ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE
Fakulta elektrotechnická
Katedra počítačů

Mobilní aplikace poskytující informace o obcích

Mobile application providing information about municipalities

Diplomová práce

Studijní program: Otevřená informatika
Studijní obor: Softwarové inženýrství

Vedoucí práce: Ing. Božena Mannová, PhD.

Bc. Václav Štemberg
Praha 2020

ZADÁNÍ DIPLOMOVÉ PRÁCE

I. OSOBNÍ A STUDIJNÍ ÚDAJE

Příjmení: **Štemberg** Jméno: **Václav** Osobní číslo: **421051**

Fakulta/ústav: **Fakulta elektrotechnická**

Zadávající katedra/ústav: **Katedra počítačů**

Studijní program: **Otevřená informatika**

Specializace: **Softwarové inženýrství**

II. ÚDAJE K DIPLOMOVÉ PRÁCI

Název diplomové práce:

Mobilní aplikace poskytující informace o obcích

Název diplomové práce anglicky:

Mobile application providing information about municipalities

Pokyny pro vypracování:

Navrhněte a vytvořte mobilní aplikaci, která bude poskytovat důležité a aktuální informace z obcí. Potřebná data budou uložena na serveru, který je bude automaticky generovat z webových stránek obcí. Uživatel si bude moct nastavit preferované obce, které chce sledovat. Systém se bude skládat ze serveru s databází, administrátorského webu a samotné mobilní aplikace.
Pokyny pro zpracování:
1)Vyhledejte a analyzujte existující systémy poskytující podobnou funkcionalitu.
2) Na základě vyhodnocení těchto poznatků, specifikujte požadavky na funkcionalitu systému.
3) Seznamte se s technologiemi potřebnými pro vytvoření mobilní aplikace komunikující se serverem prostřednictvím síťového spojení po internetu splňující běžné standardy.
4) Navrhněte vhodné grafické rozhraní aplikace.
5) Navrhněte a implementujte základní části systému.
6) Otestujte aplikaci včetně uživatelských testů.
7) Výsledky vyhodnoťte.
8) Uveďte možná budoucí vylepšení aplikace.

Seznam doporučené literatury:

[1] Houssein Djirdeh, Anthony Accomazzo, Sophia Shoemaker: Fullstack React Native: Create beautiful mobile apps with JavaScript and React Native Kindle Edition. Independently Published, 2019. 9781728995557
[2] Bonnie Eisenman: Learning React Native: Building Native Mobile Apps with JavaScript 2nd Edition. O'Reilly Media, Inc., 2015. 9781491929070
[3] Biehl, Matthias: RESTful API Design (API-University Series) (Volume 3). CreateSpace Independent Publishing Platform; 1 edition, 2016. 9781514735169
[4] Ted Hunter, Steven Porter: Google Cloud Platform for Developers: Build highly scalable cloud solutions with the power of Google Cloud Platform. Packt Publishing, 2018. 9781788837675

Jméno a pracoviště vedoucí(ho) diplomové práce:

Ing. Božena Mannová, Ph.D., kabinet výuky informatiky FEL

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

Datum zadání diplomové práce: **10.02.2020** Termín odevzdání diplomové práce: **22.05.2020**

Platnost zadání diplomové práce: **30.09.2021**

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

III. PŘEVZETÍ ZADÁNÍ

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

. | Datum převzetí zadání

Podpis studenta

Prohlášení

„Prohlašuji, že jsem předloženou práci vypracoval samostatně a že jsem uvedl veškeré použité informační zdroje v souladu s Metodickým pokynem o dodržování etických principů při přípravě vysokoškolských závěrečných prací.“

Praha, 11. května 2020

……

podpis autora práce

Poděkování

Zde bych rád vyjádřil poděkování vedoucí mé diplomové práce paní Ing. Boženě Mannové, PhD. za pomoc, ochotu a povzbuzování během vedení této práce.

Dále bych rád poděkoval své rodině a blízkým za podporu během studia. V neposlední řadě bych rád vyjádřil poděkování všem účastníkům mého závěrečného testování mobilní aplikace.

Abstrakt

Diplomová práce se zaměřuje na návrh a vývoj obecního informačního systému. Ten se skládá ze serveru, mobilní aplikace a webové aplikace. Server generuje aktuality obcí na základě jejich webových stránek a ukládá je do databáze. Mobilní aplikace následně umožňuje tyto aktuality zobrazit. Součástí systému je také návrh webové aplikace, která slouží k administraci aktualit ze strany obcí. Práce popisuje zpracování současných řešení, návrh systému a také implementaci určitých komponent. Poslední částí práce je testování mobilní aplikace. Obecní informační systém usnadňuje přístup občanů k důležitým událostem a informacím z lokalit, o které se zajímají.

Klíčová slova: mobilní aplikace, server, Google Cloud, obce, React Native

Abstract

This thesis focuses on the design and development of municipalities information system. It consists of server, mobile application and web application. Server generates actualities of municipalities from their websites and saves them into the database. Mobile application then allows end user to view these actualities. The system also includes a design of web application, which is used for actualities administration by municipalities. The thesis describes analysis of existing solutions, system design and implementation of certain components. The last part of the work is testing of mobile application. The municipalities information system facilitates people's access to important events and information from their localities they are interested in.

Keywords: mobile application, server, Google Cloud, municipalities, React Native

Obsah

Terminologie

API	Application Programming Interface - aplikační rozhraní, na které přichází požadavky
Cloud	Jde o služby, aplikace či programy poskytované na internetu, kterých člověk může využít vzdáleně pomocí určitého rozhraní (často webový prohlížeč)
Endpoint	Koncový bod, používaný u adres URL (na serveru se používá při filtraci příchozích požadavků)
Framework	Také knihovna, usnadňující práci při programování (většinou se každý framework zaměřuje na konkrétní problematiku)
GCP	Google Cloud Platform - cloudová služba společnosti Google
GUI	Graphical User Interface - uživatelské grafické rozhraní
Handler	Funkce reagující na určitou událost (zpracovává požadavky uživatelů na serveru)
Header	Hlavička, používající se u HTTP požadavků a také JWT tokenů
HTML	Hyper Text Markup Language - značkovací jazyk určený pro tvorbu webových stránek
HTTP	Hyper Text Transfer Protocol - internetový protokol určený pro komunikaci
HTTPS	HTTP Secure - zabezpečený internetový protokol HTTP
JSON	JavaScript Object Notation - formát pro reprezentaci dat
REST	Representational State Transfer - architektura rozhraní API
URL	Uniform Resource Locator - specifikuje adresu zdroje v internetu

Obrázky

Tabulky

1 Úvod

Tato práce se zaměřuje na vývoj obecního informačního systému. Jeho součástí je univerzální mobilní aplikace, která poskytuje aktuální informace z co největšího možného počtu obcí, jejichž obsah je generován automaticky. Podíl na vzniku tohoto systému má do určité míry cestování. Lidé se tak ocitají na různých místech, kde převážně tráví svůj čas. Příkladem je bydlení na trvalé, nebo přechodné adrese, dojíždění do práce, či pobyt na víkendové chatě. S tím je často spojena malá informovanost o dění v daných lokalitách. Z pravidla je to tím, že lidé nemají na sledování novinek čas, zvláště pokud jde o více lokalit. Otevřít si webové stránky obce a sledovat, co je nového dnes nedělá každý, zvlášť mladší lidé. Člověk webové stránky otevře většinou až když něco konkrétního hledá. A právě na tuto problematiku informovanosti občanů se zaměřuje tato diplomová práce. Cílem je tedy zlepšit povědomí obyvatel o aktuálních informacích a událostech z lokalit, které je zajímají. Tito obyvatelé jsou současně cílová skupina této práce. Na základě analýzy odpovědí z dotazníku zveřejněného mému širšímu okolí, bylo zřejmé, že vyvíjená aplikace má smysl a že by o ni byl zájem.

V současnosti je tento problém řešen individuálními mobilními aplikacemi navrženými speciálně pro dané obce. Dále existují aplikace, které se snaží poskytovat důležité informace z více obcí najednou. Často jde ale o ne zcela zdařilé projekty. Nejčastěji je problém v tom, že aplikace poskytuje informace pouze o několika obcích (např. jen u těch, které si tuto službu zaplatí). Pokud pak nabízí informace např. o jedné ze tří požadovaných lokalit, ztrácí pro uživatele smysl. Další problém bývá v kompatibilitě s různými typy zařízení či v plynulosti ovládání.

Můj systém tuto problematiku řeší elegantním způsobem. A to tak, že bez nutnosti zásahu obcí jsou aktuální informace poskytnuty občanům, kteří si mobilní aplikaci nainstalují. Základní částí systému je server, který pravidelně generuje aktuality obcí z jejich webových stránek a ukládá je do databáze. Aktuality jsou následně poskytovány občanům prostřednictvím mobilní aplikace. Součástí práce je i návrh webové aplikace, která by sloužila jako možnost pro manuální správu obsahu ze strany obcí. Výhodou mobilní aplikace je možnost založení účtu, se kterým je spojeno veškeré uživatelské nastavení.

Diplomová práce je rozdělena do jednotlivých částí. První se zabývá podrobným rozborem současných řešení a jejich porovnáním. Následuje část věnována návrhu systému. Dále si popíšeme použité technologie, které představují hlavně služby Google Cloud Platform[1] na straně serveru a React Native[2] na straně mobilní aplikace. Závěr práce je tvořen popisem implementace serveru a mobilní aplikace včetně jejího závěrečného testování na uživatelích.

1.1 Požadavky

Zde jsou popsány základní požadavky na diplomovou práci. Konkrétně se jedná o požadavky na vyvíjený obecní informační systém.

- Funkční mobilní aplikace pro zařízení s platformou Android a iOS
- Funkční server, který bude generovat aktuality z webových stránek obcí a následně je ukládat do databáze
- Možnost vytvoření uživatelského účtu v mobilní aplikaci pro ukládání uživatelských dat
- Přehledné grafické rozhraní a snadné ovládání mobilní aplikace
- Zobrazování aktualit vybraných obcí v mobilní aplikaci
- Zabezpečení systému a zachování jeho rozšiřitelnosti

[1] Cloudové služby společnosti Google (https://cloud.google.com/)
[2] Technologie pro tvorbu multiplatformních aplikací (https://reactnative.dev/)

2 Současný stav

Tato kapitola se zaměřuje na současný stav a existující řešení dané problematiky. Problém je v současnosti řešen různě: obecním rozhlasem, vývěskami, webovými stránkami, nebo aplikacemi přímo pro konkrétní obce. Také existují aplikace, které se snaží shromažďovat informace o více obcích najednou. Bohužel to platí jen u těch obcí, které si tuto službu zakoupí. To spolu s faktem, že většina těchto služeb přináší obcím další administrativní zátěž, má za následek, že zdaleka nejsou pokryty všechny obce. Cílovému uživateli tak mohou některé lokality chybět. Toto je zásadní nedostatek, který se diplomová práce snaží řešit.

2.1 Existující systémy

Následující část pojednává o současných řešeních a existujících aplikacích, které se snaží informace z obcí shromažďovat. Sledujeme u nich komplexnost řešení, návrh uživatelského rozhraní, plynulost používání, kompatibilitu s mobilními operačními systémy (OS) apod. Zkoumané programy jsou běžně poskytovány a byly staženy pomocí distribučních služeb Google Play [1] a App Store [2]. Google Play je webová distribuční služba nabízející aplikace pro zařízení s operačním systémem Android. Na druhé straně App Store je určen pro poskytování aplikací na zařízení s OS iOS[3]. Pro co nejrelevantnější výsledky bylo zkoušení (testování) existujících aplikací prováděno na následujících zařízeních:

- Samsung Galaxy S5 Neo (Android 6.0.1)
- Samsung Galaxy Note 10.1 (Android 4.4.2)
- Asus Memo Pad (Android 4.2.2)
- Huawei MediaPad T1 8.0 Pro (Android 4.4.4)
- Honor 10 (Android 9.0)
- iPhone 7 Plus (iOS 13.2.2)

Zkoušení funkcionalit následujících aplikací bylo prováděno od října do listopadu 2019. Zároveň bylo zaznamenáno hodnocení uživatelů k patřičnému datu dle webových služeb, které aplikace oficiálně poskytují (Google Play, App Store).

2.1.1 CityApp

Jedná se o platformu společnosti BSSHOP [3] [4], která nabízí služby vývoje aplikací pro obce a města na míru. Jde však o vývoj aplikací pro každou obec zvlášť za účelem lepší personalizace ze strany obcí. Tím však nesplňuje požadavek, aby informovanost obyvatel zajišťovala jediná aplikace. Také jde o dražší záležitost, protože vývoj aplikace není v dnešní době příliš levný. Natož aby byla vyvíjena pro každou obec a město zvlášť, když ve velké míře budou mít všechny podobné funkce.

2.1.2 Česká obec

První aplikací poskytující informace z více obcí najednou je Česká obec [5] [6] [7]. Bohužel většina obcí nemá na této aplikaci profil. Krom malého množství podporovaných obcí, disponuje malou škálou třiditelnosti příspěvků (Oznámení, Událost, Úřední deska). Také má horší design a její ovládání není intuitivní. Například pro změnu obce neexistuje jednoduchý přepínač, ale je potřeba jít do přehledu obcí, který se nachází pod ikonou plus (což je velmi nepřirozené). Aplikace

[3] Operační systém společnosti Apple

nemá žádné hlavní menu, které by vyřešilo značnou část těchto nedostatků. Také nelze vyhledávat v příspěvcích obce. Co se týče základních informací o obci, jsou zde pouze kontakty např. na starostu, místostarostu nebo účetní. To jsou informace, které člověk prakticky nepoužije. V sekci Kontakty je také ukryto zasílání zpráv obci, což by zde hledal málokdo. Až na občasnou nestabilitu aplikace, funguje dobře jak na operačním systému iOS tak Android. Bohužel má nedostatky v malé funkcionalitě, designu a hlavně podporou malého množství obcí.

2.1.3 Mobilní rozhlas / Zlepšeme Česko

Mobilní rozhlas [8] [9] [10] je nejspíše nejrozsáhlejší aplikací ze všech testovaných. Jde o systém vyvíjený společností Neogenia [11]. Obecně disponuje velkou škálou funkcionalit. Při instalaci vyžaduje zbytečně velké množství oprávnění. To může některé uživatele odradit. Další překážkou pro uživatele může být nutná registrace, bez které se aplikace nedá použít. Registrace je možná přes Facebook nebo telefonní číslo. Vizuálně vypadá aplikace velice dobře avšak rozvržení jejích komponent je poněkud různorodé a nepřehledné. Třeba nedisponuje žádným hlavním menu, na které je většina uživatelů zvyklá (např. do správy sledovaných míst, je zapotřebí se složitěji proklikat).

Značným problémem je nestabilita aplikace na určitých zařízení s OS Android verze 6.0.1 a nižší. Vzhledem k tomu, že tyto verze používá stále přes 40 % všech uživatelů Android [12] (údaje k dni 23.9.2019), jde o nezanedbatelnou vlastnost. V některých případech je aplikace pro tyto uživatele téměř nepoužitelná. Konkrétní problémy s nestabilitou jsou popsány v následujícím odstavci.

V aplikaci dochází k extrémnímu „sekání“ při ukládání nových obcí nebo při jejich mazání (na displeji je bílé okno s načítáním, které je prakticky zamrzlé a nic se neděje). Ve velkém počtu takovýchto zaseknutí dojde k pádu aplikace. Po její znovuspuštění dochází opět k sekání a velmi pomalému načítání dat (občas následuje opět pád aplikace). Během těchto fází dokonce na chvíli zcela zčernala obrazovka a aplikace vůbec nereagovala. Občas se také z nepochopitelných důvodů zobrazí hláška “Něco se pokazilo”. Příčinou např. bylo, že nebyla zvolena žádná obec k odebírání příspěvků. V takovém případě je ale na místě zobrazit nějakou smysluplnější hlášku. Dále jsou v aplikaci různé chyby, kdy např. po odebrání obce ze seznamu oblíbených se obec stále zobrazuje v daném seznamu. Pro jeho aktualizaci je potřeba odejít a opět zobrazit seznam obcí.

Vzhledem k tomu, že jde technicky o ne příliš složitou aplikaci, dochází k jejímu častému sekání a neplynulosti jejího ovládání. Použití na zařízení iPhone a novějších verzích Android je vcelku bezproblémové. Avšak po delším používání na iPhone došlo k podobnému zaseknutí jako na Androidu. Konkrétně proces načítání preferovaných obcí trval tak dlouho, až bylo nutné operaci přerušit ručně. Stále to ale nebylo v takové míře jako popisováno dříve. Co se týče grafického rozhraní, přehledně působí např. zobrazení základních údajů o obci. Celkově ale aplikace postrádá určitou základní strukturu v uspořádání.

Výhodou aplikace je široký záběr, co se týče služeb, které obcím nabízí. Jednou z nich je zadávání podnětů od občanů (např. černé skládky, ztracená zvířata a ztracené věci). V kombinaci se špatnou kompatibilitou s některými Android zařízeními jde však o nepodstatnou vlastnost. Nejzásadnějším problémem je ale fakt, že ani tato aplikace nepodchycuje všechny obce.

2.1.4 MojeObec

Aplikace MojeObec [13] [14] je vyvinuta společností Macron Software [15] a je podporována službou Mobisys [16]. Pro její použití není vyžadováno přihlášení. Obecně funguje dobře jak na zařízení iPhone, tak i starších zařízení Android. Nelze však sledovat příspěvky z více

obcí najednou. Vždy se načtou pouze ty příspěvky obce, u které máme nastavený odběr. MojeObec má málo funkcí a příspěvky nejde třídit, čímž budou uživatelé dostávat informace, o které nestojí. Na to, jak je aplikace jednoduchá, zabírá příliš mnoho místa (okolo 100 MB na iPhone, 64 MB na Android - údaje platné k 12. 11. 2019). Výhodou je poskytnutí funkce na hlášení nepořádku v obci. Největším problémem však zůstává opět velký nedostatek podporovaných obcí.

2.1.5 My Village

Jak již název napovídá i tato aplikace [17] má poskytovat informace z oblíbených lokalit. Bohužel disponuje pouze několika cizími městy. Zásadním problémem je nedokončenost a podpora pouze Android zařízení. Dále vyžaduje zbytečná oprávnění v telefonu (jako např. Telefon, Fotky, Identita, ID zařízení, Historie zařízení). Při zkoušení zobrazení mapy obcí aplikace spadla a už se ji nepodařilo spustit. Na jiném Android zařízení se ani nespustila, přestože šla nainstalovat. Jde tedy o nepoužitelný software.

2.1.6 V OBRAZE

Autorem této aplikace [18] [19] je společnost Online-Team s.r.o. [20]. Výhodou programu je funkčnost bez jakéhokoliv přihlašování, které vyžadují některé výše zmíněné aplikace. Na druhou stranu nevýhodou je, že aplikace tyto účty vůbec nepodporuje. Ty by mohly lidem ušetřit práci, pokud vlastní více zařízení. Nelze tedy založit účet, nebo zadat telefonní číslo s kterým by byly preferované lokality svázány.

Příspěvky aplikace třídí do 4 kategorií (Zprávy, Akce, Úřední deska, Galerie). Vyšší personalizaci zpráv bohužel neposkytuje. Tyto příspěvky však lze volit zvlášť pro jednotlivé obce. Kladně hodnotím design, který v porovnání s ostatními aplikacemi působí přehledněji. Jako u jedné z mála testovaných aplikací bylo funkční upozorňování pomocí notifikací. Dobře funguje např. vyhledávání v příspěvcích, nebo zobrazení akcí dle data. Jako jediná z testovaných aplikací má klasické boční menu, na které je většina lidí zvyklá. Bohužel i tato aplikace však podporuje malé množství obcí.

2.2 Porovnání aplikací

V následující tabulce jsou porovnány jednotlivé aplikace zmíněné výše. Konkrétně je porovnáváno uživatelské hodnocení na distribučních službách Google Play a App Store. Tato hodnocení jsou velmi dobrým měřítkem toho, jak je daná aplikace kvalitní. Další porovnávanou vlastností jsou minimální požadavky na OS daného zařízení (Android nebo iOS). Pomocí Android aplikace ManifestViewer [21] jsou zjištěny podporované verze API level pro Android zařízení.

Tabulka 1 - Porovnání aplikací

Aplikace (datum hodnocení)	Hodnocení uživatelů		Minimální požadavky na OS	
	Google Play	App Store	Android min. API level	iOS
Česká obec (29.10.2019)	3,3 / 5	3,1 / 5	16	9.0
Mobilní rozhlas (29.10.2019)	3,6 / 5	4,0 / 5	16	9.3
Moje obec (29.10.2019)	3,4 / 5	4,1 / 5	19	10
My Village (12.11.2019)	3,0 / 5	-	19	-
V obraze (29.10.2019)	3,5 / 5	3,5 / 5	19	9.0

Z tabulky lze vyčíst, že např. všechna uživatelská hodnocení na službě Google Play ale i na App Store jsou velice nízká.

Na základě porovnávaných aplikací lze vidět, že již několik systémů se snaží shromažďovat informace z více lokalit na jedno místo a zlepšit tak informovanost občanů. Bohužel všechny aplikace podporují pouze několik obcí v ČR. To je největší problém všech testovaných aplikací. Tím, že disponují např. pouze jednou ze tří lokalit, kde se uživatel nachází, stanou se pro uživatele bezpředmětnými. Kromě tohoto nejzásadnějšího nedostatku nemá většina aplikací ideálně navržené uživatelské rozhraní. To je pro dosažení co nejvyšší použitelnosti občany klíčovou vlastností. Vzhledem k tomu, že má být aplikace pro všechny občany, žádná z testovaných nepodporuje např. režim pro ty, kteří hůře vidí, nebo mají špatnou motoriku. Další otázkou je cena. Systémy si účtují od obcí měsíční poplatky za provoz a využívání jejich služeb. Zda je částka úměrná nabízeným službám či ne a zda se to obcím vyplatí, je otázkou na každou obec zvlášť. Existují však i menší obce, pro které i menší částka hraje roli. O obsah v systému je potřeba se starat a pravidelně jej aktualizovat. To přináší další administrativní zátěž. Vyvíjená aplikace se snaží zmiňované nedostatky eliminovat.

