FAKULTA INFORMAČNÍCH TECHNOLOGIÍ ČVUT V PRAZE

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

Název:	Sociální síť pro výzkumníky Apakrychlí - backend a mobilní klient
Student:	Marek Kodr
Vedoucí:	Ing. Michal Valenta, Ph.D.
Studijní program:	Informatika
Studijní obor:	Webové a softwarové inženýrství
Katedra:	Katedra softwarového inženýrství
Platnost zadání:	Do konce letního semestru 2019/20

Pokyny pro vypracování

Cílem práce je navrhnout a implementovat sociální síť na míru potřebám výzkumníků Apakrychlí. Dalším cílem práce je vytvořit ukázkový funkční projekt pro mobilní platformu za použití moderních technologií. Postupujte v těchto krocích:
1. Společně s vedoucím práce rozpracujte funkční a nefunkční požadavky na systém. Základní požadavky jsou uvedeny níže v zadání.
2. Pomocí metod softwarového inženýrství proveďte návrh celého systému.
3. Návrh implementujte, řádně otestujte a zdokumentujte.

Základní požadavky:
- systém bude používat cloudové úložiště Firebase
- bude podporována autentizace a autorizace uživatelů buď přímo nebo pomocí existujících účtů na sociálních sítích (Google apod.)
- klient bude implementován pro mobilní zařízení na platformě android a bude vyvinut v jazyku Kotlin
- aplikace bude nabídnuta uživatelům přes standardní aplikaci GooglePlay
- funkcionalita bude inspirovaná možnostmi, které poskytuje platforma Facebook pro zájmovou skupinu

Seznam odborné literatury

Dodá vedoucí práce.

Ing. Michal Valenta, Ph.D.
vedoucí katedry

doc. RNDr. Ing. Marcel Jiřina, Ph.D.
děkan

V Praze dne 24. října 2018

FAKULTA
INFORMAČNÍCH
TECHNOLOGIÍ
ČVUT V PRAZE

Bakalářská práce

APAnet

Marek Kodr

Katedra softwarového inženýrství
Vedoucí práce: Ing. Michal Valenta, Ph.D.

15. května 2019

Poděkování

Zde bych rád bych poděkoval vedoucímu práce Ing. Michalu Valentovi, Ph.D. za podporu a cenné rady při tvorbě a odborné vedení. Děkuji také odborníkovi na Apakrychle Ing. Tomáši Nováčkovi za připomínky a gramatické korektury práce.

Prohlášení

Prohlašuji, že jsem předloženou práci vypracoval samostatně a že jsem uvedl veškeré použité informační zdroje v souladu s Metodickým pokynem o etické přípravě vysokoškolských závěrečných prací.

Beru na vědomí, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorského zákona, ve znění pozdějších předpisů. V souladu s ust. § 46 odst. 6 tohoto zákona tímto uděluji nevýhradní oprávnění (licenci) k užití této mojí práce, a to včetně všech počítačových programů, jež jsou její součástí či přílohou, a veškeré jejich dokumentace (dále souhrnně jen „Dílo"), a to všem osobám, které si přejí Dílo užít. Tyto osoby jsou oprávněny Dílo užít jakýmkoli způsobem, který nesnižuje hodnotu Díla, a za jakýmkoli účelem (včetně užití k výdělečným účelům). Toto oprávnění je časově, teritoriálně i množstevně neomezené. Každá osoba, která využije výše uvedenou licenci, se však zavazuje udělit ke každému dílu, které vznikne (byť jen zčásti) na základě Díla, úpravou Díla, spojením Díla s jiným dílem, zařazením Díla do díla souborného či zpracováním Díla (včetně překladu), licenci alespoň ve výše uvedeném rozsahu a zároveň zpřístupnit zdrojový kód takového díla alespoň srovnatelným způsobem a ve srovnatelném rozsahu, jako je zpřístupněn zdrojový kód Díla.

V Praze dne 15. května 2019 ……

Odkaz na tuto práci

Kodr, Marek. *APAnet*. Bakalářská práce. Praha: České vysoké učení technické v Praze, Fakulta informačních technologií, 2019.

Abstrakt

Výzkum Apakrychlí je velmi složitá vědecká činnost, jejíž nedílnou součástí je sdílení informací mezi výzkumníky. Existuje mnoho způsobů, jak v dnešní době sdílet informace, avšak doposud žádná sociální síť se nevěnovala této problematice. Zde přichází sociální síť APAnet. APAnet umožní snadné sdílení a prohlížení informací, obrázků i míst podezřelých z výskytu Apakrychlí, čímž urychlí šíření povědomosti o nich. Tím pomůže přilákat jak odborné, tak i amatérské publikum k tématu Apakrychlí a jejich výzkumu.

Klíčová slova Apakrychle, Android, Kotlin, sociální síť, APAnet

Abstract

Research of Apakrychle is a very complex scientific activity, with information sharing among researchers being an integral part of it. There are many ways to share information nowadays, but so far no social network has focused on this topic. Here comes the social network APAnet. APAnet will provide an easy way to share and view information, images, and suspicious locations of Apakrychle occurrence, speeding up the spread of knowledge about them.

This will help attract both professional and amateur audiences to the topic of Apakrychle and its research.

Keywords Apakrychle, Android, Kotlin, social network, APAnet

Obsah

Seznam obrázků

Seznam tabulek

Úvod

V dnešní době je primární zařízení pro sdílení informací chytrý telefon. Vznikají aplikace pro tyto zařízení cílené na jednotlivé skupiny s určitým zájmem. Existují aplikace, které sdružují komunity, kuchařů, domácích kutilů, nadšenců vesmírných misí, ale zatím neexistuje žádná platforma, která by sdružovala výzkumníky Apakrychlí. Považuji toto téma za důležité a rád bych svou prací usnadnil jejich výzkum a zdůraznil nezbytnost dalšího výzkumu. Cílem práce je navrhnout a implementovat sociální síť na míru potřebám výzkumníků Apakrychlí [9], která by sjednocovala nejen odbornou, ale také amatérskou komunitu příznivců Apakrychlí. Název APAnet byl vytvořen jako zkrácená verze pro „Apakrychle social network“, tedy sociální síť pro Apakrychle. APAnet umožní sdílet nová věděcká zjištění, fotografie Apakrychlí a také důležité dokumenty. Aplikace bude sloužit také jako demonstrační ukázka mobilní aplikace za použití jazyka Kotlin, propojení s multiplatformní službou Firebase a možností přihlášení pomocí účtu třetích stran, jako například Facebook nebo Google. APAnet klade důraz na anonymitu hodnocení, tedy uživatelé nebudou moci zjistit, kdo a jak jejich příspěvek ohodnotil, nebudou moci smazat příspěvek, tedy pokud jednou uživatel nějaký příspěvek sdílí, nebude ho moci smazat, ani editovat. Důvodem je přiblížení ke skutečnému životu. Aplikace se také zavazuje, že nebude poskytovat data o uživatelích aplikacím třetích stran.

V následujících kapitolách budu podrobně popisovat tvorbu aplikace APAnet od prvotního návrhu, přes analýzu a implementaci až po nasazení do obchodu Google Play.

Kapitola 1

Analýza

Jednou z nejdůležitějších částí je analýza. Zde budu zkoumat, pro jakou platformu je perspektivní vytvořit aplikaci, analyzuji funkční a nefunkční požadavky a provedu srovnání výsledků analýzy s aktuální konkurencí na trhu.

1.1 Analýza požadavků

Po konzultaci s výzkumníky Apakrychlí Michalem Valentou a Tomášem Nováčkem jsem sestavil následující seznam požadavků.

1.1.1 Funkční požadavky

- Sdílení příspěvků *[FP1]*
- Prohlížení příspěvků *[FP2]*
- Komentování příspěvku *[FP3]*
- Nahrávání a stahování příloh *[FP4]*
- Editace profilu *[FP5]*
- Sdílení souřadnic výskytu Apakrychlí *[FP6]*
- Tvorba a vyhledávání tagů *[FP7]*
- Hodnocení příspěvků *[FP8]*

1.1.1.1 Sdílení příspěvků

Aplikace umožní vytváření a sdílení příspěvků.

1.1.1.2 Prohlížení příspěvků

Příspěvky bude možné prohlížet nejen na domovské obrazovce, ale i výsledcích vyhledávání či profilu uživatele.

1.1.1.3 Komentování příspěvku

K příspěvku bude možné přidat komentář na jeho detailním zobrazení. I na komentáře k příspěvku bude možné reagovat pozitivně, či negativně, případně pod něj přidat další komentář.

1.1.1.4 Nahrávání a stahování příloh

Dokument, nebo obrázek bude možné připojit k příspěvku. Příloha bude výlučná, tedy jeden příspěvek nebude moci mít jak obrázek, tak i přiložený dokument. Při zobrazení příspěvku s dokumentem bude možné po kliknutí onen dokument stáhnout. V případě obrázku se po kliknutí na něj zobrazí jeho detail.

1.1.1.5 Editace profilu

Uživatel bude mít možnost upravovat svůj profil, a to tak, že bude moci měnit jméno, které bude zobrazené, a nahrávat nové profilové obrázky.

1.1.1.6 Sdílení souřadnic výskytu Apakrychlí

Bude existovat speciální typ příspěvku, který bude obsahovat souřadnice místa podezřelého z výskytu Apakrychlí.

1.1.1.7 Tvorba a vyhledávání tagů

K příspěvku bude možné do textu připojit tag, podle kterého bude příspěvek možné později vyhledat.

1.1.1.8 Hodnocení příspěvků

Uživatelé budou moci hodnotit příspěvky ostatních uživatelů, a to jak pozitivně, tak i negativně. Hodnocení bude pro uživatele anonymní, tedy nebude nikde graficky viditelné, kdo koho jak hodnotil.

1.1.2 Nefunkční požadavky

- Aplikace pro mobilní platformu
- Nahrávání dat na Firebase
- Přehlednost
- Git
- Automatizovaná distribuce
- Přihlášení přes účty třetích stran

1.1.2.1 Aplikace pro mobilní platformu

Aplikace bude navržena pro mobilní operační systém, který bude nejvíce používaný v cílové komunitě uživatelů.

1.1.2.2 Napojení na Firebase

Data budou nahrávána na cloudové úložiště Firebase. Bude využívána databáze a úložiště pro sdílení dat a souborů.

1.1.2.3 Přehlednost

Bude kladen důraz na jednoduchý a přehledný design. Aplikace bude obsahovat co nejméně skrytých menu, která by vedla k hlubokému zanoření.

1.1.2.4 Git

Aplikace bude verzována přes školní verzovací systém GitLab.

1.1.2.5 Automatizovaná distribuce

Aplikaci bude možné nahrát do obchodu s mobilními aplikacemi.

1.1.2.6 Přihlášení přes účty třetích stran

K aplikaci APAnet bude možné se přihlásit nejen pomocí email a hesla, ale také i pomocí účtů třetích stran, jako je Google a nebo Facebook.

1.2 Výběr platformy

První krok je zvolení nejvhodnější a nejvíce rozšířené mobilní platformy, pro kterou bude aplikace narhována.

1.2.1 Mobilní platformy

Seznam nejpoužívanějších operačních systému pro mobilní zařízení podle serveru StatCounter. Z daných výsledků tabulky jasně plyne, že nejperspektivnější platformou pro tvorbu této aplikace je Android. Aplikace tedy bude napsána pro mobilní platformu Android a bude distribuována skrz obchod Google Play.

Operační systém	Podíl ve světě	Podíl v Evropě	
Android	74.15%	71.20%	
iOS	23.28%	27.69%	
Windows	0.29%	0.53%	

Tabulka 1.1: Podíl mobilních operačních systému na trhu [1]

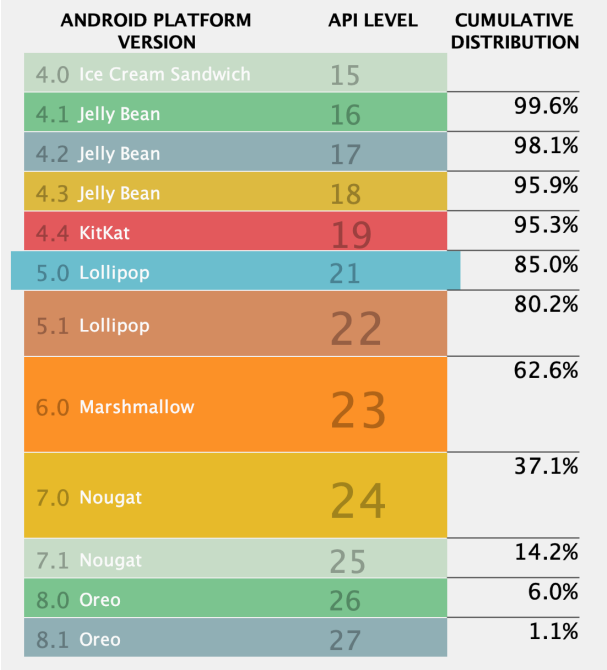
1.2.2 Android SDK

Důležitou součástí je volba minimální verze Android API. Dle statistik uvedených v tabulce 1.1 85 % zařízení používá verzi Androidu 5.0, respektive API 21 a vyšší. Z toho důvodu jsem také zvolil jako minimální verzi Android SDK 21.

1.3 Konkurence v Google Play

Hledání bylo zaměřeno pouze na aplikace pro Android v rámci obchodu Google Play, kde existuje mnoho konkurenčních aplikací. Zde se budu zabývat několika vybranými z nich a srovnávacím kritériem budou následující prvky:

- Sdílení fotografií bez komprimace
- Sdílení souborů
- Sdílení souřadnic
- Hodnocení příspěvků
- Možnost vytvoření tagů
- Přehlednost zobrazení příspěvků



Obrázek 1.1: Zastoupení verzí OS Android [15, snímek pořídil autor]

1.3.1 Podobné aplikace

Pro podrobnější analýzu jsem vybral aplikace Facebook, Google+, Line a Twitter. Kritéria pro výběr aplikací byla ta, aby obsahovala možnost sdílení textových příspěvků a obsahovala zeď, tedy místo pro zobrazení všech příspěvků.

