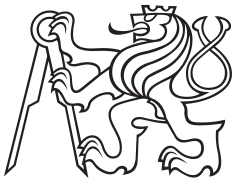
Bakalářská práce



České
vysoké
učení technické
v Praze

F3

Fakulta elektrotechnická
Katedra počítačů

Nástroj pro generování mobilních klientských aplikací pro tvůrce virtuálních naučných stezek v systému EduARd

Dominik Prokš

Vedoucí: Ing. Ivo Malý, Ph.D.
Studijní program: Softwarové inženýrství a technologie
Květen 2021

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

I. OSOBNÍ A STUDIJNÍ ÚDAJE

Příjmení: **Prokš** Jméno: **Dominik** Osobní číslo: **483840**

Fakulta/ústav: **Fakulta elektrotechnická**

Zadávající katedra/ústav: **Katedra počítačů**

Studijní program: **Softwarové inženýrství a technologie**

II. ÚDAJE K BAKALÁŘSKÉ PRÁCI

Název bakalářské práce:

Nástroj pro generování mobilních klientských aplikací pro tvůrce virtuálních naučných stezek v systému EduARd

Název bakalářské práce anglicky:

Tool for generation of mobile client applications for creators of Virtual educational trails in EduARd system

Pokyny pro vypracování:

Proveďte analýzu projektu EduARd. Zaměřte se jak na existující API, které poskytuje přístup k virtuálním naučným stezkám a učebnicím, tak i na existující mobilní klientské aplikace.
Na základě analýzy navrhněte rozhraní mobilní aplikace pro prohlížení virtuálních naučných stezek. Uživatelské rozhraní aplikace bude přizpůsobené k získávání učebnic pomocí API klíče, který je svázán s tvůrcem virtuální stezky a s grafickým vzhledem mobilní aplikace.
Výslednou mobilní aplikaci implementujte s využitím vývojové platformy Flutter. Aplikace bude primárně zaměřena na cílovou platformu Android. Optimalizujte strukturu aplikace tak, aby mohla sloužit jako šablona pro různé tvůrce virtuálních stezek. Pro tvůrce pak vytvořte webovou aplikaci pro naplnění dat do šablony a vygenerování finální aplikace.
Výsledné řešení ověřte pomocí uživatelských testů s alespoň 5 uživateli. Generování výsledné aplikace ze šablony ověřte pro alespoň 2 tvůrce a 4 virtuální naučné stezky.

Seznam doporučené literatury:

1. M. Jones, G. Marsden, Mobile Interaction Design, Wiley, 2006
2. T. Lowdermilk, User-Centered Design, O'Reilly Media, 2013
3. Š. Maňour, Virtuální naučné stezky v React Native, Bakalářská práce, České vysoké učení technické v Praze, 2020.
4. J. Skála, Systém pro správu uživatelů projektu EduARd, Bakalářská práce, České vysoké učení technické v Praze, 2020.
5. D. Truong, Georeferencovaná rozšířená realita, Bakalářská práce, České vysoké učení technické v Praze, 2020.

Jméno a pracoviště vedoucí(ho) bakalářské práce:

Ing. Ivo Malý, Ph.D., katedra počítačové grafiky a interakce FEL

Jméno a pracoviště druhé(ho) vedoucí(ho) nebo konzultanta(ky) bakalářské práce:

Datum zadání bakalářské práce: **12.02.2021** Termín odevzdání bakalářské práce: **21.05.2021**

Platnost zadání bakalářské práce: **30.09.2022**

Ing. Ivo Malý, Ph.D. podpis vedoucí(ho) práce	podpis vedoucí(ho) ústavu/katedry	prof. Mgr. Petr Páta, Ph.D. podpis děkana(ky)

III. PŘEVZETÍ ZADÁNÍ

Student bere na vědomí, že je povinen vypracovat bakalářskou práci samostatně, bez cizí pomoci, s výjimkou poskytnutých konzultací. Seznam použité literatury, jiných pramenů a jmen konzultantů je třeba uvést v bakalářské práci.

.	
Datum převzetí zadání	Podpis studenta

Poděkování

Poděkování patří hlavně panu Ing. Ivu Malému, Ph.D. za jeho skvělé vedení a rady. Také bych chtěl poděkovat všem, kteří byly ochotni se podílet na testování mobilní aplikace.

Prohlášení

Prohlašuji, že jsem předloženou práci vypracoval samostatně, a že jsem uvedl veškerou použitou literaturu.

V Praze, 21. května 2021

Abstrakt

Tato práce se zabývá analýzou a návrhem mobilní aplikace pro řešení virtuálních učebnic. Aplikace je vyvíjena pro školství, aby bylo možné učitelem již na existující webové stránce projektu EduARd vytvořit virtuální učebnici, kterou by po sléze bylo možné stáhnout do mobilních zařízení žáků. V neposlední řadě tato práce obsahuje tvorbu webové aplikace, která umožní učitelům vytvořit instalační soubor mobilní aplikace spojený s API klíčem tvůrce učebnic. Tento soubor je možné použít k instalaci aplikace v mobilních zařízeních s operačním systémem Android. Práce je zakončena uživatelským testováním mobilním aplikace.

Klíčová slova: Flutter, vývoj mobilních aplikací, framework, učebnice

Vedoucí: Ing. Ivo Malý, Ph.D.

Abstract

This work deals with the analysis and design of a mobile application for the solution of virtual textbooks. The application is being developed for education so that the teacher can create a virtual textbook on the existing EduARd project website, which could be downloaded and installed to the pupils' mobile devices. Last but not least, this work contains the creation of a web application that will allow teachers to create a mobile application installation file associated with the API key of the textbook maker. This file can be used to install the application on Android mobile devices. The work ends with user testing of the mobile application.

Keywords: Flutter, mobile application development, framework, textbook

Title translation: Tool for generation of mobile client applications for creators of Virtual educational trails in EduARd system

Obsah

Obrázky

Tabulky

Kapitola 1

Úvod

1.1 Úvod do problému

Cílem této bakalářské práce je vytvořit návrh a analýzu mobilní aplikace pro výukový systém EduARd.

Mým úkolem je vytvořit mobilní aplikaci, jenž by dokázala stáhnout virtuální učebnice ze serveru a následně je zobrazila uživateli. Jednotlivé učebnice v sobě obsahují výukový materiál, úkoly a otázky, které má uživatel vypracovat.

Učebnice se rozdělují do tří druhů, jelikož aplikace má možnost snímat uživatelovu polohu GPS. Těmito druhy jsou učebnice s mapou, se seznamem anebo s obojím. Učebnice s mapou obsahuje sbírku pozic na mapě. Tyto jednotlivé pozice představují úkoly a jsou na mapě vykresleny pomocí značek. Pokud uživatel na jednu ze značek klikne, je přesměrován na daný úkol.

Učebnice se seznamem žádnou takovouto implementaci mapy neobsahuje. Obsahuje pouze seznam úkolů, mezi kterými uživatel může listovat a otevírat jednotlivé úkoly. Tento druh učebnice je zamýšlen pro úkoly, které nejsou svázány s konkrétními součadnicemi.

V poslední řadě existuje učebnice, která kombinuje oba druhy učebnic. Jak si můžeme domyslet, učebnice obsahuje jak mapu, tak soubor otázek, které může uživatel kdykoliv otevřít.

Projekt EduARd aktuálně nemá plně funkční aplikaci, která by dokázala stáhnout data ze serveru a následně je využila v aplikaci. Existuje aplikace vytvořená Michaelou Kubišovou. Tato aplikace momentálně již není udržována ani vyvíjena a tudíž není schopna zpracovat aktuální definice učebnic a úkolů. Proto aplikace není zcela funkční.

1.2 Existující řešení

V této sekci popíši několik existujících řešení podobných mobilních aplikací. Těchto aplikací není k dispozici mnoho a většinou jsou svázány s určitým místem České republiky, pro které jsou vytvořeny. Popisy jednotlivých aplikací vycházejí z funkčních požadavků popsaných v kapitole „Aplikační požadavky a případy užití“2.

1.2.1 Naučné stezky FSC

Aplikace Naučné stezky FSC[1] neobsahuje správu uživatelského účtu. Počet naučných stezek je omezen, jelikož neexistuje možnost přidat uživatelem novou naučnou stezku. Výpis naučných stezek obsažených v aplikaci je vidět na obrázku 1.1a. Aplikace neumožňuje filtrování mezi naučnými stezkami. Tato skutečnost je nejspíše podpořena faktem, že aplikace neobsahuje vysoký počet naučných stezek a ani nelze přidávat další naučné stezky. Po vybrání naučné stezky není uživateli ukázána úvodní stránka, která by obsahovala shrnutí o naučné stezce. Místo toho je přesměrován na obrazovku s mapou naučné stezky. Mapa je vyobrazena na obrázku 1.1b. Klepnutí na bod na mapě je uživatel přesměrován na obrazovku detailu bodu na mapě 1.1c. Na obrazovce detailu bodu na mapě je možné klepnout ve spodní části na tlačítko, jenž přesměruje uživatele na doplňující informace o bodu na mapě. Tato obrazovka je ukázána na obrázku 1.1d. Mapa je schopna pracovat s uživatelovou polohou, jelikož aplikace podporuje využití lokace pomocí GPS. Naučné stezky neobsahují otázky, na které by uživatel odpovídal a tím by rozšířil interakci s aplikací.

[1] `https://play.google.com/store/apps/details?id=cz.azami.stezky&hl=cs`

(a) : Úvodní obrazovka

(b) : Zobrazení na mapě

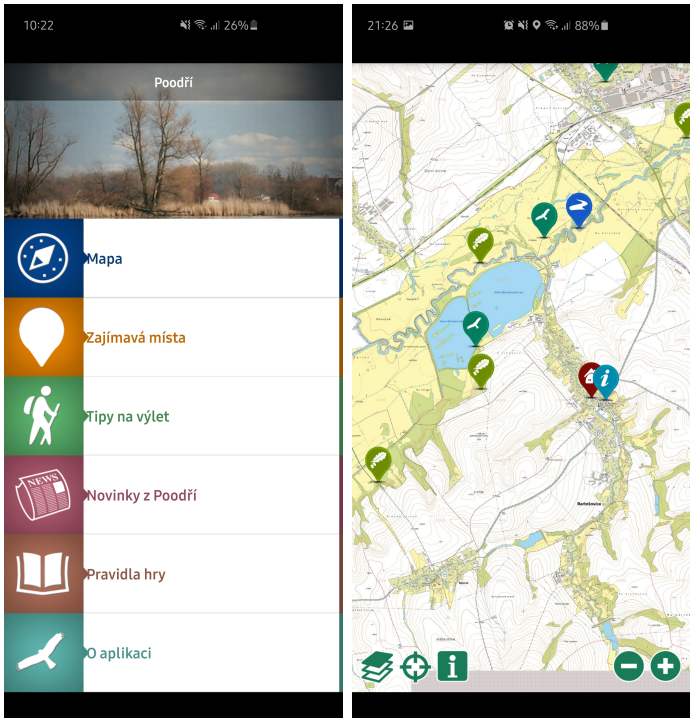
(c) : Detail bodu na mapě

(d) : Další informace k bodu na mapě

Obrázek 1.1: Naučné stezky FSC

1.2.2 Poodří

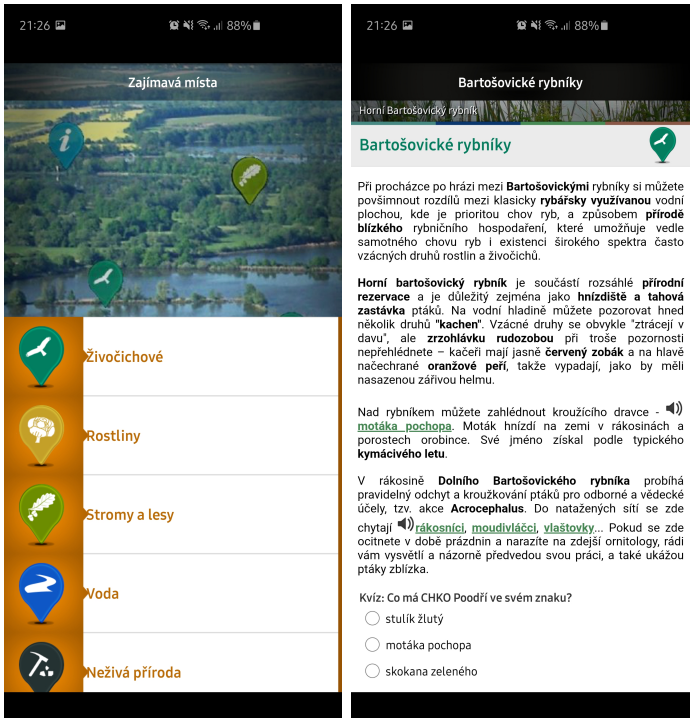
Po zapnutí aplikace Poodří[2] je uživateli ukázána úvodní obrazovka s rozcestníkem aplikace 1.2a. Mapa je ukázána na obrázku 1.2b. Mapa obsahuje pouze krátký popis místa, které zobrazuje. Klepnutí na popis na mapě přesměruje uživatele na obrazovku s popisem místa. Tato obrazovka je znázorněna na obrázku 1.3b. Mimo mapu je možné najít zajímavá místa i v jejich kategoriích. Tyto kategorie můžeme najít v rozcestníku pod možností Zajímavá místa. Výpis kategorií zajímavých míst je ukázán na obrázku 1.3a. Uživatel nemá možnost stažení dalších naučných stezek a zároveň neexistuje možnost filtrování mezi jednotlivými body, tudíž dohledávání popisu místa nebo živočichů apod. může být dosti zdlouhavé.



