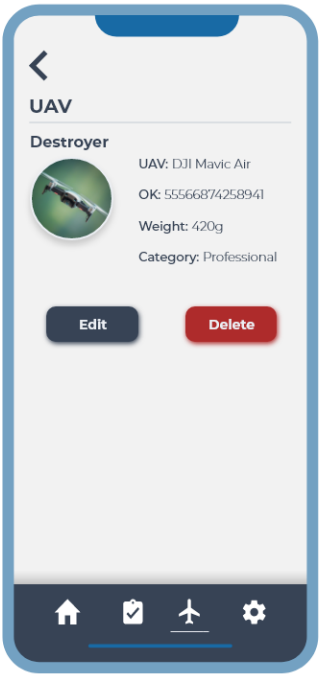
UAV - ListScreen



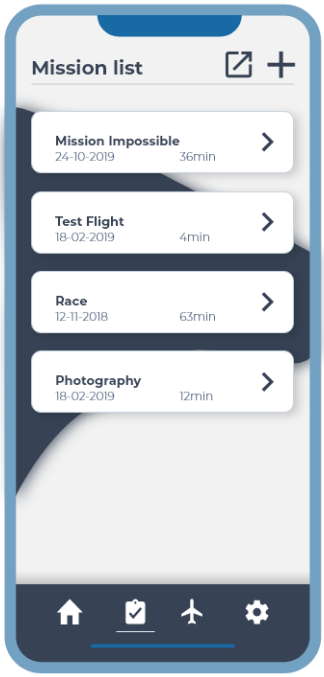
UAV - AddUAVScreen



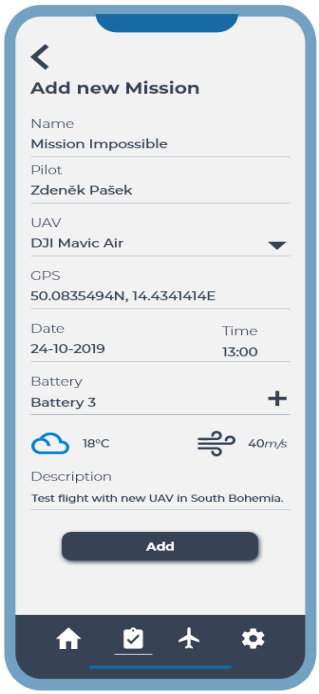
UAV - SelectedUAVScreen



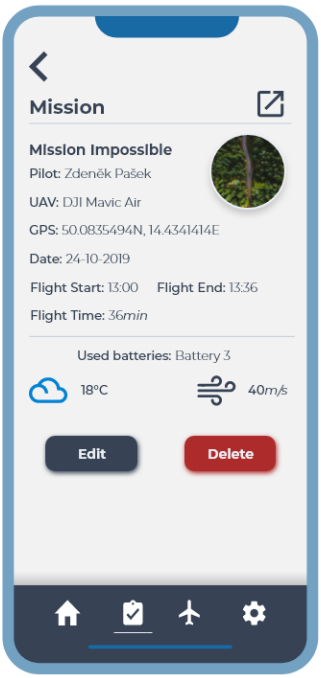
Mission - ListScreen



Mission - AddMissionScreen



Mission - SelectedMissionScreen



SettingsScreen