1.3.1.1 Facebook

Aplikace Facebook (viz obrázek 1.2a) nabízí velmi široké možnosti, co se týče sdílení příspěvků. Umožňuje sdílet souřadnice a vytvářet tagy. Všechny příspěvky jsou zobrazeny na hlavní obrazovce, tedy dobře přístupné pro uživatele. Příspěvky je možné vyhledávat podle tagu i podle obsahu příspěvku. Je zde ovšem velmi komplikované sdílení dokumentů. Co se týče hodnocení příspěvků, je možné vybírat z několika možných reakcí. Je to velký krok kupředu, jelikož uživatel už není limitován pouze na reakci, že se mu něco líbí, ale může také vyjádřit negativní reakci, což je podle mě velmi důležité.

1.3.1.2 Google+

Google+ (viz obrázek 1.2b) je dle mého názoru nejlepší z vybraných aplikací. Nabízí nejen sdílení obrázků a souborů pomocí Google Drive, tedy je možné sdílet i nekomprimované obrázky. Je možné vytvářet vlastní tagy a vyhledávat podle nich. Zobrazení příspěvků je velmi přehledné a i zde je možné reagovat a hodnotit příspěvky, opět pouze však pozitivní reakcí. Také jsem zde nenašel možnost sdílení lokace v příspěvku.

I přes to, že tato služba nabízí velmi pokročilé možnosti a dle mého názoru je velmi povedená, byla ukončena 2. dubna 2019.

1.3.1.3 Line

Primární účel aplikace Line (viz obrázek 1.2c), nejvíce používané v Asii, není sdílení příspěvků, ale posílání zpráv, ale i tak je sdílení velmi významnou částí. Aplikace umožňuje sdílení obrázků i polohy. Reakce na příspěvky jsou možné v podobě velkých emoji[1]. Neobsahuje možnost sdílení dokumentů v příspěvku. Tvorba tagů funguje podobně jako u ostatních služeb, tedy je možné vytvářet nové tagy a i podle nich vyhledávat.

1.3.1.4 Twitter

Twitter je jedna z nejpopulárnějších aplikací pro sdílení textových příspěvků. Twitter (viz obrázek 1.2d) umožňuje velmi pokročilé vyhledávání podle příspěvků, dokonce už i při vytváření napovídá tagy. Sdílení souborů zde není možné, ovšem obrázky i lokaci možné sdílet je. Hodnocení probíhá prostřednictvím pozitivní reakce v podobě červeného srdíčka. Příspěvky je také možné libovolně filtrovat a velmi přehledně zobrazovat.

1.3.2 Porovnání

Stanoveným požadavkům byla nejblíže aplikace Google+, ovšem i tato aplikace je stále zaměřená na velmi širokou veřejnost, proto nenabízí specifické funkce pro jednotlivé skupiny. Je zde tedy velký prostor pro novou službu zaměřenou na skupinu apakrychliků.

[1] Malý obrázek imitující náladu tvůrce.

1.4 Analýza uživatele

Jelikož tato aplikace je stavěna na míru výzkumníků Apakrychlí, tak zde pro návrh aplikace použiji návrh podobný User Centered Design, který se zaměřuje na podrobnou analýzu potřeb koncového uživatele a přiblížení prostředí, ve kterém aplikace bude nasazena vývojářům. Pro lepší ilustraci koncového uživatele zde vytvořím modelové persony.

1.4.1 Persony výzkumníků Apakrychlí

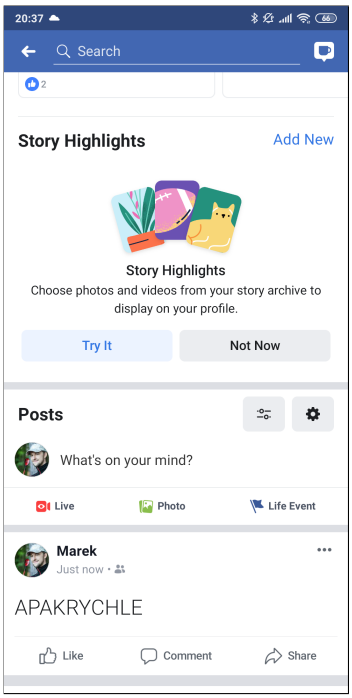
Největší popularitě se výzkum Apakrychlí těší na půdě vysokých škol, tedy jako modelové persony zvolím studenta a vyučujícího.

1.4.1.1 Student

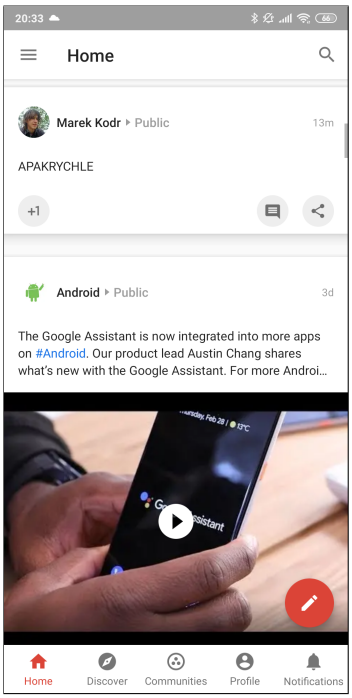
Jan Novák je studentem bakalářského programu Fakulty informačních technologií Českého vysokého učení technického v Praze. Pracuje na několika projektech pro školu a je známý tím, že klade důraz na použití nejmodernější technologie a designu, který je aktuálně dostupný. S novým softwarem se seznamuje velice rychle a rád objevuje všechny možnosti nastavení aplikace a vždy stahuje nejnovější aktualizace. Tvorbu příspěvků rád doprovází obrázky.

1.4.1.2 Vyučující

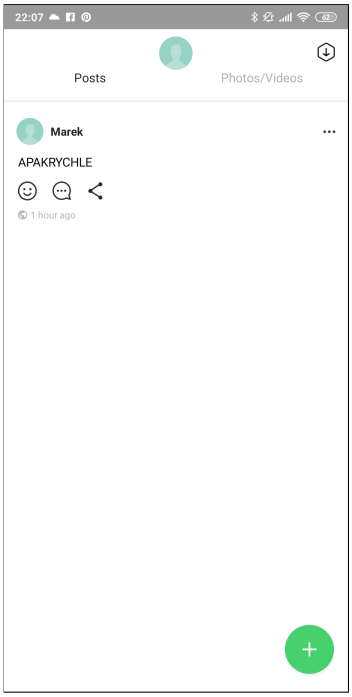
Jaroslav Krátký je vyučující na vysoké škole. Vyučuje zároveň několik předmětů a díky tomu často využívá tagy, aby rozlišil informace týkajících se jednotlivých témat. Při vědecké práci často objevuje zmínky o Apakrychlích a proto je pro něj velmi důležité, aby mohl tyto dokumenty sdílet. Na nové technologie nespěchá, ale klade důraz na detail a na spolehlivost aplikace. Také by ocenil, kdyby mohl sdílet souřadnice míst podezřelých z výskytu Apakrychlí, jelikož při své práci často nějaké objeví.



(a) Facebook



(b) Google+



(c) Line

(d) Twitter

Obrázek 1.2: Konkurenční aplikace

Kapitola 2

Návrh

Zde se budu věnovat návrhu aplikace. Popíšu zde případy užití a návrh databáze, zdůvodním výběr použitých technologií a představím návrh uživatelského rozhraní.

2.1 Případy užití

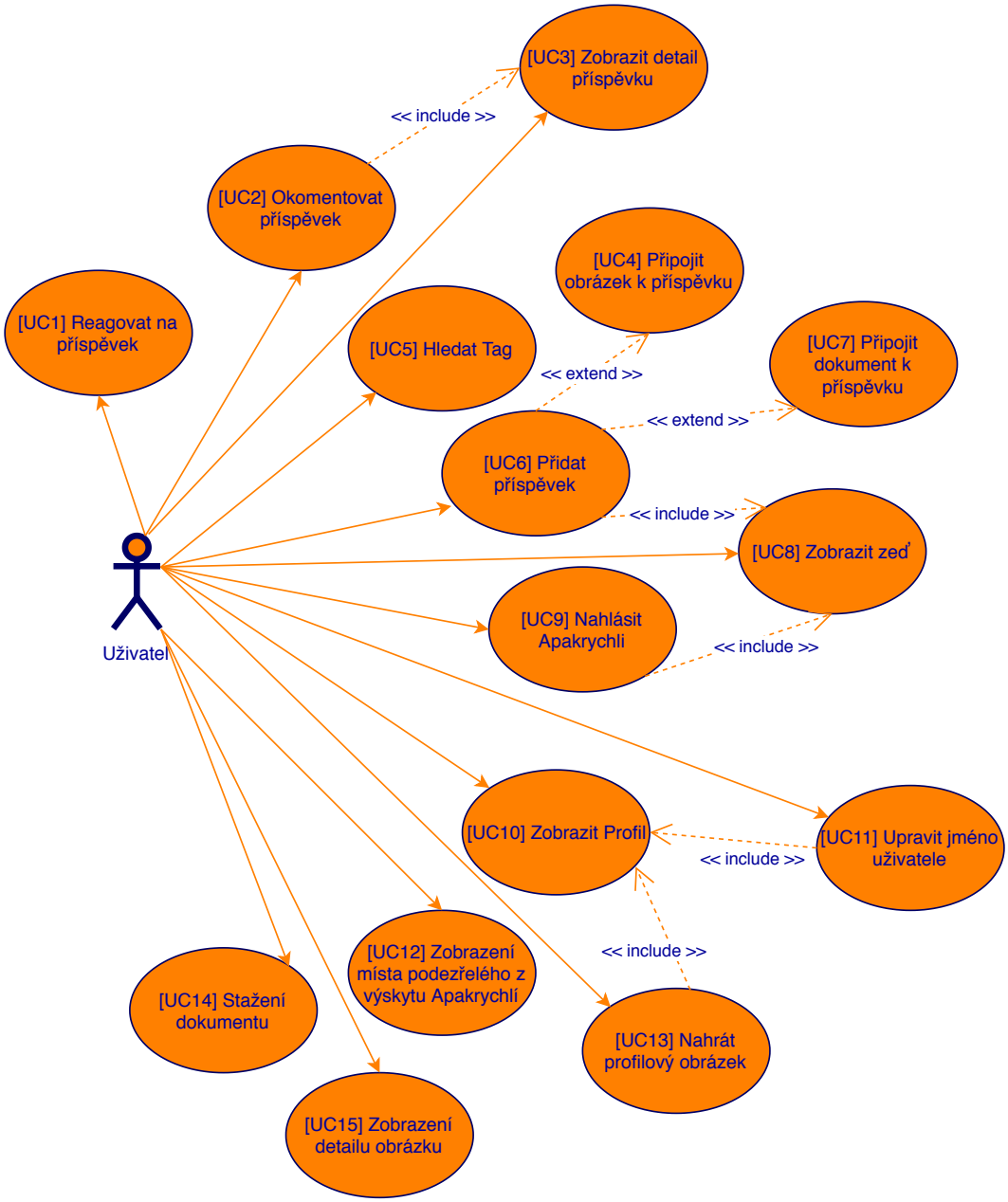
Uživatel si bude moci v aplikaci zobrazit příspěvky chronologicky podle toho, jak byly tvořeny. U příspěvku, který obsahuje obrázek, bude možné si zobrazit detail obrázku. U těch, které obsahují dokument, bude možné jeho stažení. Pokud se bude jednat o nahlášené místo podezřelé z výskytu Apakrychlí, bude možné zobrazit jeho lokaci v mapách. Dále bude umožněno vyhledávání příspěvků podle tagu, který obsahují. Příspěvky bude uživatel moci nejen prohlížet, ale i vytvářet a v případě potřeby připojit dokument, či obrázek. Pokud bude potřeba, je možné nahlásit místo výskytu Apakrychlí. Profil uživatele je editovatelný a uživatel bude moci nahrát nový obrázek, případně i změnit zobrazené jméno. Všechny klíčové případy užití jsou zachyceny na obrázku případů užití 2.1.

2.2 Výběr technologie

Pro tvorbu aplikace jsem musel udělat několik rozhodnutí ohledně použité technologie. Klíčové bylo zvolit technologie, která bude podporovaná i v budoucnu.

2.2.1 Nativní vs. hybridní

V dnešní době je možné vyvíjet aplikace pro Android různými způsoby. Pro nativní způsob je možné využít Android Studio IDE [12], případně velmi poruchové a zastaralé IDE NetBeans [17]. Pro hybridní vývoj existuje několik



Obrázek 2.1: Model případů užití

možností. Nejpoužívanější je Flutter a Xamarin. Problém vývoje hybridních aplikací je ten, že komponenty musí být univerzální pro všechny platformy. Na jednu stranu to může být výhodou pro investora, který chce ušetřit na vývoji dané aplikace, na druhou stranu je velmi těžké takovou aplikaci odladit a udržovat její běh. Z toho důvodu jsem se rozhodl pro nativní způsob vývoje.

2.2.2 Implementační jazyk

Při tvorbě aplikací pro Android je možné zvolit jazyk Java, nebo nově i Kotlin. Zásadní výhodou Kotlinu je null-safety jazyk, který slouží k eliminaci null referencí v kódu. [16] To usnadní kontrolu obsahu proměnných a vynechání bloků try-catch. Kotlin také obsahuje mnoho lambda výrazů, což usnadní čitelnost a implementaci. V případě, že by v Kotlinu něco chybělo a bylo potřeba využít implementaci v Javě, je to možné, díky tomu, že oba jazyky používají stejnou JVM[2]. Tedy není jediný důvod pro použití jazyku Java, který se v tuto chvíli stal již zastaralým pro vývoj aplikací pro platformu Android.

