FAKULTA INFORMAČNÍCH TECHNOLOGIÍ ČVUT V PRAZE

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

Název:	Android Aplikace sloužící ke sledování aktivit
Student:	Dušan Janiš
Vedoucí:	Ing. Miroslav Balík, 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

Vytvořte Android aplikaci pro sledování aktivit.
1. Celková správa uživatelů bude realizována pomocí cloudové služby Google Firebase (registrace uživatele, přihlášení uživatele, odhlášení uživatele).
2. Android aplikace bude umožňovat přihlášení pomocí emailu, účtu Google a účtu Facebook.
3. Android aplikace bude komunikovat s existující REST API částí, kde komunikací je myšleno:
a. založení nových záznamů (skupin nebo aktivit),
b. editace existujících záznamů,
c. čtení existujících záznamů,
d. mazání existujících záznamů,
To vše dle obecně známých principů popsaných v https://www.restapitutorial.com
4. Android aplikace bude umět odesílat a přijímat notifikace pomocí služby Google Firebase.
5. Student nastuduje, popíše a aplikuje nahrání android aplikace Google play a to i do její testovací verze.
6. Student nastuduje, popíše a aplikuje různé způsoby testování android aplikace. (unit testy, integrační testy, akceptační testy).

Seznam odborné literatury

Dodá vedoucí práce.

Ing. Michal Valenta, Ph.D.
vedoucí katedry

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

V Praze dne 5. prosince 2018

FAKULTA
INFORMAČNÍCH
TECHNOLOGIÍ
ČVUT V PRAZE

Bakalářská práce

Android Aplikace sloužící ke sledování aktivit

Dušan Janiš

Katedra softwarového inženýrství
Vedoucí práce: Ing. Miroslav Balík, Ph.D.

12. února 2020

Poděkování

Především chci poděkovat svému vedoucímu práce, který mě svým pozitivním přístupem a konstruktivní kritikou podporoval po celou dobu zpracování této práce. Zároveň bych rád poděkoval svému oponentovi a spolu s ním i společnosti Skills Fighters za poskytnutou podporu, co se týče nasazování backendu.

Prohlášení

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

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

V Praze dne 12. února 2020 .....

Odkaz na tuto práci

Janiš, Dušan. *Android Aplikace sloužící ke sledování aktivit.* Bakalářská práce. Praha: České vysoké učení technické v Praze, Fakulta informačních technologií, 2020.

Abstrakt

Cílem této bakalářské práce je vytvořit aplikaci pro systém Android sloužící ke sledování aktivit uživatelů. Tato mobilní aplikace komunikuje s již existující backend aplikací a poskytuje uživateli rozhraní, které mu umožňuje ukládat a následně sledovat své aktivity. Součástí práce je i nahrání aplikace do služby Google Play a vypracování testů pro aplikaci.

K samotné komunikaci s backend aplikací je použito rozhraní REST. Pro ověřování identity uživatelů za účelem ukládání dat na backend, je implementováno přihlašování pomocí služby Google Firebase. Google Firebase je také využíván pro přijímání notifikací.

Výsledné řešení splňuje funkcionalitu vymezenou v zadání práce. Aplikace poskytuje přihlašování pomocí všech hlavních přihlašovacích služeb (Google, Facebook), aplikace umožňuje uživateli sledovat své aktivity a aplikace je i schopna přijímat notifikace zasílané backendem. K aplikaci jsou vypracovány testy. Tato funkcionalita byla následně dále rozšiřována nad rámec vymezený zadáním.

Klíčová slova vývoj aplikace, mobilní aplikace, Android, Google Firebase, REST, Kotlin

Abstract

The goal of this bachelor thesis is to create an Android application for activity tracking. This mobile application is to communicate with an already existing backend application and it provides the user with an interface which allows him to save and subsequently track his activities. The application will be a published on the Google Play service. Part of the thesis is implementation of tests for this application along with its upload to the Google Play service.

REST interface is used for communication with the backend application. To save user data to backend a user authentication is required. Authentication is implemented using Google Firebase service. Google Firebase is also used for receiving of notifications.

The final solution implements the functionality as described in the thesis assignment. Application provides using sign up and sing in using all the main identity providers (Google, Facebook), the user can track his activities and the application can receive notifications sent by the backend application. Tests for this application are implemented. This functionality has been further extended above what is required by the thesis assignment.

Keywords application development, mobile application, Android, Google Firebase, REST, Kotlin

Obsah

Seznam obrázků

Seznam tabulek

Úvod

V dnešní hektické době je pro mnoho lidí snadné ztratit se ve všech svých povinnostech a aktivitách. Cílem této práce je tedy vytvořit mobilní aplikaci, která bude umožňovat uživatelům lépe si své aktivity sledovat, aby si byli více vědomi toho, jak nakládají se svým drahocenným časem.

Nelze v dnešní době upřít důležitost mobilních technologií. Ty se v posledních letech posunuly značně dopředu. Telefon už uživateli neslouží pouze jako komunikační zařízení, ale jedná se o přenosný počítač, který lze použít pro organizaci času a jako zábavní platformu. Navíc díky masovému rozšíření chytrých telefonů jsou mobilní aplikace dostupné většině lidí.

Práce je zejména určena pro společnost Skills Fighters. Jedná se o komunitu programátorů, kterým je problematika správy času důvěrně známá. To však neznamená, že tato aplikace nebude využitelná i pro širokou veřejnost.