(a) : Úvodní obrazovka **(b) :** Mapa

Obrázek 1.2: Poodří

[2] https://play.google.com/store/apps/details?id=cz.ubik.poodri&hl=cs



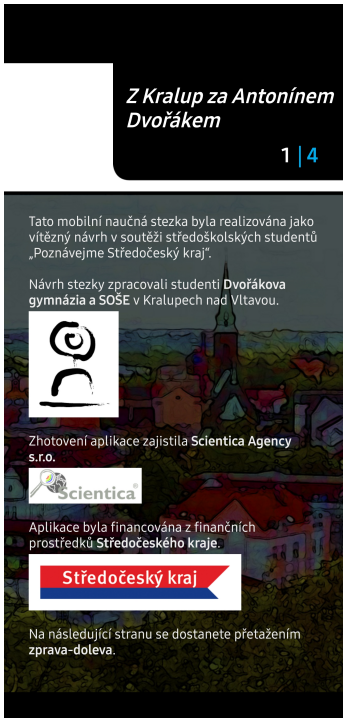
(a) : Zajímavá místa **(b) :** Popis zajímavého místa

Obrázek 1.3: Poodří - doplněk

1.2.3 Z Kralup za Antonínem Dvořákem

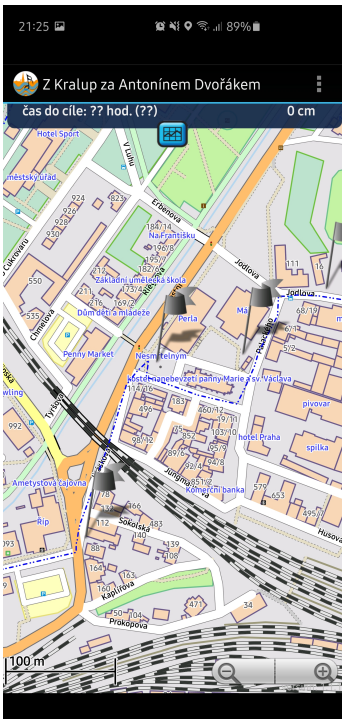
Aplikace Z Kralup za Antonínem Dvořákem[3] rovněž neobsahuje žádnou správu účtu. Úvodní obrazovka obsahuje 4 strany s popisem aplikace. První strana úvodní obrazovky je zachycena na obrázku 1.4a. Obrazovka s mapou, obsahující jednotlivé body na mapě, je ukázána na obrázku 1.4b. Pro otevření bodu na mapě je nutné, aby se uživatel přiblížil místu se zapnutou GPS. Tato aplikace sloužila jakožto předloha pro předchozí verzi aplikace projektu EduARd vytvořenou Michaelou Kubišovou. Další skutečnosti jsem nebyl schopný dohledat, jelikož aplikace neumožňuje otevření bodů na mapě, pokud se uživatel na tomto místě podle GPS souřadnice nenachází.

[3] `https://play.google.com/store/apps/details?id=cz.scientica.kralupy&hl=cs`



(a) : Úvodní obrazovka

(b) : Mapa



(c) : Mapa se zasunutým panelem s dalšími informacemi

Obrázek 1.4: Z Kralup za Antonínem Dvořákem

Kapitola 2

Aplikační požadavky a případy užití

2.1 Funkční požadavky

Tato sekce pojednává o požadavcích, které musí být v aplikaci zahrnuty. Následující funčkní požadavky vyplynuly z analýzy již vytvořené aplikace Michaelou Kubišovou [10] a z průběžných konzultací s Ing. Ivem Malým, Ph.D.

Mobilní aplikace:

- Práce s učebnicemi
  - Aplikace musí umožňit uživateli otevřít učebnici.
  - Aplikace musí umožnit uživateli smazat učebnici.
  - Aplikace musí umožnit uživateli vyhledávat mezi staženými učebnicemi pomocí názvu učebnice.
  - Aplikace musí umožnit uživateli využívat učebnici, tím je myšleno odpovídat na otázky.
  - Aplikace musí umožnit uživateli zobrazit mapu s jeho aktuální polohou.
- Rozdělení obrazovek
  - Aplikace musí umět vykreslit všechny obrazovky učebnice dle jejich nastaveného rozpoložení.

- Rozdělení učebnic
  - Aplikace musí rozdělovat druhy učebnic na *se seznamem*, *s mapou* a *s obojím*.
  - Dle výše uvedených typů musí aplikace měnit práci s učebnicí. To znamená, pokud se jedná o učebnici s mapou, aplikace uživateli nedovolí otevřít otázky z přehledu otázek.
- Stahování učebnic
  - Aplikace musí umět ze serveru EduARd stáhnout učebnici do zařízení pomocí API klíče.

Webová aplikace:

- Aplikace musí umožnit vygenerování instalačního balíčku apk pro operační systém Android

2.2 Nefunkční požadavky

V této sekci jsou obsaženy požadavky na aplikaci, jenž uživatel nijakým způsobem nevidí, ale na práci s aplikací mají také vliv. Tyto požadavky rovněž vyplynuly z konzultací s Ing. Ivem Malým, Ph.D.

- Otevření učebnice nesmí být závislé na internetovém připojení.

2.3 Případy užití

Následující tabulka obsahuje uživatelské scénáře, které vycházejí z již popsaných funkčních a nefunkčních požadavků. Tabulka obsahuje uživatelské scénáře pro mobilní i webovou aplikaci

ID	Název	Aktér	Popis
UC1	Zobrazení stažitelných učebnic	Uživatel	Uživatel má možnost na spodním navigačním panelu klepnout na tlačítko *Ke stažení* a je přesměrován na obrazovku s výpisem stažitelných učebnic.
UC2	Stažení učebnice	Uživatel	Uživatel má možnost klepnout na učebnici na stránce s vypsanými učebnicemi ke stažení. Klepnutí zobrazí dialogové okno s informacemi o stahování a učebnice se stáhne do zařízení.
UC3	Otevření učebnice	Uživatel	Uživatel klepne na obrazovce stažených učebnic na kartu představující učebnici. Tento akt přesměruje uživatele do vybrané učebnice. Pro otevření učebnice není třeba internetového připojení.
UC4	Filtrování	Uživatel	Na obrazovce se staženými učebnicemi zadá uživatel do textového pole název učebnice. Aplikace vyfiltruje učebnice dle zadaného názvu.
UC5	Zobrazení mapy	Uživatel	Uživatel v učebnici s mapou klepne na tlačítko, jenž jej přesune na obrazovku s mapou a vykreslenými body (points of interest).
UC6	Zobrazení přehledu otázek	Uživatel	Uživatel na obrazovce s mapou klepne na tlačítko, které jej přesune na obrazovku s přehledem otázek obsažených v učebnici. Zde záleží na druhu učebnice.
UC7	Zobrazení úkolu	Uživatel	Uživatel má možnost zobrazit si jednotlivé úkoly obsažené v učebnici a pracovat s nimi.
UC8	Odpovězení na otázku	Uživatel	Uživatel má možnost odpovědět na otázku obsaženou v úkolu učebnice.
UC9	Uložení odpovědi	Uživatel	Uživatelova odpověď na otázku je uložena v zařízení.
UC10	Zamknutí odpovědi na otázku	Uživatel	Po odpovězení otázky nemá uživatel možnost znovu odpovědět.
UC11	Smazání úkolu	Uživatel	Uživatel má možnost smazat postup v uložených úkolech učebnice.
UC12	Smazání učebnice	Uživatel	Uživatel má možnost smazat ze zařízení staženou učebnici.
UC13	Vygenerování aplikace	Uživatel	Uživatel má možnost vygenerovat instalační soubor mobilní aplikace pomocí vstupních údajů

Tabulka 2.1: Tabulka s případy užití

Kapitola 3

Design

Tato kapitola pojednává o základních vlastnostech a krátkém obecném popisu Material Designu.

Využití Material Designu v implementaci aplikace odůvodňuji hned několika důvody. Jedním z nich je již obsažená implementace komponent Material Designu ve Flutteru, což výrazně zkrátí dobu vývoje. Maximálně bude třeba nějaké lehké úpravy nebo obalení již připravených komponent. Dalším důvodem je fakt, že s komponenty Material Designu jsem již v minulosti pracoval.

V poslední řadě bych rád podotkl, že je Material Design využit na spoustě vysoce používaných stránek. Jedná se například o stránky Youtube nebo GMail. Pevně věřím, že uživatelé, kteří s výše zmíněnými stránkami pracují skoro na denním pořádku, nebudou mít s používáním aplikace sebevětší problémy.

3.1 Material Design

Material Design byl vytvořen společností Google pro využití v mobilních aplikacích a webovém prohlížeči. Jak je psáno v popisu principů Material Designu [8], je inspirován reálným světem a způsobem jakým spolu objekty interagují . Material Design se skládá z komponent, mezi které se řadí například Appbar, Bottom navigation, Card, Dialog, Snackbar, List, Card, Text

field. Komponenty jsou vytvořeny s ohledem na uživatele a snaží se pomocí vlastní interakce ulehčit uživateli práci a pochopení aplikace (postupná změna barvy, probliknutí), oznamují uživateli, že se něco děje. Pokrývají širokou škálu rozhraní, dle pravidel [7] se řadí na:

- Zobrazení - zobrazování a vkládání obsahu aplikace (Card, List, Sheet)
- Navigace - umožňuje uživateli pohybovat se skrze aplikaci (Tab, Navigation drawer)
- Akce - umožňuje uživateli uskutečnit úkony uvnitř aplikace (Button, Floating action button)
- Vstupy - umožňuje uživateli vkládat vlastní vstup do aplikace nebo vybírat z prvků (Text field, Checkbox)
- Komunikace - zobrazení zpětné vazby uživateli (Snackbar, Banner, Dialog)

Layout Material Designu se musí řídit třemi principy. Těmito principy jsou [9]:

- Předvídatelnost - užití intuitivního a předvídatelného rozložení rozhraní s konzistentními oblastmi uživatelského rozhraní a prostorovou organizací
- Konzistence - rozložení mají být konzistentní s mřížkou, ohraničením znaků a paddingem (odsazením)
- Responzivita - layout je adaptivní a reaguje na vstup od uživatele, zařízení a prvků obrazovky (např. layout se sám přizpůsobí šířce nebo výšce zařízení)

3.2 Návrh uživatelského rozhraní

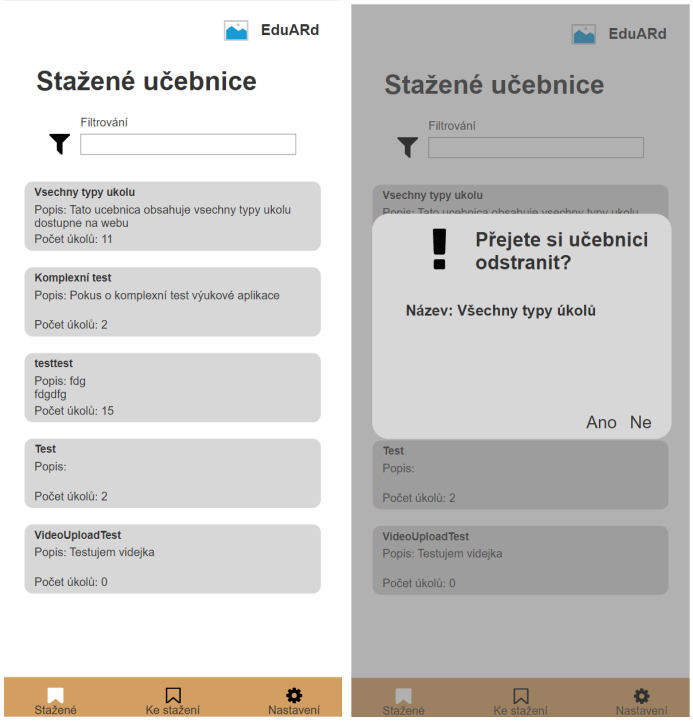
Následující kapitola popisuje vytvořený prototyp, podle kterého bude implementována výsledná aplikace. U každé obrazovky bude popsáno její rozložení, jaké komponenty obsahuje a jaké interakce budou vystaveny uživateli. Nicméně všechny vlastnosti a funkčnost prototypu a jeho obrazovek, které odpovídají požadavkům, budou v následujících podkapitolách popsány u příslušných obrazovek.