2.2.2.1 Kotlin extension

Velkou výhodou pro vývoj moderních aplikací pro Android je podpora jazyku Kotlin. Kromě syntaktických výhod nabízí ale i možnost rozšíření a přináší nový způsob interakce s grafickými prvky. Pro srovnání zde uvedu rozdíl mezi implementací v Javě a Kotlinu pro vytvoření akce tlačítka. Kdysi v Javě bylo potřeba každý grafický prvek nejprve namapovat přes volání findViewById. V Kotlinu díky Kotlin extension[6] můžu přímo přistupovat k jednotlivým prvkům. Přístup k prvku mi zajistí importování knihovny synthetic z kotlinx, tedy Kotlin extension.

```
Button button = (Button)rootView.findViewById(R.id.myButton);
buttonClick.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
           // some action
        }
});
```

Výpis 2.1: Implementace v Javě

```
import kotlinx.android.synthetic.main.fragment.myButton

myButton.setOnClickListener {
       // some action
}
```

Výpis 2.2: Implementace v Kotlinu

2.2.3 Vývojové prostředí

Pro nativní způsob vývoje aplikace, jak již bylo zmíněno výše, rozhodoval jsem se mezi aplikacemi Android Studio, NetBeans, Eclipse, Qt Creator a další. Z nabídky jsem vybral dvě vývojová prostředí a týden obě používal za

[2] Java Virtual Machine

účelem zjistit, které je vhodnější. Jelikož Android Studio, které je založené na platformě IntelliJ, je přímo určené pro vývoj aplikací pro Android, bylo ve všech ohledech vhodnější volbou a nenalezl jsem jediný případ, kde by NetBeans bylo lepší nebo vhodnější.

2.2.3.1 Android Studio

Android Studio [12] je oficiální vývojové prostředí pro tvorbu aplikací pro zařízení Android, založené na platformě IntelliJ IDEA, vyvíjené společností Jetbrains. Nativně podporuje jazyky Java, Kotlin a C++ a obsahuje možnost spuštění v emulátoru. Obsahuje specifické funkce pro vývoj pro Android, jako například podepsání aplikace klíčem, vykreslení layoutu při tvorbě nebo sledování běhu aplikace pomocí nástroje Profiler.

2.2.4 Typ databáze

Pro projekt APAnet jsem se rozhodl použít NoSQL databázi, která ukládá data ve formátu JSON[3]. Jednou z hlavních výhod je flexibilita. Je jednoduché postupně přidávat další informace do databáze jelikož není žádné pevné schéma. To mi umožní dynamicky rozšiřovat uložené objekty o další informace a jelikož sociální sítě často přidávají nové možnosti s tím, jak komunita roste, je tato vlastnost velkou výhodou. Například kdybych k uživateli chtěl do databázi přidat novou informaci, „kolik Apakrychlí viděl“, nemusel bych měnit celé databázové schéma, ale stačilo by pouze tuto informaci do objektu „Uživatel“ přidat.

2.2.5 Firebase

Firebase je cloudová služba provozovaná společností Google, která slouží jako backend pro vývoj moderních aplikací. Nejedná se však o jednu službu, kterou by poskytovala, ale o více služeb týkajících se vývoje aplikací pospolu. Firebase nabízí hosting, autentifikaci, NoSQL databázi, Crashlytics, úložiště a mnoho dalších služeb, a to zdarma pro malé projekty do určitého datového limitu. Jednotlivé služby na sobě nejsou závislé a mohou být používány zvlášť. Například bych mohl využít jen Crashlytics a databázi a autentizaci bych mohl mít úplně jinde. V aplikaci APAnet jsem se ale rozhodl využít co nejvíce možností nabízených Firebase. Důvodem je dobrá spolupráce mezi jednotlivými částmi a možnost použití zdarma pro malé aplikace do určitého limit. Pro integraci do projektu je potřeba pouze stáhnout konfigurační soubor a v nastavení zaregistrovat aplikace podle názvu balíčku a klíče. [13]

[3] JavaScript Object Notation

2.2.6 Material Design

V roce 2014 Google na konferenci Google I/O představil nový design, respektive vizuální jazyk pro tvorbu aplikací, který se snaží sjednotit design aplikací nezávisle na platformě. [14] Tento jazyk je možné použít nejen v mobilních aplikacích, ale i ve webových službách a dalších. Samotný Google se od té doby snaží své aplikace a služby převést do tohoto designu, a tím sjednotit roztříštěnost designu svých služeb. Nyní je již většina jím provozovaných služeb spuštěna v tomto stylu. Aktuálně vzniká nová verze, Material Design 2, která ještě není plně definovaná, ale oproti první verzi klade důraz na zakulacené rohy a podporu denního a nočního režimu. V aplikaci se pokusím najít propojení mezi starým a novým designem a implementovat ho.

2.2.7 Testování

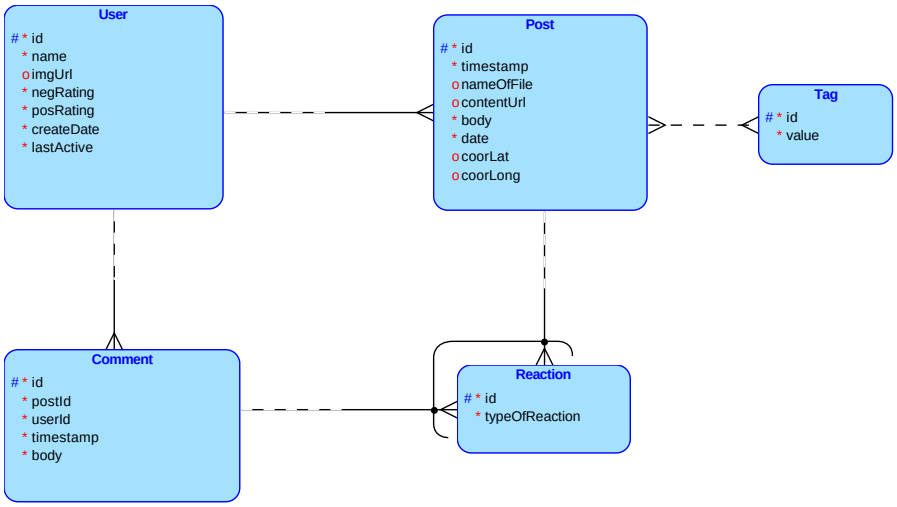
Pro testování aplikace bych chtěl využít možnost automatického průchodu aplikace, abych mohl po každém vydání nechat aplikaci otestovat zda neobsahuje nějaká kritická místa, kde by došlo k pádu. Účelem těchto testů tedy není průchod specifického scénáře, ale volba náhodného průchodu aplikace ideálně na několika různých zařízeních. Pro tyto účely je naprosto ideální služba Google Play Console, která nabízí přesně tuto funkcionalitu, a je zde možné otestovat aplikaci na více než 12 000 různých virtualizovaných zařízeních. Testování se spustí automaticky po nahrání nové verze aplikace, před vypuštěním do Alpha, Beta, nebo vypuštění do produkčního kanálu.

2.2.8 Crashlytics

Jakmile je aplikace spuštěna, je velmi složité sledovat její případné chyby. K tomu jsem se rozhodl do aplikace integrovat Crashlytics systém, který případné pády aplikace nahlásí a uloží anonymizovaný záznam situace, kde chyba vznikla. Opět zde použiji funkci od společnosti Google, a to konkrétně Firebase Crashlytics. Všechna hlášení chyb tedy budu mít zaznamenána na Firebase, a je i možnost nechat si nahlásit chybu pomocí emailu. Firebase následně vytvoří i statistiky chyb na časové ose.

2.3 Databázový model

Základem aplikace bude databázová struktura rozdělená do 5 hlavních částí: Uživatel, Příspěvek, Tag, Komentář a Reakce, kterou znázorňuje schéma 2.2. Každý uživatel může vytvářet více komentářů a příspěvků. Na komentáře a příspěvky je možné reagovat, tedy přidat reakci buď k příspěvku nebo ke komentáři. Komentář může obsahovat jeden nebo více tagů a stejně tak i opačně jeden tag může být použit ve více příspěvcích.



Obrázek 2.2: Databázové schéma

2.4 Návrh UI

UI, tedy User Interface, přeloženo jako uživatelské rozhraní, bylo navrhované tak, aby uživatel měl možnost projít celou aplikaci s co nejmenší hloubkou zanoření, tedy aby uživatel musel proklikat co nejméně kroků, aby dosáhl cíle. Domovskou obrazovka obsahuje navigační prvek BottomNavigationView[4], který se stará o přepínání mezi 3 obrazovkami: Home, Search a Me.

2.4.1 Domovská obrazovka

Na obrazovce Home jsou zobrazeny všechny příspěvky od nejaktuálnějšího po nejstarší. Každý příspěvek je složen z hlavičky, ve které se nachází obrázek autora a jeho jméno, těla příspěvku obsahující text a případně přílohy v podobě souřadnic, obrázku, nebo dokumentu a spodní lištu, ve které jsou zobrazeny pozitivní i negativní reakce a případně i ikona přidání komentáře.

2.4.2 Vyhledávání

V horní části obrazovky Vyhledávání je okno, do kterého je možné zadat hledaný tag. Těsně pod tím jsou zobrazeny 3 nejvíce používané tagy. Při zadávání textu se zobrazuje nápověda těsně pod nejpopulárnějšími výsledky a dynamicky se mění. Po kliknutí na vybraný tag se zobrazí obrazovka obsahující výsledky hledání.

[4]Navigační prvek umístěný na spodní části obrazovky.

2.4.3 Profil

Záložka Profil obsahuje základní informace o uživateli a jeho obrázek. Pod informacemi se nachází seznam sdílených příspěvků od daného uživatele. Vpravo nahoře je ikona nastavení, pomocí které ho můžeme zobrazit.

2.4.3.1 Statistiky uživatele

Na profilu uživatele jsou viditelné dvě statistické informace, a to reputace a počet sdílených příspěvků. Počet sdílených příspěvků je tedy číslo, které ukazuje, kolik příspěvků uživatel vytvořil. Reputace je vypočítána jako rozdíl pozitivních reakcí a negativních.

2.4.4 Nastavení

V nastavení má uživatel možnost měnit celý vzhled aplikace volbou mezi temným, nebo světlým režimem. Je zde také možnost změnit jméno uživatele, případně se z aplikace odhlásit.

Kapitola 3

Implementace Backendu

Implementace backendu pro mě byla velkou výzvou. Nebylo to jen tím, že jsem backend pro aplikaci v takové míře ještě netvořil, ale také z toho důvodu, že jsem se rozhodl tvořit aplikace v NoSQL databázi Firebase, která se velmi liší od více používaných relačních databází, se kterými jsem měl možnost se seznámit dříve.

Jak jíž bylo zmíněno, pro projekt APAnet jsem se rozhodl využít cloudovou platformu Firebase. V projektu budu implementovat Firebase Database, Crashlytics, Storage a Authentication.

3.1 Firebase Realtime Databáze

Data jsou ve Firebase uložená ve formátu JSON. To umožňuje rychlé úpravy a snadné vyhledávání podle id. Celá databáze je realizovaná pomocí klíčů, tedy každý záznam má vlastní klíč, podle kterého je možné přistoupit ke konkrétnímu objektu. Každý prvek drží odkaz v podobě klíče na sebe a ideálně i na svého předchůdce, aby zajistil možnost svého případného odebrání. Níže budu popisovat vnitřní implementaci databáze. Jelikož jsem nenašel žádný zavedený způsob zobrazení NoSQL databáze, definoval jsem zde vlastní.

3.1.1 Notace

V této notaci používám tabulku s modrou hlavičkou, se zelenou hlavičkou a propojení pomocí čar. Tabulka s modrou hlavičkou značí objekt, tabulka se zelenou hlavičkou značí Enum[5] a čára, která asociaci tabulkami těmito mezi prvky.

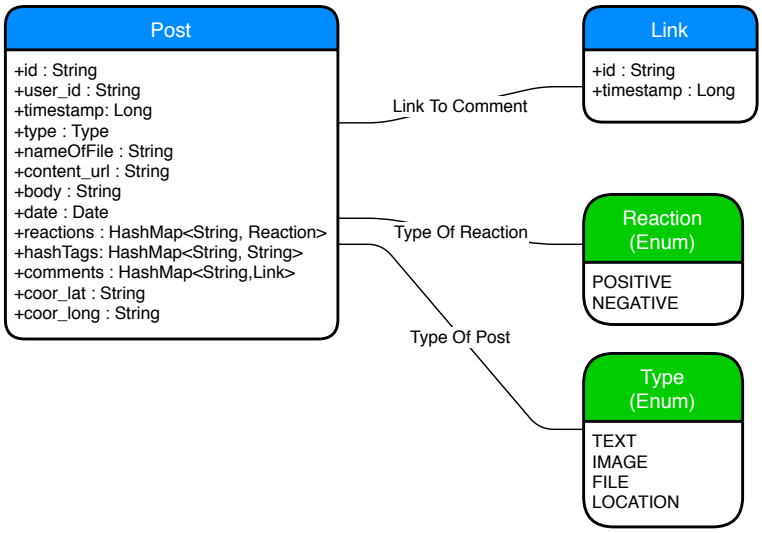
3.1.2 Popis databáze

Zde popíši strukturu jednotlivých částí databáze.