Z výše uvedených důvodů jsem shledal téma jako zajímavé a rozhodl jsem se na něm pracovat. Velice oceňuji fakt, že téma je praktického rázu a v každodenním životě může být pro aplikaci využití.

Kapitola 1

Cíl práce

Cílem této bakalářské práce je vytvořit Android aplikaci sloužící ke sledování aktivit. V aplikaci je třeba identifikovat uživatele za účelem ukládání dat na backend. Správa uživatelů bude tedy realizována pomocí služby Google Firebase, kde si bude uživatel moci vytvořit účet na základě svého emailu nebo se přihlásit pomocí účtu Google či účtu Facebook. Dále bude aplikace komunikovat s již existující backend aplikací a to pomocí rozhraní REST, přičemž komunikací je myšleno zakládání, editace, čtení a mazání záznamů a to skupin či aktivit.

Aplikace bude schopna odesílat a přijímat notifikace, přičemž toto bude implementováno opět pomocí služby Google Firebase. Dílčím cílem práce je pak aplikace různých způsobů automatizovaného testování jako jsou unit testy, integrační testy či testy akceptační. Na závěr bude aplikace nahrána do služby Google Play a to i do její testovací verze.

Kapitola 2

Analýza

2.1 Analýza použitých technologií

Android je operační systém s linuxovým jádrem, který se v dnešní době vyskytuje na valné většině mobilních zařízení na trhu. Díky jeho rozšíření a široké uživatelské základně může aplikace dosáhnout většího úspěchu než na systémech ostatních. Další výhodou vysokého množství uživatelů je i velká vývojářská komunita. Vývoj aplikací pro systém Android usnadňuje tedy i velká míra pomocí ze strany komunity vývojářů[1].

2.1.1 Jazyky pro vývoj aplikací

Oficiální podporu pro vývoj aplikací měla při vydání systému Android pouze Java. Java je multiplatformní a velmi rozšířená. Díky tomu pro ni existuje velké množství knihoven a není problém s podporou ze strany ostatních vývojářů, vzhledem k již zmíněné velké uživatelské základně. Java je ovšem jazyk navržený již před více než dvaceti lety, a proto má řadu nevýhod. Při změnách v samotném jazyce, je nutné hledět na zpětnou kompatibilitu a je tedy problematické zavádět funkce, které obsahují modernější jazyky.

Proto se v poslední době začal používat jazyk Kotlin. Jazyk Kotlin vyvinutý společností Jetbrains je od základů navržen k plné kompatibilitě s jazykem Java. Je plně schopen využívat knihoven Javy, ale zároveň obsahuje funkce, které usnadňují vývoj aplikací a podporují psaní čistého kódu. V roce 2017 byl na konferenci Google I/O prohlášen za oficiálně podporovaný jazyk pro vývoj Android aplikací, nejsou tedy problémy s podporou a všechny oficiální návody pro vývojáře, jsou psané jak pro jazyk Java, tak pro jazyk Kotlin[2].

2.1.2 Verze systému Android

Jako nejnižší kompatibilní verze systému Android byla zvolena verze 5.0. Tímto je zajištěna kompatibilita aplikace pro 90% uživatelů systému Android. Verze byla volena jednak na základě počtu uživatelů, jednak na základě požadované funkcionality. Služba Firebase Cloud Messaging totiž vyžaduje Android API alespoň verze 21. Ta je poskytována právě verzí 5.0 systému Android.

Zároveň díky snaze zvolit verzi systému Android co nejnižší možnou, je aplikace potencionálně dostupná velkému množství uživatelů. Na základě dat o užívání systému Android, která jsou dostupná z oficiálních zdrojů, je množství uživatelů používajících verze Androidu nižší než 5.0 pouhých 10,7%[3].

Version	Codename	API	Distribution
2.3.3 - 2.3.7	Gingerbread	10	0.3%
4.0.3 - 4.0.4	Ice Cream Sandwich	15	0.3%
4.1.x	Jelly Bean	16	1.2%
4.2.x		17	1.5%
4.3		18	0.5%
4.4	KitKat	19	6.9%
5.0	Lollipop	21	3.0%
5.1		22	11.5%
6.0	Marshmallow	23	16.9%
7.0	Nougat	24	11.4%
7.1		25	7.8%
8.0	Oreo	26	12.9%
8.1		27	15.4%
9	Pie	28	10.4%

Obrázek 2.1: Tabulka ukazující procentuální zastoupení jednotlivých verzí Androidu na uživatelských zařízeních

2.1.3 Firebase

Google Firebase je platforma pro vývoj mobilních aplikací s podporou pro všechny hlavní druhy mobilních zařízení, tedy jak Android, tak i iOS. V rámci této platformy jsou vývojářům poskytovány nástroje a služby k rozšíření možností a k usnadnění vývoje mobilních aplikací. Většina služeb je poskytována alespoň v základní variantě zcela zdarma[4].

2.1.3.1 Firebase Authentication

V aplikaci je nutná identifikace uživatelů, za účelem ukládání dat na cloudu a jejich synchronizaci mezi zařízeními. Proto je třeba zajistit, aby se každý uživatel mohl přihlásit. Při zajištění přihlášení je nutné se zaměřit primárně na bezpečnost. Obecně implementovat si autentizační systém zcela sám, je řešením špatným, jelikož je zde velký prostor pro chyby, které by mohly ohrozit osobní data uživatelů.