3 Návrh systému

Tato kapitola se zaměřuje na návrh systému. Návrh je klíčovou částí při vývoji. V případě špatného navržení systému se mohou tyto nedostatky projevit během implementace nebo později při rozšiřování systému. Může to zapříčinit nepoužitelnost aplikace nebo neschopnost reagovat na budoucí požadavky a rozšíření. Tato skutečnost je častým problémem a systémy tak mohou přestat být konkurence schopnými vůči ostatním systémům, které na nové požadavky dokáží reagovat.

3.1 Požadavky

Veškeré systémové požadavky, které od systému požadujeme lze rozdělit do dvou základních kategorií. Těmito kategoriemi jsou funkční požadavky a nefunkční požadavky. V následujících podkapitolách si je popíšeme.

3.1.1 Funkční

Funkční požadavky popisují chování systému. Jde o služby a akce, které po systému požadujeme, aby splňoval. Vzhledem k tomu, že je systém složen ze tří částí, rozdělíme požadavky podle toho, do kterých částí systému spadají. V našem případě máme tyto části: server, mobilní aplikace, webová aplikace.

A. Server

- **Správa API**
  Server bude spravovat přicházející HTTP požadavky a vracet na ně patřičné odpovědi.
- **Správa uživatelských účtů**
  Server bude umožňovat správu uživatelských účtů a s tím spojené funkcionality jako registrace, přihlášení, autorizace, obnova hesla či změna hesla.
- **Složitost hesel**
  Při registraci bude server vyžadovat heslo s určitým stupněm složitosti, tak aby nebylo snadno prolomitelné.
- **Ověření emailové adresy**
  Server bude při registraci vyžadovat ověření emailové adresy.
- **Zasílání emailů**
  Server bude umožňovat zasílání emailů uživatelům.
- **Správa dat**
  Server bude umožňovat práci s daty v databázi jako vkládání, editaci, čtení a mazání.
- **Generování dat**
  Server bude generovat data (aktuality) z daných obcí na základě jejich webových stránek. V rámci aktualit se budou zjišťovat (pokud budou k dispozici) následující informace: nadpis, obsah, datum vytvoření, autor, datum konání, přílohy vztahující se k události a webové stránky události.
- **Ukládání generovaných dat**
  Server bude generovaná data z obcí ukládat do databáze.

B. Mobilní aplikace

- **Registrace uživatele**
  Mobilní aplikace bude umožňovat koncovému uživateli založit účet. S tímto účtem bude spojeno nastavení daného uživatele.
- **Přihlášení uživatele**
  Mobilní aplikace bude umožňovat přihlášení daného uživatele.
- **Odhlášení uživatele**
  Mobilní aplikace bude umožňovat odhlášení již přihlášeného uživatele.
- **Obnova hesla**
  Mobilní aplikace bude umožňovat resetovat heslo pro případ, když ho uživatel zapomene.
- **Změna hesla**
  Mobilní aplikace bude umožňovat změnu uživatelského hesla.
- **Ukládání uživatelských dat**
  Mobilní aplikace bude ukládat data uživatele, aby při znovuspuštění aplikace byly opět k dispozici. Během startu aplikace bude tedy kontrolováno, zda jsou uložena nějaká uživatelská data. Pokud ano, pak budou nastavena v aplikaci.
- **Kontrola internetového připojení**
  Mobilní aplikace upozorní uživatele, pokud bude vyžadováno připojení k internetu, ale v danou chvíli nebude dostupné.
- **Postranní menu**
  Mobilní aplikace bude disponovat postranním menu. To bude vysouvatelné prstem nebo tlačítkem v levém horním rohu displeje.
- **Vyhledávání obcí**
  Mobilní aplikace bude umožňovat vyhledávání obcí podle země a jména obce. Vyhledávání bude fungovat nezávisle na velikosti písmen či interpunkci. Ve výsledcích vyhledávání země se bude nacházet jakákoliv shoda s oficiálním názvem, běžným názvem či kódem, který má každá země unikátní. Během vyhledávání bude aplikace zobrazovat (napovídat) možné výsledky z dosud zadaných znaků.
- **Přidání oblíbené obce**
  Mobilní aplikace bude umožňovat přidávání obcí do seznamu oblíbených obcí uživatele.
- **Odebrání oblíbené obce**
  Mobilní aplikace bude umožňovat odebrání obcí ze seznamu oblíbených obcí uživatele.
- **Zaslání hlášení konkrétní obci**
  Mobilní aplikace bude umožňovat přihlášenému uživateli odeslat hlášení dané obci. Hlášení bude mít formát předmět a popis.
- **Zobrazení událostí**
  Mobilní aplikace bude umožňovat zobrazení událostí z obcí, které má uživatel v seznamu oblíbených.
- **Personalizace událostí podle kategorií**
  Mobilní aplikace bude umožňovat filtrování událostí podle zvolených kategorií pro každou obec zvlášť.
- **Zobrazení podrobnosti dané události**
  Mobilní aplikace bude umožňovat zobrazení podrobností o dané obci.

- **Změna velikosti zobrazených událostí**
  Mobilní aplikace bude umožňovat zobrazení událostí ve třech různých velikostech (hustotách). Malá velikost událostí bude mít pouze nadpis, datum vytvoření a obec, ke které se událost vztahuje. Střední bude obsahovat navíc část obsahu události a velká velikost bude mít zobrazenou část obsahu události větší než střední.
- **Řazení událostí**
  Mobilní aplikace bude umožňovat řazení zobrazených událostí podle jejich dostupných dat vytvoření.
- **Změna jazyka**
  Mobilní aplikace bude umožňovat změnu uživatelského jazyka. Dostupnými jazyky bude čeština a angličtina.
- **Zaslání zpětné vazby**
  Mobilní aplikace bude umožňovat přihlášenému uživateli zaslání zpětné vazby. To obnáší volbu hvězdiček od 1 do 5 nebo slovní ohodnocení.
- **Příručka k ovládání**
  Mobilní aplikace bude poskytovat příručku, jak aplikaci správně používat.
- **Smazání účtu**
  Mobilní aplikace bude umožňovat smazání celého uživatelského účtu. Tato akce bude nevratná a všechna uložená uživatelská data budou odstraněna.

C. Webová aplikace

- **Přihlášení do aplikace**
  Webová aplikace bude umožňovat přihlášení pověřené osoby pro správu dané obce.
- **Odhlášení z aplikace**
  Webová aplikace bude umožňovat odhlášení již přihlášené osoby.
- **Přidání události dané obce**
  Webová aplikace bude umožňovat manuální přidání události dané obci. Náležitosti události jsou totožné s údaji popsané výše u generování dat.
- **Úprava události dané obce**
  Webová aplikace bude umožňovat úpravu již existující události.
- **Smazání události dané obce**
  Webová aplikace bude umožňovat smazání již existující události.
- **Zobrazení hlášení**
  Webová aplikace bude umožňovat zobrazení hlášení od uživatelů.

3.1.2 Nefunkční

Nefunkčními požadavky jsou ty, podle kterých lze měřit kvalitu vyvíjeného produktu. Jsou to určitá pravidla a omezení kladená na samotný produkt nebo na jeho vývoj. Doplňují funkční požadavky.

A. Server

Část systému poběží na vzdáleném serveru. Server bude poskytovat rozhraní API, kam budou směřovány požadavky z mobilní a webové aplikace. Dále bude sloužit pro správu uživatelů a databázi.

B. Mobilní aplikace

Součástí systému je vytvoření Mobilní aplikace pro koncové uživatele. Mobilní aplikace musí podporovat dvě nejrozšířenější mobilní platformy, a to Android a iOS. Pro plnohodnotné využití mobilní aplikace bude zapotřebí přístup k internetu.

C. Webová aplikace

Webové rozhraní určené pro správu obsahu z daných obcí. Tato část systému bude vyžadovat přístup k internetu. Bez něho nebude možné do aplikace pokračovat.

D. Databáze

Systém bude disponovat databází pro ukládání všech potřebných dat (data obcí, události, uživatelská nastavení, hlášení).

E. Dostupnost

Data systému budou dostupná 24 hodin 7 dní v týdnu.

F. Bezpečnost

Veškerá data budou posílána pomocí zabezpečeného protokolu HTTPS. Určité akce budou vyžadovat ověření uživatele. Toto ověření bude prováděno pomocí přihlášení, kde přihlašovací údaje musí splňovat požadované vlastnosti tak, aby uživatelské účty nebyly snadno prolomitelné.

G. Škálovatelnost

Server zvládne zpracovat vyšší počet požadavků tak, aby tento jev minimálně dopadl na dobu odezvy serveru.

H. Rozšiřitelnost

Systém bude modulární pro jeho snadnou správu a budoucí rozšiřitelnost či vylepšení.

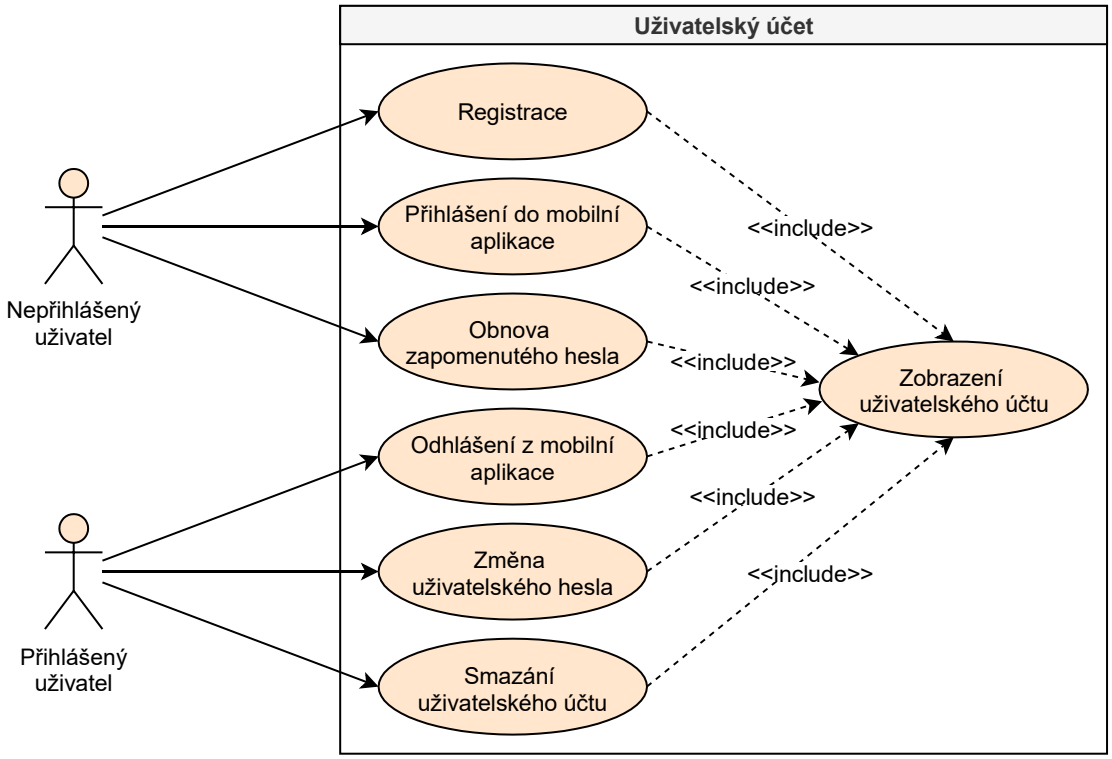
3.2 Případy užití

V této sekci se zaměříme na případy užití, což je popis chování systému z pohledu uživatele. Uživatel je považován za aktéra, který určitým způsobem se systémem interaguje za účel dosažení určitého cíle. Právě tuto komunikaci (aktéra se systémem) se snažíme zachytit. Případy užití znázorňují to, kteří aktéři komunikují s danými částmi systému a které akce smí vykonávat. Obecně případy užití popisují pouze funkční požadavky. Tato technika je velice důležitá a v softwarovém inženýrství nezbytná. Základním cílem těchto případů užití je, že velice dobře zachycují podrobnosti jednotlivých funkcionalit. Usnadňují odhad, kolik času a práce zabere vývoj. Dále jsou velice nápomocné při tvorbě uživatelských dokumentací a pro tvorbu testovacích scénářů (víme co přesně máme testovat).

Případy užití lze popsat pomocí diagramů. V těch, uživatel vystupuje jako aktér komunikující se systémem. Ovšem aktérem může být např. i jiný systém nebo třeba čas, který také může ovlivnit stav. Na následujících diagramech si ukážeme, jak aktéři používají náš systému. V našem případě budeme pracovat s přihlášeným a nepřihlášeným uživatelem a také s administrátorem obce. V diagramech se také nachází vazba `<<include>>` spojující dva případy užití. Vazba říká, že případ užití, ze kterého vazba vychází, zahrnuje i případ užití do kterého vazba směřuje.

3.2.1 Mobilní aplikace

V této části se zabýváme akcemi, které lze provádět v mobilní aplikaci. Následujícími aktéry jsou nepřihlášený a přihlášený uživatel. Na následujícím obrázku (Obrázek 1) lze vidět konkrétní operace prováděné uživateli, které se vztahují k uživatelskému účtu. Poté následují případy užití vztahující se k základním úkonům, které lze v aplikaci provádět. Ty jsou znázorněny na dalším obrázku (Obrázek 2)



Obrázek 1 - Případy užití - uživatelský účet

A. Zobrazení uživatelského účtu

Nepřihlášený uživatel vysune postranní menu a klikne na tlačítko týkající se uživatelského účtu. Následně zobrazený obsah o uživatelském účtu se přizpůsobí podle aktuální situace, zda je přihlášený či nikoliv.

B. Registrace uživatele

Nepřihlášený uživatel klikne v postranním menu na tlačítko týkající se uživatelského účtu. Zde se zobrazí možnosti pro přihlášení a registraci. Pokračováním na registraci se zobrazí formulář pro vyplnění e-mailové adresy a hesla. Toto heslo je následně potřeba ještě potvrdit. Pokud jsou uživatelská vstupní data validní, neuvidí uživatel žádnou chybovou hlášku. Potvrzením formuláře tlačítkem se data zkontrolují a odešlou na server, kde se zpracují. Následně uživatel obdrží informace o tom, jak proces vytváření účtu dopadl. V případě úspěchu je potřeba, aby uživatel potvrdil svoji e-mailovou adresu kliknutím na odkaz, který mu byl zaslán ze serveru. Tím se ověří totožnost uživatele a přihlášení do aplikace již bude funkční.

C. Přihlášení do mobilní aplikace

Nepřihlášený uživatel zvolí v postranním menu možnost uživatelského účtu. Zde vybere možnost pro přihlášení a tím se zobrazí formulář pro vyplnění údajů. Uživatel vyplní e-mailovou

adresu, heslo a potvrdí přihlášení. V případě nevalidních údajů, bude uživatel upozorněn chybovou hláškou. Pokud vše proběhne v pořádku, bude uživatel přesměrován na domovskou stránku uživatelského účtu s hláškou, že přihlášení proběhlo úspěšně. Současně se překreslí nabídka na dané stránce. Konkrétně zmizí nabídka přihlášení a registrace a místo nich se zobrazí možnosti pro odhlášení, změnu hesla a smazání účtu.

D. Odhlášení z mobilní aplikace

Na stránce uživatelského účtu přihlášeného uživatele je možnost pro odhlášení. Kliknutím na tuto možnost dojde k odhlášení uživatele a současně se smažou veškerá uživatelská data, která dosud byla uložena v paměti telefonu.

E. Změna uživatelského hesla

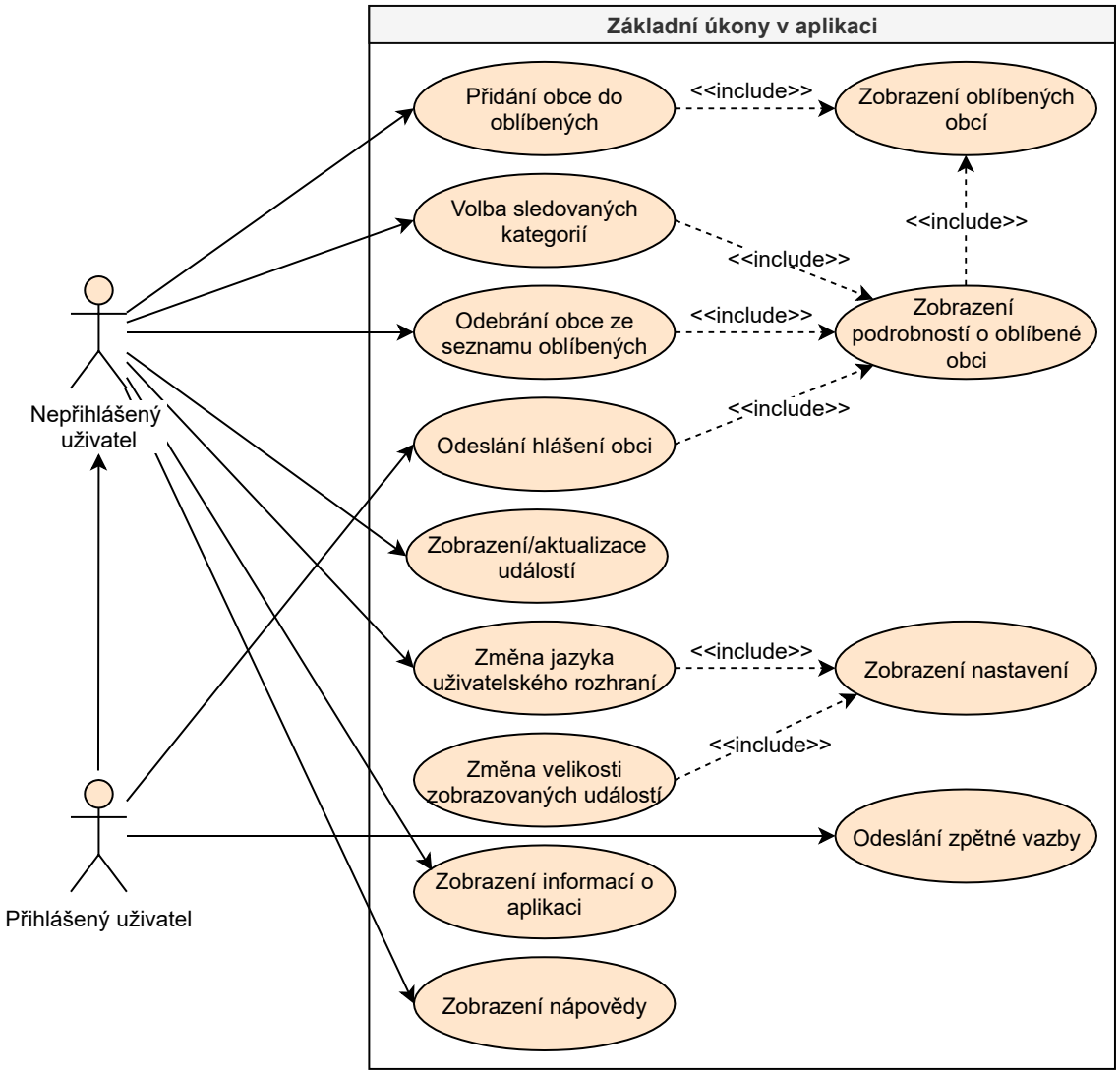
Přihlášený uživatel zvolí na stránce s uživatelským účtem možnost změny hesla. Tím se zobrazí formulář, kde uživatel vyplní své stávající heslo a dvakrát nové. Opět tato hesla musí splňovat určitá bezpečnostní kritéria. Potvrzením na tlačítko se odešle požadavek na server a následně se uživateli zobrazí hláška, jak akce dopadla. V případě úspěchu bude od této chvíle platit nové heslo.

F. Obnova zapomenutého hesla

V případě, že uživatel zapomene své heslo k účtu, lze ho obnovit. Na přihlašovací stránce klikne uživatel na možnost pro případ zapomenutého hesla. Následně se zobrazí stránka s instrukcemi a místem pro zadání e-mailové adresy. Tu uživatel vyplní a potvrdí kliknutím na tlačítko. Pokud server nalezne shodu s e-mailovou adresou mezi uživatelskými účty, odešle na zmiňovanou adresu e-mail. V tomto e-mailu se bude nacházet obnovovací odkaz, na který musí uživatel kliknout. Tím dojde k přesměrování na webovou stránku, kde bude vyzván zadání nového hesla. To uživatel zadá a potvrdí akci. Tím je heslo obnoveno.

G. Smazání uživatelského účtu

V postranním menu zvolí přihlášený uživatel možnost uživatelského účtu. Zde bude mít k dispozici možnost pro smazání účtu. Kliknutím na dané tlačítko se zobrazí potvrzovací formulář. Na něm bude uživatel vyzván, aby zadal svoji e-mailovou adresu. Po její zadání a kliknutí na potvrzovací tlačítko dojde k nevratnému smazání účtu. Potvrzovací formulář lze po celou dobu zrušit a vyhnout se tak smazání účtu.



Obrázek 2 - Případy užití - základní funkce

H. Odeslání zpětné vazby

Přihlášený uživatel zvolí v postranní nabídce možnost zpětné vazby. Zobrazí se stránka, kde může uživatel ohodnotit aplikaci pomocí hvězdiček od 1 do 5. Další možností je slovní ohodnocení, kde může uživatel vytknout nebo pochválit to, co se mu na aplikaci líbí nebo nelíbí. Jakmile je ohodnocení hotové potvrdí uživatel odeslání kliknutím na tlačítko Výsledek zpracování zpětné vazby (a jeho odeslání na server) je následně uživateli zobrazen.