[5] Výčtový typ

3.1.2.1 Příspěvek

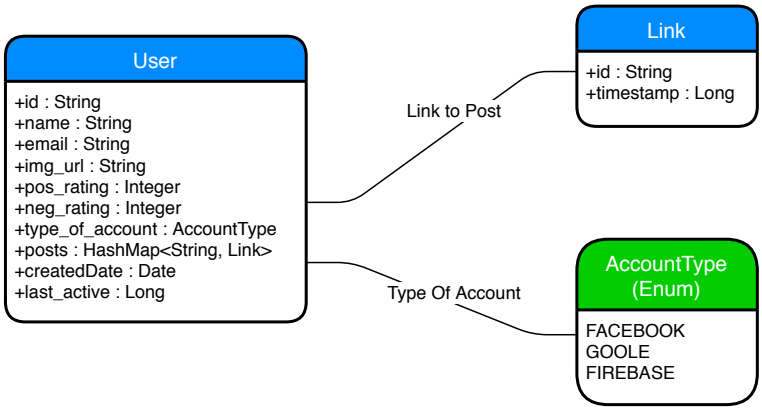
Seznam všech příspěvků v aplikaci.



Obrázek 3.1: Příspěvek

3.1.2.2 Uživatel

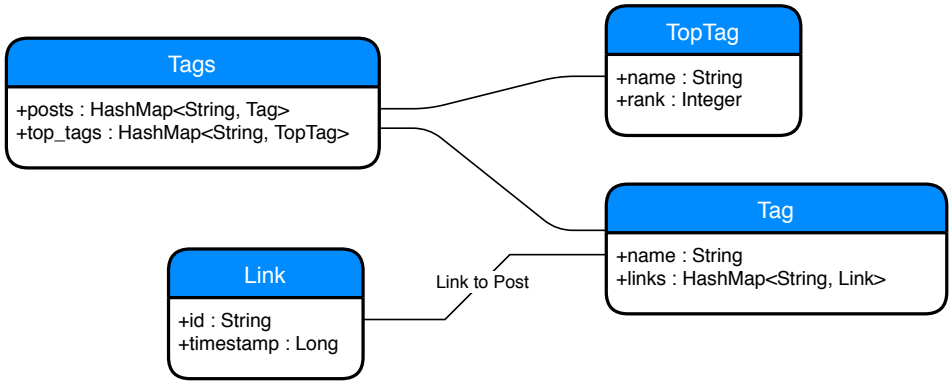
Místo, kde jsou uloženy všechny informace o uživateli a seznam odkazů na sdílené příspěvky.



Obrázek 3.2: Uživatel

3.1.2.3 Tag

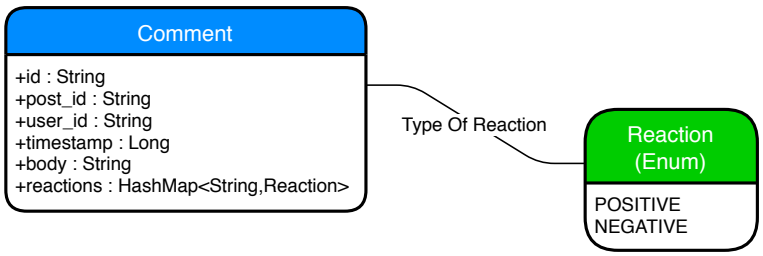
Tag (viz obrázek 3.3) slouží k identifikaci příspěvků s podobným tématem. Seznam všech použitých tagů v aplikaci je rozdělen na seznam odkazů na příspěvky obsahující onen tag a na seznam nejčastěji používaných tagů. Toto rozdělení primárně slouží k rychlému zobrazení nejpoužívanějších tagů, tedy místo toho, aby se vždy musely projít všechny tagy a zjistit počet použití, seznam je rovnou řazen, pouze z databáze vezme se prvních n prvků s nejvyšším počtem použití řazených sestupně.



Obrázek 3.3: Tag

3.1.2.4 Komentář

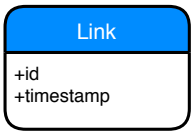
Seznam všech komentářů (viz obrázek 4.6) pod příspěvky v aplikaci. Později je na ně odkazováno pomocí odkazů.



Obrázek 3.4: Komentář

3.1.2.5 Odkaz

Odkaz (viz obrázek 3.5) na nějaký nějaký objekt v databázi. Obsahuje pouze identifikátor cílového objetu, který je identický s jeho identifikátorem, a časové razítko, díky kterému je možné odkazy chronologicky řadit.



Obrázek 3.5: Odkaz

3.2 Firebase úložiště

Struktura úložiště (viz obrázek 3.6) je navržena přesně pro potřeby aplikace APAnet. Základní rozdělení je na data související s profilem uživatele a na data související s daty jednotlivých příspěvků. Unikátním identifikátorem je zde klíč uživatele, respektive příspěvku. Ve složce identifikované klíčem jsou již nahraná odpovídající data.

```
post
    postId ..... identifikátor příspěvku
        file ..... dokument nebo obrázek
profile
    userId ..... identifikátor uživatele
        profileImg ..... profilový obrázek
```

Obrázek 3.6: Struktura úložiště Firebase Storage

3.3 Firebase authentication

Pro přístup do databáze je nutné, aby uživatel byl přihlášen. Jak již bylo zmíněno, jsou podporovány tři typy přihlášení, a to přes účet Facebook, Google a pomocí emailu a hesla. Všechny tyto způsoby přihlášení musí být explicitně v rozhraní povoleny. Zde se také spravují všichni uživatelé, kteří mají přístup do aplikace. Pro přihlášení pomocí emailu a hesla jsou zde vytvořeny vzory emailu, které se odešlou na uvedenou emailovou adresu uživatele pro ověření účtu a resetování hesla.

Kapitola 4

Implementace Android

Pro implementaci aplikace jsem zvolil Android Studio ve verzi Canary. V době, kdy jsem začínal s implementací aplikace APAnet, ještě nebyly novinky představené na Google I/O 2018 ve stabilní verzi, proto jsem se rozhodl použít Android Studio Canary, tedy lehce testovanou verzi vypuštěnou v předstihu před stabilní verzí. Pro implementaci jsem chtěl využít nový způsob navigace mezi fragmenty a nový způsob komprimace dat. V této kapitole se budu věnovat implementaci nejen designu, ale i jednotlivých částí aplikace.

4.1 Základní komponenty

Zde představím důležité komponenty Androidu použité v aplikaci pro uvedení čtenáře do kontextu.

- *Aktivita* je klíčovou komponentou aplikace pro systém Android. [18]
- *Fragment* představuje chování nebo část uživatelského rozhraní v aktivitě. [19]
- *RecyclerView* je flexibilní zobrazení v omezeném okně pro velké množství dat. [20]
- *Dialog* je malé okno, které uživatele vyzve k rozhodnutí nebo k zadání dalších informací. [21]
- *ViewHolder* popisuje pohled na položku a metadata o jejím umístění v RecyclerView. [22]
- *Adapter* poskytuje přístup k jednotlivým datovým položkám. [23]

4.2 Uživatelské rozhraní

APAnet je zaměřen na zařízení s API 21 a vyšším. Minimální verze Androidu je tedy Android 5.0. V této verzi byla přidána podpora Material Designu. To bylo nezbytné pro použití vlastností layout, jako je například tag „elevation“, který zajišťuje vizuální vyzdvižení prvku nad ostatní. I přes to, že stále ještě není plně definovaný, snažil jsem se podobu jednotlivých komponent navrhovat podle Material Designu 2.

Problém, který jsem zde řešil, byl, že při otočení přihlašovací obrazovky do pozice Landscape tlačítka pro přihlášení zabírala příliš mnoho prostoru. Bylo tedy nutné navrhnout specifický layout pro Portrait[6] i Landscape[7] obrazovky, tedy vytvořit layout, který obsahoval stejné prvky se stejnými id daných prvků, ale s jiným rozložením na obrazovce.

Další náročný krok bylo nastavení temného režimu. Standardně se používá metoda, kdy do Shared Preferences[8] uložím informaci, zda používám světlý režim, nebo temný režim, a při vytváření layoutu pro každý prvek zvolit odpovídající barvy. Tato metoda by ovšem přinesla mnoho zbytečného kódu navíc. Zde jsem využil nový DayNight motiv, který se po zavolání postará o vykreslení odpovídajících barev podle hodnot z balíčku „values“ nebo „values-night“. Metodu pro zobrazení nočního režimu jsem ještě musel rozšířit o změnu barvy status baru, který se tímto způsobem nedají změnit.

4.2.1 Font

Namísto základního fontu jsem použil volně dostupný font Poppins [5] Regular od společnosti Google.

4.2.2 Barvy

APAnet obsahuje dvě základní barevné palety. Jednu pro světlý (výchozí) režim a druhou pro temný režim.

Název	Světlý režim	Vizualizace	Temný režim	Vizualizace
Background	#FAFAFA		#424242	
Post Background	#FFFFFF		#323232	
Tint	#40A934		#FFC400	
Text	#6B6B6B		#FEFEFE	

Tabulka 4.1: Základní barvy aplikace

[6] Orientace obrazovky na výšku

[7] Orientace obrazovky na šířku

[8] Shared Preferences je jedno z míst, kam uložit data aplikace, aby i po zavření zůstala v paměti.

(a) Ikona

(b) Kulatá ikona

Obrázek 4.1: Ikony

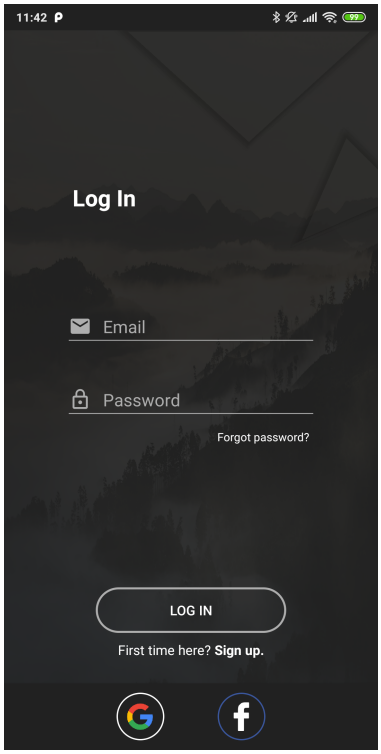
4.2.3 Ikona

Pro aplikace pro operační systém Android je vždy potřeba vytvořit kulaté i čtvercové ikony, jelikož existuje více grafických nadstaveb a můžou používat různé typy ikon. Popředí ikony je vytvořeno z loga Apakrychle [4] a pozadí je vyplněno barvou #000000, tedy černou. Takto jsem vygeneroval všechny typy ikon pro aplikaci za použití nástroje Image Asset v Android Studiu (viz obrázek 4.1). Použití ikony bylo se souhlasem autora.

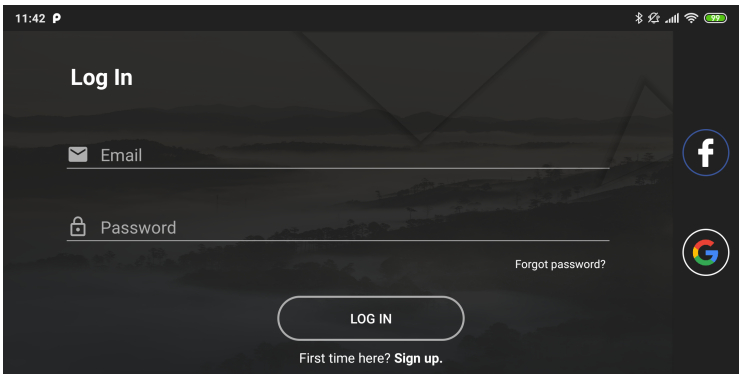
4.2.4 Přihlášení

Jak již bylo zmíněno, do aplikace je možné se přihlásit pomocí několik různých účtů. Aby bylo možné všechny tyto možnosti poskytnout na jedné obrazovce, musel jsem vytvořit dvě různá rozložení obrazovky pro tuto akci. Jedno je pro zobrazení na výšku (viz obrázek 4.2a) a druhé pro zobrazení na šířku (viz obrázek 4.2b), tedy vytvořit soubory se stejným názvem a stejnými prvky, jen jiným rozložením, jeden ve složce „layout“ a druhý ve složce „layout-land“.

Pokud uživatel ještě nemá účet, je možné zvolit možnost registrace pomocí posunutí doprava, případně kliknutím na tlačítko. Mechanismus pro ono posouvání bylo poměrně složité implementovat, jelikož abych zajistil možnost posunutí v jakémkoliv místě na obrazovce, musel jsem přidat vrchní vrstvu, která reaguje na posunutí. V případě, že uživatel chce zvolit účet třetích stran, je zde zobrazena ikona Googlu i Facebooku. Po kliknutí na ni je automaticky přihlášen k aktivnímu účtu. V případě, že je aktivních účtů přihlášeno více, zobrazí se dialog 4.3b, ve kterém si uživatel vybere účet, který chce použít pro přihlášení.