3.2.1 Stažené učebnice

Na této obrazovce, kterou znázorňuje obrázek 3.1a, budou uživateli vypsány všechny jeho stažené učebnice, mezi kterými bude moci pomocí textové pole filtrovat. Textové pole přebírá uživatelem zadaný text a hledá názvy učebnic, které zadanému textu odpovídají.

Hlavní obsah této obrazovky, jimiž jsou jednotlivé učebnice, bude realizován pomocí komponent Card, které budou vloženy do Flutter ListView. Tímto docílím jednoduchého vykreslení celé kolekce stažených učebnic. Ve spodní části obrazovky je komponenta Bottom navigation bar, který se objevuje na dalších obrazovkách.

Klepnutím na kartu s učebnicí se uživatel přesune na Úvodní stránku učebnice. Pomocí dlouhého podržení nebo přetáhnutí doleva (swipe doleva) se uživateli zobrazí dialogové okno, znázorněné obrázkem 3.1b, které se uživatele zeptá, zda chce učebnici smazat ze zařízení. Při klepnutí na *Ano* v dialogovém okně se učebnice společně se svými obrázky a jinými přílohami odstraní.



(a) : Stažené učebnice

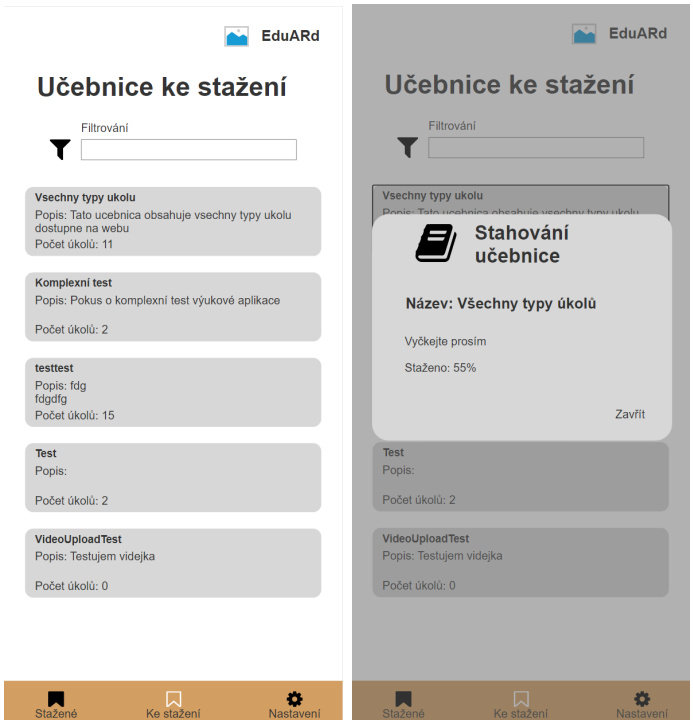
(b) : Dialogové okno pro smazání učebnice

Obrázek 3.1: Obrazovka pro stažené učebnice

3.2.2 Učebnice ke stažení

Na obrázku 3.2a je vyobrazena obrazovka pro učebnice ke stažení. Tato obrazovka je velice podobná již probrané obrazovce se staženými učebnicemi. Jediným rozdílem je, že se zde zobrazují uživateli dostupné učebnice, které si může ze serveru stáhnout. Stejně tak jako na obrazovce se staženými učebnicemi, i zde může uživatel filtrovat pomocí názvů učebnic a přesouvat se v aplikaci pomocí komponenty Bottom navigation panel.

Uživatel může stáhnout libovolnou učebnici tím, že na kartu s učebnicí klepne. Po klepnutí se zobrazí dialogové okno, na kterém budou zobrazeny informace o stahování. Dialogové okno je znázorněno na obrázku 3.2b.



(a) : Zodpovězené otázky **(b) :** Nezodpovězené otázky

Obrázek 3.2: Obrazovka s výpisem stažitelných učebnic

3.2.3 Úvodní obrazovka učebnice

Tato obrazovka, znázorněna obrázkem 3.3, zobrazuje uživateli úvodní informace k učebnici. Pokud bude učebnice obsahovat více úvodních obrazovek, tlačítko pro přesun do nápovědy k učebnicím bude ve spodním navigačním

panelu posunuto více doleva a na jeho původní místo přibude další šipka, která bude směřovat doprava a po klepnutí přesune uživatele na další úvodní stránku učebnice.

Autor učebnice má možnost vybrat rozložení této stránky. Na telefonních zařízeních jsou dvě možnosti a na tabletech jsou možnosti čtyři. Autor má následující možnosti rozložení:

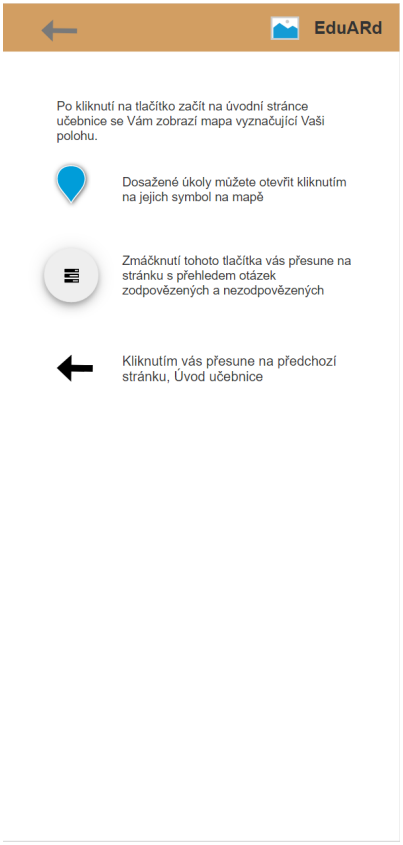
- Obrázky nad textem
- Obrázky pod textem
- Obrázky vlevo od textu
- Obrázky vpravo od textu

Možnosti rozložení obrazovek s obrázkem vedle textu jsou k dispozici pouze na tabletové verzi aplikace. Kliknutí na komponentu Floating action button, který se nachází uprostřed komponenty Bottom navigation panel, zapne uživateli práci s učebnicí.

Obrázek 3.3: Úvodní obrazovka učebnice

3.2.4 Nápověda k učebnicím

Obrázek 3.4 ukazuje rozpoložení obrazovky pro nápovědu k učebnicím. Tato obrazovka slouží jakožto návod k práci s učebnicemi. Uživatel zde může vidět jakým způsobem jsou na mapě zobrazeny body, jenž označují lokaci, kde se nachází otázka nebo úkol. Také zde může vidět jak vypadá tlačítko, jenž jej přesune na obrazovku s výpisem úkolů. Uživatel se klepnutím na šipku zpět v horním navigačním panelu (AppBar) přesune do úvodu učebnice.



Obrázek 3.4: Nápověda k práci s učebnicemi

3.2.5 Mapa

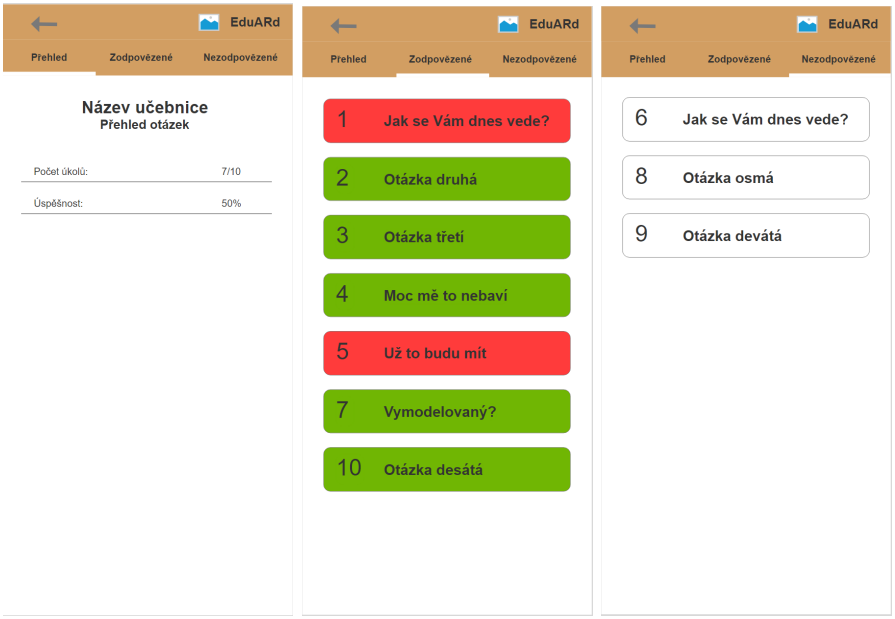
Tato obrazovka, znázorněna na obrázku 3.5, bude uživateli otevřena pouze u učebnic, které s mapou mají pracovat. Na mapě budou vyznačeny body, u kterých se uživateli po příchodu zobrazí otázka nebo úkol. Ve spodní části aplikace se znovu nachází komponenta Bottom navigation panel. Na tomto panelu je vlevo umístěna šipka zpět, která uživatele přesune na úvod k

učebnici. Uprostřed je umístěn Floating action button, jenž uživatele přesune na obrazovku s výpisem otázek.

Obrázek 3.5: Mapa s body

3.2.6 Přehled otázek

Tato obrazovka se dělí na tři záložky. Obrázek 3.6a popisuje první záložku. První záložkou je celkový přehled zodpovězených otázek a jejich procentuální úspěšnost při plnění. Druhou záložkou je výpis všech zodpovězených otázek, znázorněna na obrázku 3.6b. Třetí záložkou je výpis všech nezodpovězených otázek, znázorněna na obrázku 3.6c. Kliknutí na kartu s otázkou přesune uživatele na obrazovku s otázkou, kde může na otázku odpovědět. Možnost přesunu na stránku s otázkou bude povolena pouze u učebnic typu *se seznamem* nebo *s obojím*.

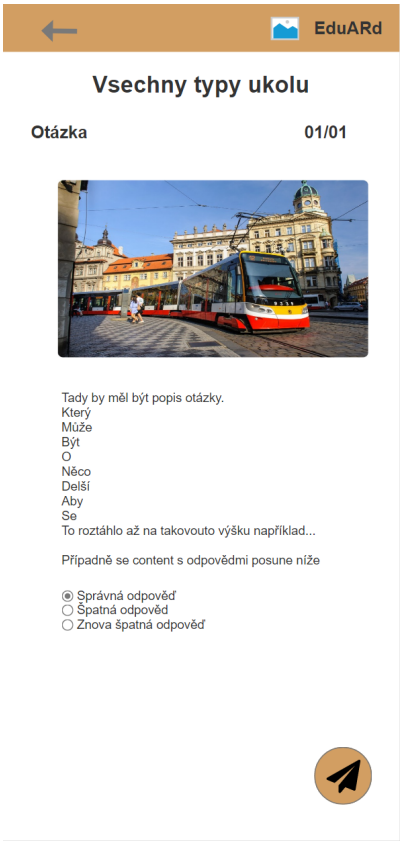


(a) : Přehled otázek **(b) :** Zodpovězené otázky **(c) :** Nezodpovězené otázky

Obrázek 3.6: Přehled otázek

3.2.7 Obrazovka s úkolem

Tato obrazovka zobrazuje uživateli text otázky s případnými doplňujícími obrázky. Obrázek 3.7 je znázorněním této obrazovky. Jelikož si autor knihy může vybrat jaký druh odpovědi chce, bude třeba napsat tuto obrazovku pro všechny možné druhy odpovědí. Může se jednat o textová pole, výběr z checkboxů nebo radio buttonů. Ve spodní části obrazovky se nachází Floating action button, který slouží jako tlačítko pro odeslání odpovědí k vyhodnocení. Toto tlačítko zobrazí uživateli správnou odpověď a přesune jej na přehled otázek.