I. Zobrazení nápovědy

Uživatel zvolí v postranním menu možnost pro zobrazení nápovědy. Tím se zobrazí stránka, která uživateli usnadní orientaci v aplikaci a také její ovládání.

J. Zobrazení informací o aplikaci

V postranním menu zvolí uživatel možnost pro zobrazení informací o aplikaci. Následně se zobrazí patřičné informace.

K. Zobrazení oblíbených obcí

V postranním menu zvolí uživatel možnost oblíbených obcí. Tím se zobrazí stránka, kde uživatel uvidí všechny své oblíbené obce.

L. Přidání obce do oblíbených

Uživatel zvolí v postranní nabídce možnost oblíbených obcí. Na následující stránce klikne na tlačítko pro přidání obce. Tím se uživatel dostane na stránku, kde je již možné obce vyhledávat. V horní části obrazovky se nachází pole pro zadání země, ve které se obec nachází. Uživatel toto pole vyplní a během psaní se mu rovnou zobrazují možné země z dosud napsaných znaků. Uživatel vybere zemi a následně se mu zobrazí další pole pro zadání jména obce. Při jeho vyplňování opět funguje možnost napovídání z dosud napsaných znaků. Po vybrání obce se zobrazí tlačítko pro potvrzení. Kliknutím na něj dojde k pokusu o přidání obce do seznamu oblíbených. Pokud se obec v seznamu již nachází, zobrazí se patřičná hláška. V případě, že se obec v seznamu nenachází, dojde k jejímu úspěšnému přidání. Současně s tím dojde k návratu na stránku s oblíbenými obcemi a zobrazení hlášky, že přidání proběhlo úspěšně.

M. Zobrazení podrobností o oblíbené obci

Uživatel zvolí v postranním menu možnost oblíbených obcí. Tím se mu zobrazí všechny jeho oblíbené obce. Kliknutím na danou obec se zobrazí její podrobnosti.

N. Volba sledovaných kategorií

Uživatel si zobrazí podrobnosti obce, u níž si přeje zvolit sledované kategorie. V podrobnostech obce uživatel najde sekci s těmito kategoriemi. Ty si vybere zaškrtáváním políček (checkboxů), nebo přepínáním tlačítkem.

O. Odeslání hlášení obci

Přihlášený uživatel si zobrazí podrobnosti obce, jíž chce odeslat hlášení. V dolní části této stránky se nachází tlačítko pro odeslání hlášení. Kliknutím na něj se uživatel ocitne na stránce s formulářem pro vyplnění. Tento formulář uživatel vyplní a potvrdí jeho odeslání. V případě nesprávných vstupních dat bude uživatel upozorněn patřičnou hláškou. Pokud dojde k odeslání hlášení v pořádku opět se zobrazí patřičná hláška.

P. Odebrání obce ze seznamu oblíbených

Uživatel si zobrazí podrobnosti obce, již si přeje odstranit. Úplně na konci této stránky se nachází tlačítko pro odstranění obce ze seznamu oblíbených. Kliknutím na něj se zobrazí potvrzovací dialog. Dalším potvrzením se obec ze seznamu odebere. Opačnou možností v dialogu je odebrání obce zrušit.

Q. Zobrazení/aktualizace událostí

Uživatel si zobrazí domovskou stránku aplikace. Jedná se o stránku s událostmi, která se také nachází v postranním menu. Na této stránce se automaticky zobrazí události z obcí, které má uživatel v seznamu oblíbených. Tyto události může uživatel aktualizovat manuálně tažením stránky směrem dolů. Tím se aktivuje aktualizační proces, který načte znovu dostupné události z oblíbených obcí.

R. Zobrazení nastavení

Uživatel vysune postranní menu, kde klikne na tlačítko indikující nastavení (ozubené kolečko). Poté se mu zobrazí veškerá aktuální nastavení, která mobilní aplikace nabízí.

S. Změna jazyka uživatelského rozhraní

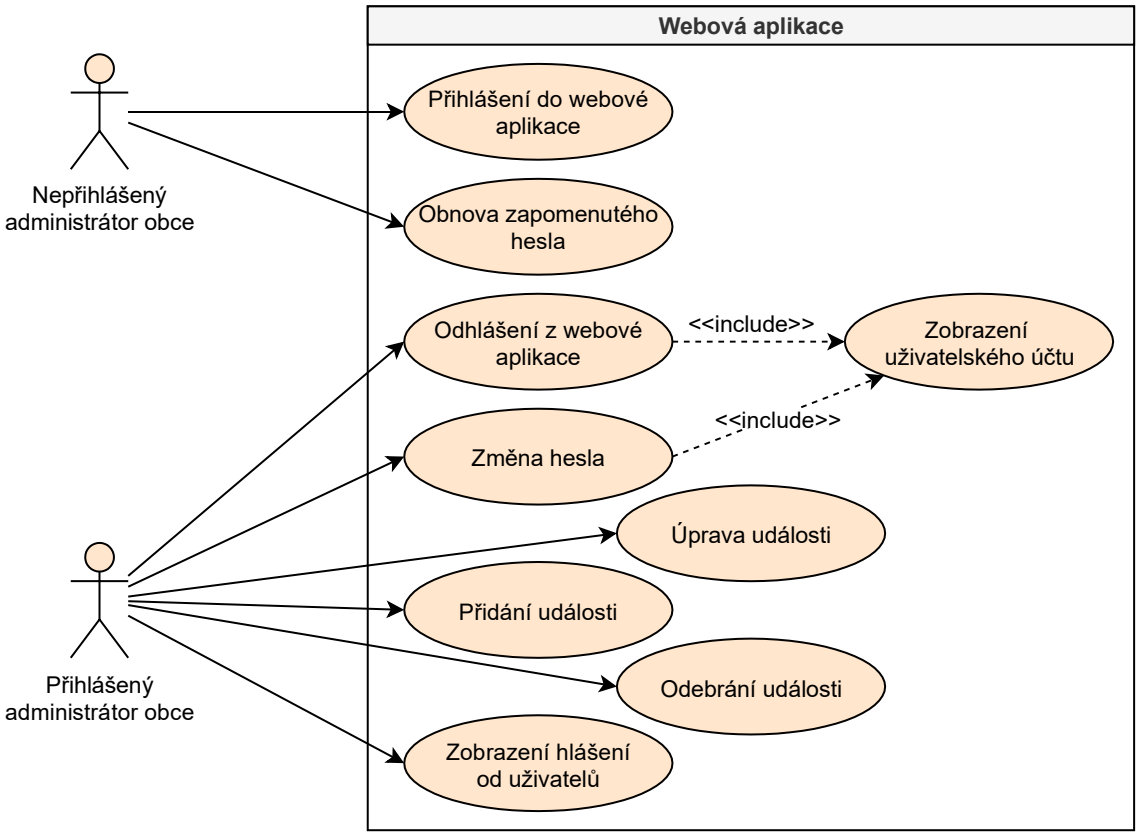
Uživatel zvolí v postranním menu možnost nastavení. Zde bude mít kromě jiného k dispozici volbu jazyka. Jeho volbou se okamžitě překreslí aktuální uživatelské rozhraní do patřičného jazyka.

T. Změna velikosti zobrazovaných událostí

Uživatel zvolí v postranním menu možnost nastavení. V něm nalezne možnost změny velikosti zobrazovaných událostí. Jakmile uživatel velikost změní, budou mít při dalším zobrazení požadovanou velikost.

3.2.2 Webová aplikace

V této části se zaměřujeme na aktivity, které lze provádět ve webové aplikaci. Zde máme jednoho aktéra, kterým je administrátor obce. Veškeré úkony lze vidět na obrázku níže (Obrázek 3).



Obrázek 3 - Případy užití - webová aplikace

A. Přihlášení do webové aplikace

Administrátor obce zadá patřičnou adresu URL a zvolí možnost přihlášení. Zadá přihlašovací údaje a pokud jsou správná bude vpuštěn do systému. V opačném případě se zobrazí patřičná hláška

B. Obnova zapomenutého hesla

Uživatel klikne v přihlašovacím formuláři na možnost pro zapomenutí hesla. Dále vyplní svoji e-mailovou adresu, s kterou se registroval v aplikaci. Po potvrzení tlačítkem pošle server

na zadanou e-mailovou adresu obnovovací odkaz. Kliknutím na něj dojde k možnosti nastavit si své heslo.

C. Zobrazení uživatelského účtu

Přihlášený uživatel klikne na ikonu účtu v horní části obrazovky. Tím se otevře menu, ve kterém najde další možnosti, které lze vykonat.

D. Odhlášení z webové aplikace

Administrátor obce si zobrazí podrobnosti o účtu v horním části obrazovky. Následně klikne na tlačítko odhlášení. Tím bude automaticky odhlášen a přesměrován na úvodní stránku s přihlášením.

E. Změna hesla

Přihlášený uživatel klikne na možnosti účtu v horní části stránky. Po rozbalení nabídky zvolí možnost změny hesla. Tím se zobrazí formulář pro zadání současného a 2x nového hesla. Pokud je současné heslo správné a nové splňuje bezpečnostní kritéria, pak bude heslo změněno.

F. Přidání události

Administrátor obce si zobrazí v postranní nabídce možnost událostí. Následně se mu zobrazí jednotlivé události. Současně se také zobrazí tlačítko pro přidání nové. Kliknutím na něj se otevře formulář, který administrátor vyplní a potvrdí. V případě, že některé údaje budou chybět nebo nebudou validní, bude na to upozorněn. V případě úspěchu dojde k přidání nové události mezi ostatní.

G. Úprava události

Administrátor obce si zobrazí události. Poté vybere tu, kterou si přeje upravit. Tlačítkem pro úpravy se událost zobrazí v upravovatelné formě. Administrátor provede patřičné změny a uloží je.

H. Odebrání události

Administrátor obce si zobrazí události. Nalezne tu, kterou si přeje odebrat a klikne na tlačítko pro odstranění. Následně potvrdí smazání kliknutím na tlačítko v dialogovém oknu a událost je tím odebrána.

I. Zobrazení hlášení od uživatelů

Administrátor obce vybere v postranním menu možnost hlášení od uživatelů. Poté se zobrazí jednotlivá hlášení. Ty si může administrátor zobrazit.

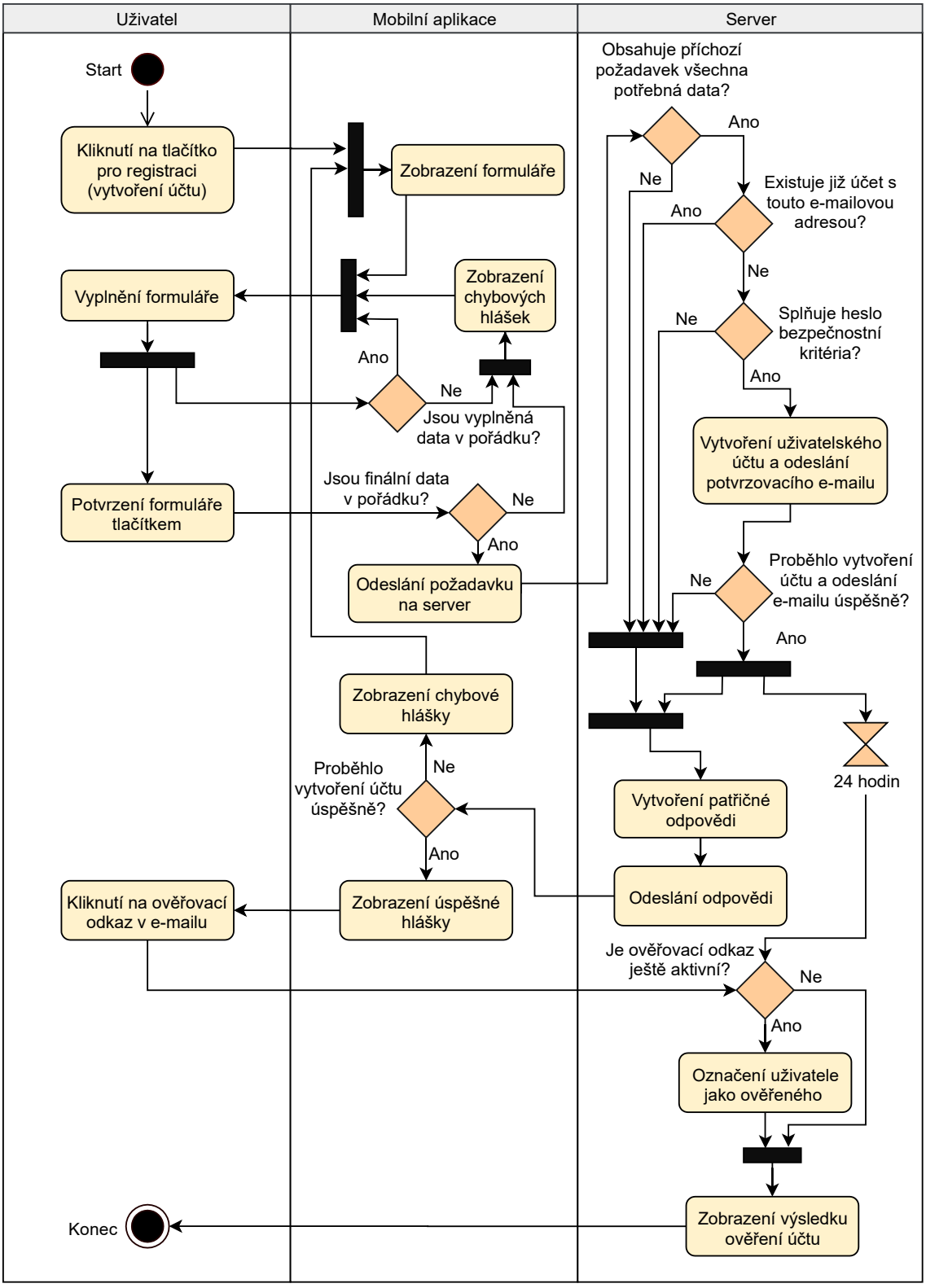
3.3 Diagramy aktivit

Na následujících diagramech aktivit si zobrazíme interakci uživatele se systémem. Tyto diagramy popisují procedurální logiku jednotlivých procesů. Úkony lze rozdělit mezi uživatele a části systému, které je později zpracovávají. Lze říci, že jde o popis případů užití pomocí posloupnosti akcí (workflow), které se v systému odehrávají. V rámci diagramů aktivit si ukážeme proces registrace a změnu hesla.

3.3.1 Registrace uživatele

Registrace neboli vytvoření uživatelského účtu je jednou z velmi důležitých součástí systému. Celý tento proces je popsán na obrázku níže (Obrázek 4) a začíná kliknutím na tlačítko pro registraci. Mobilní aplikace zobrazí formulář k vyplnění (jedná se o e-mailový učet a heslo). Uživatel formulář vyplní. Během vyplňování údajů mobilní aplikace průběžně kontroluje validitu vyplňovaných údajů. Pokud nejsou validní, aplikace místo zvýrazní tak, aby uživatel snáze identifikoval chybu. Jakmile uživatel potvrdí odeslání formuláře, dojde k finální kontrole ze strany aplikace. Pokud je vše v pořádku, odešle se požadavek na server. V opačném případě se označí chybně zadané údaje. Na serveru dojde ke kontrole, zda příchozí požadavek obsahuje všechna náležitá data. Pokud ano, dojde k dalším kontrolám. Mezi těmito kontrolami jsou: kontrola, zda existuje již účet se stejnou e-mailovou adresou a zda splňuje heslo bezpečnostní kritéria. V případě, že je vše v pořádku, dojde k vytvoření uživatelského účtu a odeslání potvrzovacího e-mailu na e-mailovou adresu vytvářeného účtu. Pokud jakákoliv kontrola na serveru selže, vytvoří se na základě této chyby patřičná odpověď. Následuje sestavení odpovědi s patřičným obsahem a odeslání zpět do mobilní aplikace. Zde proběhne kontrola zda se vytvoření účtu zdařilo či nikoliv. Podle toho aplikace zobrazí patřičné hlášky.

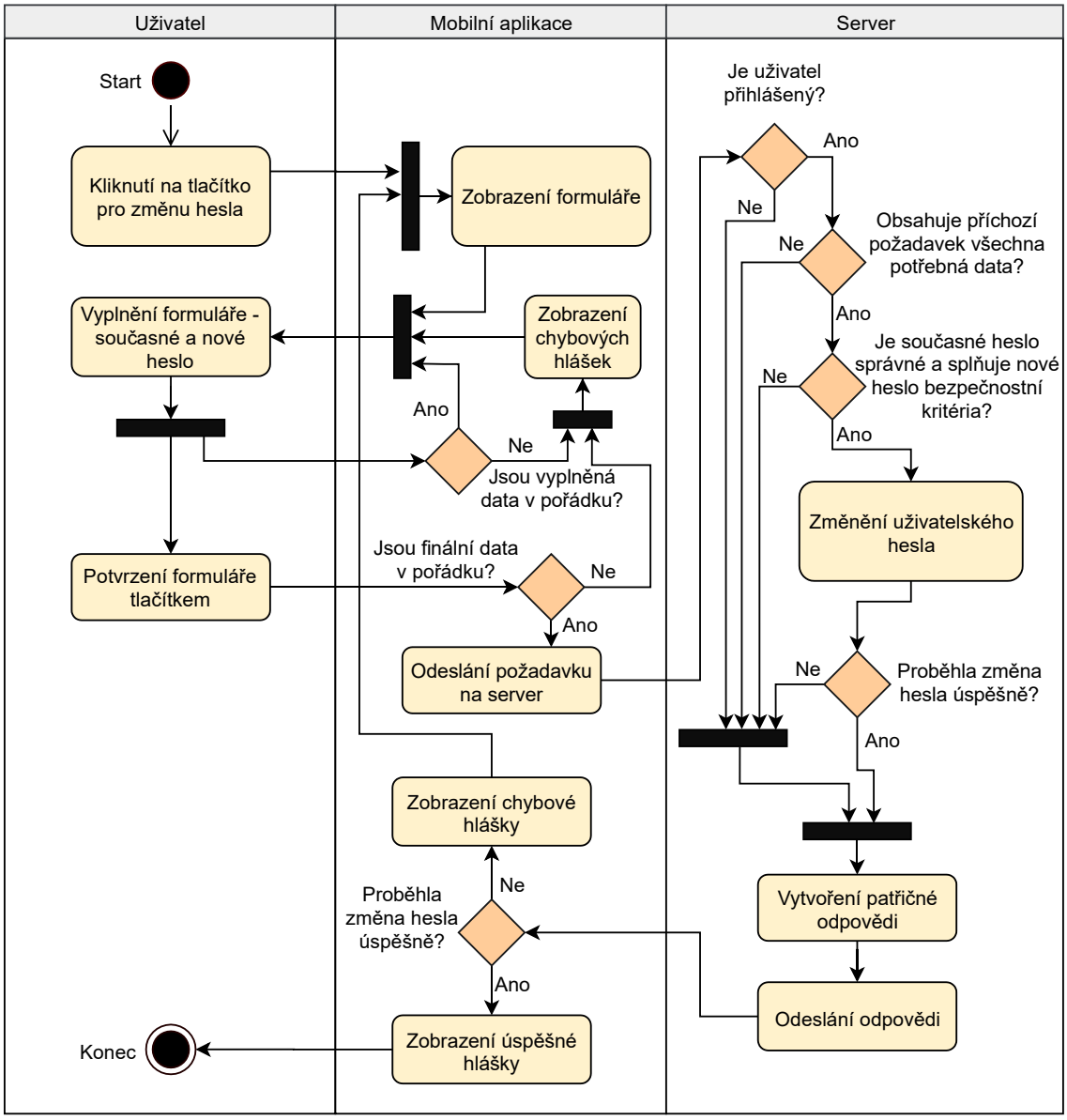
V tuto chvíli je uživatelský účet potřeba potvrdit kliknutím na odkaz, jež byl zaslán na uživatelovu e-mailovou adresu. Jakmile dojde ke kliknutí, na serveru se zkontroluje, zda odkaz již expiroval. Pokud ne, server označí uživatele jako ověřeného. Tím je účet vytvořený a aktivovaný. V opačném případě server informuje uživatele, že došlo k expiraci daného odkazu.



Obrázek 4 - Diagram aktivit - proces registrace uživatele

3.3.2 Změna hesla

Pro změnu hesla musí uživatel podstoupit následující proces (Obrázek 5). Nejprve klikne na tlačítko pro změnu hesla. Aplikace mu vykreslí formulář pro zadání současného a nového hesla. Uživatel tento formulář vyplní. Během psaní aplikace průběžně validuje jeho vstupní údaje, zda splňují bezpečnostní omezení. Pokud nesplňují, zobrazí se chybové hlášky. Jakmile uživatel potvrdí změnu hesla kliknutím na tlačítko, proběhne v mobilní aplikaci závěrečná validace. Pokud je vše v pořádku, odešle se požadavek na server. Na serveru se nejprve zjistí, zda je uživatel již přihlášený. Dále, zda požadavek obsahuje všechna potřebná data a na závěr, jestli je současné heslo platné a nové splňuje bezpečnostní kritéria. Pokud je vše splněno, server změní uživatelovo heslo a vytvoří úspěšnou odpověď. V opačném případě sestaví chybovou odpověď. Server následně zkompletuje odpověď a odešle ji zpět do aplikace. V aplikaci se zjistí, jestli se změna hesla podařila či nikoliv. V obou případech se zobrazí patřičná hláška s výsledkem změny hesla. V případě neúspěchu dojde k překreslení formuláře pro nový pokus.