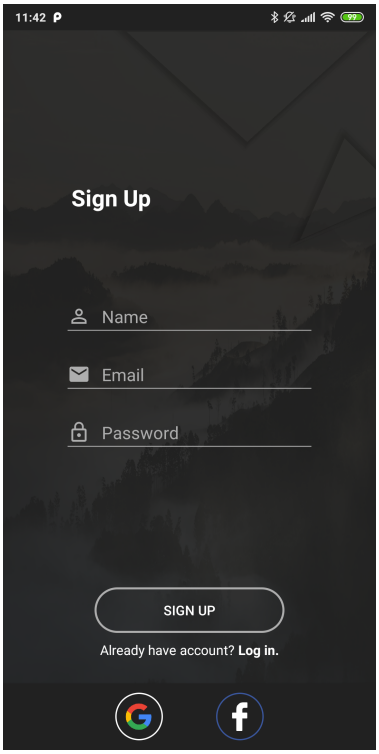


(a) Přihlášení

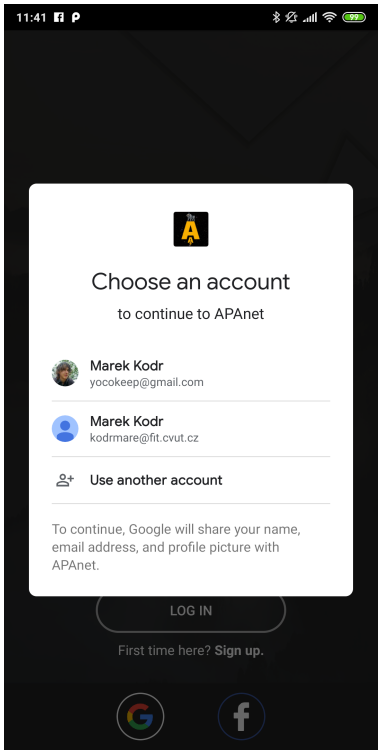


(b) Přihlášení Landscape

Obrázek 4.2: Přihlášení



(a) Registrace



(b) Přihlášení přes Goodle

Obrázek 4.3: Registrace a přihlášení

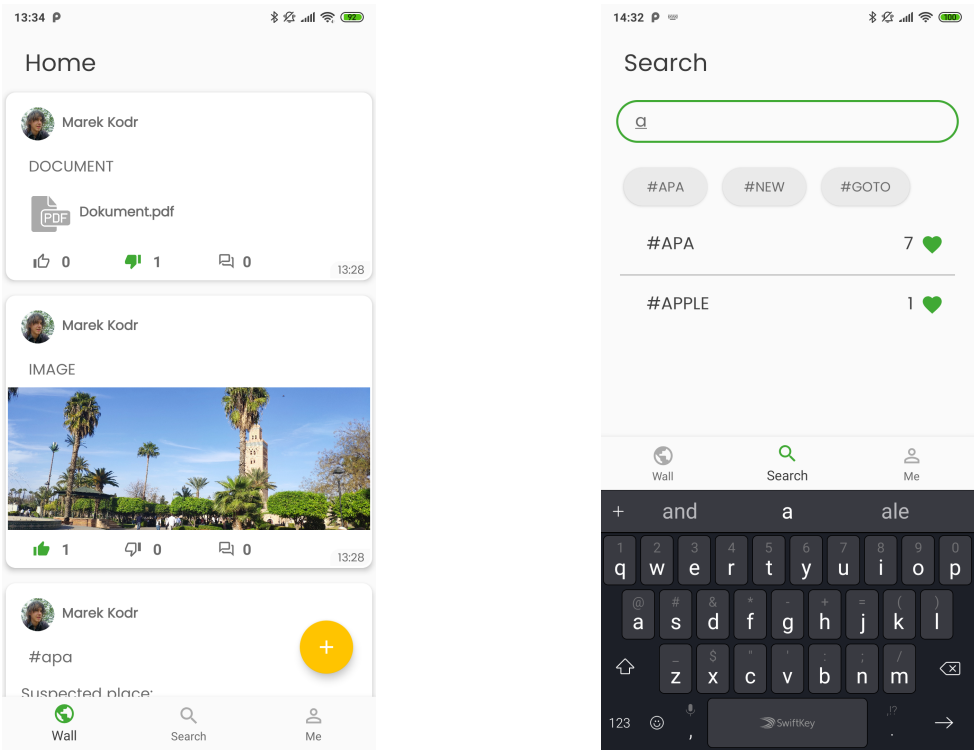
4.2.5 Domovská obrazovka

Nejvýraznější komponentou domovské obrazovky je RecyclerView[9], který se stará o zobrazování jednotlivých příspěvků. V pravém dolním rohu se nachází SpeedDial[10] tlačítko a nabídne buď nahlášení výskutu Apakrychlí nebo sdílení příspěvku (viz obrázek 4.4a).

4.2.6 Vyhledávání

Při začátku psaní vyhledávaného tagu se začne zobrazovat kontextová nápověda podle aktuálního zadaného řetězce. Při kliknutí na napovězený výsledek nebo na navržený tag následuje přechod do seznamu vyhledávaných tagů (viz obrázek 4.4b).

[9] Android komponenta, která umožňuje znovu používat již vytvořený layout.
[10] Tlačítko které se po kliknutí zobrazí rychlé menu.



(a) Domovská obrazovka (b) Vyhledávání

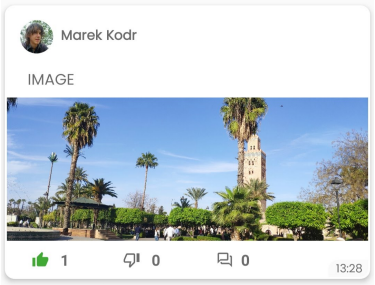
Obrázek 4.4: Výřezy obrazovek

4.2.7 Nastavení

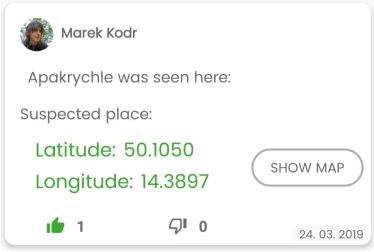
Nastavení (viz obrázek 4.8) je rozloženo na 3 části: General, Personal a Others. Každá z částí obsahuje nastavení specifické pro danou kategorii. V kategorii General je možné měnit motiv aplikace a dimenzi, pokud na to máme dostatečná oprávnění. V Personal je možné měnit nastavení vztažené k uživateli, zde lze tedy měnit jméno, a v kategorii Others se nachází možnost odhlášení z aplikace.

4.2.8 Profil

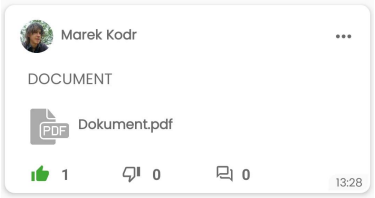
Profil uživatele je sestaven z několika různých prvků, zobrazených pomocí recyclerView, držící různé typy zobrazených prvků. Jako první zobrazený prvek je vždy Profil uživatele, tedy speciálně navržený layout zobrazující jméno, profilový obrázek, reputaci a počet sdílených příspěvků. V případě, že se nejedná o profil aktuálně přihlášeného uživatele, tak je ještě zobrazena informace, kdy byl uživatel naposledy aktivní, a je odebrána možnost měnit údaje o uživateli. Další položky v recyclerView jsou příspěvky, které uživatel sdílel. Z implementačního hlediska je toto realizované tak, že načtu z Firebase



(a) Obrázek



(b) Lokace



(c) Dokument

Obrázek 4.5: Typy příspěvků

databáze informace o uživateli, které obsahují nejen jméno, url obrázku a statistiky, ale také seznam odkazů na příspěvky s časovým údajem. To mi umožní stáhnout chronologicky seřazené příspěvky bez toho, aniž bych věděl cokoliv o jejich obsahu. Následně pro každý příspěvek v daném ViewHolderu[11] načtu data o příspěvku pomocí jeho klíče a příspěvek zobrazím, jak je možné vidět na výřezu obrazovky profilu (viz obrázek 4.7).

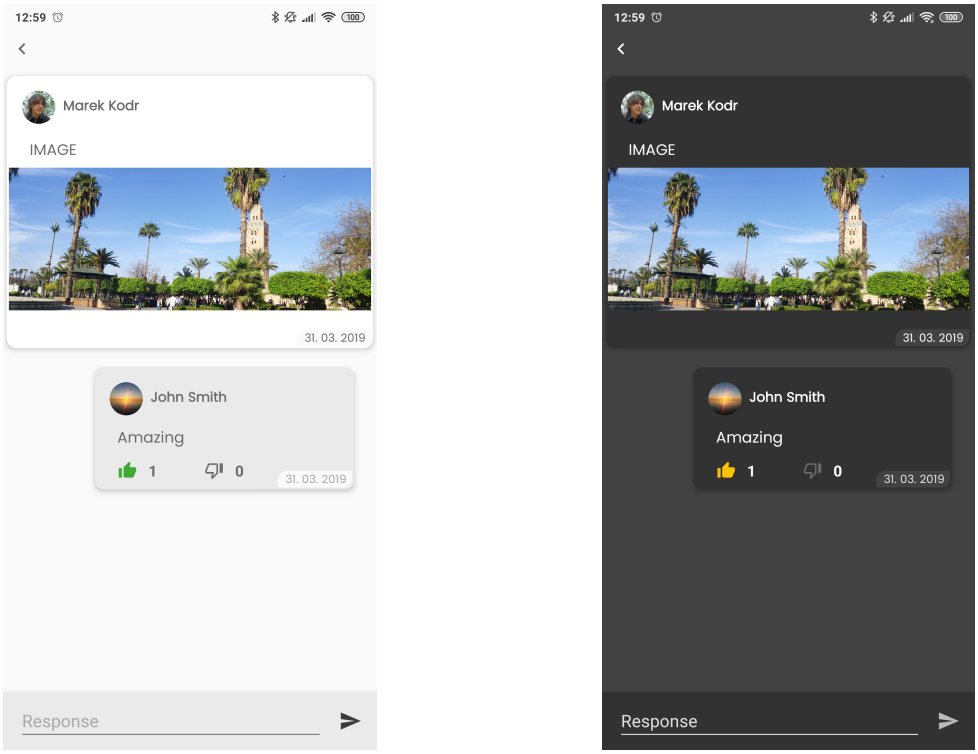
4.2.9 Příspěvek

V aplikaci existují 3 typy příloh k příspěvku: obrázek (viz obrázek 4.8a), dokument (viz obrázek 4.8d) a lokace (viz obrázek 4.5b). Každý z nich má svůj specifický layout [12] a specifické vlastnosti odpovídající obsahu.

4.2.10 Komentář

Přidávání komentářů je možné v detailu příspěvku. Do toho je možné se dostat kliknutím na ikonu komentáře u příspěvku, nebo přes menu a zvolení možnosti *komentář*. Následně je zobrazena obrazovka obsahující příspěvek jako první položku adaptéru a pod ním jednotlivé komentáře, které jsou druhým typem layoutu tohoto RecyclerView. Pro přidání komentáře je nutné na spodní liště vyplnit text a zmáčknout tlačítko *odeslat*. Tímto je komentář přidán a zobrazen pod ostatními (viz obrázek 4.6).

[11] Třída, která mi drží instanci zobrazeného prvku.
[12] Rozložení



(a) Komentář světlý

(b) Komentář tmavý

Obrázek 4.6: Komentář

4.2.11 Temný režim

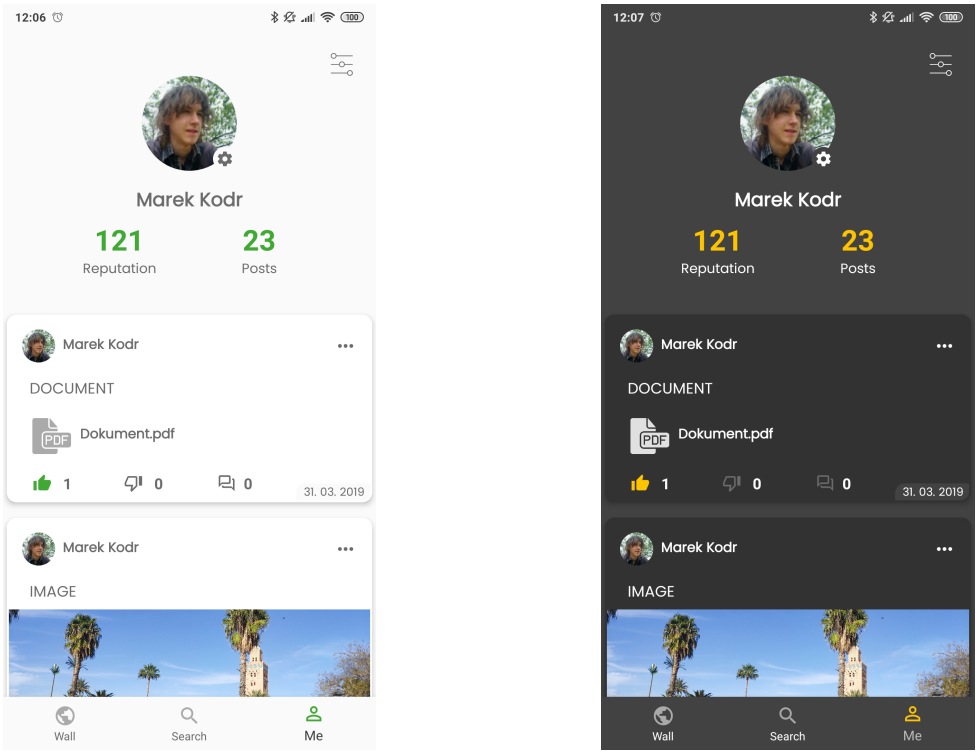
Aplikace je navržena tak, aby nativně podporovala dva grafické režimy zobrazení – jeden světlý a druhý temný. V Shared preferences je uložena informace o tom, jaký režim je aktuálně aktivní, a při vytváření jednotlivých fragmentů se na začátku zvolí odpovídající zdroj barvy podle aktuálního režimu. Zobrazování a porovnání jednotlivých barevných změn by bylo příliš objemné na počet obrázků, proto zde uvedu pouze pár příkladů na Profilu (viz obrázek 4.7) a Nastavení (viz obrázek 4.8).

4.3 Přihlášení pomocí účtu třetích stran

Do aplikace je možné se přihlásit pomocí emailu a hesla, Google účtu a nebo Facebook účtu.

Pro vytvoření možnosti přihlášení Google účtu bylo zapotřebí pouze povolit tyto přihlašovací metody v rozhraní Firebase a přes integrované API GoogleSignIn klienta si vyžádat token.