Obrázek 3.7: Obrazovka s úkolem

Kapitola 4

Analýza nástrojů a struktury

4.1 Použité vývojové nástroje (SDK) pro mobilní aplikaci

Již v zadání práce bylo řečeno, že aplikace musí být implementována pomocí UI toolkitu Flutter od společnosti Google. Tento fakt je pro mne velice vyhovující, jelikož s Flutterem již delší dobu pracuji a navíc je zcela multiplatformní. Ke psaní zdrojového kódu pro Flutter je využíván programovací jazyk Dart, který je taktéž vyvíjen společností Google. Flutter kompiluje kód pro operační systémy Android a iOS. Flutter, právě tím, že kompiluje kód pro obě skupiny zařízení, zkracuje vývoj mobilních aplikací na polovinu a je o hodně jednodušší vytvořit aplikaci pro obě platformy bez nutnosti učení se dalších programovacích jazyků.

V následujících podsekcích porovnám Flutter s jinými druhy vývoje mobilních aplikací, abych ukázal, že Flutter je konkurenceschopný pro vývoj mobilních aplikací.

4.1.1 Nativní aplikace

Jednou z možností vývoje je vytvoření aplikace přímo v nativních programovacích jazycích pro dané platformy. Obrovskou výhodou nativního vývoje

je optimalizace pro jednotlivá zařízení. Bohužel tento fakt má i své úskalí. Řekněme, že budeme potřebovat opravit chybu, která se objevuje na více platformách (např. iOS a Android). Pro její opravu tedy budeme muset zasahovat do kódu pro danou platformu. Zároveň je tedy potřeba znát syntax programovacích jazyků pro danou platformu.

- Výhody
  - Vysoká možnost optimalizace
  - Využití nejnovějších API pro zařízení

- Nevýhody
  - Potřeba znalostí více programovacích jazyků
  - Delší doba vývoje
  - Cena za vývoj

4.1.2 React Native

React Native je jedním z vysoce prosazovaných frameworků pro multiplatformní vývoj. Je vyvíjen společností Facebook a značně používán vývojářskou komunitou. Velikou předností React Native je mimo multiplatformního vývoje i Fast Refresh [3], který vývojáři dovolí vidět změny ihned po uložení souboru s kódem a není tak nutné znovu zdlouhavě kompilovat aplikaci. Pro vývoj v React Native je použit jazyk JavaScript.

- Výhody
  - Multiplatformní vývoj
  - Zkrácení doby vývoje
  - Fast Refresh
  - Obrovská komunita JavaScriptu

- Nevýhody
  - Nikdy jsem s React Native nepracoval
  - Horší optimalizace
  - Některé vývojářské funkce z nativních API nemusejí být podporovány

4.1.3 Flutter

Flutter je open-source framework vytvořený společností Google. Jedná se o multiplatformní framework, který ke svému chodu potřebuje SDK platforem, pro které se bude kód kompilovat. Flutter je UI framework, který vývojáři dovolí rychlé a jednoduché napsání layoutu aplikace. Nápodobně jako React Native i Flutter obsahuje svou verzi Fast Refresh, která je nazvána Hot Reload [4]. UI komponenty Flutteru se nazývají widgety [6]. Z widgetů se tvoří stromová struktura, dále widget tree, která je zpracována a vykreslena. Pro psaní ve Flutter frameworku je použit programovací jazyk Dart.

- Výhody
  - Hot Reload
  - Vyvíjen společností jenž vyvíjí samotný operační systém Android
  - Skvělá dokumentace a návody
  - S Flutterem již delší dobu pracuji
  - Možnost volání nativního kódu z Dartu a vice versa

- Nevýhody
  - Poměrně nový
  - Menší vývojářská komunita
  - Jeden design pro více platforem
  - Neexistují žádné pokyny, kterými by se měli vývojáři řídit

4.2 State management

Tato podkapitola pojednává o výběru možných prostředků pro zpracování uživatelských interakcí a jejich případné uložení.

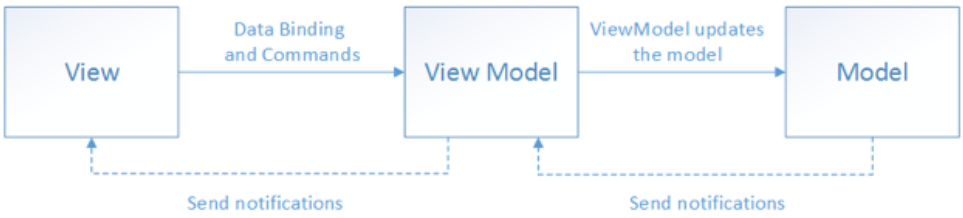
Pro udržování stavu aplikace je z její obsáhlejší velikosti a obnosu dat, se kterými bude potřeba pracovat, ať už se jedná o stažené učebnice nebo učebnice ke stažení, použit architektonický vzor MVVM. Další možností bylo využití architektonického vzoru BLoC[15].

Pro integraci architektonického vzoru MVVM je dobré použít balíček Provider[2], který se stará o propagování notifikací o změnách stavu objektů. Balíček Provider je obalem (wrapperem) pro InheritedWidget[5], který je již součástí Flutter toolkitu.

Rovněž je dobré využít balíček GetIt[13], který se stará o správu jediných instancí objektů a jejich případnou „lazy“ inicializaci.

4.2.1 Architektonický vzor MVVM

Jedná se o způsob oddělení business logiky od vykreslovací logiky aplikace. Architektura vzoru MVVM [1] je následující. Model představuje doménový model, jenž má nějaké své vlastnosti. View představuje vykreslovací část aplikace. View je reprezentací modelu a zachytává uživatelovu interakci. Ta je dále předána do ViewModelu. ViewModel zpracovává uživatelovi interakce podle kterých mění Model. ViewModel je popisován jako stav modelu.



Obrázek 4.1: Rozdělení architektonického vzoru MVVM, obrázek převzat ze článku popisující architekturu vzoru MVVM [1]

- Výhody
  - Větší přehlednost a čitelnost kódu
  - Jednodušší psaní unit testů vůči vnitřní logice aplikace
  - Jsem s ním poměrně dobře obeznámen

- Nevýhody
  - Nejedná se o kompletní oddělení business logiky od zobrazovací části aplikace
  - Pozdější úpravy UI jsou častěji více náročné a většinou se musí sáhnout i do jiné části aplikace

4.2.2 Architektonický vzor BLoC

Stejně jako architektonický vzor MVVM je BLoC určen k oddělení business logiky od zobrazovací logiky. Architektonický vzor BLoC byl navržen s třemi základními hodnotami, kterých se drží. Těmito hodnotami jsou:

- Jednoduchý
- Mocný
- Testovatelný

BLoC se snaží, aby změny stavu byly předvídatelné tím, že reguluje kdy mohou změny nastat [15].

Architektura architektonického vzoru BLoC [14] je dost podobná jiným architektonickým vzorům, které se zabývají oddělení business logiky a zobrazovací logiky. Skládá se z UI, komponenty bloc a dat.

Prezentační vrstva, nebo-li UI, je zodpovědná za vykreslování. Má za úkol vědět, co všechno se má vykreslit na základě jednoho nebo více stavů. Zároveň má za úkol zacházet s uživatelskými vstupy a životním cyklem událostí.

Vrstva business logiky, nebo-li bloc, je zodpovědná za reagování na vstupy od prezentační vrstvy novými stavy. Tato vrstva může spoléhat na jedno či více úložišť, které získávají data potřebná k vytvoření aplikačního stavu.

Datová vrsta je zodpovědná za získávání a manipulaci dat z jednoho či více zdrojů. Skládá se ze dvou částí, úložiště a data provider (poskytovatel dat).

- Data Provider (poskytovatel dat)
  - Odpovědnost Data provideru je poskytnout nezpracovaná data. Data provider by měl být obecný a všestranný
- Repository
  - Repository je wrapper jednoho či více Data providerů, se kterými vrstva business logiky komunikuje.

- Výhody
  - Kompletní oddělení business logiky od zobrazovací logiky aplikace
  - Aktuálně převládající trend Flutteru pro state management

- Nevýhody
  - Těžší na implementaci než architektonický vzor MVVM
  - Nejsem s ním dobře obeznámen

4.2.3 InheritedWidget

InheritedWidget je druh widgetu, jenž umožňuje všem svým potomkům sáhnout si na vystavenou hodnotu. Místo obsáhlého a zdlouhavého předávání hodnoty napříč konstruktory a build metodami skrze widget tree si může jakýkoliv widget, který vychází z InheritedWidgetu, sáhnout na zděděnou hodnotu. Dalo by se říci, že se jedná o způsob state managementu aplikace, který je součástí Flutter frameworku.

- Výhody
  - Zabudován ve Flutteru
  - Jednoduchý na použití

- Nevýhody
  - Používán pouze na menší aplikace
  - Nejedná se o oddělení business logiky od zobrazovací logiky aplikace

4.2.4 Provider

Jak je napsáno v popisu balíčku Provider [2]. Je konstruován jakožto wrapper nad vestavěným InheritedWidgetem. Díky jeho vlastnostem vystavení, vytváření, naslouchání a likvidaci hodnot nebo objektů je vysoce užitečný pro implementaci architektonického vzoru MVVM. Pokud jsou jím držené hodnoty změněny, ihned o tom informuje své posluchače, kteří na tuto změnu mohou reagovat. Může se jednat například o změnu stavu widgetu, načež se daný widget překreslí s aktuálními daty.

- Výhody
  - Zjednodušená alokace a likvidace prostředků
  - Méně boilerplate
  - Lazy-loading
  - Nádherně zapadá do architektonického vzoru MVVM

- Nevýhody
  - Těžší na pochopení

4.2.5 GetIt

Volně dostupný balíček pro získávání objektů z různých částí aplikace. GetIt podporuje registraci jediných instancí objektů, čímž usnadňuje od vytváření nových instancí, které by musel Garbage Collector po chvilce smazat. Typickým použitím je např. přístup k objektům použitým k přístupu na REST API a databáze.

- Výhody
  - Jednoduchý na použití
  - Rychlý - O(1) [13]

- Nevýhody
  - Není přímo určen pro state management

4.3 Webová aplikace

Poslední částí zadání bakalářské práce je vytvoření webové aplikace k jednoduchému zkompilování kódu mobilní aplikace a vytvoření instalačního balíčku pro platformu Android. Webová aplikace má na svém vstupu několik formulářových prvků.

Mezi tyto vstupy patří:

- Název aplikace
- ID aplikace
- API klíč tvůrce učebnic
- Logo aplikace (nepovinné)

Existuje několik možných frameworků, které by tvorbu webové aplikace výrazně ulehčili. Mezi tyto frameworky patří například Laravel, ASP.NET Core a Spring Boot. Kde každý z těchto frameworků využívá jiný programovací jazyk.

- Laravel - PHP
- ASP.NET Core - C#
- Spring Boot - Java

Výběr frameworku nebyl nijak složitý. K vytvoření webové aplikace byl použit framework Spring Boot[1]. Spring Boot jsem vybral hlavně z následujících důvodů:

- Zkušeností s jazykem Java a Spring frameworkem
- Velký počet open-source Java EE serverů

Hlavně základní znalosti Spring frameworku mě přiměly k využití právě Spring Boot frameworku.

[1] `https://spring.io/projects/spring-boot`

Kapitola 5

Implementace

V následující kapitole popíši jakým způsobem probíhala implementace aplikace a za použití jakých prostředků tak bylo učiněno. Popíši vytvoření architektury kódu aplikace, způsob komunikace různých objektů mezi sebou a způsob ukládání dat do databáze pro offline použití aplikace.

5.1 Architektura

5.1.1 Architektonický vzor MVVM

Jak jsem se již zmiňoval v minulé kapitole v části State management, k implementaci aplikace jsem použil architektonický vzor MVVM s pomocí Provider[2] balíčku. Pro propojení vzoru MVVM a balíčku Provider jsem vycházel z článku Jiteshe Mohite[11].

Strukturu projektu jsem tedy rozdělil na views, viewmodels a models. Views představují jednotlivé stránky aplikace, které se zobrazují uživateli. Views jsou skládány z widgetů, ať už mnou vytvořených nebo poupravených, tak z již existujících widgetů Flutteru. Zároveň každé View vytváří svůj ViewModel a uchovává si referenci na tento objekt. ViewModels reprezentují stav Views a provádějí potřebnou logiku zpracování vstupních dat/interakcí od uživatele. ViewModels v sobě narozdíl od Views nenesou referenci na View,

pro které jsou vytvořeny. Models v našem případě představují modely knih a jednotlivých druhů otázek a jejich odpovědí. Detailní popis vlastností a funkčností jednotlivých viewmodels bude obsažen dále v dokumentu.