Obrázek 5 - Diagram aktivit - změna uživatelského hesla

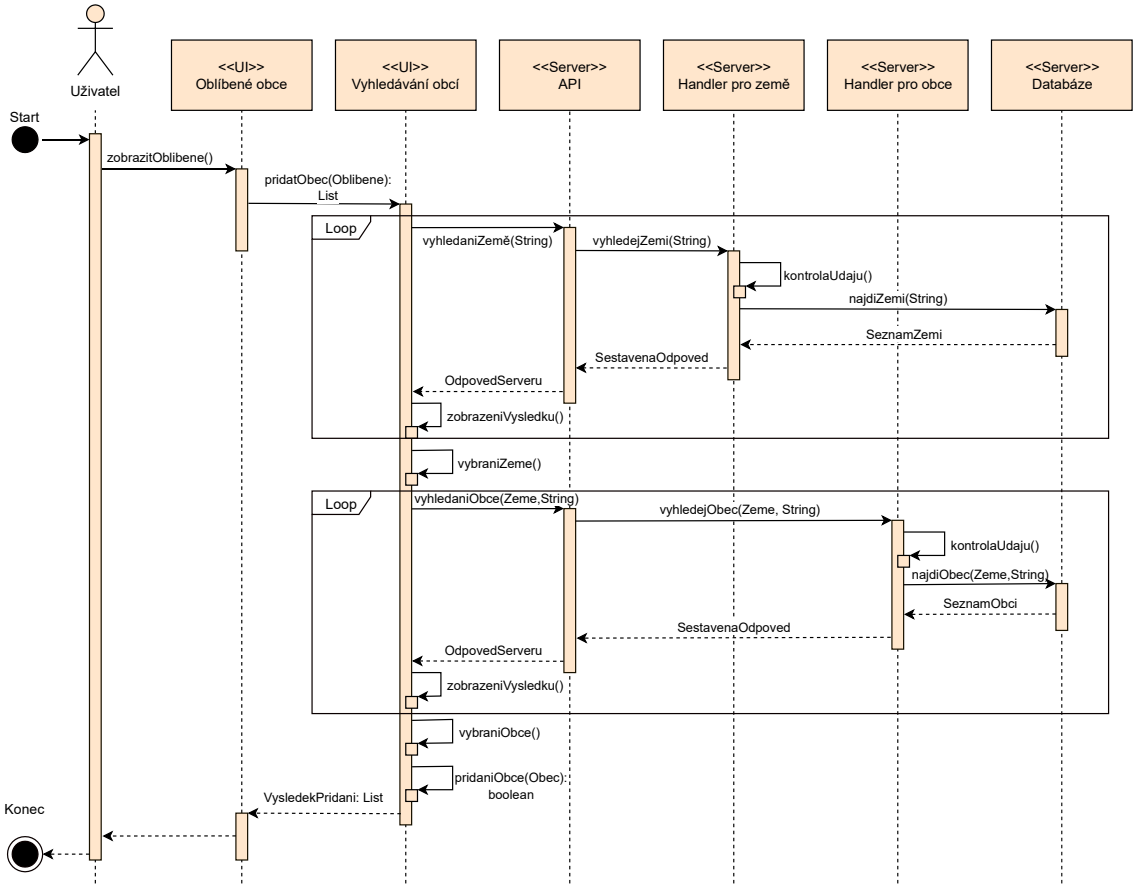
3.4 Sekvenční diagramy

Jde o jeden z diagramů UML, který je zaměřen na chování. Sekvenční diagramy znázorňují komunikaci mezi objekty v rámci určitého procesu. Zachycují posloupnost této komunikace (posílání zpráv) v čase. Současně popisují i to, s jakými objekty se v rámci komunikace pracuje. Na ukázku je vypracován diagram pro přidání obce do seznamu oblíbených.

3.4.1 Přidání oblíbené obce

Na následující diagramu (Obrázek 6) je znázorněn proces přidání obce do seznamu oblíbených. V rámci vykonávání jednotlivých úkonů se zde setkáváme s objekty jako jsou: uživatel, části mobilní aplikace a serverové komponenty.

Vše začíná poté, co uživatel klikne na tlačítko pro zobrazení oblíbených obcí a následně na tlačítko pro přidání obce. Zde se uživateli zobrazí možnost pro vyhledávání země. Psaním do vyhledávacího pole se automaticky posílají dosud napsané znaky na API serveru. Zde dojde k filtraci požadavku a jeho přesměrování tzv. Handleru (funkce reagující na určitou událost – v tomto případě na příchozí požadavek), který požadavek obslouží. Handler zkontroluje přítomnost potřebných dat k vyhledání zemí. Pokud je vše v pořádku, odešle požadavek na vyhledání země do databáze. Databáze provede samotné vyhledání a vrátí výsledek Handleru. Ten sestaví odpověď, kterou odešle zpět do API, a to následně do mobilní aplikace. Přijaté výsledky vyhledávání se ihned zobrazí uživateli, takže má snazší výběr. Podobný proces funguje i pro vyhledávání samotných obcí. Zde se však s požadavkem na server posílá také země, ve které se má obec vyhledávat. API serveru přesměruje požadavek na Handler, ale jiný, než tomu bylo v předchozím případě. Tentokrát jde o Handler spravující vyhledávání obcí. Pokud jsou příchozí údaje v pořádku, odešle dotaz do databáze. Z následných výsledků je opět sestavena odpověď, která je přes API odeslána do mobilní aplikace. Zde se výsledky zobrazí a uživatel vybere obec, kterou si přeje přidat do seznamu oblíbených. Výsledek přidání se uživateli zobrazí a tím celý proces končí.



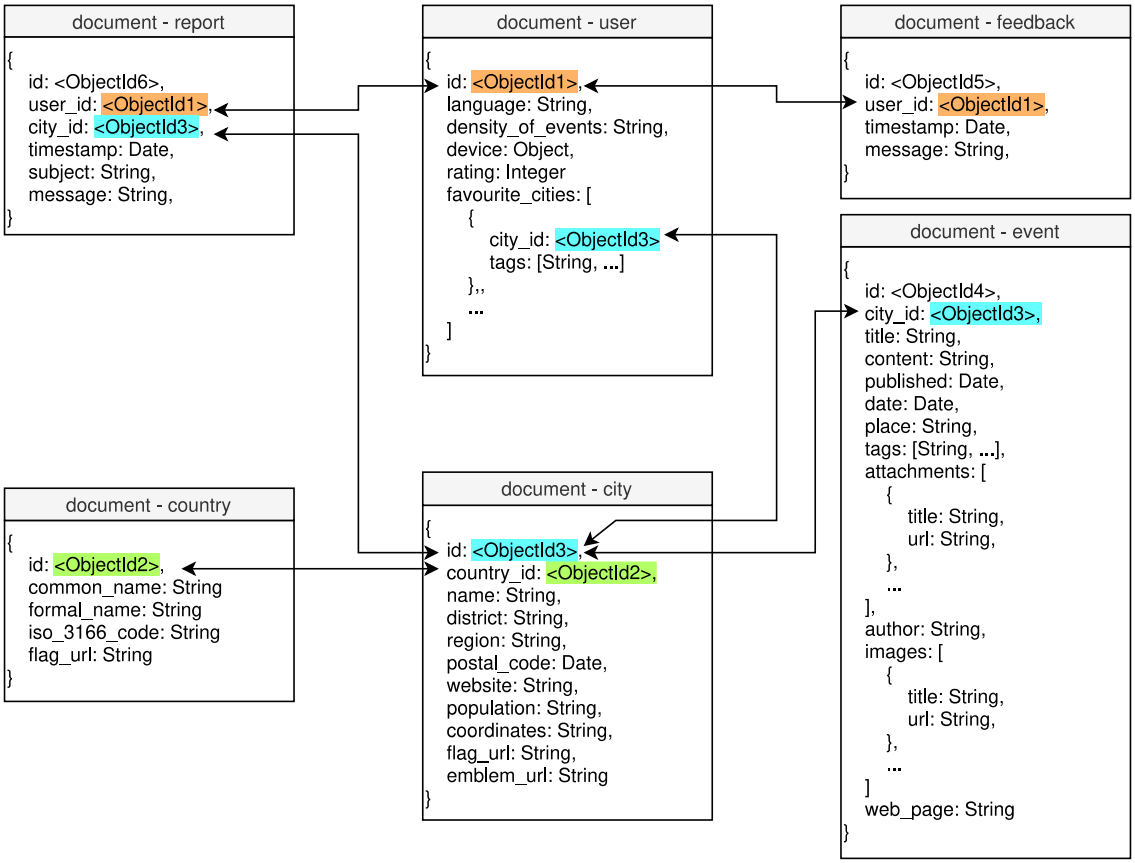
Obrázek 6 - Sekvenční diagram (přidání oblíbené obce)

3.5 Databáze

Další základní komponentou systému je databáze, kde budou uložena veškerá data. Pro návrh databáze je velice důležité zvolit správný model. Jejich hlavní rozdíly jsou ve struktuře ukládání dat a vazbami mezi nimi. Dále je rozdílné i následné dotazování nad daty. Mezi základní modely patří relační a NoSQL databáze. Mezi často používané systémy relačních databází jsou např. MySQL, PostgreSQL, Oracle nebo Microsoft SQL Server. Na druhé straně existuje také několik typů NoSQL databází. Mezi nejčastější patří databáze typu klíč-hodnota, dokumentové, grafové nebo sloupcové.

Volba správného modelu se odvozuje od požadavků na systém a struktury používaných dat. Dalšími faktory jsou např. způsob kterým se bude systém používat, popř. na které funkcionality bude kladen důraz. V našem případě se bude často pracovat s aktualitami obcí, které představují velké množství textu a různé struktury (ne vždy budeme mít o každé aktualitě všechna data, která bychom chtěli). Vzhledem k tomu, že nebude docházet k častým úpravám těchto záznamů (značná část dat zůstane, tak jak je vytvoříme), vychází lépe NoSQL databáze. Současně je systém více flexibilní pro budoucí rozšíření. To znamená, že stará a nová data mohou bez problému existovat současně vedle sebe. Vzhledem k tomu, že v rámci komunikace mobilní aplikace a serveru jsou data uložena v JSON formátu, je ideální využít takový databázový systém, jež JSON formát využívá také. Tím je dokumentový systémem MongoDB. Celá tato myšlenka volby dokumentové NoSQL databáze mi byla potvrzena vyučujícím panem RNDr. Martinem Svobodou, Ph.D., který pracuje na ČVUT.

Na následujícím obrázku (Obrázek 7) lze vidět základní schéma databáze vytvářeného systému. Základními prvky databáze jsou kolekce obsahující dokumenty. V těchto dokumentech jsou pak uložena data. V rámci systému máme následující kolekce: uživatelé, země, obce, události, hlášení a zpětné vazby. Na obrázku lze vidět atributy jednotlivých dokumentů a vazby toho, které dokumenty uchovávají informace o jiných (barevně označené identifikátory).



Obrázek 7 - Schéma databáze

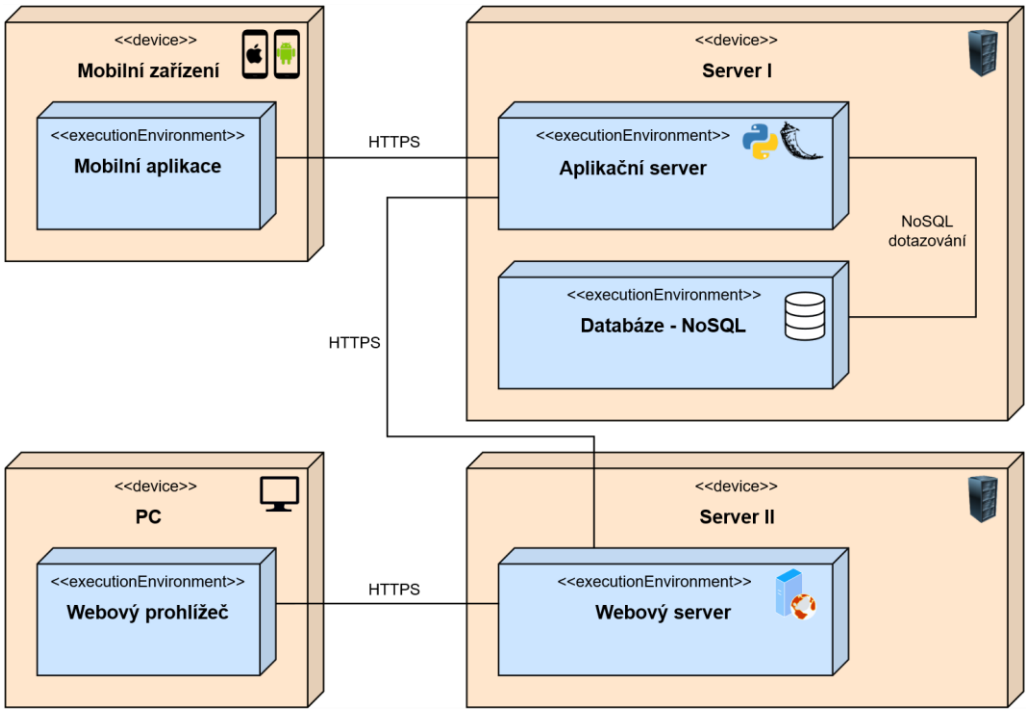
3.6 Diagram nasazení

V rámci diagramu nasazení se zaměřujeme na hardwarovou strukturu systému a také samotné nasazení softwaru na patřičný hardware. Setkáváme se zde s dvěma základními prvky (uzly). Těmi jsou <<device>> označující hardwarová zařízení a <<executionEnvironment>> popisující softwarovou část.

Následující diagram (Obrázek 8) popisuje obecný návrh nasazení vyvíjeného systému. V levé horní části se nachází mobilní zařízení, na kterém poběží aplikace. Ta bude komunikovat s API serveru pomocí protokolu HTTPS. Na serveru se bude veškerá logika provádět na aplikačním serveru, který poběží v programovacím jazyce Python. Konkrétně bude použit framework Flask. Aplikační server bude mít současně přístup do databáze. Tato databáze bude dokumentová, tedy NoSQL.

Pro administraci obsahu ze strany obcí bude sloužit webový prohlížeč na libovolném počítači. Ten bude komunikovat s webovým serverem také pomocí protokolu HTTPS. Tento webový server by mohl běžet na nějakém jiném místě (serveru) než je aplikační server s databází. Proto je v diagramu znázorněn v odděleném serveru. Jeho základní funkcí je posbírat všechna potřebná data a vygenerovat HTML kód, který pošle do prohlížeče. Požadavky z webového

prohlížeče (od administrátorů obcí) se na webovém serveru odchytí a přepošlou na aplikační server. Také zde bude využito protokolu HTTPS. Aplikační server požadavky zpracuje a může dojít i k dotazování do databáze. Následné výsledky jsou poslány z aplikačního na webový server. Ten na základě odpovědi vygeneruje patřičný HTML kód, který odešle zpět do webového prohlížeče a ten ho vykreslí.



Obrázek 8 - Diagram nasazení

4 Použité technologie

V této kapitole jsou představeny technologie, pomocí kterých je systém sestaven a které byly použity. Ve velké míře se jedná o službu Google Cloud Platform, pomocí které je realizována celá serverová část. Dalšími technologiemi jsou např. různé postupy, standardy použité při vývoji či způsob realizace mobilní aplikace.

4.1 Google Cloud Platform

Jednou ze základních technologií byla volba cloudové služby, na které poběží server. Kromě použití cloudových služeb byla také možnost hostit si server sám. Tuto možnost jsem však vzhledem k náročnosti správy zavrhl. Při reálném nasazení by se stejně použil nějaký existující cloud, takže by bylo užitečné využít nějaký existující. V rámci profesionálních cloudových služeb, se nabízely 3 velmi známé služby, které jsou v této oblasti nejrozšířenější. Těmito službami jsou:

- Amazon Web Services (AWS)
- Microsoft Azure (MS Azure)
- Google Cloud Platform (GCP)

Tyto služby zároveň poskytují tzv. „Free tier“ účet, který umožňuje uživateli vyzkoušet některé funkce na rok zdarma nebo alespoň v omezeném rozsahu. Některé tyto společnosti současně dávají uživateli určitou peněžní částku tak, aby uživatel mohl některé placené služby testovat a zkoušet. Všechny tři tyto služby se stávají vůči sobě čím dál vyrovnanější, co se funkcionalit týče. Podrobnější porovnání těchto služeb je znázorněno v tabulce níže (Tabulka 2).

Tabulka 2 - Porovnání cloudových „Free tier“ služeb jednotlivých společností k 10. 12. 2019

Služba	Amazon Web Services	Microsoft Azure	Google Cloud Platform
Kredit	-	$ 200 / 12 měsíců	$ 300 / 12 měsíců
Serverless Funkce[4]	Lambda	Azure Functions	Cloud Functions
	1 mil volání / měsíc	1 mil volání / měsíc	2 mil volání / měsíc
Relační databáze	RDS	SQL Database	Cloud SQL
	20 GB	250 GB	$ 7,67 / měsíc
NoSQL databáze	DynamoDB	Azure Cosmos DB	Firestore
	25 GB	5 GB	1 GB
Úložiště	S3	File Storage	Cloud Storage
	5 GB	5 GB	5 GB
API	API Gateway	API Management	AppEngine, Cloud run
	1 mil volání / měsíc	není zdarma	různé ceny dle prostředí
Autentifikace uživatelů	Cognito	Active Directory	Firebase Authentication
	50000 ověření / měsíc	50000 ověření / měsíc	10000 ověření / měsíc

Vzhledem k tomu, že již v bakalářské práci jsem pracoval s AWS, jsem se rozhodl tuto možnost vyloučit. Chtěl jsem poznat technologie jiných společností, tak abych se naučil něco nového. Tím mi zbývaly pouze 2 možnosti (MS Azure nebo GCP).

[4] Služby běžící pouze tehdy když jsou potřeba.

Původně jsem zamýšlel použít službu MS Azure, která vyloženě podporuje konkrétní typ dokumentové databáze MongoDB, který jsem plánoval použít. Bohužel při další práci s MS Azure jsem zjistil, že služba, která by sloužila jako API (API Management) je placená. Po krátké době testování částka za využívání API stále rostla. Ve spojení s ne zcela přehledným uživatelským rozhraním dalších MS Azure služeb jsem se rozhodl zvolit službu GCP. Google Cloud Platform sice nepodporuje konkrétně databázi MongoDB, ale má alternativní databázovou službu Firestore. Jde také o NoSQL dokumentovou databázi. Poskytuje vysoký výkon a snadnou použitelnost. Od klasických NoSQL databází se Firestore liší způsobem popisování vztahů mezi objekty [22]. Dalším hlavním důvodem pro volbu GCP bylo to, že ani po otestování značného množství jiných funkcí se nenačítala žádná peněžní částka, jako tomu bylo v případě MS Azure. Následující podsekce se vztahují k jednotlivým službám GCP, které byly v práci použity.

4.1.1 App Engine

App Engine je jednou ze služeb GCP, která je zdarma. Ovšem jen do míry zatížení a využití. Na rozměry této diplomové práce je však bohatě dostačující. Základní funkcí je, že služba umožňuje provoz aplikací. Tyto aplikace mohou být napsány v mnoha jazycích, a i nadále se toto portfolio podporovaných jazyků rozrůstá. Další důležitou vlastností je, že služba je tzv. „serverless“. To znamená, že aplikace běží pouze tehdy, když je požadována. V opačném případě vyčkává a tím tak šetří zdroje i peníze.

4.1.2 Firebase Authentication

Služba sloužící ke správě uživatelských účtů a jejich identit. Součástí je zakládání účtů, ověřování uživatelů, obnova hesla a další akce spojené s uživatelským účtem. Firebase Authentication umožňuje i přihlašování pomocí služeb třetích stran. Těmi jsou např. Google, Facebook, Twitter, Github, Yahoo, Microsoft či Apple. V mém případě však využívám pouze možnost ověření pomocí emailové adresy a hesla.

4.1.3 Firestore

Tato služba zastupuje databázi. Firestore je NoSQL databáze, která je velice podobná databázi MongoDB. Na rozdíl od SQL jde tedy o dokumentovou databázi. Její základní strukturou jsou kolekce. V těchto kolekcích jsou dokumenty a v těchto dokumentech jsou naše finální data, která potřebujeme uložit. To je základní struktura NoSQL databází. Samozřejmostí je vnořování kolekcí a vytváření tak složitějších databázových struktur. Služba umožňuje správu dat pomocí několika programovacích jazyků. Mezi nimi jsou např. C++, Unity, Node.js, Java, Python nebo Go [23]. Kromě Firestore by šlo použít také realtime databázi. To však pro účely této práce není nutné. Správa dat v rámci této práce se provádí přes jednotné API.

4.1.4 Cloud Storage

Storage slouží jako úložiště dat. V rámci mého účtu na GCP je kapacita úložiště 5 GB. Základní účel je ukládání jakýchkoliv dat. Lze sem tak ukládat multimediální data, jako jsou např. fotky uživatelů či obecních aktualit. Kromě multimédií (jako jsou fotky a videa) [24] se do tohoto úložiště nahrávají i zdrojové kódy pro App Engine. Do úložiště se tak počítají veškeré věci nahrané na cloud. Výhodou je, že se platí za takové množství, které člověk využívá. Ovšem dnes se tímto modelem účtování setkáváme čím dál častěji.

4.1.5 Cloud SDK

Celým názvem Google Cloud Software Development Kit. Zajišťuje správu služeb a aplikací hostovaných na Google Cloud. Tato správa je prováděna formou příkazové řádky. Pomocí příkazů jsem např. nahrával aktuální verze mého serveru do služby App Engine popsané výše. Google Cloud SDK je však potřeba nejprve nainstalovat do PC a integrovat ho tak do místního příkazového řádku. Podporován je Linux, Windows, macOS a další systémy.

4.2 Git

Jde o verzovací systém, pomocí kterého lze spravovat jednotlivé verze vyvíjeného systému. Výhodou je ukládání verzí jednotlivých stavů a tím mít možnost se k některé verzi třeba vrátit. Další hlavní výhodou je kontrola nad projektem, pokud na něm pracuje více osob. Mezi další výhody patří možnost experimentování s projektem, bezpečnost, přenositelnost mezi zařízeními a to, že je bezplatný. V rámci diplomové práce jsem využíval aplikaci Bitbucket[5], která tuto možnost poskytuje. Verzování jsem aplikoval na všechny programátorské části této práce.