Přihlášení pomocí emailu a hesla probíhalo taktéž přes rozhraní Firebase,



(a) Profil světlý

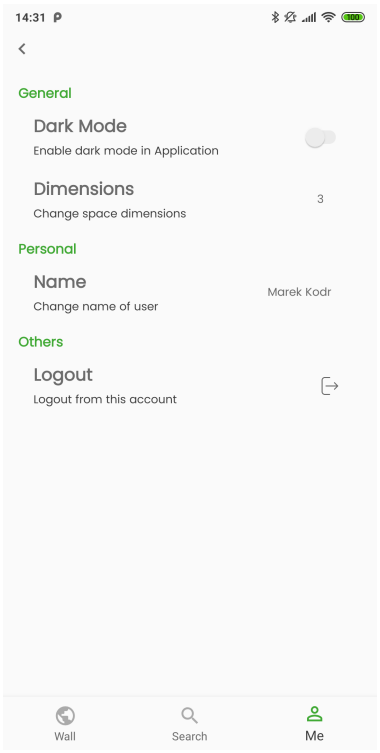
(b) Profil tmavý

Obrázek 4.7: Profil

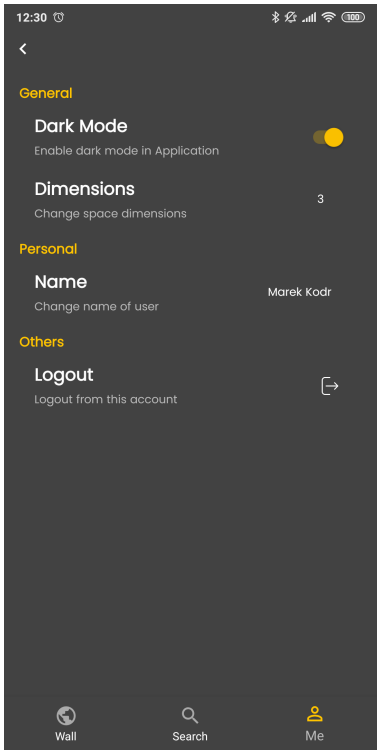
kdy nejprve bylo nutné rozhodnout, zda uživatel je již registrovaný, či ne. V případě, kdy uživatel ještě neměl účet, bylo nutné jej vytvořit. Přes Firebase rozhraní jsem tedy nejprve vytvořil uživatelský účet, odeslal ověřovací email na zadanou adresu a následně uživatele do aplikace přihlásil. V případě, že uživatel již účet měl, po zadání validního jména a hesla stačilo uživatele přihlásit přes metody rozhraní.

Nejtěžší ovšem byla možnost přihlášení přes Facebook. První nutnou věcí bylo vytvořit profil pro aplikaci v portálu Facebook for developers. Následně importovat specifické Facebook SDK pro Android, vygenerovat tajné klíče a propojit s aplikací. Dokumentace nebyla úplně aktualizovaná, tedy implementace vyžadovala značné pátraní.

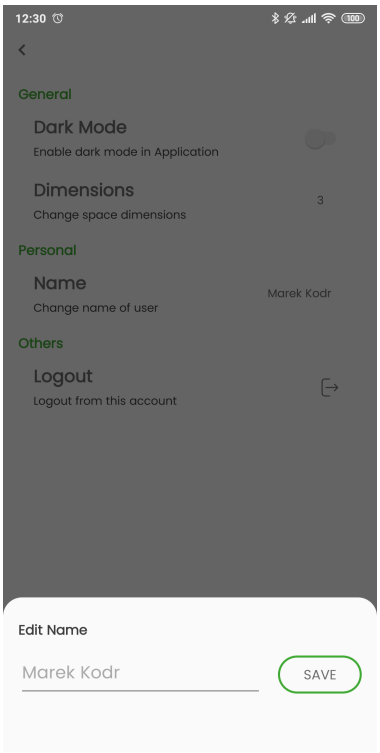
Po úspěšném prvním přihlášení pomocí jakékoliv metody se vytvořil uživatel s informacemi získatelnými z Googlu, Facebooku, případně z formuláře pro vytvoření interního účtu. Informace o uživateli se následně nahrály do Firebase databáze.



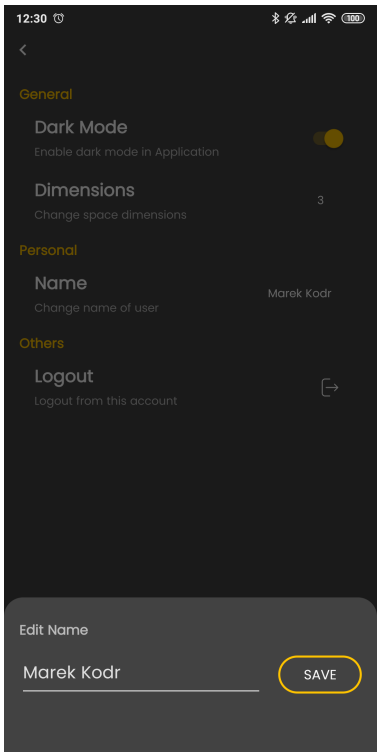
(a) Nastavení světlé



(b) Nastavení tmavé



(c) Nastavení dialog světlé



(d) Nastavení dialog tmavé

Obrázek 4.8: Nastavení

4.4 Dynamické načítání příspěvků

Stránkování a dynamické načítání příspěvků byla jedna z nejtěžších částí implementace grafické části. Adaptér, který zobrazuje aktuální seznam načtených příspěvků, obsahuje 3 různé prvky, které může vykreslit.

První z nich je samotný příspěvek, tedy grafická reprezentace načteného příspěvku. Když se již blížil konec aktuálně načteného seznamu, spustila se metoda pro stažení další stránky dat a zobrazil se ProgressBar[13] jako poslední prvek seznamu. Pokud se nepodařilo další data načíst, indikátor načítání je odebrán, aby uživateli dal najevo, že načítání bylo ukončeno. Pokud již byla načtena všechna data z databáze, zobrazí se poslední typ prvku, upozornění, že již další data v databázi nejsou.

4.5 Vyhledávání podle tagu

V sekci vyhledávání je možné vyhledat příspěvek podle tagu, který příspěvek obsahuje ve svém textovém těle. Z hlediska implementace by bylo velmi výpočetně náročné prohledávat všechny příspěvky, zda obsahují hledaný tag, a následně ho zobrazovat, proto jsem zde zvolil jiný způsob. Při uložení příspěvku si uložím do vedlejší struktury informace o tagu, který daný příspěvek obsahuje. Tuto informaci rozdělím do dvou rozdílných struktur, jak již bylo popsané výše. V jedné z nich mám uložený počet příspěvků k jednotlivým tagům, to mi umožní zobrazit nejpoužívanější tagy, a v druhé struktuře mám uložené odkazy na jednotlivé příspěvky ke každému tagu. Ve výsledku tedy zadám hledaný tag, případně zobrazím kontextovou nápovědu, pokud existuje, a po vybrání hledaného tagu již mám přístup k seznamu odkazů na jednotlivé příspěvky, které můžu takto zobrazit.

4.6 Nahrávání obrázků a dokumentů

Obrázky i příspěvky jsou nahrávány do úložiště (viz obrázek 3.6) Firebase Storage. Celý přenos je řízený pomocí knihoven přímo od Firebase. Po úspěšném nahrání zjistím URL nahraného souboru a uložím ji buď do profilu uživatele, nebo k příslušnému příspěvku. Odtud je možné následně data získat přes uloženou URL.

4.7 Stateful layout

Každá aplikace připojená k internetu má určité stavy. Stateful layout se stará o jejich zobrazení. Konkrétně se zde jedná o stav načítání, stav, kdy jsem data načetl a zobrazuji je, a stav, kdy nemám připojení k internetu. V aplikaci jsem

[13] Indikátor stavu načítání

tento přístup realizoval pomocí broadcast receiveru[14], který sleduje, zda je zařízení připojeno k internetu. Při každé změně stavu pak obdrží notifikaci, se kterou je možné dále pracovat, v tomto případě je to informace, zda je zařízení připojené k internetu, nebo ne. Pokud obdržím z broadcast receiveru zprávu o tom, že zařízení není připojeno, zobrazím tuto informaci na obrazovku a skryji ostatní data. V případě, že přejdu ze stavu nepřipojeno do stavu připojeno, začnu načítat data, tedy zobrazím indikátor načítání, a ve chvíli, kdy obdržím data, tak je zobrazím.

4.8 Android Jetpack

Android Jetpack je sada nástrojů doporučených pro vývoj aplikací. V projektu APAnet několik z nich také využívám.

4.8.1 Navigation Component

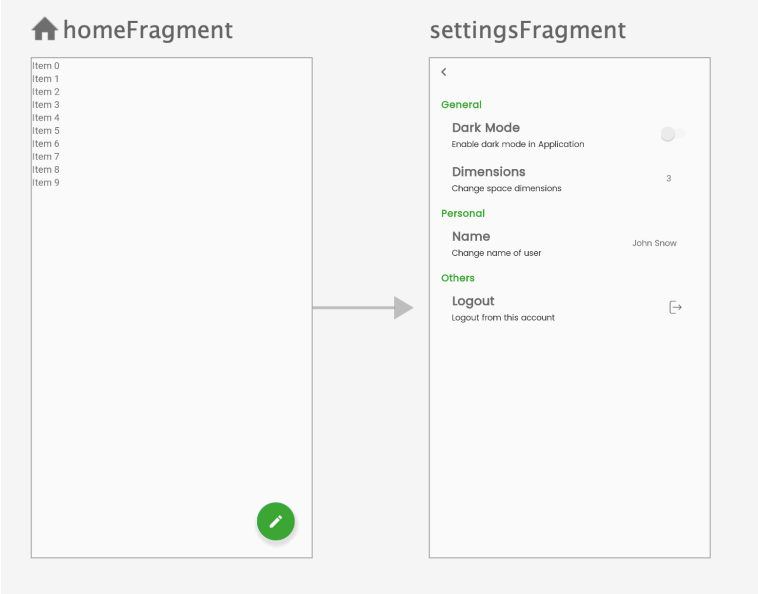
Navigation Component[8] byl představen v Android Studiu 3.3. Jedná se velký krok kupředu, co se týče navigace mezi fragmenty, jelikož doposud nebylo možné nijak graficky zachytit přechody mezi jednotlivými obrazovkami. Jde tedy o velké zpřehlednění navigace v aplikaci. Podobný způsob grafického znázornění navigace mezi fragmenty byl již několik let součástí vývojového prostředí XCode[15] pro vývoj mobilních aplikací, ale do Android Studia se dostal až na podzim 2018.

```
<fragment
       android:id="@+id/homeFragment"
       android:name="cz.cvut.kodr.marek.apanet.fragment.HomeFragment"
       android:label="fragment_home"
       tools:layout="@layout/fragment_home" >
       <action
           android:id="@+id/action_homeFragment_to_settingsFragment"
           app:destination="@id/settingsFragment" />
</fragment>

<fragment
       android:id="@+id/settingsFragment"
       android:name="cz.cvut.kodr.marek.apanet.fragment.SettingsFragment"
       android:label="fragment_settings"
       tools:layout="@layout/fragment_settings" />
```

Výpis 4.1: Textová reprezentace obrázku Navigation Component výše 4.9

[14] Metoda, která obdrží notifikace o události, pro které je registrovaná.
[15] Vývojové prostředí pro tvorbu aplikací pro iOS



Obrázek 4.9: Ukázka použití Navigation Component – Editor

4.8.2 LiveData a ViewModel

ViewModel [10] a LiveData [7] jsou další velmi užitečné komponenty z této sady knihoven. Umožňují a skoro i nutí uživatele do extrahování logiky z aktivit a fragmentů do ViewModelu a za použití Live Dat je předávat do aktivit. Největší výhodou tohoto přístupu je to, že aktivita se při otočení obrazovky zničí a znovu vytvoří. Kdyby data byla uložena v aktivitě, byla by při otočení obrazovky zničena, ale při použití ViewModelu jsou data uložena v něm a pomocí LiveDat, která mají v sobě implementovaný Observer pattern[16], tak se zpropadují do aktivity. Při opětovném vytvoření aktivity se data opět zobrazí. V následující ukázce kódu předvedu interakci mezi ViewModelem a Fragmentem za použití Live Dat. Podobný způsob interakce jsem využíval i aplikaci.

[16]Návrhový vzor, který umožňuje sledovat změny v hodnotách proměnných.

4.8.2.1 Ukázkový ViewModel

Nejprve definuji proměnnou data typu MutableLiveData. To je sledovatelná proměnná, kterou vrátím ve funkci downloadData. Jakmile se obsah proměnné data změní, sledující této funkce bude upozorněn.

```
class SampleViewModel : ViewModel(){

        private val data: MutableLiveData<String> by lazy {
            MutableLiveData<String>() }

        fun downloadData() : LiveData<String>{
        \\ post data
            data.postValue("Some Data")

            return data
        }
}
```

Výpis 4.2: Ukázka použití ViewModelu

4.8.2.2 Ukázkový Fragment

Ve fragmentu nejprve musím zvolit správný ViewModel a připnout ho k aktuálnímu fragmentu. Následně mohu sledovat sledovatelné funkce a proměnné třídy ViewModel a být upozorňován na změny.

```
class SampleFragment : Fragment() {

        private lateinit var viewModel: SampleViewModel

        override fun onCreateView(
            inflater: LayoutInflater, container: ViewGroup?,
            savedInstanceState: Bundle?
        ): View? {
            // set ViewModel Provider
            viewModel = ViewModelProviders.of(this)
                            .get(SampleViewModel::class.java)

            viewModel.downloadData().observe(this, Observer {
                //do something
            })
            return inflater.inflate(R.layout.fragment_sample,
                                    container, false)
        }
}
```

Výpis 4.3: Ukázka použití ViewModelu ve Fragment

Kapitola 5

Testování

V této kapitole se budu zabývat procesem testování aplikace APAnet.