Firebase Authentication poskytuje možnost autentizovat uživatele pomocí hlavních poskytovatelů přihlášení - Google, Facebook či Twitter, a dále poskytuje možnost autentizovat uživatele jen na základě jejich emailu. V tomto případě si Google Firebase ukládá email i heslo ve své databázi. Vývojář se tedy nemusí starat o zabezpečení přihlašovacích údajů, neboť se o vše stará Google Firebase. Ten po přihlášení poskytne vývojáři „token“, který slouží jako identifikátor, pomocí nějž mohou další služby identifikovat přihlášeného uživatele[5].

2.1.3.2 Firebase Cloud Messaging

Jedná se o službu poskytující posílání zpráv a upozornění mezi zařízeními. Funguje napříč platformami, je tedy použitelná jak na Androidu tak i na systému iOS a lze ji použít i pro webové klientské aplikace. Zajišťuje bezpečné doručení zpráv z backend aplikace skrz webovou službu Firebase Cloud Messaging až ke klientům. Klienty identifikuje pomocí unikátních identifikátorů, které jsou přiděleny každému zařízení, na němž je aplikace nainstalována[6].

2.1.4 REST API

Representational State Transfer, ve zkratce REST je architektura rozhraní určená pro distribuované prostředí. Pod pojmem distribuovaná prostředí je myšleno to, že části programu běží na různých strojích a spolu následně komunikují s využitím sítě. REST navrhnul a popsal v roce 2000 Roy Fielding v rámci své disertační práce. V této své práci, Fielding také odvozuje principy RESTu na základě známých přístupů k architektuře[7].
Za základní principy RESTu stanovil, že

- stav aplikace a její chování je vyjádřeno takzvaným zdrojem;
- každý zdroj má unikátní identifikátor (URL či URN);
- je definován jednotný přístup pro získání a manipulaci se zdrojem v podobě operací CRUD;
- zdroj může mít různé reprezentace (XML, HTML, JSON).

Pro komunikační protokol pak platí, že

- klient / server - oddělení odpovědnosti;
- bezestavovost - každý požadavek musí obsahovat všechny informace nutné k jeho vykonání;
- cache - každý požadavek lze explicitně označit, zdali má být možnost jej uložit do cache. Toto umožňuje zvýšit výkonnost komunikace mezi klientem a serverem;
- Code-On-Demand - funkcionalita klienta lze rozšířit kódem, který zašle server;
- vrstevnatost - umožňuje skládání vrstev poskytující služby za účelem zvýšení variabilnosti.

Krom RESTu existují i další přístupy k řešení distribuované architektury, jako je Remote Procedure Call (RPC).

2.2 Funkční požadavky

- Přihlášení do Firebase
  - Pomocí emailu
  - Pomocí účtu Google
  - Pomocí účtu Facebook
- Odhlášení z Firebase
- Smazání účtu ve Firebase
- Komunikace s backend aplikací pomocí REST API
  - Založení skupiny
  - Založení aktivity
  - Smazání skupiny
  - Smazání aktivity
- Přijímání notifikací s využitím Google Firebase

2.3 Nefunkční požadavky

- Testování aplikace
  - Unit testy
  - Integrační testy
  - Akceptační testy
- Publikace na Google Play
- Minimální verze systému Android 5.0

2.4 Analýza konkurenčních aplikací

Hledání konkurenčních aplikací v Google Play bylo prováděno na základě klíčových slov pomocí výrazů „activity tracker“, „habit tracker“ a „daily log“.

Pod pojmem „activity tracker“ se nachází aplikace sloužící primárně ke sledování fyzických aktivit tedy toho, jak moc se uživatel hýbe a v jaké je fyzické kondici. Často to bývají aplikace spolupracující se smart hodinkami a podobným příslušenstvím, které sledují momentální stav tělesných funkcí nositele, tj. tep, krevní tlak či tělesnou teplotu. Toto ovšem nebude účelem vyvíjené aplikace, a proto žádné takto nalezené aplikace ke srovnání vybrány nebyly.

Při hledání pomocí výrazu „habit tracker“ byly nalezeny aplikace, které jsou účelově blíže aplikaci, která je vytvářena v rámci této práce. Jsou to aplikace, které mají za úkol sledovat a ukládat, kdy nějaká aktivita nastala a tedy i jak často. Tyto aplikace slouží k vytvoření návyků a k jejich zaznamenávání. Uživatel si pak může vybírat z různých aktivit a plánů, které aplikace nabízejí. Poskytují možnosti plánovat si život dopředu. Často obsažená funkce je kalendář, ve kterém si uživatel sám zaklikává, jestli svůj cíl splnil nebo ne.

Poslední výraz použitý při průzkumu je „daily log“. Takto nalezené aplikace, byly často aplikace sloužící buď jako deníčky, nebo aplikace, které poskytovaly možnost vytvářet reporty ve firemním prostředí.

Při výběru konkurenčních aplikací pro srovnání byly zvoleny tyto tři aplikace, *onda*, *HabitBull* a *Daily Log*.

2.4.1 Aplikace onda

Aplikace onda je zcela zdarma a poskytuje sledování aktivit. Uživatel si může sám vytvořit aktivitu, zvolit si její kategorii z předem daného seznamu a nastavit čas, jak dlouho má tato aktivita trvat. Následně může uživatel tuto aktivitu spustit. To má za následek spuštění časovače, což by mělo uživatele motivovat k vykonávání dané aktivity. Uživatel má dále možnost zobrazit si seznam jím vytvořených aktivit. Aplikace je ryze offline.