4.3 JSON Web Tokens

Zkráceně JWT je způsob bezpečné výměny informací mezi dvěma stranami. Jak již název napovídá tento proces je obstarán pomocí tokenů. Základní princip používání je následující. Uživatel se přihlásí na serveru pomocí přihlašovacích údajů nebo jiné služby. Zpravidla se jedná o autentizační server. Server v případě úspěšného přihlášení vygeneruje JWT token a přidá ho do odpovědi uživateli. Uživatel si token uloží a při dalších požadavcích na server (API aplikace) ho přidá do hlavičky HTTP požadavku. Server pak díky tokenu ví, že daný uživatel je přihlášen.

Každý tento token se skládá ze tří částí. Všechny části jsou od sebe odděleny tečkou. První částí je hlavička (Header) obsahující informace o typu algoritmu, který je použit pro podpis. Druhou částí jsou data (Payload) týkající se daného uživatele. Mezi tato data patří např. jméno, uživatelské jméno, uživatelský identifikátor, email a další údaje. Poslední částí je podpis (Signature). Ten slouží k ověření pravosti dat.

4.4 Postman

Jedná se o program, pomocí kterého lze vytvářet HTTP požadavky a testovat tím tak HTTP API. Aplikace je velice jednoduchá a intuitivní. Po odeslání požadavku, aplikace zobrazí i odpověď od serveru. Lze tak tedy velice dobře testovat API a zjistit, jak v různých situacích reaguje. Při vytváření požadavků se nabízí široká škála nastavení. Od nastavení různých atributů v hlavičce požadavku, až po různé způsoby zápisu těla (body). Samozřejmostí je volba různých HTTP metod jako jsou POST, GET, PATCH a další.

4.5 Python

Základní technologií při vytváření systému je volba programovacího jazyka. Pro serverovou část jsem zvolil programovací jazyk Python. Jde o skriptovací jazyk, který je v dnešní době velice oblíbený a jednoduchý. Vzhledem k zamýšleným úkonům, které je na serveru potřeba vykonávat,

[5] Verzovací systém Bitbucket (https://bitbucket.org/)

byla jeho volba jednoduchá. Python zároveň přispívá k přehlednosti, protože jeho kód musí být správně odsazován, jinak nefunguje správně.

4.5.1 Flask

Hlavní částí implementace serveru je použití frameworku Flask programovacího jazyka Python. Ten umožňuje mapování HTTP dotazů, které přichází na server. Toto mapování příchozích dotazů se provádí podle adresy URL a použité HTTP metody.

4.5.2 Beautiful Soup

Jde o další framework Pythonu, který slouží ke snadnému získávání informací z HTML nebo XML souborů. Tomuto procesu získávání dat se anglicky říká „Web scraping“. Tento framework z dané stránky a nalezených značek vytvoří strom. Ten je pak snadné procházet a hledat ty věci, které člověka zajímají. Jeho použití je velice jednoduché a poskytuje široké možnosti při vyhledávání. Výhodou je, že dokáže pracovat i s nevalidními HTML stránkami, což je výhoda, protože ne vždy lze se stránkou pracovat jako s validním XML formátem.

4.6 Javascript

JavaScript [25] je, jak již název napovídá, skriptovací programovací jazyk a je na něm postavena celá mobilní aplikace. Jeho základními vlastnostmi je, že je multiplatformní a objektový. Zásadně se však odlišuje od jazyků jako jsou Java, nebo C/C++. JavaScript může běžet jak na klientské, tak serverové straně. Nejčastější klientskou stranou je webový prohlížeč.

4.6.1 React

React [26] je knihovna (framework) programovacího jazyka JavaScript. Primárně je určena pro tvorbu uživatelského rozhraní. Z pohledu MVC (Model View Controller) architektury jde tedy o část View. React byl vyvinut společností Facebook. Základním prvkem jsou komponenty, které představují modifikovatelné a znovupoužitelné HTML elementy. Těm lze nastavovat určité vlastnosti. Komponenty mají také svůj vnitřní stav. Framework React se nejčastěji používá při vytváření samostatných webových stránek, vykreslování stránek na straně serveru a v neposlední řadě při tvorbě multiplatformních nativních mobilních aplikací.

4.6.2 React Native

Jedná se o další framework společnosti Facebook. Je založen na programovacím jazyce JavaScript. Jeho hlavní výhodou je, že pomocí něho lze vytvářet multiplatformní mobilní aplikace pro Android a iOS. Výsledná aplikace je tvořena nativními komponentami, na rozdíl od konkurenčních frameworků (např. Cordova), u kterých je veškerá funkčnost (i jednotlivé komponenty) zajištěna webovou stránkou běžící v nativní systémové komponentě WebView. Z toho plyne, že použitím nativních komponent obou platforem (Android a iOS) je dosaženo vyšší rychlosti a plynulosti chodu aplikace. I nástroje jako React Native mají však své úskalí. Pro složité a náročné aplikace s hlubší integrací do operačního systému by tato technologie nemusela být dostačující. Vzhledem k povaze vyvíjené aplikace je však pro toto řešení ideální.

4.6.3 Redux

Také se jedná o knihovnu programovacího jazyka JavaScript. Ta je určenou pro správu dat a stavu aplikace. Díky této knihovně odpadá složité provázání mezi jednotlivými obrazovkami.

To by se normálně řešilo pomocí parametrů při přechodu na jinou obrazovku. Pokud by pak došlo k přesunutí dané obrazovky jinam, došlo by k ztrátě vazby a následně i dat. Výhodou této knihovny je, že dokáže měnit data současně na všech místech, kde jsou použita najednou. Není tak potřeba si složitě předávat jednotlivé údaje přes parametry či jiným způsobem. Další výhodou je, že při správném nastavení dokáže ukládat všechna data, která uchovává do tzv. perzistentního úložiště daného zařízení. V případě mobilních zařízení to je paměť telefonu, ve webových aplikacích jde o úložiště daného prohlížeče. Díky těmto vlastnostem je vyvíjená aplikace více flexibilní. Redux má 3 základní prvky. První z nich je tzv. Store. Jde o objekt, který uchovává data naší aplikace. V rámci celé aplikace máme pouze jeden. Store může uchovávat data jakéhokoliv typu. Lze nad ním také volat určité funkce. Těmi jsou: `getState()` vracící uložená data, `subscribe(callback)` pro zjištění, zda se data změnila a `dispatch(action)` pomocí které lze data ve Store změnit. Dalším důležitým prvkem Reduxu je Action, který voláme z funkce `subscribe()`. Action má povinný atribut *type* a popisuje, ke kterým všem změnám může dojít. Posledním prvkem je Reducer. Ten se zaměřuje na vykonání změny dat v úložišti Store. Reducer popisuje, jakým způsobem se budou daná data měnit.

4.7 REST

Celým názvem Representational State Transfer je architektura rozhraní, která je navržena pro distribuované prostředí. Tato architektura nám pomáhá definovat rozhraní API a využívá protokolu HTTP. Současně zavádí určitá pravidla (standardy) a způsoby jakými se s rozhraním má komunikovat. Mezi ně spadá adresa zdroje URI a použitá metoda. Tím by měl být identifikovatelný každý zdroj, ke kterému se snažíme přistoupit. REST definuje 4 základní HTTP metody pro přístup ke zdrojům (GET, POST, DELETE, PUT).

4.8 SendGrid

SendGrid je webová aplikace poskytující emailové služby. Mezi základní služby se řadí např. zpracování příchozích emailů či jejich posílání. Obsah emailů si může uživatel nastavit dle libosti. SendGrid také disponuje přehledným webovým rozhraním, kde lze vidět veškeré své nastavení, statistiky, aktivní služby a jejich využití. Na základě tohoto využití se pak vypočítává cena za využívání služby.

4.9 Vývojová prostředí

Jednou z posledních technologií jsou vývojová prostředí, která byla využita při vytváření serveru a mobilní aplikace.

4.9.1 PyCharm

Kompletní severovou část jsem programoval pomocí aplikace PyCharm. Jde o aplikaci společnosti JetBrains s.r.o.[6]. Pycharm je navržen speciálně pro vývoj v programovacím jazyce Python. Vzhledem k tomu, že celý server je napsaný v tomto jazyce, byl Pycharm snadnou volbou. V rámci diplomové práce byla využita školní licence tohoto programu.

[6] Společnost vyvíjející software pro programátory

4.9.2 Webstorm

Webstorm je další aplikací společnosti JetBrains. Tato aplikace se však zaměřuje hlavně na vývoj webových aplikací. To znamená, že i na jiné programovací jazyky. V rámci diplomové práce jsem převážnou část mobilní aplikace vyvíjel právě v aplikaci Webstorm.

4.9.3 Visual Studio Code

Posledním vývojovým prostředím je Visual Studio Code. To pochází od společnosti Microsoft a podporuje celou řadu programovacích jazyků [27]. Během vývoje mobilní aplikace jsem ze začátku používal i tuto aplikaci. Výhoda oproti konkurenci je ta, že je zdarma.

5 Implementace

V rámci implementace byly po dohodě s vedoucí práce vypracovány dvě základní komponenty systému a to server a mobilní aplikace. V následujících podkapitolách si rozebereme každou z těchto dvou komponent.

Jednou ze základních funkcionalit pro obě systémové části je způsob komunikace. Ta je zajištěna pomocí zabezpečeného protokolu HTTPS, který na rozdíl od obyčejného protokolu HTTP využívá asymetrickou kryptografii k navázání zabezpečeného spojení. Přenášený obsah pak není pro případné útočníky čitelný. Tím máme zajištěno, že veškerá přenášená data aplikace jsou v bezpečí. Samozřejmě vše závisí na důvěryhodnosti certifikačních autorit, které digitální certifikáty vydávají. Struktura přenášených dat má své specifické vlastnosti. Základním prvkem je adresa URL. V případě požadavků z mobilní aplikace na server se URL skládá ze základní části identifikující serverovou aplikaci, druhé části označované jako koncový bod (Endpoint) a parametrů. Základní část je vygenerována službou Google a identifikuje projekt vytvořený na GCP. Koncový bod se pak odvíjí od požadované akce, kde každá akce má svůj koncový bod. Volitelnou možností jsou parametry, které se přidají za dosud vytvořenou URL a znak „?“. Jednotlivé parametry jsou vždy typu klíč-hodnota a oddělují se mezi sebou znakem „&“. Současně s URL se určuje metoda posílaného požadavku. V rámci architektury REST máme čtyři základní metody označovány pojmem „CRUD“. Jsou to zkratky jednotlivých názvů operací v angličtině (Create, Read, Update, Delete). URL spolu s metodou tvoří základní vlastnosti, podle kterých se požadavky na serveru třídí. Dalším důležitým prvkem HTTP požadavků jsou použité hlavičky. Tu tvoří položky typu klíč-hodnota. Naše požadavky obsahují informace o tom, že posílaná data jsou ve formátu JSON. Pokud je u daného požadavku vyžadována autorizace, zasílá se v hlavičce JWT token identifikující uživatele a obnovovací token.

Odpověď serveru pak tvoří stavový kód a tělo odpovědi. Stavové kódy jsou rozděleny podle čísel, kde každá skupina čísel charakterizuje něco jiného. Konkrétní rozdělení čísel stavových kódů je následující:

- 1xx mají informační charakter
- 2xx značí úspěch
- 3xx označují přesměrování
- 4xx identifikují chybu na straně klienta
- 5xx identifikují chybu na straně serveru

Tělo odpovědi je ve formátu JSON a obsahuje určitá data, podle kterých aplikace ví, co dělat. Jedním atributem odpovědi je v našem případě status, říkající, jak byl požadavek zpracován (zpravidla jde o úspěch nebo neúspěch). Pokud jde o neúspěch následuje další atribut, kterým označuji konkrétní typ chyby. Tento atribut nazývám interní kód. Dalším atributem v případě neúspěchu je případný chybový výstup, pokud došlo k pádu aplikace. Mezi další atributy v těle odpovědi se řadí zpráva, která slouží pouze jako informační údaj k jakým akcím došlo. Pokud byl požadavek úspěšný a v odpovědi mají být vrácena nějaká data, následuje ještě atribut obsahující tato data. Mobilní aplikace pak dokáže na základě všech těchto údajů v odpovědi adekvátně reagovat.

5.1 Server

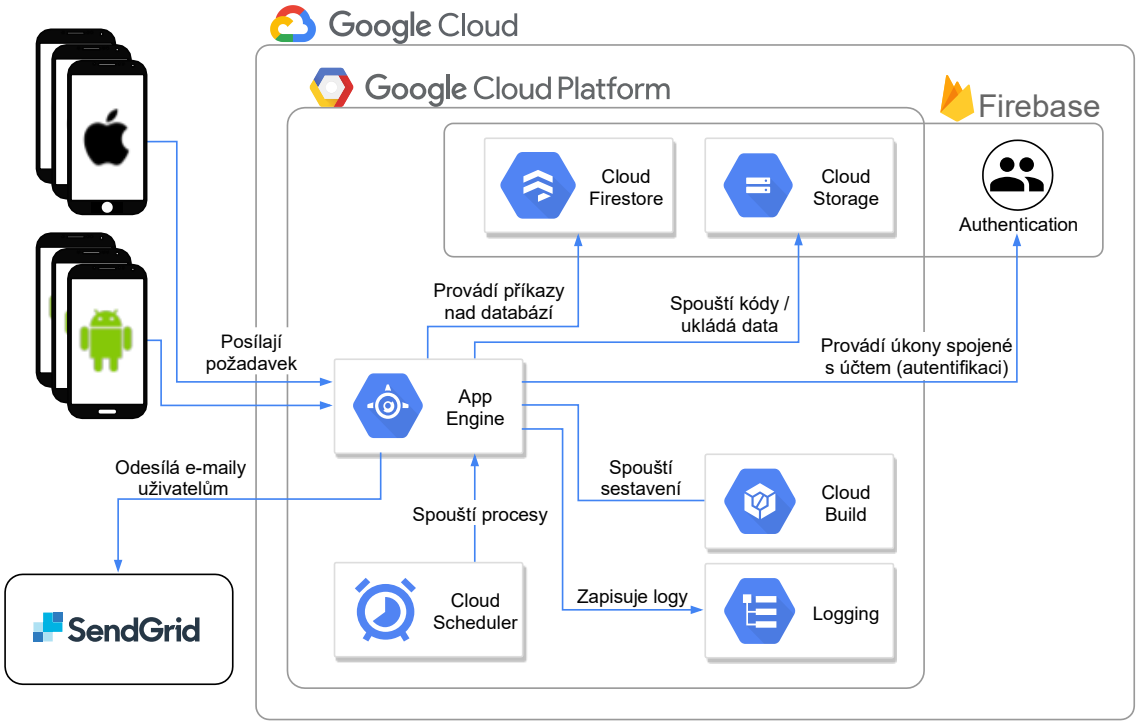
Celá serverová část běží na službě Google Cloud Platform (GCP). Jde o jedno z nejlepších cloudových řešení, které současný trh nabízí. Poskytuje velké množství nejrůznějších služeb s širokou mírou využití. V našem případě se využívá ročního tzv. „Free Tier“ účtu, který zahrnuje

několik základních služeb zdarma, nebo využití dalších služeb v omezeném rozsahu. Další vlastností tohoto účtu je částka 300 dolarů, za které lze využívat některé služby plnohodnotně, nebo některé již placené služby. Tato částka je platná po dobu 12 měsíců od dne založení účtu.

Pro diplomovou práci využíváme několika základních služeb, které však současně využívají automaticky dalších. Celkový počet využitých služeb je pak vyšší. Výhod použití cloudových služeb je celá řada. Mezi základní se řadí dostupnost, kterou se společnosti snaží držet těsně pod 100 %. Další výhodou je škálovatelnost. Tu lze aplikovat ať už na úložiště nebo na jiné běžící služby. Pokud se tedy zvýší vytížení služeb, Google navýší zdroje tak, aby zatížení zvládl. Podobně to platí pro databáze. Jakmile by uživateli došlo místo, lze kapacitu úložiště navýšit. Jakékoliv tyto změny či aktualizace by byly pro člověka velkou zátěží, pokud by si to měl spravovat sám. Ale v případě takovýchto cloudových služeb jde o záležitost několika „kliknutí“. Google tedy spravuje veškeré služby za uživatele a zajišťuje jejich bezproblémový chod. Další důležitou mírou je cena. Ta je v GCP řešena podle aktuálního využívání služeb. Uživatel tak platí pouze za to, co opravdu využívá. V neposlední řadě je velkou výhodou automatické zálohování, nebo uživatelská podpora. Díky těmto všem výhodám, které cloudové služby nabízí, se lze čistě soustředit na vývoj daného systému či aplikace.

V rámci diplomové práce jsem založil v GCP projekt. Tento projekt je sdílen mezi dalšími službami, jako je např. Firebase. Co je Firebase a co má společného s GCP je popsáno podrobněji níže. Celá serverová část je tedy spojena s tímto vytvořeným projektem. Takovýchto projektů lze ve „Free Tier“ účtu vytvořit 25.

Server diplomové práce se skládá z několika základních komponent (služeb). Hlavní z nich je služba App Engine zajišťující API a následné zpracování požadavků. Zde probíhá hlavní logika. Další důležitou částí je databáze, kterou v našem případě zajišťuje služba Firestore. Jde o NoSQL databázi. Veškeré nahrané soubory včetně kódů pro App Engine jsou uloženy ve službě Cloud Storage. Ta tedy slouží jako úložiště. Pro správu uživatelů využíváme službu Firebase Authentication a pro logování službu Cloud Logging. Jedinou komponentou, která nespadá pod Google Cloud je webová aplikace SendGrid. Ta zprostředkovává e-mailové služby jako např. posílání e-mailů. Celá architektura serveru včetně použitých služeb je znázorněna na obrázku níže (Obrázek 9). Jsou zde znázorněny vzájemné vztahy jednotlivých částí (služeb). Konkrétní použití a nastavení jednotlivých služeb je popsáno v podkapitolách.



Obrázek 9 - Struktura použitých služeb Google Cloud

5.1.1 App Engine

App Engine[7] je serverless[8] platforma určena pro provoz aplikací. V našem případě na této službě běží server v programovacím jazyce Python využívající framework Flask. Ten je velice rychlý a jednoduchý na použití. App Engine je zároveň prvním bodem při komunikace s koncovým uživatelem. Slouží tedy i jako API. Díky svým vlastnostem umožňuje velice snadné filtrování příchozích požadavků. Tyto požadavky jsou tříděny pomocí koncových bodů (Endpointů) a metod. Na obrázku níže (Obrázek 10) lze vidět ukázku tohoto filtrování. Každému filtru náleží vlastní tzv. „Handler“, který požadavek zpracuje dále. Každý má tedy na starosti rozdílné typy požadavků.

[7] https://cloud.google.com/appengine
[8] Serverless znamená, že služba běží pouze tehdy pokud ji někdo vyžaduje

```
38  @app.route('/firebase/sign_in', methods=['POST'])
39  def firebase_sign_in():
40      return sign_in(request)
41
42
43  @app.route('/firebase/password_reset', methods=['POST'])
44  def firebase_password_reset():
45      return password_reset(request)
46
47
48  @app.route('/firebase/password_change', methods=['POST'])
49  def firebase_password_change():
50      return password_change(request)
51
52
53  @app.route('/firebase/delete_account', methods=['DELETE'])
54  def firebase_delete_account():
55      return delete_account(request)
```

Obrázek 10 - Ukázka filtrování příchozích požadavků na serveru

Úkolem Handlerů je zpracovat požadavek a vrátit adekvátní odpověď. Tento proces zahrnuje analýzu příchozího požadavku, zda splňuje všechny náležitosti. Tyto náležitosti se mohou mezi Handlery lišit podle toho, které požadavky obstarávají. Jednou z náležitostí je kontrola adresy a přítomných parametrů. To je často spojeno s požadavky s metodou GET. U akcí, které vyžadují, aby byl uživatel přihlášen, se kontroluje přítomnost JWT tokenu. Jak token vypadá a co obsahuje, je popsáno níže v podkapitole pojednávající o správě uživatelů. Pokud je token přítomen, nachází se v hlavičce požadavku pod klíčem „Authorization“. Další věc, kterou Handlery kontrolují, je obsah těla požadavku. To se děje především u požadavků, které mají metodu POST, nebo UPDATE. V těle těchto požadavků se často nachází data, která je třeba uložit do databáze. Ovšem i strukturu těchto dat je potřeba kontrolovat, tak aby nedocházelo v databázi k nekonzistencím. Všechny tyto kontroly jsou prováděny v jednotlivých Handlerech. Pokud je požadavek detekován jako nevalidní (např. chybí určitá data, nebo uživatel není přihlášen), je vytvořena patřičná odpověď a ta následně vrácena. Klient pak ví, kvůli čemu byl požadavek zamítnut. Vytváření těchto odpovědí je funkční i pro případ, pokud by selhala některá funkce serveru. Např. pokud by došlo k chybě při dotazování do databáze, nebo by databáze vůbec nebyla dostupná. V těchto případech by se vytvořila odpověď se stavovým kódem 5xx identifikující chybu na straně serveru. Handlery tedy pracují i s databází. Konkrétně je však tento proces obstarán mezivrstvou, která jediná má přímý přístup do databáze. Tato mezivrstva se nazývá „DAO“ a jde o návrhový vzor jež poskytuje rozhraní (interface) k databázi. Zkratka DAO vychází z anglického Data Access Object. Díky tomuto přístupu máme zajištěno, že veškerá práce s databází je spravována jednotně právě přes tuto vrstvu. V rámci DAO vrstvy se nachází několik souborů, kde každý obstarává přístup k jiné kolekci dat. Pojmem přístup se rozumí, že zajišťuje různé operace nad danou kolekcí (např. čtení, zápis, nebo mazání). Jakmile Handler požadavek zpracuje, vytvoří odpověď a vrátí ji na API. To pak pošle odpověď zpět klientské straně.