5.1.2 Models

Pro práci s učebnicemi bylo hlavně třeba vymodelovat třídy, představující strukturu otázek. To znamená, vytvořit mapování XML na objekty, aby se dalo později s úkoly pracovat, odpovídat na ně a ukládat odpovědi do databáze. Mapování jsem dosáhl pomocí balíčku XML[1]. Tento balíček mi pomocí svých funkcí umožnil vytvářet objekty přímo z XML. Způsob jakým se mapování provádí je vidět na ukázce kódu 3. Pomocí dokumentace na GitLabu projektu EduARd jsem postupně vytvořil mapování všech objektů. Některé objekty bohužel nemají početní omezení, např. objekt Task. Proto bylo třeba pomocí funkce getElements z balíčku XML najít všechny tagy task a vytvořit z nich objekty, čehož si můžeme také všimnout na ukázce kódu 3.

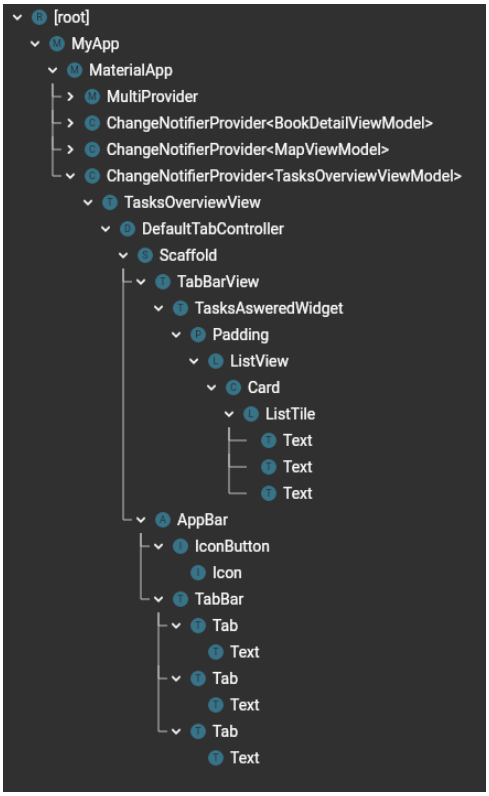
Pro každý XML tag bylo třeba vytvořit k němu odpovídající objekt. U některých objektů to nebylo zcela jednoduché, jelikož např. objekt Images má dva způsoby jakým může být vytvořen. Tag images totiž může být buď jako list a nebo jen jedna hodnota. Během mapování je tedy třeba zjistit o jaký typ se jedná a následně použít správnou metodu mapování. V obou případech se hodnota/hodnoty mapují do listu hodnot. Příklady zápisů tagu images si můžeme povšimnout na ukázce xml 1.

5.1.3 Views

Každé View je představováno objektem dědícím ze třídy StatelessWidget[2] nebo StatefulWidget[3]. Objektům, které jsou vykresleny na obrazovce zařízení, se říká widgety. Právě tyto objekty Flutter vykresluje a přidává do widget tree. Každý takovýto objekt obsahuje metodu build, která zajišťuje vizuální reprezentaci daného objektu. Pod pojmem widget tree si představme stromovou strukturu jednotlivých widgetů.

Ukázka widget tree právě v mnou vytvořené mobilní aplikaci je k vidění na obrázku 5.1.

[1] `https://pub.dev/packages/xml`
[2] `https://api.flutter.dev/flutter/widgets/StatelessWidget-class.html`
[3] `https://api.flutter.dev/flutter/widgets/StatefulWidget-class.html`



Obrázek 5.1: Widget tree

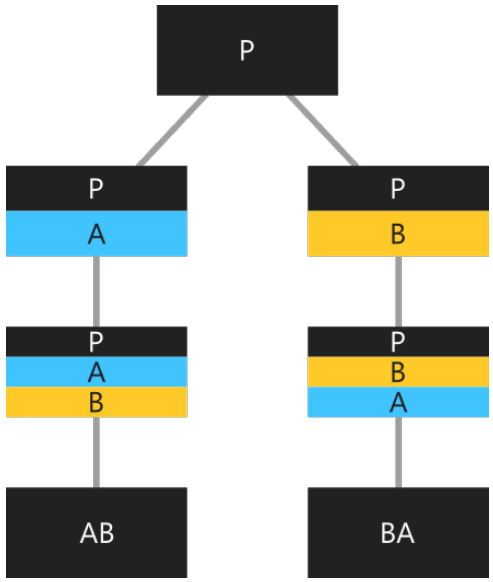
5.1.4 ViewModels

Jak jsem se již zmiňoval výše, ViewModely mají představovat stav jednotlivých View a zároveň mají zpracovávat vstupy od uživatele. Abych tohoto mohl docílit a propojit View s ViewModelem, bylo třeba, aby ViewModel využíval tzv. „mixin“. Pro mé potřeby bylo potřeba použít mixin ChangeNotifier, který ViewModelu umožní notifikovat View o změnách stavu pomocí metody notifyListeners.

Mixin je v programovacím jazyce Dart normální třídou. Důležité ale je, jak tuto třídu Dart používá pro vytváření dědění objektů. Uvažujme stejný případ jako v článku [12] od Romaina Rastela. Mějme následující třídy:

```
1 class AB extends P with A, B {}
2 class BA extends P with B, A {}
```

Klíčové slovo „with“ znamená využití mixinu. Dart tyto třídy a jejich stromovou reprezentaci vytvoří způsobem znázorněným na obrázku 5.2.



Obrázek 5.2: Strom objektů Mixin, obrázek převzat z článku Romaina Rastela [12]

Jedná se o způsob znovupoužití kódu napříč několika hierachiemi objektů, které spolu nesouvisí. Podrobněji je mixin popsán přímo v dokumentaci[4] jazyka Dart a již zmíněném článku od Romaina Rastela.

Ke správnému propojení View a ViewModelu je zároveň třeba, aby View naslouchalo změnám napřímo a nebo využívalo objektu Consumer, který automaticky naslouchá změnám určitých obejktů. V případech, kdy jsem chtěl, aby celé View naslouchalo změnám, bylo potřeba do build metody View přidat tento řádek „T viewModel = Provider.of<T>(context, listen: true);“, který zajistí, že se View napojí jakožto observer objektu T. Takového napojení si můžeme povšimnout na ukázce kódu 4, zde ovšem není vidět parametr listen, jelikož je automaticky nastaven na hodnotu true, neurčí-li se ve volání metody jinak. Zároveň je třeba, aby se ve widget tree objevil nad daným View objekt ChangeNotifierProvider, který propaguje právě změny a notifikace pomocí metody notifyListeners níže do widget tree.

[4] `https://dart.dev/guides/language/language-tour`

5.1.5 Services

V další řadě bylo potřeba vytvořit jednoduché objekty, které by plnily služby jakožto například stažení dat z API nebo uložení dat do lokální databáze. Pro takovéto případy jsem vytvořil objekty APIService, AssetsService, BookService, FileService a TaskDTOService.

Jak již z názvu napovídá BookService se stará o stažení učebnice jako takové. Toho jsem docílil rozčleněním stahování na dílčí podčásti, které jsou zprostředkovány dalšími services. BookService se stará o stažení celku knih a zároveň popisujících informací o učebnicích. Tyto popisující informace, jako je název, počet pokusů atd. je zobrazen uživateli na úvodní stránce aplikace (obrazovky „Stažené učebnice“ a „Ke stažení“). Kompletní stažení učebnice je v BookService implementováno pomocí dalších dílčích services. Pomocí AssetsService jsou staženy assety (obrázky, videa a audionahrávky) a následně uloženy pomocí FileService. Otázky k učebnici jsou staženy v podobě XML pomocí BookService a posléze je XML uloženo na disk zařízení. Cesty ke staženému XML a k assetům jsou zároveň uloženy do lokální databáze v zařízení.

Všechny services jsou přístupné z jakékoliv části aplikace a to díky balíčku GetIt, který se stará o správu singleton instancí zaregistrovaných objektů. Pomocí GetIt jsem schopen přistoupit v jakékoliv části aplikace právě k zaregistrovaným objektům. Způsob zaregistrování objektů je vidět na ukázce kódu 5.

5.1.6 Hive databáze

Pro offline ukládání dat jsem se rozhodl použít NoSQL databázi Hive, která byla vyvinuta čistě v programovacím jazyku Dart a aktuálně se jedná o jedno z nejrychlejších NoSQL databázových řešení pro Flutter.

Pro Hive jsem se rozhodl zároveň také proto, že je možné nechat vygenerovat mapování objektů mezi databází a objekty, tzv. TypeAdapters. O generování se stará build task, který bohužel není přímo součástí balíčku Hive, ale dá se snadno naimportovat skrze balíček hive_generator[5]. Tento build task vygeneruje ke každé třídě, která nese anotaci HiveType a má unikátní typeId, soubor, který se stará o transformace objektů na byty uložitelné do databáze

[5] `https://pub.dev/packages/hive_generator`

a zpětnou transformaci z bytů na objekty. Pro správné vygenerování je třeba objektům přidat anotace, které budou build tasku říkat, jaké vlastnosti mají být zahrnuty a jaké nikoliv. Jednoduchý návod na vytvoření TypeAdapteru je k dispozici přímo na stránkách HiveDB[6]. Za další je potřeba TypeAdaptery zaregistrovat, aby aplikace věděla, že s nimi má pracovat. Jedná se o podobný princip jako při využívání balíčku GetIt. Jakým způsobem se TypeAdaptery registrují je ukázáno na ukázce kódu 7. TypeAdaptery je potřeba zaregistrovat ještě před spuštěním Flutter aplikace. Pro jednoduchou ilustraci jsem přidal úkázku kódu 6, která znázorňuje jednu ze tříd modelu a obsahuje anotace pro Hive.

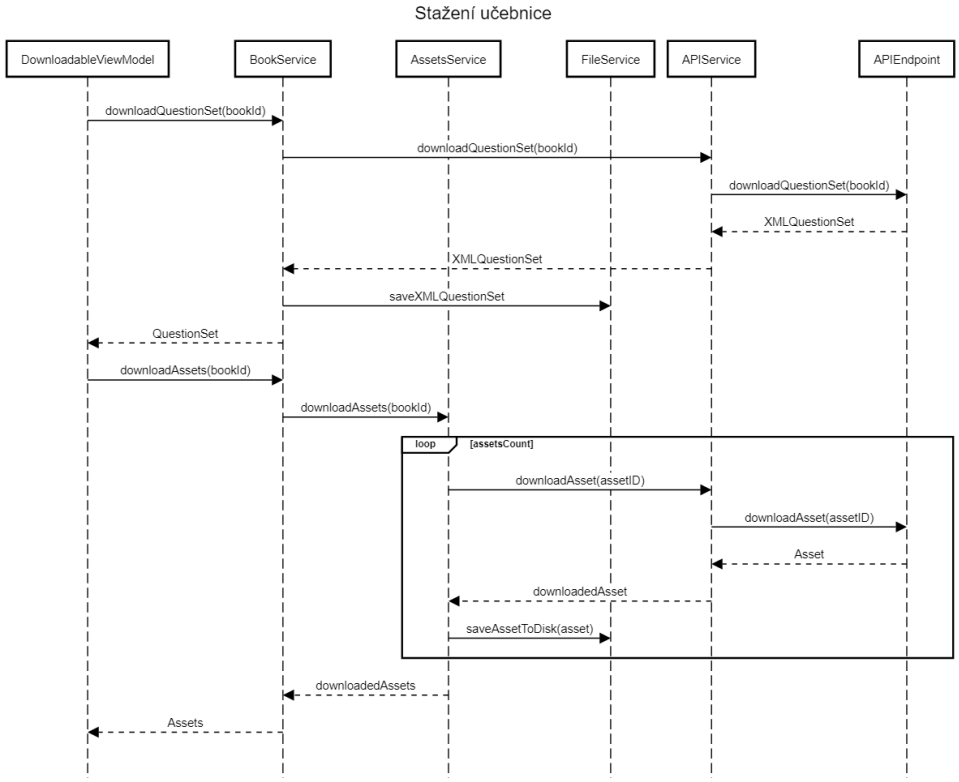
5.2 Práce s daty

5.2.1 Stahování učebnic

Proces stahování učebnic je zahájen po klepnutí na kartu představující učebnici na obrazovce „Učebnice ke stažení“. Tento proces je znázorněn na sekvenčním diagramu 5.3 Po klepnutí se na obrazovce vykreslí dialogové okno, které zobrazuje komponentu CircularProgressIndicator[7], v případě, že se provádí stahování a ukládání dat. Pokud během procesu stahování dojde k jakékoliv chybě, je o ní uživatel v dialogovém okně informován a stisknutím tlačítka „Rozumím“ dialogové okno uzavře. V opačném případě se uživateli zobrazí poznámka o tom, že je učebnice stažena a po 1 vteřině se dialogové okno samo uzavře.