Aplikace je rozdělena do tří obrazovek. První z nich zobrazuje seznam již proběhlých aktivit. U každé takto proběhlé aktivity je zobrazeno její datum a trvání. Druhá obrazovka nabízí několik základních již vytvořených aktivit a možnost vytvářet nové. Na této obrazovce je uživatel také spouští. Poslední obrazovka pak uživateli poskytuje základní statistiku o tom, kolik času strávil vybranou aktivitou[8].

2.4.2 HabitBull

HabitBull umožňuje uživateli vytvořit si vlastní aktivity. Při vytváření aktivity si uživatel volí, do jaké skupiny aktivit ji chce zařadit, jak má aplikace sledovat plnění této aktivity a název zvolené aktivity. Následné plnění aktivit je zapisován do kalendáře obsaženého v aplikaci. V tomto kalendáři je vidět, nakolik je uživatel v plnění jím vymyšlených aktivit úspěšný. Cílem této aplikace je budovat v člověku návyky tím, že mu ukazuje, jestli se mu daří plnit aktivity každý den, nebo zda některé dny vynechává. Aplikace poskytuje i možnost vytvoření profilu, toto ovšem ve verzi zdarma poskytuje pouze možnost uzamčení aplikace heslem. V prémiové verzi je poskytnuto i zálohování dat na cloud.

Tato aplikace má poměrně jednoduché uživatelské rozhraní. Na hlavní stránce zobrazuje pro každou zvolenou aktivitu kalendář s úspěšnými dny a pod ním graf znázorňující, zdali každý den plní svou tuto aktivitu či nikoliv. V hlavním menu aplikace se pak nachází tlačítko přidat další aktivitu nebo změnit zobrazení kalendáře z týdenního na měsíční. Další možnosti, jako zakoupení prémiové verze, se nacházejí v menu na levé straně obrazovky[9].

2.4.3 Daily Log

Daily Log umožňuje uživateli vytvářet si kategorie, pro které si volí jeden ze tří typů, a to boolean, číslo s jednotkou či řetězec. Následně si uživatel pro jím zvolenou kategorii zaznamenává, ve které dny došlo k aktivitě, a to v kalendáři obsaženém v aplikaci. U jednotlivých kategorií si může uživatel zobrazovat jednoduché statistiky v podobě grafu znázorňujícího data zapsané v aplikaci. Aplikace sice má tlačítko Import, které by mohlo umožňovat import dat ze serializované podoby, ale nebyla nalezena možnost data z aplikace exportovat. Aplikace nevyužívá připojení k Internetu.

Hlavní obrazovka aplikace je na začátku prázdná, později se v ní zobrazují vytvořené kategorie. Stisknutím tlačítka pro přidání kategorie se uživateli zobrazí dialogové okno, v němž si vytvoří kategorii. Zvolením kategorie se otevře kalendář zobrazující aktivity. Zde může uživatel zapisovat aktivity do kalendáře a zároveň vidí aktivity už dříve zapsané[10].

2.4.4 Srovnání aplikací

Zde je srovnání aplikací na základě jejich funkcionalit. Všechny tyto aplikace fungují pouze v offline režimu. Pouze aplikace *HabitBull* poskytuje možnost ukládat si data na cloud a to pouze v prémiové verzi.

Funkce	Activity Tracker	onda	HabitBull	Daily Log
Vytváření aktivit	ANO	ANO	ANO	ANO
Vytváření skupin	ANO	NE	NE	NE
Nutnost přihlášení	ANO	NE	NE	NE
Ukládání na cloud	ANO	NE	Pouze premium	NE
Možnost práce offline	NE	ANO	ANO	ANO

Tabulka 2.1: Srovnání aplikací

Uživatelské rozhraní všech těchto aplikací má několik společných prvků. Jedna obrazovka typicky slouží k vytváření aktivit a až další k jejich zobrazování. Často je v aplikacích už předem připravených několik aktivit na výběr. Zároveň aplikace poskytují přehledný způsob, jak aktivity zpětně sledovat. Kalendář je k dispozici v aplikacích *HabitBull* a *Daily Log*.

Kapitola 3

Návrh

3.1 Architektura aplikace

Při vývoji aplikace byl kladen důraz na rozdělení aplikace na jednotlivé vrstvy a to dle architektonického vzoru MVC. Dle vzoru MVC se aplikace skládá ze tří oddělených vrstev[11]:

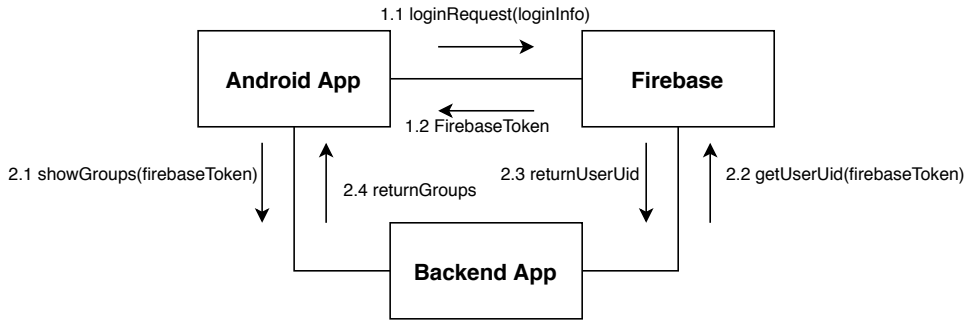
- **Model**, který ukládá a pak následně poskytuje data, s nimiž aplikace pracuje;
- **View** obsahuje vše, co se zobrazuje na obrazovku a s čím uživatel interaguje;
- **Controller** je zodpovědný za logiku aplikace a zprostředkovává komunikaci mezi vrstvami **Model** a **View**.