Zmiňovaný server zpracovávající požadavky byl vytvářen a testován lokálně na PC. Jakmile se ověřila bezchybnost funkcionalit (často jsem testoval funkčnost pomocí aplikace Postman), nahrál jsem zdrojové kódy na Google Cloud (konkrétně do služby App Engine). Nahrávání jsem prováděl z terminálu (příkazové řádky) pomocí příkazu „gcloud app deploy“. Tím se zahájil proces nahrávání souborů do cloudu a následného spuštění. Všechny problémy při tomto procesu se logují pomocí služby Cloud Build. Zde pak bylo snadné nalézt důvody proč se akce nezdařila.

5.1.2 Firebase

Jedná se o platformu pro vývoj mobilních a webových aplikací. Nabízí např. real-time databázi, náhledy statistik nebo zasílání tzv. push notifikací. Pro nás je důležité, že poskytuje službu pro správu uživatelských účtů, kterou využíváme. Platforma Firebase byla založena v roce 2011 stejnojmennou společností a v roce 2014 byla koupena společností Google [28]. Od té doby se postupně a úspěšně začleňuje do struktury Google Cloud Platform. Tím je myšleno, že Firebase a GCP sdílí některé služby. Těmi jsou třeba databáze Firestore, Cloud Functions umožňující událostmi řízený běh (event-driven) nebo uložiště dat Cloud Storage [29] [30].

Firebase [31] však stále nabízí služby, které v GCP nenajdeme. Ty se zaměřují hlavně na vývoj již zmiňovaných mobilních či webových aplikací. Kromě sdílených služeb je také sdílen uživatelský účet a projekt. Lze tedy vytvořit projekt v klasickém GCP a ve Firebase do něho později přidat další funkcionalitu. GCP projekt je tedy i Firebase projekt. V rámci diplomové práce bylo z platformy Firebase využito služby Authentication pro správu uživatelů a administrace databáze Firestore. Databázi by bylo možné spravovat také přímo z GCP, ale Firebase nabízí lepší grafické rozhraní.

5.1.3 Databáze

Jak již bylo zmíněno, databázi zajišťuje služba Firestore. Tu lze spravovat v prostředí GCP nebo Firebase. Celá databáze je NoSQL a je tedy dokumentová. Je velice podobná databázovému systému MongoDB, který byl uveden při návrhu, jako ideální způsob pro ukládání dat. Tedy i zde jsou data uložena ve formátu JSON. Díky tomu pak mají Handlery snazší práci při tvorbě odpovědí na požadavky uživatelů.

V databázi se nachází celkem 6 kolekcí dat (země, města, události jednotlivých obcí, uživatelská data, zpětné vazby uživatelů a hlášení od uživatelů směrem k obcím). Každá kolekce následně obsahuje řadu dokumentů, v nichž jsou již naše data. Pro zachování unikátnosti záznamů má každý dokument unikátní ID (identifikátor). Tyto ID jsou automaticky generovány službou Firestore při vytváření dokumentů. Bylo by však možné si tyto ID nastavovat ručně. To se děje pouze u kolekce vztahující se k uživateli. Zde nastavuji ID stejné, jako je ve službě Authentication u daného uživatele. Schéma databáze a provázání kolekcí lze vidět v kapitole o návrhu (Obrázek 7).

Databáze se používá prostřednictvím služby App Engine popsané dříve. To je zajištěno pomocí knihovny od služby Firestore určené pro tyto účely. Pro správný chod knihovny (aby věděla, do které databáze má přistupovat) je potřeba správného nastavení. To nám poskytne služba Firestore při tvorbě jedné z možných aplikací. Na základě zvolené aplikace se vygeneruje patřičný konfigurační soubor. Zde pracuji s možností webové aplikace. Díky této aplikaci se nám tedy zpřístupní veškeré konfigurační údaje potřebné pro přístup do databáze. Zmíněné údaje jsou následně použity ve službě App Engine. Nevýhodou oproti klasickým SQL databázím je např. způsob aktualizace dat u již existujícího dokumentu. V případě NoSQL databází je potřeba získat celý dokument, upravit ho a následně celý vrátit zpět do databáze. Nejde tedy změnit jeden atribut v dokumentu bez toho, aniž bychom celý dokument znovu uložili. Během používání se vyskytla potřeba vyhledávaná data řadit podle názvu nebo data. Aby se tyto řadící procesy daly vykonat rovnou při dotazování do databáze, bylo potřeba vytvořit tzv. indexy. Díky těm je takovéto vyhledávání možné a také rychlejší. Bohužel jednou z nevýhod databáze je nemožnost vyhledávání slova uvnitř jiného textu. Naštěstí tento proces není potřeba často. Řešením by bylo migrovat část databáze do relační, která již podobně složitější vyhledávání podporuje. Zabezpečení databáze je zajištěno pomocí sady pravidel definující přístupy k jednotlivým kolekcím dat. Současně s tím je databáze „schovaná“ za Handlery, které také kontrolují, zda je

uživatel oprávněn danou operaci provést. Vzhledem k požadavkům GDPR je v mobilní aplikaci zajištěna funkce pro smazání všech uživatelských dat z databáze.

5.1.4 Správa uživatelů

Další použitou službou je Firebase Authentication. Ta umožňuje správu uživatelských účtů včetně jejich souvisejících funkcionalit. Těmi jsou scénáře pro obnovu hesla při jeho zapomenutí, změna hesla, či samotné přihlášení nebo registrace. V mobilní aplikaci je možnost pro přihlášení pomocí e-mailového účtu a hesla. Služba Authentication však nabízí i další možnosti přihlášení přes účty třetích stran, jako jsou např. Facebook, Twitter, Github, Yahoo, Microsoft a další. Při registraci je vyžadována e-mailová adresa a heslo. Heslo musí splňovat bezpečnostní kritérium, jímž je délka hesla minimálně 8 znaků a alespoň 2 sady různých znaků. Po odeslání požadavku je potřeba e-mailovou adresu ověřit. Služba odešle na danou adresu e-mail, ve kterém je potvrzovací odkaz. Po jeho kliknutí dojde ve službě Authentication k aktivaci daného účtu. V opačném případě zůstává účet neověřený a nelze se na něj přihlásit.

Při úspěšném přihlášení služba vygeneruje tzv. ID token, pomocí kterého lze uživatele identifikovat. Jde o JWT token popsaný v kapitole použitých technologií. Jeho platnost trvá jednu hodinu a poté expiruje. Konkrétní strukturu JWT tokenu vygenerovaného službou Authentication lze vidět v tabulce níže (Tabulka 3). Jeho dekódování bylo provedeno pomocí webové aplikace JWT.io[9]. Kromě ID tokenu je vygenerován také tzv. Refresh token. Pomocí něho lze vygenerovat nový ID token pro daného uživatele. To nám umožní nadále pracovat s aplikací bez toho, aniž bychom se museli znovu přihlašovat. Tyto vygenerované údaje jsou poslány uživateli v odpovědi po úspěšném přihlášení. Uživatel si údaje uloží a následné požadavky na server, kde je potřeba autorizace, provádí se svým ID token. Tím ho server identifikuje a dané operace povolí. V případě, že uživatelův ID token expiroval, podívá se server, zda byl v požadavku také Refresh token. Pokud ano server podle něj nechá vygenerovat nový ID token, který vrátí uživateli se speciálními příznaky. Těmi jsou např. stavový kód 401, který říká, že uživatel nebyl autorizován (anglicky Unauthorized). Do odpovědi serveru se přidá také již nový ID token. Mobilní aplikace následně detekuje, že na serveru došlo k problému s autorizací právě podle stavového kódu 401. To zapříčiní, že se aplikace podívá, zda je v odpovědi i nový ID token. Pokud ano, uloží si ho a odešle původní požadavek znovu (již s novým ID tokenem). Jediný problém představuje proces změny hesla. Jakmile uživatel změní své heslo, služba Authentication nechá automaticky expirovat jeho Refresh token. Z tohoto důvodu je potřeba, aby se uživatel znovu plnohodnotně přihlásil a obdržel tak všechny tokeny validní.

[9] Webová aplikace, kde lze dekódovat JWT token (https://jwt.io/)

Tabulka 3 - JWT token vygenerovaný službou Firebase Authentication

<table>
<tr><th>Zakódovaný JWT token</th><th>Dekódovaný JWT token</th></tr>
<tr><td rowspan="3">eyJhbGciOiJSUzI1NiIsImtpZCI6IjVlOWVlOTdjODQwZjk3ZTAyNTM2ODhhM2I3ZTk0NDczZTUyOGE3YjUiLCJ0eXAiOiJKV1QifQ.eyJpc3MiOiJodHRwczovL3NlY3VyZXRva2VuLmdvb2dsZS5jb20vc2VjcmV0LWZvb3RpbmctMjY4NjEwIiwiYXVkIjoic2VjcmV0LWZvb3RpbmctMjY4NjEwIiwiYXV0aF90aW1lIjoxNTg3NjYxNzA4LCJ1c2VyX2lkIjoiNmp0dUpDbktYZmh5UG5NVndUMXBFS0RWVld6MSIsInN1YiI6IjZqdHVKQ25LWGZoeVBuTVZ3VDFwRUtEVlZXejEiLCJpYXQiOjE1ODc2NjE3MDgsImV4cCI6MTU4NzY2NTMwOCwiZW1haWwiOiJqZWxpbWtyYUBnbWFpbC5jb20iLCJlbWFpbF92ZXJpZmllZCI6dHJ1ZSwiZmlyZWJhc2UiOnsiaWRlbnRpdGllcyI6eyJlbWFpbCI6WyJqZWxpbWtyYUBnbWFpbC5jb20iXX0sInNpZ25faW5fcHJvdmlkZXIiOiJwYXNzd29yZCJ9fQ.I2tUchcgmFqUOWP95Yc6hG60n-DtdMArFRJcdwUs1hIapfFDELWkyt2GU94PdElsvbcwObhASwQYx4hIZqy3ED5DZ-RgcEzHtX4jmFP1bUD52INqSgxEnlZu-56ws5yTKZ0gaSCZxcdo1aEYQxuWSlSEh7x8Q5X5TPmXBYCTJucZmpPb1ZBa1ly2yXu_tIs_cWr7Owr2e9Vzzg1GRA6R1NAdS_e6PacRSkd_6D4TQAqUwBdQ-5QtxBCaULq3eMTVSgC5XlDZG12nsbVJazFV__8vi4HIov9tHcTH-eZmJJjUCsbilWqfZTn6sAIzWEn_vB_gEihbZXQ3u-omTntwOA</td><td>Hlavička (Header)
{
"alg": "RS256",
"kid": "5e9ee97c840f97e0253688a3b7e94473e528a7b5",
"typ": "JWT"
}</td></tr>
<tr><td>Data (Payload)
{
"iss": "https://securetoken.google.com/secret-footing-268610",
"aud": "secret-footing-268610",
"auth_time": 1587661708,
"user_id": "6jtuJCnKXfhyPnMVwT1pEKDVVWz1",
"sub": "6jtuJCnKXfhyPnMVwT1pEKDVVWz1",
"iat": 1587661708,
"exp": 1587665308,
"email": "jelimkra@gmail.com",
"email_verified": true,
"firebase": {
"identities": {
"email": [
jelimkra@gmail.com
]
},
"sign_in_provider": "password"
}
}</td></tr>
<tr><td>Podpis (Signature)
RSASHA256(
base64UrlEncode(header) + "." +
base64UrlEncode(payload),
[Public Key or Certificate],
[Private Key]
)</td></tr>
</table>

Co se týče procesu obnovy hesla, ten probíhá pomocí zadání e-mailové adresy v mobilní aplikaci a následného odeslání na server. Ten na danou adresu pošle e-mail s odkazem pro obnovení. Uživatel na odkaz klikne a následně se mu zobrazí formulář pro zadání nového hesla.

Veškeré e-maily posílané serverem jsou zprostředkovány pomocí služby Authentication. Během vytváření serveru jsem měl však s touto službou problémy kvůli mé neznalosti. Díky tomu jsem našel jinou službu, pomocí které jsem tyto e-maily posílal. Jednalo se o službu SendGrid, kterou Google Cloud uváděl jako jednu z možností pro zasílání e-mailů. Na jejich webových stránkách jsem si založil účet. Následně jsem zjistil konfigurační údaje, které jsem zanesl do svého serveru, tak aby mohl službu SendGrid používat. Vše fungovalo velmi dobře a později se mi podařilo zprovoznit i odesílání ze služby Authentication. Tím mám tedy dvě funkční možnosti pro odesílání e-mailů. Výhoda SendGrid je v nastavení vzhledu a obsahu e-mailů. Ten si lze nastavit zcela libovolně na rozdíl od služby Authentication, která nastavení vzhledu prakticky neumožňuje. Authentication také neumožňuje zasílat libovolné e-maily netýkající se uživatelského účtu. Nelze tak např. uživatele libovolně kontaktovat či informovat. Vzhledem k tomu, že to zatím není od aplikace vyžadováno, odesílání e-mailů je řešeno službou Authentication, aby vše zůstalo u služby Google. Pokud by se však nároky na aplikaci změnily a bylo by potřeba uživatele informovat, nebyl by problém zapojit službu SendGrid.

5.1.5 Generování aktualit z webových stránek

Nyní se dostáváme k jedné z nejdůležitějších částí serveru. Tím je generování obecních aktualit z jejich webových stránek. Celý proces je založený na analýze webové stránky dané obce a následnému nalezení obecních aktualit či událostí. Následující algoritmy a postupy byly vypracovány především pomocí knihovny Beautiful Soup (popsané výše v kapitole použitých technologií). Pomocí té jsem detekoval potřebné HTML elementy na stránkách a následně zkoumal, které třídy, nebo identifikátory element obsahuje. Další užitečnou vlastností byl obsah samotného textu daného elementu. Další výhodou knihovny je, že dokáže vyhledávat elementy, které jsou např. potomkem nějakého jiného. To také velice usnadňuje vyhledávání. Obecně se tomuto postupu (analýze webových stránek) říká „Web scraping".

Prvním krokem bylo zjištění informací o obcích včetně jejich webových stránek. Původním plánem bylo získat data z Českého statistického úřadu[10]. Ten ale neposkytoval data v ideálním formátu, tak aby je bylo snadné zpracovat a zanést do databáze. Konkrétně bylo vše v tabulkách Excel a data nebyla kompletní. Další možností byly stránky Města a obce online[11], které poskytují informace o velkém množství obcí v ČR. Bohužel však neposkytují informace o webových stránkách. Nakonec jsem vše vyřešil přes stránky Wikipedie[12]. Zde jsem nalezl seznam všech obcí ČR včetně jejich základních údajů i webových stránek. Následně jsem napsal skript v Pythonu, který pomocí knihovny Beautiful Soup zpracoval všechny nalezené obce a vytvořil jejich záznam do databáze. Principem skriptu bylo nalézt na stránce s obcemi všechny odkazy vedoucí na okresy. Následný blok údajů pak obsahoval všechny obce v daném okrese. Tyto obce byly odkazy, takže kliknutím na ně došlo k zobrazení podrobností dané obce. Poté stačilo v pravé části stránky detekovat informační blok dat z něhož byly „vytaženy" základní údaje o obci. Na základě těchto údajů byla vytvořena instance obce a uložena do databáze. Obdobný proces byl aplikován na vytvoření instancí zemí. I ty byly generovány z Wikipedie [32], která poskytuje seznam všech zemí světa včetně jejich základních údajů. Země byly do databáze přidány z důvodu, aby se aplikace dala snadněji rozšířit do zahraničí.

Jakmile jsem měl obce uloženy v databázi následoval proces získávání jednotlivých aktualit (událostí). Zpočátku to vypadalo jako nemožné, protože webové stránky obcí byly ještě více rozdílné, než jsem čekal. Postupem času jsem se však i s těmito problémy vypořádal. Celý proces generování aktualit se dá rozdělit na několik fází. První fází je nalezení podstránky obce týkající se aktualit. Další fází je vyhledání konkrétních aktualit, které jsou samy podstránkou webu na nalezené stránce z fáze první. Třetí fází je rozbor stránek s konkrétními aktualitami z fáze druhé. To zahrnuje nalezení bloku dat na stránce, ve kterém se již nachází údaje o aktualitě. Poslední fází je rozbor bloku dat z fáze třetí. Jde tedy o nalézání všech potřebných atributů zjišťovaných u aktualit. Závěrečným úkonem je vytvoření jejich instancí a následné uložení do databáze. Všechny tyto fáze si nyní rozebereme podrobněji.

První fází tedy hledáme takovou podstránku obce, na níž je s největší pravděpodobností seznam aktualit. Na hlavní stránce obce tak hledáme odkazy. Po určitém zkoumání struktur webových stránek byl sestaven slovník klíčových slov, který později použijeme jako heuristickou funkci pro vybrání nejlepšího odkazu. Současně s tím byl vytvořen slovník těch slov, které naopak nechceme. Odkazy jsou v HTML kódu označeny značkou `<a></a>`. Pomocí knihovny Beautiful Soup jsou vyhledány všechny odkazy na úvodní stránce obce. Ty jsou následně pomocí

[10] Webové stránky Českého statistického úřadu (https://www.czso.cz/csu/czso/domov)
[11] Webové stránky *Města a obce online* poskytující informace o obcích (https://mesta.obce.cz/)
[12] Webová stránka odkud byla generována data obcí (https://cs.wikipedia.org/wiki/Seznam_obc%C3%AD_v_%C4%8Cesku)

heuristické funkce ohodnoceny a seřazeny. Odkazy vedoucí na jinou stránku, než je stránka obce, jsou filtrovány podle domény. Poté je vybrán odkaz s nejvyšším ohodnocením a předán dál.

Nyní začíná proces vyhledávání odkazů vedoucí na jednotlivé aktuality. V tuto chvíli máme k dispozici webovou stránku (z fáze první), na které by měl být s největší pravděpodobností seznam aktualit. V rámci práce uvažujeme, že každá aktualita má svoji vlastní stránku na webu obce. Tím vyhledáváme opět odkazy. Nyní však začínají komplikace, neboť ne všechny stránky obcí mají své aktuality na podstránce s aktualitami. Proto musíme brát ohled na daleko širší spektrum možností, které odkazy jsou ty správné a které ne. Kvůli tomu jsem vytvořil další slovník, který nám pomůže tyto odkazy rozlišit. Jednou možností však je, že aktualita je na podstránce stránky s aktualitami. Tím máme téměř jisté, že takovýto odkaz chceme, protože s velkou pravděpodobností vede na aktualitu. Velká část obcí však tuto podmínku nesplňuje a aktuality mají někde jinde. V tuto chvíli musíme hledání odkazů více specifikovat. A např. tak, že většina těchto aktualit se nachází v nadpisech. Tyto nadpisy se označují v HTML kódu značkou `<h[1-5]></h[1-5]>`. Pokud se tedy odkaz vyskytuje pod tímto nadpisem je to indicie, že vede pravděpodobně na aktualitu. Výsledkem těchto operací je seznam odkazů, který postupuje do fáze třetí. Na následujícím obrázku (Obrázek 11) lze vidět ukázku dobře napsaného kódu s odkazem na aktualitu.

Obrázek 11 - Ukázka HTML kódu stránky s odkazem na aktualitu

V třetí fázi máme k dispozici seznam odkazů, kde by měl každý představovat jednu aktualitu. Tím začíná vyhledávání bloku dat, ve kterých se nalézají informace o aktualitě. Po zkoumání jednotlivých stránek byl vytvořen slovník klíčových slov. Ten následně slouží jako heuristická funkce při vyhodnocování jednotlivých elementů stránky. Podle shody s klíčovými slovy ze slovníku se jednotlivým elementům udělují body. Ve většině případů se aktualita nachází v bloku dat označeného HTML značkou `<div></div>`. Tyto elementy jsou vyhodnoceny heuristickou funkcí a následně seřazeny. Heuristická funkce se zaměřuje na hodnoty tříd, které může element `<div>` obsahovat. Stejně tak může obsahovat identifikátor, podle kterého lze usoudit, zda se vztahuje k aktualitě. Jakmile jsou možnosti ohodnoceny a seřazeny, vybere se ten s největší vahou.

V poslední fázi následuje rozbor nalezeného bloku dat. To zahrnuje nalezení všech dostupných atributů, které nás u aktualit mohou zajímat. Tento proces se odehrává pouze v bloku dat, který nám vyplynul z fáze třetí. Hlavním atributem, který nás zajímá je nadpis. Ten bývá často již v samotném odkazu, kterým jsme se na stránku s aktualitou dostali. Každopádně to se ještě jednou kontroluje přímo ve vybraném bloku dat pomocí HTML značek pro nadpis.

Dalším prvkem, který zjišťujeme je samotný obsah aktuality. Jde o text, který bohužel ne vždy je u aktualit přítomen. Pokud však je, nalézá se v různých podobách. Nejčastěji se nachází v elementu `<div>`. Proto existuje slovník klíčových slov zaměřující se na detekci těchto míst s obsahem. Jakmile jsou tyto elementy ohodnoceny, opět vyberu ten s nevyšším ohodnocením a ten dál zkoumám. Nyní jsme již velice blízko cílenému obsahu. Ten však může být ještě ukryt pod dalšími elementy, jako jsou nadpisy či odstavce `<p>`. Po prozkoumání všech těchto elementů je sestavena jedna velká zpráva představující obsah zkoumané aktuality. Zpráva je současně upravena tak, aby neobsahovala zbytečně velké množství tzv. bílých znaků jimiž jsou např. mezery, tabulátory a nové řádky. Tím je zpráva značně pročištěna.