[6] `https://docs.hivedb.dev/#/custom-objects/generate_adapter`

[7] `https://api.flutter.dev/flutter/material/CircularProgressIndicator-class.html`



Obrázek 5.3: Proces stahování učebnice

5.2.2 Možnosti spuštění úkolu

Další v řadě, co bylo třeba implementovat, byl způsob, jakým si uživatel může spustit úkol. Jak jsem již dříve zmiňoval, existují 2 způsoby spuštění a těmi jsou spuštění z mapy a spuštění z přehledu úkolů. Tento fakt vychází z vlastnosti učebnice, kde se nastaví o jaký druh učebnice se jedná, jestli s mapou, se seznamem a nebo s obojím.

V první řadě za účelem testování jsem implementoval obrazovku seznamu úkolů. Tato obrazovka se dělí na 3 záložky, Přehled, Zodpovězené a Nezodpovězené. Na záložce Přehled je možná vidět počet zodpovězených úkolů a úspěšnost zodpovězených úkolů. Zbylé dvě záložky, jak už názvy napovídají, obsahují seznamy zodpovězených a nezodpovězených úkolů.

Poté co jsem naimplementoval jednoduchou obrazovku seznamu úkolů, začal jsem implementovat obrazovku výběru úkolu na mapě. Jakožto API pro mapu a její implementaci jsem vybral MapBox, který taktéž existuje

jakožto balíček pro Flutter. Pomocí mapbox_gl[8] balíčku jsem implemetoval obrazovku, jenž vykresluje MapBox mapu a využívá MapBox API. Následně jsem vytvořil metodu pro přidání symbolů na mapu a každému symbolu jsem přidal možnost při klepnutí přesunout uživatele na obrazovku s úkolem. V poslední řadě jsem nastavil fixní pozici kamery a přiblížení nad mapou. Mezi jednotlivé symboly na mapě se nepřidává „intro“ úkol. Jediná možnost, jak tento úkol zobrazit, je ze seznamu úkolů, kde jsem jej ponechal.

Mapu lze používat i v offline režimu aplikace, ale je nutné aby uživatel spustil mapu se zapnutým připojením k internetové síti. Pokud tak učiní, mapa se nacachuje do mobilního zařízení po dobu běhu aplikace. Když uživatel aplikaci vypne, nacachovaná mapa je ze zařízení odstraněna a nedá se bez internetového připojení znovu zobrazit. Místo mapy se v tomto případě zobrazí pouze černá obrazovka.

5.2.3 Vykreslení úkolu

Vykreslování úkolů nebylo zcela jednoduchou záležitostí. Jelikož každý objekt Task může mít nekonečně mnoho QuestionSlides (ukázka kódu 8). QuestionSlide představuje obrazovku, která obsahuje otázky, jejich popis, obrázky, videa a audionahrávky, jak bylo k povšimnutí na hrubé struktuře XML (ukázka xml 2). Vzhledem k tomu, že při vytváření učebnice je možnost změnit pořadí jednotlivých prvků QuestionSlidu, bylo nutné nejdříve přečíst všechny prvky QuestionSlidu, poté z nich vytvořit korespondující widgety a následně je poskládat dle atributu order do správného pořadí a vykreslit je.

Struktura vykreslování úkolů je poměrně jednoduchá, pro každý QuestionSlide se vytvoří widget PageView[9] a daty, jenž QuestionSlide obsahuje, se PageView naplní. Pro vykreslování otázek bylo třeba vytvořit widgety pro každou specifickou otázku zvlášť. Nicméně, potřeboval jsem, aby každý tento widget obsahoval metodu, kterou by bylo možné spustit z jiné části aplikace (TaskViewModel) a zprostředkovávala by vyhodnocení dané odpovědi na otázku. Tohoto jsem docílil tím, že jsem vytvořil Mixin QuestionWidget, který obsahuje pouze prázdnou metodu evaluate. Implementace této metody se naplní až při vytváření specifického otázkového widgetu. Společně s nastavením metody evaluate se uloží reference na daný widget do TaskViewModelu pro vyhodnocení odpovědi na otázku.

[8] `https://pub.dev/packages/mapbox_gl`
[9] `https://api.flutter.dev/flutter/widgets/PageView-class.html`

5.2.4 Ukládání vypracovaných úkolů

Pro ukládání úkolů jsem postupoval následovně. Zprvu jsem vytvořil objekty, které by představovaly odpověď na jednotlivé druhy otázek. Tyto objekty by se následně uložily do Hive databáze. Také bylo potřeba vyřešit integritu dat v databázi. Proto se každá otázka ukládá do boxu „tasks“ pod ID učebnice. Každé ID učebnice pod sebou uchovává List záznamů datového typu TaskDBO.

Objekt TaskDBO (ukázka kódu 9) jsem vytvořil vyloženě jakožto objekt ukládaný do databáze a pomocí Hive anotací jsem vytvořil adaptér, který mi umožní jej do databáze uložit. Objekt TaskDBO nese několik informací, těmi jsou název úkolu, zda byl úkol již zodpovězen, zda byl zodpovězen úspěšně a v poslední řadě nese mapu záznamů typu Map<String, Map<Question,BaseAnswer». Klíč v podobě datového typu String představuje název QuestionSlidu, který je pro každý úkol unikátní. Hodnoty této mapy jsou opět mapami a to z toho důvodu, že se pro každý QuestionSlide ukládají otázky zvlášť. Klíčem této druhé mapy (Map<Question,BaseAnswer>) je otázka, pro kterou se odpověď ukládá.

Jak jsem již zmínil, pro každou otázku bylo zapotřebí vytvořit objekt, jenž by představoval odpověď na tuto otázku a bylo by možné ho uložit do Hive databáze. Abych ale mohl ukládat otázky do jedné mapy, bylo nutné vytvořit objekt BaseAnswer 10, ze kterého všechny objekty odpovědí dědí. Objekt obsahuje jedinou vlastnost a to bool isOkay, který značí, zda byla odpověď správně nebo špatně.

V poslední řadě bylo potřeba vytvořit proces, který by po kliknutí na tlačítko automaticky prošel všechny odpovědi na otázky, pro QuestionSlide na kterém se aktuálně uživatel nachází, vyhodnotil je a uložil do databáze. Jak jsem již zmínil, každý widget, který představuje otázku, obsahuje mixin QuestionWidget a proto jsem schopen uložit si všechny widgety obsahující QuestionWidget do stejného listu a následně ke všem přistoupit skrze TaskViewModel. Mixin QuestionWidget obsahuje metodu evaluate, která sama o sobě nemá žádnou implementaci. Každý specifický QuestionWidget při svém založení nastaví implementaci právě této metody evaluate, která je po klepnutí na tlačítko „Vyhodnotit“ provedena. Po vyhodnocení odpovědi je samotná odpověď uložena do databáze za využití TaskViewModelu.

5.3 Webová aplikace

Pro tvorbu webové aplikace jsem se rozhodl použít frameworku Spring Boot[10]. Toto rozhodnutí podporoval fakt, že jsem již s frameworkem Spring pracoval v rámci implementace jiných školních projektů, což mi značně usnadnilo práci.

5.3.1 Implementace

Prvním krokem implementace webové aplikace bylo vytvoření webové stránky obsahující jednoduchý formulář, kde uživatel vyplní potřebné údaje pro build aplikace. Mezi tyto údaje patří ID aplikace (potřebné k rozlišení aplikací v telefonu), název aplikace, API klíč tvůrce učebnice a nepovinný obrázek aplikace.

Následně, co jsem měl vytvořený formulář, jsem přistoupil ke zpracování dat odeslaných na server. V první řadě jsem se snažil jen o samotné spuštění buildu aplikace. Build aplikace se provádí následovně. Ve složce, kde je umístěn zdrojový kód aplikace na webovém serveru, jsou umístěny 2 batch scripty. Oba provádí ve směs stejné akce, ale jeden z nich nepřidává aplikaci logo, v případě, že uživatel ve formuláři žádný obrázek nepřidal. Posloupnost akcí prováděných scripty je následovná:

1. Vyčištění složky s předchozími buildy aplikace
2. Stažení potřebných balíčků pro build aplikace
3. Vygenerování loga aplikace díky balíčku flutter_launcher_icons[11] (pouze v případě, že uživatel zaslal i obrázek)
4. Změna názvu aplikace v zařízení díky balíčku flutter_launcher_name[12]
5. Build aplikace

Dalším krokem bylo vytvoření jednoduchého souboru, který se využívá ve zdrojovém kódu mobilní aplikace. Tohoto jsem docílil tak, že jsem do určité složky (lib/config/) vytvořil nový soubor s názvem web_config.dart. Tento

[10] `https://spring.io/projects/spring-boot`
[11] `https://pub.dev/packages/flutter_launcher_icons`
[12] `https://pub.dev/packages/flutter_launcher_name`

soubor obsahuje třídu WebConfig, která obsahuje 2 vlastnosti, APIKey a Name. Právě z tohoto souboru se v aplikaci využívá API klíč tvůrce učebnice, pro stahování učebnic. Jakým způsobem se soubor web_config.dart vytváří si můžeme všimnout na ukázce kódu 11

Poté, co jsem měl tvorbu souboru web_config.dart hotovou, začal jsem pracovat na změně názvu aplikace v mobilním zařízení. Jak jsem již psal, toto se děje díky balíčku flutter_launcher_name, který přejímá název aplikace ze souboru pubspec.yaml. Soubor pubspec.yaml je automaticky vytvářen při generování adresáře Flutter projektu. Abych byl schopen zapsat název aplikace přenesený z formuláře, bylo třeba přidat závislost pro jackson-dataformat-yaml do pom.xml Spring Boot projektu. Poté jsem díky této závislost načetl data z pubspec.yaml, přepsal název aplikace pro script flutter_launcher_name a soubor uložil. Metoda, která se právě o tuto funkčnost stará, je ukázána na ukázce kódu 12.

V předposlední řadě bylo třeba dopracovat proces vytváření loga aplikace. Jak jsem již psal výše, k tomu se využívá script z balíčku flutter_launcher_icons. Stejně jako flutter_launcher_name, i flutter_launcher_icons má své nastavení v souboru pubspec.yaml, nicméně v tomto případě jsem zde nic měnit nemusel. Jelikož jsem nechal pevně nastavený název souboru pro použití vygenerování ikony aplikace. Název tohoto souboru jsem ponechal jakožto icon.png. Tudíž všechny obrázky, které uživatel zašle skrze formulář na server, se ukládájí pod stejným názvem a to právě icon.png. Tyto soubory se zároveň ukládají do specifického adresáře assets/icon v projektu mobilní aplikace. Odtud je script flutter_launcher_icons nastaven k načítá obrázku.

V poslední řadě bylo nutné nastavit ID aplikace. Kdyby se toto ID nenastavilo a zůstalo stále stejné ID, tak by se pokaždé aplikace v mobilním zařízení přeinstalovala. Nicméně by si uchovala předchozí data, tedy stažené učebnice a předchozí práci s nimi. V celku by se pouze změnil API klíč tvůrce učebnic. Změny ID jsem docílil tím, že jsem k build scriptu gradle, umístěném v adresáři android/app/build.gradle, přidal pár řádků kódu, které načtou proměnnou appID ze souboru local.properties. Právě do tohoto souboru zapisuji ID aplikace z formuláře. Proměnnou appID v souboru local.properties zapisuji pomocí regulárního výrazu „appID=\w*“. Tento regulární výraz nalezne v souboru řádek obsahující právě proměnnou appID a následně část za znakem „=“ přepíši na hodnotu ID z dat formuláře a soubor uložím. Konkrétní kód tohoto řešení je k vidění na ukázce kódu 13.

5.3.2 Nasazení

Webovou aplikaci jsem psal s myšlenkou multiplatformnosti. Tím myslím spustitelnost webové aplikace více webovými servery. Proto jsem webovou aplikaci tvořil jakožto tzv. Java Servlet, který jsem archivoval do .WAR souborů. Díky tomuto jsem schopný webovou aplikaci nasadit na jakýkoliv Java EE server. Pro vlastní testování jsem použil webový server Apache Tomcat, který je jedním z Java EE webových serverů.

Aby webová aplikace fungovala správně, je nutné aby na zařízení, na němž běží tato webové aplikace, bylo nainstalováno AndroidSDK a FlutterSDK.