5.1 Testovací zařízení

Seznam zařízení použitých k testování aplikace

Název	Velikost	Rozlišení (pixels)	Android	Firmware
Xiaomi Mi Mix 2s	6.0	2160 x 1080	9.0	MIUI
Sony Xperia Z2	5.2	1080 x 1920	6.0	Xperia UI
Nokia 6.1	5.5	1080 x 1920	8.0	Android One

Tabulka 5.1: Parametry testovacích zařízení

5.2 Testování a optimalizace autorem

Aplikaci jsem průběžně testoval při implementaci, a to buď v emulátoru, nebo na zařízení Xiaomi Mi Mix 2s, které jsem používal přímo pro vývoj. Po každé větší změně v aplikaci jsem také testoval použitelnost aplikace a správné zobrazení na zařízení Sony Xperia Z2 pro ověření, zda aplikace plynule běží i na starších zařízeních se starší verzí OS Android.

5.2.1 Manuální testování

Manuální testování probíhalo tak, že po implementaci každé větší části jsem prošel různé cesty v aplikaci a případy, které mohly nastat po kliknutí na dané prvky. Například pro přidávání příspěvku jsem se pokusil přidat dokument a zároveň obrázek a následně ho odebral, abych zjistil, zda správně funguje přidávání přílohy.

5.2.2 Testování implementace

Pro testování implementace jsem připravil JUnit testy, které testují, zda na daném zařízení správně fungují jednotlivé knihovny a funkce, například kontrola validity emailové adresy nebo extrahování tagů z těla příspěvku. Spouštění testů oproti klasickým JUnit testům bylo závislé na konkrétním zařízením, jelikož zde také testuji i specifické metody pro Android SDK.

5.2.3 Testy správy paměti

Testování, zda aplikace správně zachází s pamětí, jsem prováděl pomocí aplikace LeakCanary. Tu jsem přidal do závislostí v Gradle a následně inicializoval při spuštění aplikace. Následně se do systému vedle aplikace APAnet nainstalovala ještě aplikace LeakCanary „*A memory leak detection library for Android and Java*“ [2], která při zaznamenání problému zobrazila notifikaci a uložila údaje, které napomohly k jeho identifikaci.

5.2.4 Optimalizace grafického rozhraní

Na zařízení Xiaomi Mi Mix 2s jsem využil nástroj z pokročilých možností pro vývojaře, a to Debug GPU Overdraw, volně přeloženo jako analyzování míst, kde se překrývají jednotlivé elementy. Tento nástroj zapne vizuální zobrazení překreslení prvků. Čím více prvků se překrývá, tím tmavší barva je zobrazena.

5.2.5 Profiler

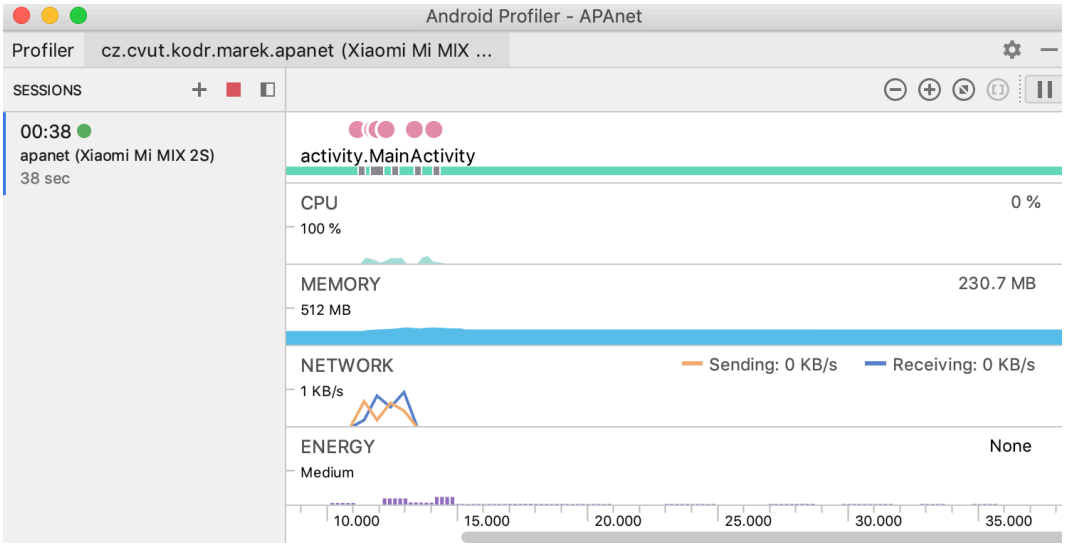
Optimalizaci běhu aplikace jsem prováděl přes vestavěný nástroj Profiler. Tento nástroj v závislosti na verzi Androidu na daném zařízení umožňuje pokročilou analýzu paměti, sítě, využití energie a další za běhu programu [3].

5.3 Testování přes Google Play

Pro testování aplikace přes Google Play Console a Google Play je nutné nejprve vytvořit vývojářský účet. Nejprve je tedy potřeba založit obyčejný Google účet a následně ho zařadit do vývojářského programu a uhradit registrační poplatek. Následně získám přístup do Google Play Console, což je místo, kde mohu nahrát aplikaci na Google Play, možnost alpha a beta testování a další.

5.3.1 Google Play

Google Play je obchod s aplikacemi a mediálním obsahem pro zařízení Android. Je zde možné aplikace stahovat zdarma, případně zakoupit, tedy místo, kde se s aplikací setkává koncový uživatel.



Obrázek 5.1: Profiler

5.3.2 Google Play Console

Do obchodu Google Play je možné aplikace nahrát pomocí Google Play Console. Pro správné uvedení na trh je potřeba vyplnit popis aplikace, zvolit správné ikony a pozadí, určit omezení pro tuto aplikaci, (například kdyby byl obsah nevhodný pro mladší děti) a samozřejmě nahrání a otestování podepsané aplikace. V případě, že aplikace obsahuje povolení pro GPS, kameru nebo další zdroj citlivých informací, je nutné sepsat právní dokument o pravidlech použití a připojit ho k aplikaci.

5.3.3 Generování klíčů

Každá aplikace nahraná na Google Play musí být podepsána autorem. Tento podpis[11] vyžaduje vygenerování podpisových klíčů. To je umožněno přímo pomocí nástroje v Android Studiu pro generování podepsané aplikace, kde je potřeba vytvořit unikátní klíč s unikátním jménem vázaným na uživatele. Následně je potřeba tyto klíče nahrát na Firebase a Facebook for developers portály, aby byl umožněn přístup aplikace i s tímto novým podpisem. Kdyby tento klíč nebyl přidán, nebylo by možné se k těmto službám přihlásit. Vygenerovaná a podepsaná aplikace je tímto připravena pro nahrání na Google Play.

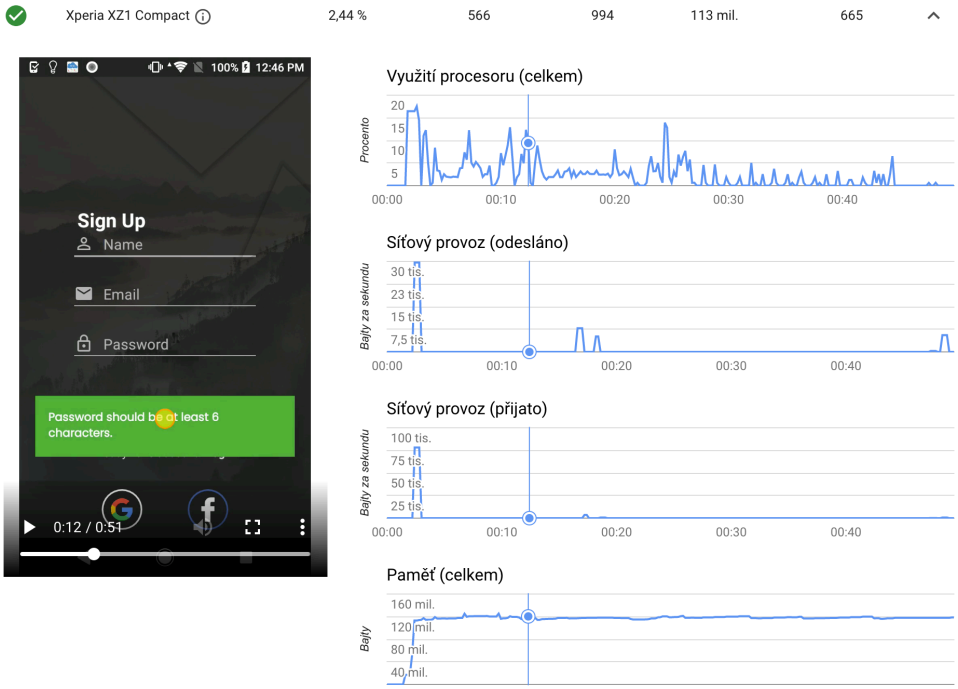
5.3.4 App Release

Aplikaci je možné vydat v několika různých verzích. Pro prvotní testování jsem využil Alpha testovací kanál. To mi umožnilo nahrát aplikaci na Google Play tak, aby aplikace byla přístupná pouze vybraným uživatelům, které jsem

zařadil do testovacího programu zadáním specifické emailové adresy. Následně tomuto uživateli byla odeslána pozvánka do testovacího programu. Po přijetí pozvánky byl uživatel přesměrován na aplikace v Google Play, kde již je možné aplikaci nainstalovat do zařízení.

5.3.5 Google Play Console tests

Před vydáním aplikace je možné nechat aplikaci otestovat na různých zařízeních a sledovat jejich chování za běhu, analýzu paměti, výkonu procesoru, vytížení sítě a mnoho dalších aspektů. K analýze běhu je také připojen video záznam obrazovky, díky čemuž je možné sledovat korelaci mezi provedenou akcí a vytížením zařízení. Toho jsem provedl i u aplikace APAnet. Toto testování zajistí, že i přes to, že nevlastním nějaké zařízení, na kterém bych aplikaci mohl otestovat, můžu alespoň využít virtualizované zařízení zde. Je to skvělý nástroj pro automatickou kontrolu běhu aplikace na velké škále různých zařízení, které by se jinak muselo nahradit manuální. Pro ukázku přikládám snímek obrazovky (viz obrázek 5.2) z testování na virtualizovaném zařízení Xperia XZ1 Compact. Níže přidávám kompletní seznam (viz obrázek 5.3) použitých virtuálních zařízení využitých pro testování.



Obrázek 5.2: Google Play Console Test

Model zařízení	Průměrné využití procesoru (procenta)	Prům. odesláno v síti (bajty/s)	Prům. přijato v síti (bajty/s)	Prům. paměť (bajty)	Doba spouštění (ms)	
Moto Z	1,13 %	2,3 tis.	1,4 tis.	171 mil.	-	
Xperia XZ1 Compact	2,44 %	566	994	113 mil.	665	
Pixel 2	10,93 %	1,4 tis.	4,8 tis.	222 mil.	743	
Pixel 2	1,37 %	1,5 tis.	3,7 tis.	155 mil.	-	
Moto G4 Play	24,04 %	702	2 tis.	149 mil.	1,9 tis.	
Mate 9	11,41 %	1,1 tis.	2,3 tis.	434 mil.	767	
Pixel	25,52 %	1,2 tis.	2,6 tis.	228 mil.	898	
K3 2017	14,95 %	1,2 tis.	4,9 tis.	130 mil.	3,2 tis.	
Galaxy S9	2,47 %	396	217	205 mil.	-	

Obrázek 5.3: Použitá zařízení pro Google Play Console Test

5.4 Firebase Crashlytics

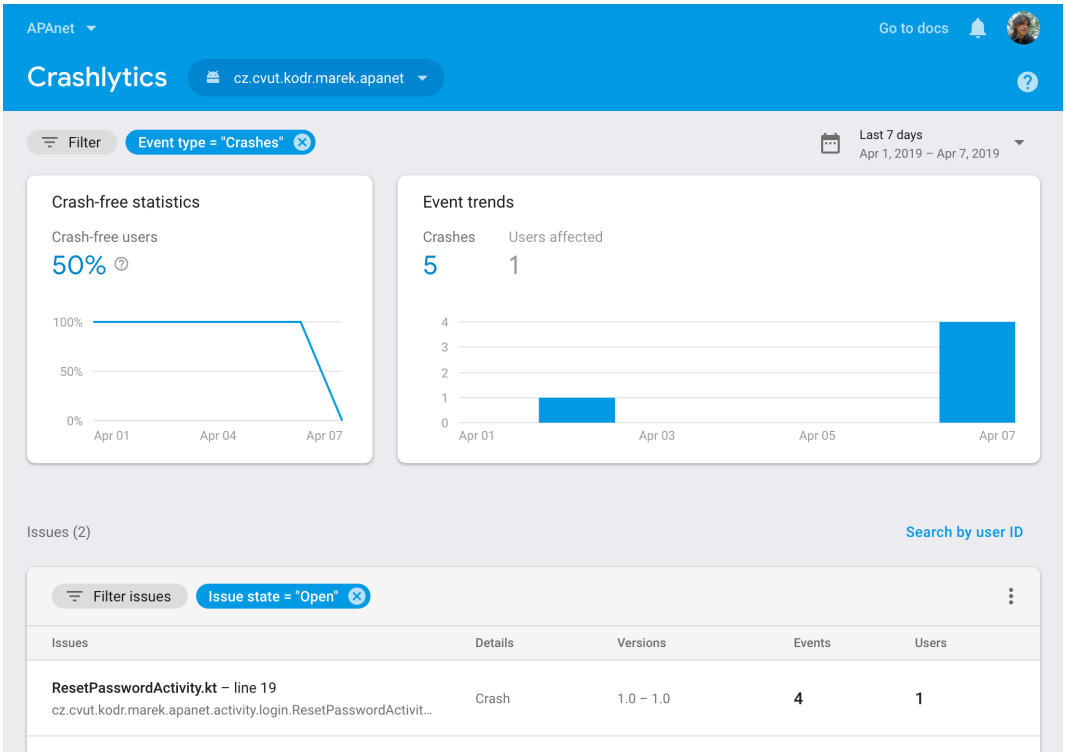
V případě, že je již aplikace nainstalována na zařízeních koncových uživatelů, nebo obecně na zařízeních, ke kterým nemá vývojař přímý přístup, je velmi obtížné sledovat, zda při běhu aplikace nedochází k chybám. Pro takové případy jsem přidal do implementace připojení na Firebase Crashlytics, což je systém umožňující hlášení pádů aplikace s detaily, jako například: Značka a model zařízení, orientace obrazovky, volná paměť RAM, verze operačního systému a další. To je velmi důležitá součást aplikace, například v situaci, kdy bude na nějakých specifických zařízení vznikat nějaká chyba, o které nebudu vědět. Díky těmto informacím budu moci nasimulovat danou chybu, případně otestovat na daném problematickém zařízení. Bez tohoto hlášení chyb by to nebylo možné. Na obrázku 5.4 je ukázána základní obrazovka s hlášením chyb.