Dalším atributem vyhledávaným v aktualitě je datum zveřejnění. To se nejčastěji vyskytuje v HTML prvcích jako jsou `<div>`, `<span>`, nebo `<time>`. Pro identifikaci data je vytvořený další slovník klíčových slov. Pokud tedy některý ze zmiňovaných elementů obsahuje slovo ze slovníku, provádíme u něj podrobnější analýzu. Ta zahrnuje zjišťování, zda text elementu obsahuje datum. Tento proces je obstarán speciální funkcí. Vzhledem k různým možnostem zapsání data je to nelehký úkol. Naštěstí pomocí správného použití regulárních výrazů se dá datum celkem snadno určit. Ukázku regulárních výrazů detekující datum lze vidět na obrázku níže (Obrázek 12). Funkce vykonávající tento proces dokáže také detekovat datum, ve kterém je měsíc napsán slovy. Jakmile je datum detekováno, je převedeno do formátu den, měsíc a rok (*%d-%m-%Y*). V rámci aktualit se vyhledává také datum konání. To je zpravidla vyhledáváno v obsahu, nalezeného dříve stejnou funkcí, jako pro datum zveřejnění.

```
\d{1,2}[. -\/]{1,2}\d{1,2}[. -\/]{1,2}\d{4}
\d{1,2}[. -\/]{1,2}[A-Za-z]{4,8}[. -\/]{1,2}\d{4}
```

Obrázek 12 - Ukázka regulárních výrazů pro datum

Mezi další vyhledávané atributy seřadí autor události. Na jeho vyhledání je opět použit slovník klíčových slov, který nám lépe umožní detekovat místo, kde se jméno nachází. V aktualitách se také často vyskytují přílohy. Proto i zde bychom je neměli opomenout. Algoritmus v daném bloku dat (z fáze třetí) hledá odkazy, protože veškeré přílohy jsou většinou nahrány někde na stránkách obce. Tyto odkazy jsou následně podrobeny testu, zda obsahují některá klíčová slova. Tato slova jsou ve značné míře koncovky možných souborů, jako jsou PDF, DOC a další. Jakmile je některý takový odkaz nalezen přidá se jeho adresa URL do seznamu nalezených příloh.

V rámci aktualit se také přiřazují kategorie, do kterých spadají. Ty se určují na základě nalezeného nadpisu. Prozatím algoritmus detekuje čtyři typy kategorií. První kategorie označuje důležité informace. Další kategorie označuje kulturní akce a předposlední se věnuje sportovním aktivitám. Poslední kategorie označuje události týkající se počasí. Pro všechny tyto kategorie existuje slovník klíčových slov, podle kterého se algoritmus rozhoduje, zda událost kategorií označí či ne.

Jakmile jsou výše uvedené atributy posbírány, provede se jejich celkové vyhodnocení. To probíhá tak, že každý atribut má určitou váhu, pokud se nám ho podařilo v aktualitě nalézt. V případě, že součet těchto vah překročí jistou hranici považujeme aktualitu za důvěryhodnou. Následně vytvoříme její instanci a uložíme do databáze. Ukládání do databáze však předchází proces kontroly, kdy detekujeme, zda již daná aktualita existuje. To je v současné chvíli řešeno kontrolou názvů událostí z poslední doby (konkrétně posledních 10 aktualit). Pokud jsou názvy shodné alespoň na 95 %, jsou vyhodnoceny jako jedna a ta samá aktualita. Poté dojde k aktualizaci, a nikoliv vytvoření nové aktuality. Kontrola podobnosti názvů je prováděna pomocí

knihovny „difflib“. Ta je vestavěná v Pythonu, takže ji není potřeba externě stahovat, jako např. knihovnu „Levenshtein“ poskytující podobné funkce. Jakmile je aktualita uložena do databáze, celý proces končí.

Při zpětném ohlédnutí lze obce rozdělit do několika kategorií podle kvality kódu svých webových stránek. První a nejlepší kategorie podchycuje kvalitně navržené stránky s jasnou strukturou a čitelným kódem. V druhé kategorii jsou obce s hůře čitelným kódem, kde se musí algoritmus více „snažit“, aby aktuality našel. Na rozdíl od následujících kategorií je algoritmus pro oba zmíněné typy úspěšný. Další kategorie obsahuje vizuálně dobře navržené stránky, ale kvůli špatně napsanému kódu je velmi obtížné aktuality nalézt. To je často zapříčiněno tím, že stránky nemají žádnou strukturu, nebo mají aktuality pouze pomocí příloh. Mezi další nepodchycené obce spadají ty s webovými stránkami, které jsou téměř neudržované a všechny aktuality mají většinou na jedné stránce bez možnosti zobrazení podrobností. V těchto případech jsou webové stránky většinou nečitelné nejen po zdrojové stránce, ale i uživatelské. Poslední kategorií, u které algoritmus na první pohled nic nenajde jsou obce, které sice mají podstránku týkající se aktualit, ale bohužel zde žádné nemají. S tímto problémem jsem se setkal poměrně často. Celkově lze říci, že pokud by obce zavedly určitý řád (protokol) do svých webových stránek, výše popsaný algoritmus by jejich aktuality nalezl a byly by součástí tohoto systému. Úspěšnost algoritmu se pohybuje okolo 55 %. To je slušné číslo, pokud si uvědomíme různorodost jednotlivých webových stránek. Občas se stane, že algoritmus události detekuje špatně (jedná se většinou o výjimky), i tak ale zůstává velice spolehlivý. V tabulce níže (Tabulka 4) jsou znázorněny výsledky pro určité okresy ČR, na kterých byl algoritmus spuštěn. V počtu obcí, u nichž se podařilo aktuality nalézt, nejsou zahrnuty ty obce, u kterých byla stránka s aktualitami nalezena, ale neobsahovala žádné příspěvky. Jde tedy čistě o obce, u kterých se podařilo nějaké aktuality nalézt a úspěšně je zpracovat. Z důvodu omezeného úložiště na Google Cloud bylo v rámci diplomové práce pravidelné generování dat prováděno pouze pro okres Beroun. To zajišťuje služba „cron“. Pomocí jejího konfiguračního souboru „cron.yaml“ lze nastavit, co se kdy má spustit. V našem případě se spouští algoritmus pro získání aktualit. Tento úkon se provádí pravidelně v 17:00 GMT (Greenwich Mean Time - greenwichský střední čas).

Tabulka 4 - Počet úspěšně zpracovaných obcí v okresech

Okres	Počet obcí v okrese	Počet obcí, u nichž se podařilo aktuality nalézt	Celková úspěšnost v daném okrese v %
Beroun	85	54	63,5
Domažlice	84	41	48,8
Hradec Králové	104	57	54,8
Jeseník	24	10	41,6
Jihlava	123	57	46,3
Karlovy Vary	56	32	57,1
Kladno	100	64	64
Mladá Boleslav	120	51	42,5
Olomouc	98	66	67,3
Opava	77	42	54,5
Prostějov	97	68	70,1
Příbram	120	60	50
Třebíč	167	80	47,9
Zlín	89	59	66,2
Znojmo	144	73	50,6
Celkem	**1488**	**814**	**55,01333**

Celý proces generování dat pro okres Beroun trval na osobním počítači přibližně 5 minut a 30 vteřin. To je způsobeno detailním procházením jednotlivých webových stránek, kde se doba

zpracování odvíjí i od času odezvy jednotlivých poskytovatelů, u kterých obce hostují své stránky. Pokud bychom uvažovali pokrytí celé ČR, trval by tento proces o mnoho déle. A to neuvažujeme okolní státy, pokud by byla aplikace v budoucnu rozšířena. Z těchto důvodů jsem se rozhodl pro paralelizaci kódu generování dat. Využil jsem knihovnu „concurrent.futures“ z programovacího jazyka Python. Proces paralelizace je následující. Pomocí knihovny jsem vytvořil tzv. „ThreadPoolExecutor“, který slouží, jako seznam možných vláken. Generování událostí pro jednotlivé obce je pak rozdistribuováno mezi jednotlivá vlákna, čímž se proces velice urychlí. Jakmile vlákno dokončí generování dat, ujme se další obce. Takto proces pokračuje, dokud se nezpracují všechny obce. Při použití 4 vláken se proces generování na osobním počítači zrychlil pro okres Beroun na 2 minuty. Podobného zrychlení bylo dosaženo i ve službě App Engine. Bohužel kvůli omezeným možnostem zdrojů jsem již nenavyšoval počet použitých vláken. Ovšem při reálném nasazení to tato technologie podporuje.

5.2 Mobilní aplikace

Mobilní aplikace slouží jako klientská část vyvíjeného systému. Je postavena na programovacím jazyce JavaScript. Mezi jeho nejdůležitější použité frameworky patří React a React Native. React slouží jako prostředek pro vytváření grafického rozhraní, kde jeho základními prvky jsou tzv. komponenty, pomocí kterých je aplikace sestavena. React Native je knihovna umožňující běh jak na zařízeních s platformou Android, tak iOS. Zároveň poskytuje prvky, které se pro obě platformy vykreslí různě. Další důležité knihovny jsou popsány v jednotlivých podkapitolách níže. Během vývoje mobilní aplikace jsem narazil na nevýhody použití React Native. Konkrétně šlo o zastavení podpory různých komponent. To znamená, že do budoucna nelze s těmito komponentami počítat jako s perspektivními. Z toho důvodu jsem byl nucen vymyslet alternativu, jak tuto komponentu nahradit.

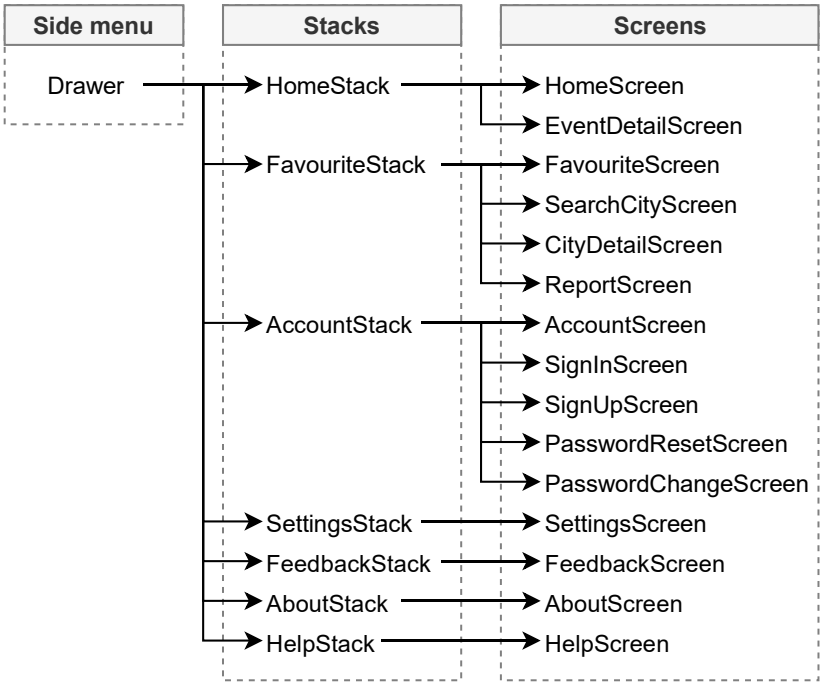
Během zkoušení aplikace bylo zjištěno, že běh na zařízeních s Android 4.x není zcela bezchybný. Proto je aplikace podporována od verze 5.x pro zařízeních Android. Pro zařízení s operačním systémem iOS je podporována verze 9.0 a vyšší. Těmito restrikcemi je pokryto 94,1 % všech zařízeních Android (zjištěno pomocí programu Android Studio) [33] [34] a necelých 100 % zařízení s platformou iOS [35] [36]. Tyto údaje jsou platné ke 30. 4. 2020.

V rámci aplikace si lze založit uživatelský účet. S tím je následně spojeno veškeré nastavení aplikace včetně oblíbených obcí a jejich zvolených kategorií či samotného nastavení aplikace. To ulehčuje uživateli přechod mezi různými zařízeními. Jakékoliv zásadní změny jsou odesílány na server, kde se ukládají k uživatelskému účtu. Současně aplikace nabízí možnost pro obnovení hesla, pokud ho uživatel zapomene, či samotnou změnu hesla. Nezbytnou podmínkou je pak možnost smazání účtu a všech souvisejících dat. K této akci je zapotřebí napsat do formuláře celou e-mailovou adresu uživatele, aby nedošlo ke smazání omylem. Pokud by uživatelův autorizační token nebyl platný, bude ještě nucen se přihlásit.

Současně s vyvíjením aplikace jsem narazil na problém se samotnou velikostí mobilní aplikace pro zařízení Android. Ta byl původně okolo 60 MB. Díky určitému nastavení při jejím vytváření došlo ke snížení. Velký podíl na úbytku velikosti má rozdělení vytvořené aplikace zvlášť pro jednotlivé typy procesorů. Pro platformu iOS lze aplikaci vyzkoušet pouze s dočasným certifikátem, po jehož uplynutí se aplikace nespustí. Platnost dočasného certifikátu je přibližně týden. Tento problém by šel vyřešit vlastním certifikátem pro jehož vytvoření je však zapotřebí mít aktivovaný vývojářský účet u Applu. Ten stojí okolo 99 dolarů a v rámci diplomové práce je zbytečné ho zakládat a platit. Pro testování na zařízeních s iOS postačily dočasné certifikáty. Pro zařízení s Android se mi podařilo vygenerovat spustitelný soubor APK, který je podepsaný vlastním certifikátem.

5.2.1 Struktura

Základním prvkem celé aplikace je vysouvací boční menu (Drawer). To je implementováno pomocí knihovny React Navigation [37] poskytující tyto navigační prvky. Zpočátku bylo velmi obtížné si toto menu přizpůsobit dle potřeb, ale vše nakonec funguje, jak bylo zamýšleno. Menu obsahuje nabídku (Obrázek 17), kde každá představuje jeden tzv. zásobník s obrazovkami. Všechny obrazovky použité v aplikaci jsou tedy rozděleny do jednotlivých zásobníků (Stacks). Každý tento zásobník má svoji výchozí stránku (Screens). Pokud je tedy v menu kliknuto na jednu z možností zobrazí se ta obrazovka, která je v daném zásobníku výchozí (první v pořadí). Mezi dalšími obrazovkami pak lze přepínat libovolně. Celá navigační struktura aplikace je znázorněna na obrázku níže (Obrázek 13).



Obrázek 13 - Navigační struktura mobilní aplikace

5.2.2 Správa dat

Veškerá správa dat je v aplikaci řešena pomocí javascriptové knihovny Redux (podrobnější popis lze najít v kapitole použitých technologií 4.6.3). Ta dokáže flexibilně měnit data všude, kde se s nimi pracuje. Díky tomu máme nastavení stále stejné, i když aplikaci opustíme a následně do ní vstoupíme. Tato vlastnost je zajištěna pomocí speciálního nastavení Reduxu, umožňující perzistentního ukládání. V rámci aplikace využívá Redux jako perzistentní úložiště AsyncStorage. V aplikaci se ukládá velké množství dat, jako jsou např. nastavení, oblíbené obce, nebo přihlášený uživatel. Zpočátku mi dělalo velké problémy danou knihovnu zprovoznit, ale nakonec se vše podařilo a ukládání dat funguje bez problému.

5.2.3 Aktuality obcí

První obrazovka po zapnutí aplikace představuje aktuality obcí. Ta dynamicky reaguje na aktuální stav aplikace, podle kterého určí, co na danou obrazovku zobrazit. V případě, že uživatel nemá vybranou žádnou oblíbenou obec, zobrazí se hláška informující o této skutečnosti. Může se však také stát, že má uživatel vybranou některou obec, ale bohužel pro ni nebyly nalezeny aktuality. I v tomto případě aplikace informuje, že žádné aktuality nebyly nalezeny.

Obrázek 14 - Obrazovka s aktualitami

Aby se uživateli zobrazily aktuality, musí si tedy nejprve vybrat nějakou obec a přidat ji do oblíbených. Seznam oblíbených lze otevřít z bočního menu nebo tlačítkem v horní liště vpravo. To slouží pro urychlení navigace mezi obrazovkami. Pokud se již uživateli zobrazí některé aktuality (Obrázek 14), lze jich načíst více kliknutím na tlačítko ve spodní části obrazovky. Tím se inkrementuje počet požadovaných aktualit a aplikace odešle na server požadavek o další aktuality. Pokud se chce uživatel přesvědčit, zda má zobrazeny aktuální data, může stáhnout obrazovku shora dolů (tzv. *Pull-to-refresh*). Tím se aktivuje vyhledávání a ze serveru se stáhnou nové aktuality.

Tato obrazovka také funguje jako výstupní bod z aplikace. Klepnutím na tlačítko zpět dojde k zobrazení hlášky, zda chce uživatel opravdu opustit aplikaci. Tuto akci lze potvrdit nebo zrušit.

Každá aktualita spadá do určité kategorie. V rámci aplikace si lze vybrat, které tyto kategorie chce uživatel zobrazovat. To lze nastavit pro každou oblíbenou obec zvlášť. Tím je zajištěno, že uživatel bude dostávat pouze ty příspěvky, které ho zajímají. Jednotlivé nastavení sledovaných kategorií lze provést v podrobnostech každé obce (Obrázek 18).

5.2.4 Vyhledávání obcí

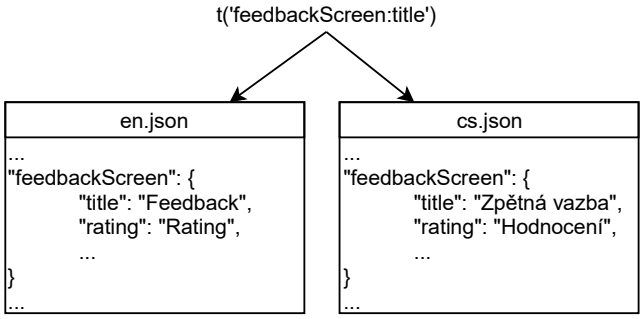
Pro vyhledání obcí musí uživatel přejít na obrazovku s oblíbenými obcemi. Zde se nachází tlačítko pro přidání. Kliknutím se dostaneme na vyhledávací obrazovku (Obrázek 19). Nyní musí uživatel zadat zemi, ve které se jeho obec nachází. To je z toho důvodu, aby se při budoucím rozšíření urychlilo vyhledávání samotných obcí. Po vybrání země se zobrazí další možnost pro vyhledání tentokrát již obce. Vyhledává se podle názvu. Původně aplikace napovídala podle dosud zadaných znaků velice detailně. Bohužel vzhledem k nově přidaným obcím a nemožnosti databáze vyhledávat v řetězcích došlo k razantnímu zpomalení. Vyhledání obce tak trvalo několik sekund což je v dnešní době nepřípustné. Proto jsem byl nucen proces vyhledávání optimalizovat. Výsledné vyhledávání nakonec napovídá dle doposud zadaných znaků, ale musí se shodovat se začátkem názvu obce. Jakmile si uživatel zvolí obec, zobrazí se mu tlačítko pro přidání. Kliknutím na něj se obec přidá do seznamu oblíbených.

5.2.5 Nastavení

V aplikaci lze také určité věci nastavit. Konkrétně jimi jsou jazyk aplikace a velikost zobrazovaných aktualit. Obě tato nastavení jsou ukládána do paměti pomocí knihovny Redux popsané dříve. Tím je zaručeno, že když uživatel aplikaci vypne a následně zapne, bude vše

nastaveno tak, jako předtím. Zde je ještě výhoda pro přihlášené uživatele, kterým se jakákoliv změna v nastavení uloží na server. Pokud se pak se svým účtem přihlásí na jiném zařízení, budou i zde mít své nastavení automaticky navolené.

Co se týče jazykového nastavení, tak to disponuje plně anglickým a českým jazykem. Německý jazyk je zatím nekompletní. Pro změnu jazyka napříč aplikací se využívá knihovny „i18next“ [38]. Ta umožňuje přepínání jazyka. Jednotlivé jazyky jsou uloženy v souborech JSON. Konfigurace zmiňované knihovny zahrne tyto soubory při své inicializaci a podle nich pak do textu dosazuje patřičné texty. Nikde v kódu tak není přímo napsaný text (tzv. plain text), ale vše je řešeno pomocí překladové funkce dané knihovny. Té se předloží řetězec podle kterého vyhledá patřičný text dle aktuálně zvoleného jazyka. Na obrázku níže (Obrázek 15) lze vidět ukázku překladové funkce a jednotlivých jazykových souborů.



Obrázek 15 - Ukázka překladové funkce

Další nastavitelnou vlastností jsou aktuality, a to konkrétně jejich zobrazovaná velikost. Nejmenší zobrazuje pouze název, datum, kategorie a obec, ke které se aktualita vztahuje. Střední velikost obsahuje navíc část popisu dané události (konkrétně 4 řádky). Třetí největší možnost zobrazuje dvojnásobek zmiňovaného obsahu, než má střední velikost).

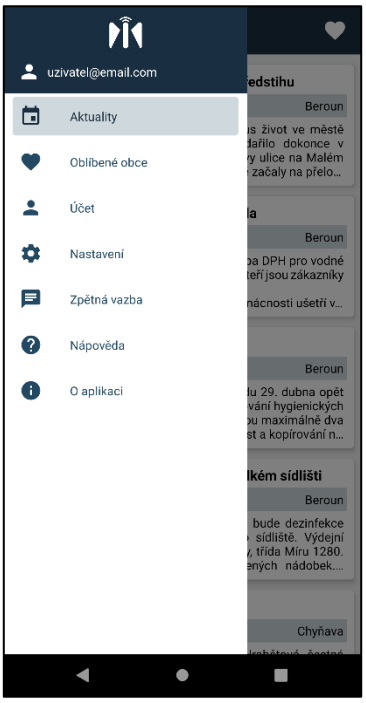
5.2.6 Grafické rozhraní

Jedním z důležitých prvků je také grafické rozhraní aplikace. Mezi jeho základní prvky se řadí boční vysunovací menu, na které je většina uživatelů zvyklá. Dalším důležitým prvkem je jednotný styl a určité typy barev. Základní barvy aplikace jsou rozděleny na primární a sekundární. Od každé této barvy pak existuje ještě světlý a tmavý odstín. Při správné kombinaci těchto barev je aplikace velice přehledná. Dále jsou pak v aplikaci jednotné barvy pro zobrazení hlášek informující uživatele. Ty jsou nastaveny podle závažnosti hlášky. Součástí grafického návrhu je také vlastní logo (Obrázek 16). V aplikaci je použit speciální prvek z knihovny React Native a to konkrétně `<SafeAreaView>`. Ten zajišťuje korektnost vykreslení na všech typech displejů. Problémy by totiž mohly nastat u zařízení mající speciální výřezy na přední fotoaparát. Díky tomuto prvku se však aplikace vždy správně přizpůsobí.