Kapitola 6

Testování mobilní aplikace

V této kapitole se budu zabývat testováním mobilní aplikace s pomocí 5 testerů. Abych zároveň neopomněl, byla testována i webová aplikace panem Ing. Ivem Malým, který zkoušel webovou aplikace pro soukromý API klíč. Mobilní aplikace, testované mnou vybranými testery, byly vytvořeny pomocí webové aplikace, o které jsem psal v sekci 5.3. Testovací scénáře byly vymyšleny tak, aby pokryly všechny funkční i nefunkční požadavky aplikace (popsány v kapitole Aplikační požadavky a případy užití 2).

6.1 Skupina testerů

Pro splnění zadání bakalářské práce bylo třeba najít 5 různých testerů ve věkové skupině 15-25 let, čili studenti středních a vysokých škol. Všichni testeři pracovali na mobilním zařízení s operačním systémem Android. Každý tester dostal testovací scénáře, dle kterých měl aplikaci otestovat a následně mi sdělit jestli testování probíhalo bez problémů či nikoliv. Také jsem testery poprosil o zpětnou vazbu jakýchkoliv postřehů během používání aplikace. Například, kde očekávali jiná gesta pro práci s aplikací.

6.2 Testovací scénáře

Jak jsem již zmínil v úvodu do kapitoly, testovací scénáře byly vymyšleny tak, aby pokryly všechny funkční i nefunkční požadavky aplikace. V testovacích scénářích jsem opomněl otestování každého druhu otázky a jejího vyhodnocení aplikací. Tester může otestovat všechny typy otázek pomocí testovací učebnice „Vsechny typy ukolu“. Tuto učebnici mají testeři k dispozici, jelikož je spojena s API klíčem, se kterým byla aplikace vytvářena. Předpokladem k testování aplikace bylo, že je tester obeznámen s názvy jednotlivých obrazovek mobilní aplikace.

Testovací scénáře byly následující:

1. Stažení učebnice B.1
2. Otevření učebnice a zobrazení seznamu úkolů B.2
3. Otevření úkolu B.3
4. Práce s úkolem B.4
5. Znovuotevření úkolu B.5
6. Smazání úkolu B.6
7. Smazání učebnice B.7
8. Vyhledávání učebnice B.8

6.3 Výsledek testování

Testování odhalilo chybu v aplikaci díky scénáři B.5. Chyba byla odhalena testerem 1. Ihned po odhalení jsem chybu opravil a poslal novou verzi aplikace k ověření správné funkčnosti. Zbytek testerů měl k dispozici již novou verzi aplikace, která měla chybu opravenou. Jejich testování potvrdilo, že chyba byla z aplikace odstraněna.

Testovací scénář č.	Tester 1	Tester 2	Tester 3	Tester 4	Tester 5
1	OK	OK	OK	OK	OK
2	OK	OK	OK	OK	OK
3	OK	OK	OK	OK	OK
4	OK	OK	OK	OK	OK
5	NOK	OK	OK	OK	OK
6	OK	OK	OK	OK	OK
7	OK	OK	OK	OK	OK
8	OK	OK	OK	OK	OK

Tabulka 6.1: Tabulka s výsledky testování

6.3.1 Poznámky testerů

Jedna ze zpětných vazeb testerů obsahovala zmínku, že aplikace zvýrazňuje kartu úkolu zeleně i přes fakt, že uživatel na žádnou otázku neodpověděl. V tomto případě se jedná o nedopracovanou funkcionalitu, kdy de facto všechny karty úkolů jsou zvýrazněny červenou barvou jen v případě, že některá z odpovědí byla vyhodnocena jako špatná. V ostatních případech, tzn. i když uživatel v úkolu nic neprovedl, svítí karta zeleně. V takovémto případě by měla karta svítit spíše žlutou nebo oranžovou barvou pro naznačení neúplnosti řešení úkolu.

Další zpětná vazba obsahovala poznámku, kdy se na obrazovce „Ke stažení“ vypíše přímo zpráva výjimky, která je vyhozena při nepřipojení k internetové síti. V tomto případě se také jedná o nedopracovanou funkcionalitu. Nicméně si myslím, že oprava této chyby by měla být poměrně jednoduchá.

V neposlední řadě mi jeden z testerů řekl, že pokud se zapne obrazovka s mapou bez připojení k internetu, je zobrazena místo mapy černá obrazovka. I v tomto případě se jedná o nedopracovanou funkcionalitu, kdy aplikace nekontroluje zda-li je uživatel připojen k internetové síti a snaží se pomocí balíčku mapbox_gl stáhnout a otevřít mapu.

Poslední poznámkou bylo přidání potvrzovacího dialogového okna před zahájením stahování učebnice.

Kapitola 7

Závěr

Již v prvotních okamžicích práce jsem si začal pomalu vytvářet jisté vize, jak by aplikace mohla vypadat a jakým způsobem by měla fungovat. Vzhledem k povaze iterativního procesu návrhu a analýze jsem postupně zrušil nebo poupravil některé své návrhy. Jelikož, jak tomu už bývá, prvotní vize neodpovídala výslednému produktu.

Během práce bylo třeba vytvořit prototyp, jenž by ukazoval funkčnost a návrh aplikace. Po zhodnocení vytvořeného prototypu mi bylo řečeno, že by bylo dobré najít design, kterým by se prototyp řídil. Vzhledem k tomu, že jsem věděl o existenci Material Designu, přečetl jsem si jeho dokumentaci a principy, kterými se řídí a zakomponoval jej do prototypu. Material design jsem vybral hlavně kvůli jeho vysoké rozšířenosti v aplikacích a principy, kterými se řídí, čímž se snaží uživateli usnadnit práci s aplikací. Při práci na tomto projektu jsem zároveň porovnal několik způsobů state managementu, které by bylo možné implementovat do aplikace. Pro udržování stavu aplikace jsem vybral kombinaci balíčků GetIt a Provider za použití architektonického vzoru MVVM. Balíček Provider svou funkčností nádherně zapadá do architektonického vzoru MVVM a je tedy dobré je využít pospolu.

V kapitole implementace jsem popsal prostředky a způsob jakým byla mobilní aplikace implementována. Zmínil jsem se o způsobu komunikace mezi jednotlivými objekty a funkčnosti klíčových objektů. V poslední sekci implementace jsem popsal i způsob implementace webové aplikace, která slouží k vytváření .apk instalačních souborů mobilní aplikace pro operační systém Android.

V poslední kapitole pojednávající o testování jsem popsal jakým způsobem byla aplikace otestována a zda-li dosahuje požadovaných požadavků. Testování odhalilo chybu, která byla po nalezení opravena. Zároveň uživatelské testování přineslo plno nových námětů ke zlepšení uživatelské přívětivosti aplikace.

7.1 Budoucí plány

V budoucnu bych se k vývoji aplikace chtěl znovu vrátit a implementovat všechny poznámky testerů, které vyplynuly z uživatelského testování. Mezi tyto poznámky patří:

- odstranění přímých popisů vyhozených výjimek na obrazovce „Ke stažení“
- lepší barevné zvýraznění odpovězených/rozpracovaných karet
- odstranění černé obrazovky mapy
- přidání dialogového potvrzovacího okna pro stahování učebnic

Zároveň bych chtěl vytvořit unit testy pro jednotlivé části aplikace. Unit testy bych chtěl pokrýt minimálně stejné funkcionality, které pokrylo uživatelské testování.

U webové aplikace bych chtěl vytvořit rozeznávání operačního systému a následné spuštění správného build scriptu. Aktuálně webová aplikace neobsahuje build scripty pro unixové operační systémy.

Literatura

[1] The model-view-viewmodel pattern. https://docs.microsoft.com/cs-cz/xamarin/xamarin-forms/enterprise-application-patterns/mvvm, 8 2017. Navštíveno: 22.12.2020.

[2] dash-overflow.net. Provider package. https://pub.dev/packages/provider. Navštíveno: 22.12.2020.

[3] Facebook. https://reactnative.dev/. Navštíveno: 22.12.2020.

[4] Flutter team. Hot reload. https://flutter.dev/docs/development/tools/hot-reload. Zobrazeno: 23.12.2020.

[5] Flutter team. Inheritedwidget. https://api.flutter.dev/flutter/widgets/InheritedWidget-class.html. Zobrazeno: 23.12.2020.

[6] Flutter team. Widgets. https://flutter.dev/docs/resources/architectural-overview. Navštíveno: 23.12.2020.

[7] Google. Components. https://material.io/design/introduction#components. Navštíveno: 21.12.2020.

[8] Google. Principles - material is the metaphor. https://material.io/design/introduction#principles. Navštíveno: 22.12.2020.

[9] Google. Understanding layout. https://material.io/design/layout/understanding-layout.html#. Navštíveno: 21.12.2020.

[10] M. Kubišová. Mobilní android aplikace pro řešení výukových uloh v systému eduard. https://dspace.cvut.cz/handle/10467/87748. Navštíveno: 20.12.2020.

[11] J. Mohite. Flutter: Mvvm architecture. https://medium.com/flutterworld/flutter-mvvm-architecture-f8bed2521958. Navštíveno: 2.5.2021.

[12] Romain Rastel. Dart: What are mixins? https://medium.com/flutter-community/dart-what-are-mixins-3a72344011f3. Zobrazeno: 14.05.2021.

[13] f. Thomas Burkhart. https://pub.dev/packages/get_it. Navštíveno: 22.12.2020.

[14] Vývojářský tým Bloc. Architecture. https://bloclibrary.dev/#/architecture. Navštíveno: 24.12.2020.

[15] Vývojářský tým Bloc. Why bloc? https://bloclibrary.dev/#/whybloc. Navštíveno: 24.12.2020.

Příloha A

Ukázky kódu

A.1 XML struktura

```
<images order="2" type="default">lavicka-litinova-c1.jpg</images>

<images order="2" type="blind" label="Popisek pro srovnani">
    <image>bg_1.png</image>
    <image>bg_2.png</image>
</images>
```

Kód 1: Ukázka různých tagů images

```
<questionset>
    <task>
        <location>
            <latitude/>
            <longitude/>
        </location>
        <questionslide>
            <questions>
                <question/>
                <description/>
                <images/>
                <audio/>
                <video/>
            </questions>
        </questionslide>
    </task>
</questionset>
```

Kód 2: Hrubá struktura XML

A.2 Mobilní aplikace

```
class QuestionSet {
    final int maxTries;
    final String layout;
    final List<Task> tasks;
    final String description;
    final String language;
    final bool isPublished;
    final String image;

    QuestionSet._(
        {this.description,
        this.language = 'cz',
        this.isPublished = false,
        this.image,
        this.maxTries,
        this.layout,
        this.tasks});

    factory QuestionSet.fromXML(XmlElement element) {
        return QuestionSet._(
            maxTries: int.tryParse(element.getAttribute('maxTries') ?? 1),
            layout: element.getAttribute('layout') ?? 'map',
            tasks:
                element.findElements('task').map((e) => Task.fromXML(e))
                .toList(),
            image: element.getAttribute('image') ?? '',
            isPublished: element.getAttribute('isPublished') == "true",
            language: element.getAttribute('language') ?? 'cz',
            description: element.getAttribute('description'));
    }
}
```

Kód 3: Příklad mapování XML QuestionSet na objekt

```
Widget build(BuildContext context) {
    if (viewModel == null) {
    viewModel = Provider.of<TaskViewModel>(context);
    }
//...
}
```

Kód 4: Propojení TaskView a TaskViewModelu

```
final GetIt locator = GetIt.instance;

Future setupLocator() async {
locator.registerLazySingleton(() => AppConfig());
locator.registerLazySingleton(() => APIService());
locator.registerLazySingleton(() => BookService());
locator.registerLazySingleton(() => FileService());
locator.registerLazySingleton(() => AssetsService());
locator.registerLazySingleton(() => TaskDB());
locator.registerLazySingleton(() => TaskDTOService());
}
```

Kód 5: Registrace objektů do GetIt

```
part 'option.g.dart';

@HiveType(typeId: 24)
class Option {
    @HiveField(0)
    String id;
    @HiveField(1)
    QuestionImage image;
    @HiveField(2)
    String innerText;
    //...
}
```