5.5 Testovací scénáře

U každého testovacího scénáře je napsáno stručné zadání pro uživatele a pod tím ideální řešení problém. Správných řešení může být více, ale vždy je uvedeno pouze jedno.

5.5.1 Testování uživateli

Jednotliví uživatelé získali přístup k aplikaci pomocí přihlášení přes účet používaný v obchodě Google Play. Přes obdržený odkaz si mohli zobrazit aplikaci APAnet a následně ji stáhnout běžným způsobem do svého zařízení.



Obrázek 5.4: Firebase Crashlytics

5.5.2 Sdílení příspěveku

Testovací scénář pro ověření přidávání příspěvku uživatelem.

5.5.2.1 Zadání

Přidejte příspěvek s tagem „#APA“ a jako přílohu zvolte obrázek.

5.5.2.2 Řešení

Vpravo dole na hlavní obrazovce klikněte na tlačítko *plus*. Následně zvolte Sdílet příspěvek. Na zobrazené obrazovce vyplňte text, do kterého vložíte řetězec „#APA“. Vlevo dole klikněte na ikonu přidat obrázek a vyberte jej. Nakonec klikněte na tlačítko *Sdílet* v pravém horním rohu.

5.5.3 Změna jména

Testovací scénář pro změnu jména uživatele.

5.5.3.1 Zadání

Změňte jméno aktuálně přihlášeného uživatele.

5.5.3.2 Řešení

Jděte na profil uživatele. V pravém horním rohu se nachází tlačítko *Nastavení*, na které klikněte. V sekci personal klikněte na kolonku jméno. Následně se zobrazí bottomSheet[17] menu. Vyplňte nové jméno a zvolte uložit.

5.5.4 Komentáře a Tagy

Testovací scénář vyhledání tagu a přidávání komentáře.

5.5.4.1 Zadání

Vyhledejte naposledy přidaný příspěvek obsahující tag „#APA“ a přidejte k němu komentář.

5.5.4.2 Řešení

V sekci vyhledávání do kolonky vyhledávání zadejte hledaný tag. Zvolte z nabízených výsledků odpovídající tag. Následně se zobrazí všechny příspěvky obsahující tento tag. Na prvním příspěvku shora kliknete na ikonu komentáře. Zobrazí se detail příspěvku, případně pod ním komentáře k němu. Do textového pole v dolní části obrazovky zadáte text odpovědi a kliknete na ikonu *odeslat*.

5.6 Výsledky testů

Testeři pro aplikaci byli vybírání z řad odborného i amatérského publika. Aplikace byly poskytnuty testerům zařazením do alpha testovacího programu aplikace v Google Play Console. Z důvodu ochrany soukromí zde nebudu uvádět informace o testerech aplikace APAnet a budou zde uvedeni pod kódovým označením.

5.6.1 Test Sdílení příspěveku

Výsledky testu přidávání příspěvku uživatelem.

5.6.1.1 Tester 1

Test proběhl bez problému a uživatel vše zvládl ideální cestou.

5.6.1.2 Tester 2

Test proběhl bez problému a uživatel vše zvládl ideální cestou.

[17] Menu, které se objeví zespoda obrazovky.

5.6.1.3 Tester 3

Tester 3 měl na začátku problém indentifikovat tlačítko na přidání příspěvku, ale následně vše zvládl.

5.6.1.4 Tester 4

Test proběhl bez problému a uživatel vše zvládl ideální cestou.

5.6.2 Test Změna jména

Výsledky testu změny jména uživatele.

5.6.2.1 Tester 1

Test proběhl bez problému a uživatel vše zvládl ideální cestou.

5.6.2.2 Tester 2

Test proběhl bez problému a uživatel vše zvládl ideální cestou.

5.6.2.3 Tester 3

Tester poměrně dlouho hledal nastavení uživatele, po několika minutách nastavení našel a úlohu zvládl.

5.6.2.4 Tester 4

Test proběhl bez problému a uživatel vše zvládl ideální cestou.

5.6.3 Test Komentáře a Tagy

Výsledky testu vyhledání tagu a přidávání komentáře.

5.6.3.1 Tester 1

Uživatel byl zmaten při vyhledávání tagu, jelikož není možné přímo stisknout tlačítko *vyhledat*, a přejít na výsledky vyhledávání, ale je nutné vybrat z nabízených nalezených výsledků. Po tomto zjištění, již bylo vše bez problému.

5.6.3.2 Tester 2

Test proběhl bez problému a uživatel vše zvládl ideální cestou.

5.6.3.3 Tester 3

Test proběhl bez problému a uživatel vše zvládl ideální cestou.

5.6.3.4 Tester 4

Test proběhl bez problému a uživatel vše zvládl ideální cestou.

5.7 Hodnocení aplikace testery

Všichni testeři po splnění testovacích scénářů vyplnili hodnocení aplikace. Hodnocení je uvedeno v procentech a platí, že čím vyšší procento, tím lepší hodnocení.

Tester	Použitelnost	Design	Přehlednost	Bezchybovost	Hodnocení
Tester 1	100 %	100 %	100 %	90 %	97,5 %
Tester 2	90 %	100 %	85 %	100 %	93,75 %
Tester 3	80 %	90 %	95 %	95 %	90 %
Tester 4	90 %	70 %	100 %	100 %	90 %
Celkem	90 %	90 %	95 %	96,25 %	92,81 %

Tabulka 5.2: Výsledky hodnocení aplikace testery

Podle výsledků hodnocení (viz obrázek 5.2) testerů aplikace získala skóre 92,81 %.

Závěr

Cílem práce bylo vytvoření sociální sítě pro výzkumníky Apakrychlí. To bylo naplněno vznikem aplikace APAnet pro nejrozšířenější mobilní platformu Android. Aplikace byla navržena přesně podle požadavků výzkumníků a její design a navigace uzpůsobena k co nejjednoduššímu použití. Mobilní klient je napsaný v aktuálně nejmodernějších technologiích a je do budoucna snadno rozšiřitelný. Backend vytvořený pro APAnet je použitelný i pro další druhy klientů, jako je webový klient nebo klient pro platformu iOS. Celá aplikace byla otestována jak testery, tak i automatickým testováním pomocí Google Play Console, a její případné pády jsou zaznamenávány pomocí Firebase Crashlytics. Aplikace je tedy připravená pro publikaci a reálné nasazení.

	FP1	FP2	FP3	FP4	FP5	FP6	FP7	FP8
UC1								x
UC2			x					
UC3		x						
UC4	x			x				
UC5							x	
UC6	x						x	
UC7	x			x				
UC8		x						
UC9						x		
UC10		x						
UC11					x			
UC12		x				x		
UC13					x			
UC14				x				
UC15		x		x				

Tabulka 5.3: Pokrytí funkčních požadavků

Budoucnost APAnet

Jelikož má počet výzkumníků Apakrychlí rostoucí tendenci, plánuje se údržba aplikace i do budoucna, aby podpořila šíření zdrojů a informací. Ovšem do budoucna bude potřeba zajistit financování provozu, jelikož Firebase je zdarma jen pro malé projekty. Určitě je také možné vytvořit aplikaci i pro méně zastoupenou mobilní platformu iOS, pro chytré hodinky, nebo i vytvořit webovou verzi této aplikace, aby byla co nejvíce přístupná všem uživatelům. Pokud by o tvorbu nové aplikace nebo rozšíření někdo projevil zájem, nabídnu mu možnost připojení ke stejné databázi a konzultace. Spuštění aplikace je plánované nejprve v České republice, a následně i do dalších zemí, kde by se nacházeli výzkumníci, případně nadšenci Apakrychlí. Aktuálně APAnet podporuje jen anglický jazyk, ale do budoucna bych rád doplnil i českou lokalizaci, případně i další. Co se týče dalších možností rozšíření, byl bych velmi rád, kdyby do budoucna přibyla možnost nějakým způsobem navrhnout změnu či vylepšení v aplikaci, aby její budoucí vývoj co nejvíce odpovídal požadavkům komunity a uspokojil jejich potřeby.

Literatura

[1] Mobile Operating System Market Share Worldwide | StatCounter Global Stats. StatCounter Global Stats - Browser, OS, Search Engine including Mobile Usage Share [online]. Copyright © StatCounter 1999 [cit. 5.04.2019]. Dostupné z: http://gs.statcounter.com/os-market-share/mobile

[2] GitHub - square/leakcanary: A memory leak detection library for Android and Kotlin.. The world's leading software development platform · GitHub [online]. Copyright © 2019 [cit. 6.04.2019]. Dostupné z: https://github.com/square/leakcanary

[3] Measure app performance with Android Profiler | Android Developers. Android Developers [online]. [cit. 6.04.2019]. Dostupné z: https://developer.android.com/studio/profile/android-profiler

[4] Apakrychle. Apakrychle – Apakrychle updated their profile picture. [online]. [cit. 6.04.2019]. Dostupné z: `https://www.facebook.com/apakrychle/photos/a.1118917031574547/1304966762969572/?type=1&theater`

[5] Google Fonts. Google Fonts [online]. [cit. 6.04.2019]. Dostupné z: https://fonts.google.com/specimen/Poppins

[6] Kotlin Android Extensions - Kotlin Programming Language. Kotlin Programming Language [online]. [cit. 6.04.2019]. Dostupné z: https://kotlinlang.org/docs/tutorials/android-plugin.html

[7] LiveData Overview | Android Developers. Android Developers [online]. [cit. 7.04.2019]. Dostupné z: https://developer.android.com/topic/libraries/architecture/livedata

[8] Navigation | Android Developers. Android Developers [online]. [cit. 7.04.2019]. Dostupné z: https://developer.android.com/guide/navigation

[9] Apakrychle. Apakrychle [online] [cit. 7.04.2019]. Dostupné z: http://apakrychle.ninja/

[10] ViewModel Overview | Android Developers. Android Developers [online]. [cit. 7.04.2019]. Dostupné z: https://developer.android.com/topic/libraries/architecture/viewmodel

[11] Use app signing by Google Play - Play Console Help. Google Help [online]. Copyright ©2019 Google [cit. 21.04.2019]. Dostupné z: https://support.google.com/googleplay/android-developer/answer/7384423?hl=en

[12] Download Android Studio and SDK tools. Android Developers [online]. [cit. 21.04.2019]. Dostupné z: https://developer.android.com/studio

[13] What is Firebase? How To Firebase [online]. [cit. 21.04.2019]. Dostupné z: https://howtofirebase.com/what-is-firebase-fcb8614ba442

[14] Introduction - Material Design. Homepage - Material Design [online]. [cit. 15.04.2019]. Dostupné z: https://material.io/design/introduction/#principles

[15] GOOGLE. Android Studio. Version 3.5 Canary 7 [software]. 2019 [cit. 15.04.2019]. Dostupné z: https://developer.android.com/studio

[16] Null Safety - Kotlin Programming Language. Kotlin Programming Language [online]. [cit. 15.02.2019]. Dostupné z: https://kotlinlang.org/docs/reference/null-safety.html

[17] Apache NetBeans. Welcome to NetBeans [online]. [cit. 17.02.2019]. Dostupné z: https://netbeans.org/

[18] Introduction to Activities | Android Developers. Android Developers [online]. [cit. 16.04.2019]. Dostupné z: https://developer.android.com/guide/components/activities/intro-activities

[19] Fragments | Android Developers. Android Developers [online]. [cit. 16.04.2019]. Dostupné z: https://developer.android.com/guide/components/fragments

[20] RecyclerView | Android Developers. Android Developers [online]. [cit. 16.04.2019]. Dostupné z: https://developer.android.com/reference/androidx/ recyclerview/widget/RecyclerView.html

[21] Dialogs | Android Developers. Android Developers [online]. [cit. 16.04.2019]. Dostupné z: https://developer.android.com/guide/topics/ui/dialogs

[22] RecyclerView.ViewHolder | Android Developers. Android Developers [online]. [cit. 16.04.2019]. Dostupné z: https://developer.android.com/reference/android/support/v7/ widget/RecyclerView.ViewHolder

[23] Adapter | Android Developers. Android Developers [online]. [cit. 16.04.2019]. Dostupné z: https://developer.android.com/reference/android/widget/Adapter

Příloha A

Slovník pojmů

Post Příspěvek

User Uživatel

Comment Komentář

Tag Krátký textový řetězec, který se používá ke zvýraznění klíčových slov v příspěvku

Layout Rozložení grafických prvků na obrazovce

Crashlytics Systém sledování chyb při běhu aplikace

Příloha B

Seznam použitých zkratek

UI User interface

UCD User Centered Design

OS Operační Systém

SDK Software Development Kit

API Application Programming Interface

Příloha C

Obsah přiloženého CD

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