Android aplikacím je toto rozdělení do jisté míry přirozené. *Controller* je reprezentován aktivitami. Ty jsou zodpovědné za logiku aplikace a komunikují s *View*, který je reprezentován „layout“ soubory ve formátu XML. V těchto souborech je popsán samotný vzhled aplikace. Typicky existuje alespoň jeden pro každou aktivitu. *Model* samotný je pak tvořen entitními třídami a klienty, kteří zajišťují komunikují s backendem.

Důležitost použití správné architektury a její dodržování tkví jednak ve vylepšení udržovatelnosti aplikace, je tedy snazší na ní v budoucnu provádět další změny a aplikaci rozšiřovat, jednak tato separace usnadňuje psaní testů.

3.2 Diagram komunikací

Diagram komunikací zde popisuje komunikaci mezi Android aplikací, která se přihlašuje do Google Firebase a následně už pak komunikuje pouze s backend aplikací. Pro identifikaci posílá Android aplikace backendu identifikační token, který Android aplikace dostala od služby Google Firebase při přihlášení. Pomocí tohoto tokenu si pak backend aplikace ověřuje uživatele, aniž by musel pracovat přímo s přihlašovacími údaji uživatele.



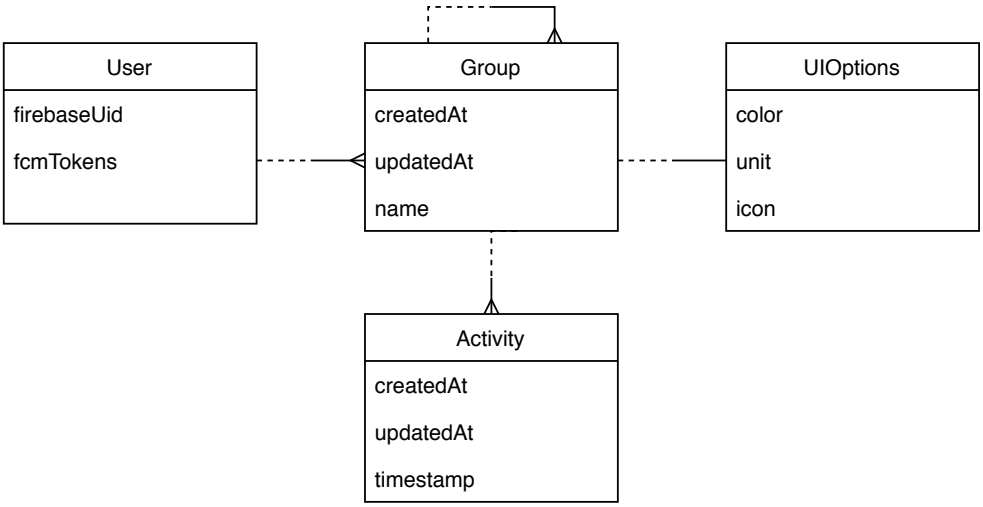
Obrázek 3.1: Komunikační diagram zobrazující přihlášení v Android aplikaci a následné načtení všech skupin uživatele z backendu

3.3 Datový model

Datový model popisuje strukturu dat, s nimiž pracuje tato aplikace. Základ modelu byl vytvořen již dříve při tvorbě backend aplikace. Datový model se skládá z:

- **Group** reprezentuje skupinu, která dále sdružuje další skupiny nebo aktivity dohromady. Pro každou skupinu je specifikován název;
- **Activity** je v podstatě pouze časový záznam. Vždy spadá do některé ze skupin. Význam aktivitě dodává skupina, do níž aktivita spadá;
- **User** je záznam o uživateli, s nímž pracuje backend aplikace. Android aplikace samotná s ním nepracuje přímo, v Android aplikaci je uživatel identifikován pouze pomocí svého Firebase identifikátoru. Se záznamem o uživateli pracuje primárně backend, který jej ukládá v databázi;
- **UIOptions** reprezentují grafické informace skupiny. **UIOptions** mohou obsahovat ikonu skupiny, její barvu nebo popřípadě jednotku skupiny. Ta popisuje, co jednotlivé aktivity ve skupině vyjadřují.

V průběhu vývoje Android aplikace byl tento model rozšířen o třídu *UIOptions*, která rozšiřuje třídu *Group* o grafická data. Třída *User* byla rozšířena o pole *fcmTokens*, obsahující seznam identifikátorů zařízení, na kterých je daný uživatel přihlášený pro účely služby Firebase Messaging.



Obrázek 3.2: Tento diagram znázorňuje třídy a jejich vazby, které tvoří datový model aplikace.

Kapitola 4

Android aplikace

Aplikace byla vyvíjena v průběhu roku 2019 ve spolupráci se společností Skills Fighters. Společnost poskytla backend aplikaci, firemní GitHub účet, firemní Google účet pro vývoj Android aplikací a přístup k firemnímu účtu na Amazon Web Services pro účely nahrání backendu. Dále byly poskytovány konzultace, co se vývoje samotné Android aplikace týče. Aplikace je napsaná kompletně v Kotlinu, který byl zvolen v rámci snahy použít co nejaktuálnější technologie a nejnovější přístupy k vývoji Android aplikací.