Kód 6: Anotace pro vytváření TypeAdapters pro Hive

```
Hive.init(AppConfig.documentsDirectory);
Hive.registerAdapter(BookStateAdapter());
Hive.registerAdapter(BookAdapter());
Hive.registerAdapter(AssetAdapter());
Hive.registerAdapter(AuthTokenAdapter());
Hive.registerAdapter(AccessTokenAdapter());
Hive.registerAdapter(TaskDBOAdapter());
Hive.registerAdapter(FilltextAnswerAdapter());
Hive.registerAdapter(NumberAnswerAdapter());
Hive.registerAdapter(ToggleAnswerAdapter());
Hive.registerAdapter(SortAnswerAdapter());
Hive.registerAdapter(SingleChoiceAnswerAdapter());
Hive.registerAdapter(MultiChoiceAnswerAdapter());
Hive.registerAdapter(IntervalAnswerAdapter());
Hive.registerAdapter(DragToMiddleAnswerAdapter());
Hive.registerAdapter(DragToLineAnswerAdapter());
Hive.registerAdapter(DescriptionAdapter());
Hive.registerAdapter(OptionAdapter());
Hive.registerAdapter(QuestionImageAdapter());
Hive.registerAdapter(QuestionAdapter());
Hive.registerAdapter(ResponseAdapter());
Hive.registerAdapter(StepAdapter());
Hive.registerAdapter(VariantAdapter());
Hive.registerAdapter(IntervalAdapter());

runApp(
MyApp(),
);
```

Kód 7: Registrování TypeAdapterů Hive

```
class QuestionSlide {
    String name;
    String title;
    String layout;
    Questions questions;

    QuestionSlide._({this.name, this.title, this.layout, this.questions});

    factory QuestionSlide.fromXML(XmlElement element) {
    return QuestionSlide._(
        name: element.getAttribute('name'),
        layout: element.getAttribute('layout'),
        title: element.getAttribute('title'),
        questions: Questions.fromXML(
        element.getElement('questions'),
        ),
    );
    }
}
```

Kód 8: Objekt QuestionSlide

```
    @HiveType(typeId: 5)
    class TaskDBO {
        @HiveField(0)
        final String name;
        @HiveField(1)
        final bool answered;
        @HiveField(2)
        final bool successful;
        @HiveField(3)
        final Map<String, Map<Question, BaseAnswer>> questionSlideAnswers;

        TaskDBO(this.name, this.answered, this.successful,
            {this.questionSlideAnswers});

        bool operator ==(o) => o is TaskDBO && o.name == name;
        int get hashCode => name.hashCode *
                            answered.hashCode *
                            successful.hashCode;
    }
```

Kód 9: Objekt TaskDBO

```dart
@HiveType(typeId: 8)
abstract class BaseAnswer {
    @HiveField(0)
    bool isOkay;

    BaseAnswer({this.isOkay});
}
```

Kód 10: Objekt BaseAnswer

A.3 Webová aplikace

```
public void createWebConfigFile(String apiKEY,String name)
    throws IOException{
    String webConfigFilePathString =
        context.getRealPath(basePath+"lib/config/web_config.dart");

    File confFile = new File(webConfigFilePathString);
    confFile.createNewFile();
    FileOutputStream fos = new FileOutputStream(webConfigFilePathString);

    StringBuilder sb = new StringBuilder();
    sb.append("class WebConfig {");
    sb.append("static String apiKey = "+"\""+apiKEY+"\""+";");
    sb.append("static String name = "+"\""+name+"\""+";");
    sb.append("}");
    fos.write(sb.toString().getBytes());

    fos.close();
}
```

Kód 11: Ukázka kódu pro tvorbu web_config.dart souboru

```java
public void setAppName(String name) throws IOException {
    String absolutePath = context.getRealPath(basePath);

    File pubspecFile = new File(absolutePath+"/pubspec.yaml");
    ObjectMapper objectMapper = new YAMLMapper();
    Map<String, Object> pubspec = objectMapper.readValue(pubspecFile,
        new TypeReference<Map<String, Object>>() { });

    Map<String,Object> launcher_name =
        (Map<String,Object>) pubspec.get("flutter_launcher_name");
    launcher_name.put("name",name);

    objectMapper.writeValue(pubspecFile, pubspec);
}
```

Kód 12: Ukázka kódu pro nastavení názvu aplikace v pubspec.yaml

```java
public void setAppID(String appID) throws IOException {
    String localPropertiesFile =
        context.getRealPath(basePath+"android/local.properties");

    File propFile = new File(localPropertiesFile);

    String content = Files.readString(propFile.toPath());
    String s = content.replaceAll("appID=\\w*","appID="+appID);
    Files.write(propFile.toPath(),s.getBytes(StandardCharsets.UTF_8));
}
```

Kód 13: Ukázka kódu pro nastavení ID aplikace v souboru local.properties

Příloha B

Testovací scénáře

B.1 Stažení učebnice

Prerekvizity:

- zařízení je připojeno k internetu

Postup:

1. Zapnout aplikaci a přejít na obrazovku „Ke stažení“
2. Klepnout na učebnici „Všechny typy úkolů“
3. Zobrazí se dialogové okno informující o stahování učebnice, na konci stahování se při úspěšném stažení zobrazí text „Staženo“ a dialog. okno se samo uzavře
4. Přejít na obrazovku „Stažené učebnice“
5. Učebnice je zobrazena v seznamu stažených učebnic

B.2 Otevření učebnice a zobrazení seznamu úkolů

Prerekvizity:

- zařízení je připojeno k internetu
- uživatel začíná na obrazovce „Stažené učebnice“

Postup:

1. Klepnout na učebnici „Všechny typy úkolů“ na obrazovce „Stažené učebnice“
2. Otevře se obrazovka popisu učebnice
3. Klepnout na tlačítko „Začít“
4. Uživatel je přenesen na obrazovku mapy s úkoly, v případě že učebnice neobsahuje mapu, je uživatel rovnou přenesen na obrazovku přehledu úkolů

B.3 Otevření úkolu

Prerekvizity:

- zařízení je připojeno k internetu
- uživatel má otevřenou učebnici, která je typu „s mapou“ nebo „s obojím“
- uživatel začíná na obrazovce mapy s úkoly
- učebnice má k úkolům přiřazené lokace na mapě

Postup:

1. Klepnout na symbol na mapě znázorňující úkol

2. Otevře se obrazovka úkolu
3. Projít všechny obrazovky úkolu a zkontrolovat, že se všechny vykreslí
4. Klepnout na šipku zpět v levém horní části obrazovky
5. Uživatel je přenesen zpět

B.4 Práce s úkolem

Prerekvizity:

- uživatel začíná na obrazovce „Přehled úkolů“
- učebnice je typu „se seznamem“ nebo „s obojím“

Postup:

1. Na obrazovce „Přehled úkolů“ přejít na záložku „Nezodpovězené“ a klepnout na kartu úkolu
2. Otevře se obrazovka s úkolem
3. Pomocí gesta swipe doleva se uživatel přesune na další stránku úkolu (pokud úkol obsahuje více obrazovek)
4. Odpovědět na otázku
5. Klepnout na tlačítko vyhodnotit a poté na šipku zpět v levém horním rohu
6. Přejít na záložku „Zodpovězené“
7. V seznamu odpovězených úkolů je zobrazena karta úkolu, jenž byla otevřena klepnutím na záložce „Nezodpovězené“

B.5 Znovuotevření úkolu

Prerekvizity:

- uživatel začíná na obrazovce „Přehled úkolů“
- učebnice je typu „se seznamem“ nebo „s obojím“

Postup:

1. Na obrazovce „Přehled úkolů“ přejít na záložku „Nezodpovězené“ a klepnout na kartu úkolu
2. Otevře se obrazovka s úkolem
3. Klepnout na šipku zpět v levé horní části obrazovky
4. Přejít na záložku „Zodpovězené“
5. V seznamu odpovězených úkolů je zobrazena karta úkolu, jenž byla otevřena klepnutím na záložce „Nezodpovězené“
6. Klepnout na kartu úkolu pro znovuotevření úkolu
7. Uživatel je přenesen na obrazovku úkolu a má možnost úkol vypracovat/dopracovat
8. Odpovědět alespoň na jednu otázku
9. Klepnout v levém horní rohu obrazovky na šipku zpět
10. Karta úkolu je na záložce „Zodpovězené“ vybarvena červeně, pokud uživatel odpověděl alespoň na jednu otázku špatně, jinak je karta vybarvena zeleně

B.6 Smazání úkolu

Prerekvizity:

- uživatel začíná na obrazovce „Přehled úkolů“
- učebnice je typu „se seznamem“
- existuje karta úkolu na záložce „Zodpovězené“

Postup:

1. Na obrazovce „Přehled úkolů“ přejít na kartu „Zodpovězené“
2. Dlouze podržet prst na jakékoliv kartě s úkolem
3. Zobrazí se dialogové okno s otázkou zda-li chceme smazat úkol
4. Klepnutí na „Ano“ smaže úkol a přenese jej na záložku „Nezodpovězené“ na konec
5. Otevřít smazaný úkol na záložce „Nezodpovězené“
6. Úkol nemá vyplňenou žádnou odpověď

B.7 Smazání učebnice

Prerekvizity:

- zařízení je připojeno k internetu
- uživatel začíná na obrazovce „Stažené učebnice“

Postup:

1. Přejít na obrazovku „Stažené učebnice“
2. Dlouze podržet prst na jakékoliv kartě s učebnicí
3. Zobrazí se dialogové okno s otázkou zda-li chceme smazat danou učebnici
4. Klepnutí na „Ano“ smaže učebnici ze zařízení a učebnice se odstraní z listu stažených učebnic
5. Přejít na obrazovku „Ke stažení“
6. Smazaná učebnice je vidět ve výpisu stažitelných učebnic

B.8 Vyhledávání učebnice

Prerekvizity:

- v zařízení je staženo několik učebnic
- uživatel začíná na obrazovce „Stažené učebnice“

Postup:

1. Do vyhledávacího pole na vrchu aplikace napsat část názvu učebnice
2. V seznamu stažených učebnic jsou k dispozici pouze učebnice ubsahující v názvu text z vyhledávacího pole

pozn.: Test lze stejným způsobem replikovat i na obrazovce „Ke stažení“, ale uživatel musí být připojen k internetové síti.

Příloha C

Návod k instalaci

C.1 Potřebné nástroje

Webový server prozatím funguje jen na operačním systému Windows, jelikož webová aplikace neobsahuje scripty pro build na unixových systémech.

Pro instalaci webové aplikace jsou potřeba následující nástroje:

- Apache Maven
- Java EE server (např. Apace Tomcat)

Pro vytvoření instalačního balíčku .apk pomocí webové aplikace jsou potřeba následující SDK:

- FlutterSDK (návod k instalaci[1])
- AndroidSDK (popsáno v části Android setup v návodu instalace FlutterSDK)

[1] `https://flutter.dev/docs/get-started/install`

C.2 Instalace webové aplikace

1. spustit příkaz „mvn clean package“ v adresáři „`./web_app/bakalarka`“ pomocí příkazového řádku
2. přejít do adresáře „`./web_app/bakalarka/target`“
3. překopírovat „bakalarkaapi.war“ na vámi zvolený webový server (v případě Apache Tomcat do adresáře „webapps“)
4. spustit webový server
5. přejít na URL serveru (s největší pravděpodobností `http://localhost:8080/bakalarkaapi`, místo bakalarkaapi zadejte název .war souboru, v případě, že jste jej měnili)

C.3 Instalace mobilní aplikace

Mobilní aplikace se vytváří pomocí webové aplikace, tudíž je třeba mít zprovozněnou webovou aplikaci. K instalaci mobilní aplikace pomocí .apk souboru je třeba, aby mobilní zařízení mělo povolené instalování z neznámých zdrojů.

1. otevřít webovou aplikaci v internetovém prohlížeči (`http://localhost:8080/bakalarkaapi`)
2. vyplnit formulář, odeslat a počkat než se na webové aplikaci objeví nápis „Data nahrána“ a tlačítko „Stáhnout apk“
3. kliknout na tlačítko „Stáhnout apk“
4. přesunout stažený apk soubor do mobilního zařízení s operačním systémem Android
5. nainstalovat v mobilním zařízení apk soubor

C.4 Kompilace mobilní aplikace bez použití webové aplikace

Je nutno mít nainstalované FlutterSDK a AndroidSDK. Pokud nechcete využít webovou aplikaci k vytvoření instalačního souboru mobilní aplikace, stačí zadat následující příkazy do příkazového řádku v adresáři /mobile_app.

V tomto pořadí:

1. flutter pub get
2. flutter build apk

Po vykonání těchto příkazů se vytvoří adresář „build“. Vytvořený .apk soubor je možné najít v adresáři /build/app/outputs/apk/release pod názvem app-release.apk.

Příloha D

Obsah elektronické přílohy

- `source_codes` ..... adresář se zdrojovými kódy
  - `mobile_app` ..... zdrojový kód mobilní aplikace
  - `web_app` ..... zdrojový kód webové aplikace
  - `README.md` ..... instalační návod
- `bakalarska_prace.pdf` ..... PDF dokument bakalářské práce