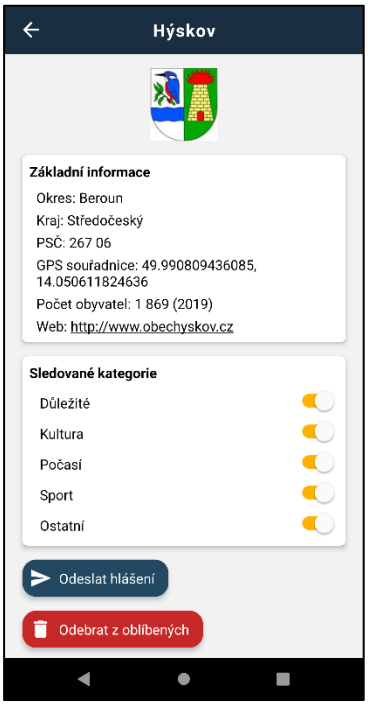
Obrázek 16 - Logo mobilní aplikace

Během práce jsem se setkal s velkým množstvím knihoven, které jsem chtěl použít. Bohužel v průběhu implementace a následného testování se přidané prvky nechovaly, tak jak bych očekával. Z toho důvodu jsem byl často nucen si komponenty napsat v sám. Příkladem je horní navigační lišta. Tu jsem se snažil napsat co nejvíce univerzální. Tato lišta obsahuje titulek uprostřed a následně na každé straně jednu ikonu a akci s ní spojenou. Lze však také uvést pouze

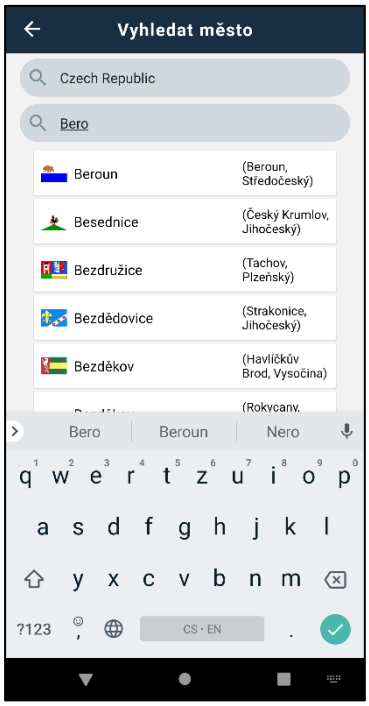
levé tlačítko a na pravé straně pak nebude nic. Dalším prvkem, který jsem si napsal sám je hodnotící okénko s hvězdičkami pro ohodnocení. I zde jsem se snažil importovat knihovnu splňující tyto požadavky, ale bohužel knihovna vypadala nedodělaně. Mezi další vlastnoručně vytvořené komponenty se řadí informační hlášky, které při konkrétních událostech „vyjedou“ na obrazovku a po chvíli zmizí. Dále jsem také napsal formulář pro vstupní data. Ten např. umožňuje nastavení chybových hlášek při špatně zadaném vstupu. V neposlední řadě jsem také vytvořil načítací komponentu, indikující že dochází k určité operaci (např. přihlašování). Pokud aplikace provádí nějakou činnost, je o tom uživatel vždy informován. Tím odpadá nejistota uživatele, zda např. již na tlačítko klikl či ne. Díky všem těmto popsaným vlastnostem je výsledná aplikace velice jednotná a působí přehledně. To bylo současně hlavním cílem vyvíjené aplikace.



Obrázek 17 - Ukázka bočního menu



Obrázek 18 - Ukázka podrobností dané obce



Obrázek 19 - Ukázka vyhledávání obce

6 Testování

Součástí diplomové práce je také testování mobilní aplikace. Konkrétně se zaměříme na testování na uživatelích. Díky tomu získáme informace o možných problémech či nedostatcích v aplikaci. Testování je zaměřeno především na grafické rozhraní a funkčnost aplikace jako celek. Mezi cílovou skupinu našeho testování spadají všichni, kdo umí používat chytré telefony či tablety a mají zájem být informování o dění v jejich lokalitách. Testování se zúčastnilo celkem 9 účastníků.

6.1 Příprava testování

Testování se odehrává v napodobené laboratoři, kde je pouze účastník testu a já v roli tzv. moderátora, který účastníka seznamuje s průběhem testování. Moderátor by však neměl účastníkovi s plněním úkolů pomáhat. Může však poskytnout potřebnou asistenci, pokud si účastník vůbec nebude vědět rady. Během testování se také provádí tzv. pre-test a post-test. Jde o krátké dotazníky, které účastník vyplní před samotným testováním (pre-test) a následně po testování (post-test). Pre-test slouží k rozšíření informací o účastníkovi. Otázky jsou zaměřeny na jeho informovanost o dění v obci. V rámci otázek post-testu se zaměřujeme na bezprostřední pocity a názory, jak se účastníkovi aplikace používala, co se líbilo či nelíbilo.

Součástí přípravy je vytvoření sady úkolů, které bude účastník plnit. Důležité je, aby náročnost těchto úkolů byla správně rozvrhnuta. Účastník začíná nejprve s jednoduššími, tak aby se seznámil s aplikací. Těmi jsou úkoly č. 1-3. Následují složitější úkoly (č. 4-10), jejichž výsledky jsou pro nás důležité a současně by měly aplikaci otestovat důkladněji. Během této fáze je totiž uživatel nejvíce soustředěný. V poslední fázi se nachází doplňkové úkoly (č. 11-15). Všechny testované úkoly jsou následující:

- Úkol 1: Přepněte aplikaci do češtiny
- Úkol 2: Zjistěte aktuální verzi aplikace
- Úkol 3: Zobrazte informaci o tom, jak vytvořit účet
- Úkol 4: Založte účet s předloženými údaji a proveďte ověření účtu
- Úkol 5: Přihlaste se do aplikace pod nově založeným účtem
- Úkol 6: Přidejte obec Týskov a Beroun do seznamu oblíbených
- Úkol 7: Zobrazte nalezené aktuality
- Úkol 8: Nastavte, že z obce Beroun chcete zobrazovat pouze aktuality spadající do kategorie „Důležité“
- Úkol 9: Zobrazte nalezené aktuality v co nejmenší velikosti, tak aby se jich na obrazovku vešlo co nejvíce
- Úkol 10: Zobrazte podrobnější informace o některé z aktualit obce Beroun
- Úkol 11: Zjistěte počet obyvatel obce Týskov
- Úkol 12: Odstraňte obec Beroun ze seznamu oblíbených
- Úkol 13: Ohodnoťte aplikaci hvězdičkami a hodnocení uložte
- Úkol 14: Odešlete Váš názor na aplikaci
- Úkol 15: Smažte používaný účet

6.2 Průběh testování

Samotné testování začalo pozváním účastníka do testovací místnosti. Zde jsem mu v roli moderátora sdělil všechny potřebné informace a ujistil ho, že výsledky testování jsou anonymní.

Také jsem účastníka ujistil v tom, že se testuje aplikace a nikoliv on sám. To je velice důležité před testováním tento fakt sdělit. Před začátkem testu vyplnil účastník spolu s moderátorem pre-test a následně jsme se pustili do samotného testování. Testování bylo rozděleno na několik úkolů. Tyto úkoly byly řádně popsány a předloženy účastníkovi. Současně musely být splněny v daném pořadí. Účastník tedy plnil jednotlivé úkoly a já jsem sledoval jeho činnost. Pokud bylo třeba upozornil jsem účastníka na to, aby své myšlenky říkal nahlas. To je jeden z velice důležitých aspektů během testování. Pokud účastník říká své myšlenky nahlas, máme mnohem lepší představu o tom, o co se v danou chvíli snaží a nad čím přemýšlí. Celé testování probíhalo 25. až 27. 4. 2020. Pro zvýšení efektivity bylo testování prováděno na libovolném zařízení, které si účastník zvolil. Tím mohly na povrch vyplynout další případné nedostatky. Po skončení testování proběhlo ještě vyplnění post-testu.

6.3 Vyhodnocení testování

Nyní se dostáváme k výsledkům samotného testování aplikace. Zde se účastníci často potýkali s problémem při vyhledání obce. Stránka totiž nabízí nejprve možnost vyhledání země a až poté nabízí vyhledání obce. To účastníky mátlo, současně s faktem, že titulek stránky zní „Vyhledání obce“. Proto se stávalo, že do kolonky země byl zadán název obce. Po prvním odzkoušení však bylo účastníkům vše jasné. Do budoucna by ale nebylo od věci předvyplnit zemi za uživatele např. podle aktuální polohy. To by ušetřilo uživateli čas. Mezi další nedostatky se řadí anglický jazyk ověřovacích e-mailů ze strany GCP při nastavené češtině v aplikaci. To bylo zapříčiněno základním nastavením služby Firebase Authentication. Kompletní seznam nálezů je popsán v tabulce níže (Tabulka 5). Každý nález je také třeba ohodnotit. Proto zavádíme kategorie závažnosti, které přiřadíme každému nálezu. Kategorie závažnosti problémů jsou následující:

1. Nízká - problém je drobností
2. Střední - problém není závažný, ale může ovlivnit ovládání
3. Vysoká - závažný problém

Tabulka 5 - Nálezy zjištěné při testování mobilní aplikace

Kategorie závažnosti	Nález (problém)	Navrhované řešení
Střední	Vyhledávání obcí nabízí nejdříve volbu země - to je pro uživatele matoucí.	Předvyplnění země za uživatele (např. podle polohy)
Střední	Ověřovací e-maily jsou anglickém jazyce.	Rozšíření této funkcionality na straně serveru (např. využitím služby SendGrid).
Nízká	Při volbě sledovaných kategorií si někteří účastníci nebyli jistí, zda se akce nemusí potvrdit. Občas tak klikali na tlačítko pro zaslání hlášení obci, které mělo stejnou barvu.	Přidání určité hlášky, že došlo ke změně, nebo změna barvy tlačítka.
Nízká	Občas hledali účastníci informace o vytvoření účtu přímo u samotné funkcionality a ne v nápovědě.	Přidání odkazu (prokliku) na nápovědu k místu s vytvořením účtu.
Nízká	Při zadávání údajů ve formulářích mizí chybové hlášky až když uživatel klikne na jiné pole.	Nastavení kontroly těchto chybových hlášek na každou změnu v daném formuláři.
Nízká	Chybí možnost některou z oblíbených obcí kompletně vypnout (neodebírat) jedním tlačítkem.	Přidání tlačítka ke každé oblíbené obci. Jeho zaškrtnutím by byly aktuality z dané obce zobrazovány.

Na základě výsledků testování byly některé drobnosti ihned opraveny. Jednalo se hlavně o gramatické chyby či nesrovnalosti v pojmenování určitých prvků. Kromě nalezených problémů bylo v rámci testování a výsledků post-testu zřejmé i několik pozitivních věcí. Jednou z nich je dobře navržené grafické rozhraní, které si účastníci pochvalovali a jednoduchost v ovládání aplikace. Další pozitivní věcí je fakt, že nebyly nalezeny žádné závažné problémy.

7 Závěr

Cíle diplomové práce byly splněny. Obecní informační systém byl navržen a jeho základní komponenty byly implementovány. Konkrétně se jedná o server generující obecní aktuality a o multiplatformní mobilní aplikaci. Vzhledem k širokému záběru použitých technologií jsem se potýkal s mnohými problémy. Ty většinou pramenily z neznalosti dané technologie. Postupem času jsem se s těmito technologiemi seznámil a problémy vyřešil. Mezi nejzásadnější úskalí se řadila práce se službami Google Cloud Platform a technologií React Native.

Zpočátku jsem se zabýval rozborem a zkoumáním existujících řešení. Z toho vzešlo několik poznatků, pomocí kterých jsem stanovil základní požadavky na systém. Dále jsem vypracoval patřičný návrh celého systému. Současně s tím jsem se seznamoval s novými technologiemi, na jejichž základě je výsledný systém sestaven.

V rámci implementace byly zrealizovány základní částí systému, a to server a mobilní aplikace. Server je provozován službou Google Cloud Platform. U té se využívá především služby pro samotný provoz serveru a jeho API, databáze, služby pro správu uživatelů a služby umožňující logování. Druhá část systému, mobilní aplikace, je realizována pomocí technologie React Native umožňující vytvoření multiplatformní aplikace jak pro Android tak iOS. S vývojem mobilní aplikace bylo spojeno správné navržení grafického rozhraní. Cílem také bylo zachování jednoduchosti používání a přehlednosti. To se na základě závěrečného testování potvrdilo jako úspěšné.

Poslední část práce byla tedy věnována testování mobilní aplikace na uživatelích. Konkrétně se testu zúčastnilo 9 účastníků, kterým byly postupně kladeny určité úkoly. Během testování bylo zaznamenáno počínání účastníků. Na základě nasbíraných dat bylo posléze provedeno vyhodnocení, z kterého vzešly nedostatky aplikace. Mezi ně se řadí např. vyhledávání obcí, které je pro uživatele občas matoucí.

Obecně lze říci, že systém funguje dobře a funguje dle očekávání. Současně byl kladen důraz na budoucí rozšiřitelnost systému. Mezi tato rozšíření by se mohlo řadit vylepšení algoritmu generování obecních aktualit např. pomocí strojového učení. Dále by mohl tento algoritmus vytvořit pro každou obec určitý vzor (template), pomocí kterého by při příštím generování věděl, jakým způsobem aktuality pro danou obec zpracovat. To by ušetřilo určité množství času. Dalším vylepšením by mohlo být rozšířené nastavení pro sledování aktualit. Konkrétní myšlenka je taková, že by si uživatel nastavil sledování např. sportovních událostí v okruhu 3 km od dané lokality. Tím by získal informace i z okolních obcí v nastavené vzdálenosti a pouze ty události s danou kategorií. Mobilní aplikace by také mohla umožňovat manuální nastavení kategorií u aktualit. Tyto změny by se zaznamenávaly na server, který by se na základě těchto změn učil, co příště označit jakou kategorií. Dalším místem pro vylepšení by byla webová aplikace pro administraci, pomocí které by administrátoři obcí mohli zasílat uživatelům sledující danou obec patřičné zprávy či informace. V neposlední řadě by mohl být systém vylepšen o zasílání notifikací serverem, informující o nových aktualitách. Důležitým krokem by také bylo nahrání aplikace na patřičné distribuční služby (Google Play a App Store), čímž by se dosáhlo značného rozšíření mezi uživatele a získání důležité zpětné vazby pro další vývoj.

8 Literatura

1. **Google Play. [Online] [Citace: 29. Říjen 2019.] https://play.google.com/store.**

2. App Store. [Online] [Citace: 28. Říjen 2019.] https://www.apple.com/cz/ios/app-store/.

3. *BSSHOP.* [Online] BSSHOP s. r. o. [Citace: 2. Listopad 2019.] https://www.bsshop.cz/blog/chytre-mesto-s-aplikaci-cityapp.

4. *CityApp.* [Online] BSSHOP s.r.o. [Citace: 1. Listopad 2019.] http://cityapp.cz/.

5. *Česká obec.* [Online] [Citace: 29. Říjen 2019.] http://www.ceskaobec.cz/.

6. Česká obec. *App Store.* [Online] [Citace: 29. Říjen 2019.] https://apps.apple.com/cz/app/%C4%8Desk%C3%A1-obec/id1027734504.

7. Česká obec. *Google Play.* [Online] [Citace: 29. Říjen 2019.] https://play.google.com/store/apps/details?id=cz.ceskaobec.

8. Mobilní Rozhlas/Zlepšeme Česko. *App Store.* [Online] [Citace: 29. Říjen 2019.] https://apps.apple.com/cz/app/mobiln%C3%AD-rozhlas-zlep%C5%A1eme-%C4%8Desko/id1378883073.

9. Mobilní Rozhlas / Zlepšeme Česko. *Google Play.* [Online] [Citace: 29. Říjen 2019.] https://play.google.com/store/apps/details?id=com.neogenia.zlepseme_cesko.

10. *Zlepšeme Česko.* [Online] [Citace: 29. Říjen 2019.] https://www.zlepsemecesko.cz/.

11. *Neogenia.* [Online] Neogenia s.r.o. [Citace: 1. Listopad 2019.] https://www.neogenia.cz/uvod.

12. Distribution dashboard. *Android Developers.* [Online] [Citace: 23. Září 2019.] https://developer.android.com/about/dashboards.

13. Moje Obec. *Google Play.* [Online] [Citace: 3. Listopad 2019.] https://play.google.com/store/apps/details?id=cz.msw.mojeobec.

14. MojeObec. *App Store.* [Online] [Citace: 1. Listopad 2019.] https://apps.apple.com/cz/app/mojeobec/id590347317.

15. *Macron Software.* [Online] [Citace: 2. Listopad 2019.] https://www.macronsoftware.cz/.

16. *Mobisys.* [Online] [Citace: 1. Listopad 2019.] https://www.mobisys.cz/co-je-mobisys.

17. My Village. *Google Play.* [Online] [Citace: 5. Listopad 2019.] https://play.google.com/store/apps/details?id=dk.minlandsby.app.

18. V OBRAZE. *Google Play.* [Online] [Citace: 3. Listopad 2019.] https://play.google.com/store/apps/details?id=cz.onlineteam.ocelot.

19. V OBRAZE. *App Store.* [Online] [Citace: 29. Říjen 2019.] https://apps.apple.com/cz/app/v-obraze-v%C3%ADm-co-se-d%C4%9Bje/id1128876118?l=cs.

20. *Online-Team.* [Online] [Citace: 29. Říjen 2019.] https://www.online-team.cz/.

21. ManifestViewer. ***Google Play.*** **[Online] [Citace: 6. Listopad 2019.] https://play.google.com/store/apps/details?id=jp.susatthi.ManifestViewer.**

22. Cloud Firestore. ***Google Cloud.*** **[Online] Google. [Citace: 14. Duben 2020.] https://cloud.google.com/firestore?hl=cs.**

23. Cloud Firestore. ***Firebase.*** **[Online] Google. [Citace: 14. Duben 2020.] https://firebase.google.com/docs/firestore#how_does_it_work.**

24. Cloud Storage. ***Firebase.*** **[Online] Google. [Citace: 12. Duben 2020.] https://firebase.google.com/docs/storage.**

25. Stefanov, Stoyan. ***Object-Oriented JavaScript.*** **: Packt Publishing, Limited, 2008. 9781847194152.**

26. A JavaScript library for building user interfaces. ***React.*** **[Online] [Citace: 25. Duben 2020.] https://reactjs.org/.**

27. Visual Studio Code. ***Microsoft Azure.*** **[Online] Microsoft. [Citace: 8. Březen 2020.] https://azure.microsoft.com/cs-cz/products/visual-studio-code/.**

28. Google Acquires Firebase To Help Developers Build Better Real-Time Apps. ***TechCrunch.*** **[Online] [Citace: 23. Duben 2020.] https://techcrunch.com/2014/10/21/google-acquires-firebase-to-help-developers-build-better-realtime-apps/.**

29. Firebase & Google Cloud Platform. ***Firebase.*** **[Online] Google. [Citace: 20. Duben 2020.] https://firebase.google.com/firebase-and-gcp/.**

30. What's the relationship between Firebase and Google Cloud? ***Medium.*** **[Online] A Medium Corporation. [Citace: 20. Duben 2020.] https://medium.com/google-developers/whats-the-relationship-between-firebase-and-google-cloud-57e268a7ff6f.**

31. Firebase. ***Firebase.*** **[Online] Google. [Citace: 21. Duben 2020.] https://firebase.google.com/.**

32. Seznam států světa. ***Wikipedie.*** **[Online] [Citace: 26. Leden 2020.] https://cs.wikipedia.org/wiki/Seznam_st%C3%A1t%C5%AF_sv%C4%9Bta.**

33. How to find Android Version Distribution statistics. ***XDA-Developers Android Forums.*** **[Online] [Citace: 25. Duben 2020.] https://www.xda-developers.com/android-version-distribution-statistics-android-studio/.**

34. Distribution Dashboard. ***Android Developers.*** **[Online] Google. [Citace: 25. Duben 2020.] https://developer.android.com/about/dashboards.**

35. App Store Support. ***Apple Developer.*** **[Online] Apple. [Citace: 25. Duben 2020.] https://developer.apple.com/support/app-store/.**

36. iOS Version Stats - David Smith. ***Independent iOS Developer.*** **[Online] [Citace: 25. Duben 2020.] https://david-smith.org/iosversionstats/.**

37. Getting Started. ***React Navigation.*** **[Online] [Citace: 21. Duben 2020.] https://reactnavigation.org/docs/getting-started.**

38. Introduction. ***i18next documentation.*** **[Online] [Citace: 2020. Březen 12.] https://www.i18next.com/.**

39. Houssein Djirdeh, Anthony Accomazzo, Sophia Shoemaker. ***Fullstack React Native: Create beautiful mobile apps with JavaScript and React Native.*** **: Independently published, 2019. 9781728995557.**

40. Eisenman, Bonnie. ***Learning React Native: Building Native Mobile Apps with JavaScript 2nd Edition.*** **: O'Reilly Media, 2015. 9781491929070.**

41. Biehl, Matthias. ***RESTful API Design (API-University Series) (Volume 3).*** **: CreateSpace Independent Publishing Platform, 2016. 9781514735169.**

42. Ted Hunter, Steven Porter. ***Google Cloud Platform for Developers: Build highly scalable cloud solutions with the power of Google Cloud Platform.*** **: Packt Publishing, 2018. 9781788837675.**

9 Přílohy

Datový nosič CD-R s následujícím obsahem:

- Diagramy
- Dokumenty
  - Diplomová práce (DOCX, PDF)
- Obrázky aplikace
- Zdrojové kódy
  - Server
  - Mobilní aplikace