4.1 Struktura aplikace

Aplikace je rozdělena do čtyř aktivit. Z pohledu uživatele aplikace funguje tak, že se uživatel pouze pohybuje mezi těmito čtyřmi aktivitami v závislosti na činnosti, kterou právě vykonává. Pokud se přihlašuje, o vše s tím související se stará pouze *AuthenticationActivity*, pokud si uživatel zobrazuje nebo manipuluje se skupinami či záznamy v nich, nachází se v *GroupActivity*. O zobrazování grafů souvisejících se skupinou se stará aktivita *GraphActivity*. Poslední aktivitou je pak *CalendarActivity*, ta zobrazuje uživateli všechny záznamy právě zvolené skupiny v kalendáři. Aktivity jsou jasně rozděleny na základě jejich účelu. Aktivity dále pracují s dalšími částmi aplikace, jako jsou klienti pro komunikaci s backendem a adaptéry pro zpracování přijatých dat od backendu do podoby vhodné k zobrazení.

4.1.1 AuthenticationActivity

Tato aktivita zajišťuje uživateli veškerou funkcionalitu spojenou s přihlášením. Jelikož je přihlášení pro funkčnost aplikace nutnost, je toto také první aktivita, která se uživateli při spuštění aplikace zobrazí. Stisknutím tlačítka přihlásit se, dojde k zavolání metody knihovny *FirebaseUI*, která se o samotné přihlášení uživatele stará.

Po přihlášení má uživatel možnost pokračovat dále do aplikace a to k zobrazování skupin. Stisknutím tlačítka *Pokračovat* se nastartuje *GroupActivity* a aplikace ji přesune do popředí. Dále se uživateli v hlavním menu aplikace na horní liště zpřístupní možnost odhlášení z aplikace a možnost smazání svého účtu v aplikaci.

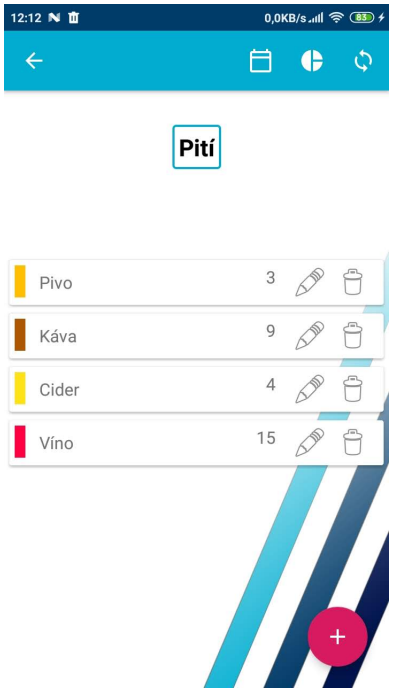
Obrázek 4.1: Přihlašovací aktivita po úspěšném přihlášení

V případě, že se uživatel z aplikace v průběhu posledního používání neodhlásil, zůstává přihlášený i při dalším spuštění aplikace a opětovné přihlášení není vyžadováno.

V hlavním menu aplikace se kromě možností odhlášení a smazání účtu nacházejí také další dvě možnosti a to *Info* a *Licence*. Možnost *Info* zobrazí momentálně nainstalovanou verzi aplikace, *Licence* pak zobrazí interaktivní seznam licencí všech knihoven používaných v aplikaci.

4.1.2 GroupActivity

Zde aplikace poskytuje svou hlavní funkcionalitu. Uživateli se v této části aplikace zobrazují jím vytvořené záznamy (tedy skupiny nebo aktivity) a to v podobě seznamu. Uživatel vždy vidí jednu skupinu a její obsah. Jejím obsahem mohou být buď další skupiny nebo aktivity. V případě, že se jedná o další skupiny, je u každé z nich vidět počet aktivit, které obsahují buď přímo ony nebo další skupiny v nich obsažené. Kliknutím na některou zobrazenou skupinu dojde k načtení obsahu této skupiny. Tímto způsobem mohou být v sobě skupiny zanořené bez omezení.



Obrázek 4.2: Skupinová aktivita zobrazující svůj obsah

Po vstupu do této aktivity se uživateli zobrazí *„Kořenová skupina“*. Tato skupina reálně v databázi neexistuje, jedná se pouze o prostor pro zobrazení všech skupin uživatele, které nenáleží pod žádnou jinou. Nemohou se zde vyskytovat aktivity. V dalších skupinách se již aktivity vyskytovat mohou.

Nachází-li se uživatel ve skupině nejnižší úrovně, tedy ve skupině, která již další skupiny neobsahuje a zobrazuje tedy aktivity, má uživatel k dispozici možnost aktivity seskupit. Namísto toho, aby uživatel viděl pouze seznam aktivit, má možnost nechat si aktivity seskupit podle dnů, týdnů, měsíců či let. Toto zobrazí uživateli zvolený časový interval a počet aktivit do něj spadajících.

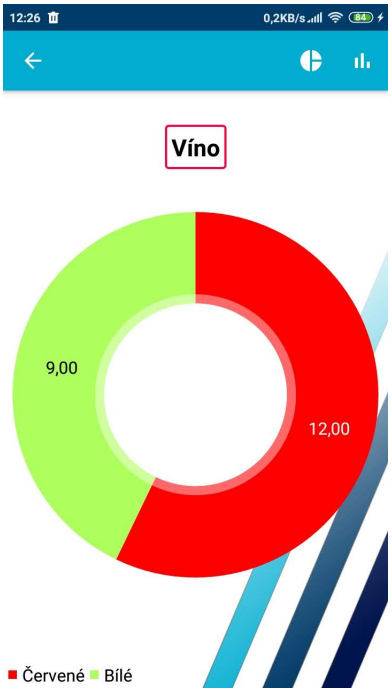
Obrázek 4.3: Skupinová aktivita zobrazující aktivity seskupené po dnech

V případě, že se v nejvyšší úrovni žádné skupiny nevyskytují, tedy je-li „Kořenová skupina“ prázdná, aplikace sama nabídne uživateli možnost si nechat vygenerovat několik základních skupin. Aplikace pošle žádost na backend, který má uložený seznam těchto skupin, vytvoří je a následně upozorní aplikaci, aby opětovně načetla data z backendu.

Pro každou skupinu je možné nastavit barvu. Jméno skupiny v horní části obrazovky funguje zároveň jako tlačítko, které vyvolá obrazovku pro výběr barvy. Tato barva je znázorněna barevným okrajem kolem jména skupiny. Po výběru je barva zaslána na backend, kde je uložena v databázi. V seznamu skupin je u každé skupiny, která má nastavenou barvu, přítomen barevný pruh.

4.1.3 GraphActivity

Tato aktivita je odpovědná pouze za zobrazování grafů. Při spuštění z *GroupActivity* je jí předána skupina, která je zrovna otevřená. K této skupině si *GraphActivity* získá z backendu veškeré skupiny, které pod spadají spolu s počtem aktivit, které tyto skupiny obsahují. Na základě tohoto aktivita zobrazuje dva druhy grafů, koláčový a sloupcový. Oba tyto grafy znázorňují počet aktivit obsažený v jednotlivých skupinách.

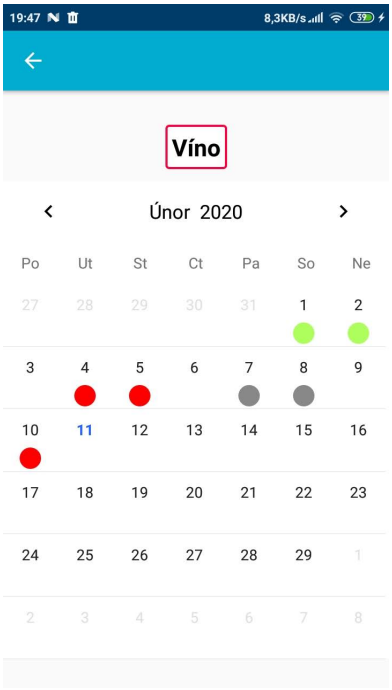


Obrázek 4.4: Grafová aktivita zobrazující koláčový graf

Při generování grafu si aplikace vyžádá informace o vzhledu skupiny, kterou zobrazuje a o vzhledu všech skupin, které pod zvolenou skupinu spadají. V případě, že je v těchto informacích o vzhledu definovaná barva pro jednotlivé skupiny, je použita pro jejich znázornění v grafech. V opačném případě aplikace zvolí náhodnou barvu z předdefinovaného seznamu barev v aplikaci.

4.1.4 CalendarActivity

Poslední vytvořená aktivita obsahuje kalendář, který zobrazuje zapsané záznamy. Podobně jako u *GraphActivity*, při spuštění aktivity je jí předána právě otevřená skupina. Následně si aktivita načte skupiny, které přímo spadají pod momentálně otevřenou skupinu. Podle těchto skupin si *CalendarActivity* načítá všechny záznamy pod ně spadající a podle nastaveného vzhledu těchto skupin je zakresluje do kalendáře. Pro každý den, pro který existuje záznam, se objeví barevné kolečko v barvě záznamu. Je-li v daný den evidováno více záznamů z různých skupin, toto kolečko má šedou barvu.



Obrázek 4.5: Kalendářová aktivita zobrazující několik záznamů

4.2 Přihlašování

Jak již je zmíněno výše v teoretické části, pro přihlašování je použita služba Firebase Authentication. Za účelem zajištění funkčnosti a spolehlivosti, byla využita knihovna *FirebaseUI*, v níž je přihlašování pomocí všech v zadání definovaných poskytovatelů přihlášení implementováno. Zároveň je tato knihovna doporučená přímo ze strany společnosti Google v oficiální dokumentaci služby Firebase.

4.3 REST API

Zásadní částí aplikace pro její funkčnost je implementace rozhraní REST pro komunikace aplikace s backendem. K implementaci tohoto rozhraní je využita knihovna *Retrofit*.

V aplikaci jsou vytvořeni čtyři klienti pro komunikaci s backendem. Tito klienti definují rozhraní, které popisuje formát a obsah jednotlivých requestů. Toto rozhraní je následně knihovnou *Retrofit* rozšířeno. Každý z těchto čtyř klientů je zodpovědný za jeden druh dat odesílaný Android aplikací na backend. Jsou jimi:

1. **ActivityApiClient** použitý pro komunikaci týkající se aktivit;
2. **GroupApiClient** pro komunikaci týkající se skupin;
3. **UIOptionsApiClient** pro zasílání a příjem informací o vzhledu skupin;
4. **TokenApiClient** pro odesílání registračního tokenu aplikace.

Toto je ukázka klienta *ActivityApiClient*. Jedná se o rozhraní popisující jednotlivé metody pro komunikaci s backendem.

```
interface ActivityApiClient {

    @GET("activity/showall")
    fun getActivities(@Query("groupid") id: Long)
        : Observable<List<Activity>>

    @GET("activity/showall")
    fun getActivities(@Query("groupid") id: Long,
       @Query("startdate") startDate: Long,
       @Query("enddate") endDate: Long)
        : Observable<List<Activity>>

    @PUT("activity/add")
    fun addActivity(@Body activity: ActivityCreated)
        : Completable

    @DELETE("activity/delete")
    fun deleteActivity(@Query("activityid") id: Long)
        : Completable

    @POST("activity/update")
    fun updateActivity(@Body activity: ActivityUpdated)
        : Completable
}
```

Každý request na backend potřebuje v hlavičce identifikační token uživatele poskytnutý knihovnou Firebase. O získání a doplnění této hlavičky se stará *interceptor*. *Interceptor* je implementován jako samostatná třída a je předán jednotlivým klientům při jejich vytváření. Tento *interceptor* je přidán ke všem čtyřem rozhraním při jejich vytváření, jelikož každý request zasílaný na backend musí identifikovat uživatele od nějž pochází. Každý request je ještě před odesláním *interceptorem* zachycen a je k němu doplněna patřičná hlavička.

4.4 Notifikace

Součástí zadání je využití služby Firebase Messaging k zasílání notifikací. V nynější verzi aplikace slouží tato služba k zasílání notifikací na všechna zařízení uživatele o tom, že byla změněna data uživatele na backendu. V reakci na přijetí této notifikace provede Android aplikace obnovení dat. Samotná Android aplikace žádné notifikace pomocí služby Firebase Messaging neposílá.

Pro přijímání těchto notifikací je v aplikaci implementována služba *MessagingService*. Tato služba běží stále v pozadí. Přijde-li do aplikace notifikace,

služba ji zpracuje a upozorní právě běžící aktivitu na příchozí notifikaci. V tuto chvíli jediná notifikace, která může přijít, je upozornění na změnu dat uživatele v backendu. Snadno lze ovšem službu upravit i pro zpracování jiných typů notifikací.

4.5 Lokalizace

Aplikace je lokalizována v angličtině a češtině. Přeloženy jsou veškeré texty a popisky tlačítek. Stránka na Google Play je taktéž k dispozici v obou jazycích.

4.6 Vývojové nástroje

K vývoji Android aplikace bylo použito *Android Studio*, oficiální vývojové prostředí doporučované a poskytované společností Google. Aplikace byla během vývoje spouštěna primárně pomocí emulátoru systému Android obsaženého v *Android Studiu* a to na virtuálním zařízení *Pixel 2* nejprve s verzí systému Android 9.0 a po jejím vydání i s verzí 10.0. V rámci vývoje byla aplikace testována i na fyzických zařízeních a to zejména na autorově zařízení *Xiaomi Redmi 4X* s verzí systému 7.1.

Pro spuštění backendu aplikace lokálně za účelem vývoje Android aplikace, je třeba lokálně nastartovat databázi. K tomuto účelu byl použit nástroj *Docker*, v němž byla nastartována databáze *MySQL*. Ke spuštění samotné backend aplikace bylo pak použito vývojové prostředí *IntelliJ IDEA*.

V pozdějších fázích byl využíván backend běžící na *AWS*. V tomto případě už byla využívána *MySQL* databáze přítomna tamtéž.

4.7 Verzování

Verzování bylo vyžadováno zadavatelem. K verzování byla použita služba *GitHub*, kde má zadavatelská firma svůj soukromý repozitář. Díky tomu byla po celou dobu vývoje zajištěno zálohování a verzování kódu. Zároveň byla pomocí nástrojů poskytovaných službou *GitHub* umožněna kontrola vypracovaného kódu zadavatelem.

4.8 Použité externí knihovny

V rámci vývoje aplikace bylo využito několik externích knihoven. Při výběru knihoven byl brán ohled na vhodnost knihovny pro tuto aplikace, na míru jejího používání k vývoji Android aplikací a na licenci pod kterou je knihovna šířena.

Knihovny byla do aplikace přidány pomocí nástroje *Gradle*, doporučeného pro vývoj Android aplikací.

4.8.1 FirebaseUI

Knihovna obsahuje uživatelské rozhraní pro jednotlivé funkcionality poskytované službou Firebase. V aplikaci je využita tato knihovna pro Firebase přihlašování. Stačí pouze použít rozhraní knihovny pro zvolení způsobů přihlášení, tedy v tomto případě *email*, *Google* a *Facebook*, a knihovna už sama vytvoří aktivitu, v níž se uživatel následně přihlásí. Tato knihovna je doporučená ze strany společnosti Google[12].

4.8.2 Retrofit

Retrofit je knihovna používaná při vývoji Android aplikací k vytváření HTTP rozhraní pro jazyky Java a tedy i Kotlin. Tato HTTP rozhraní knihovna generuje na základě rozhraní, která definuje vývojář v jazyce Java, potažmo Kotlin[13].

4.8.3 RxJava

Jedná se o knihovnu rozšiřující možnosti reaktivního programování v jazyce Java. V reaktivním programování paralelně voláme metody, které mohou mít delší dobu běhu. V této aplikaci takto voláme metody komunikující s backendem, či metody zajišťující autentizaci ve službě Firebase. Je používán *Observer* pattern k reakci na doběhnutí těchto metod. V Android aplikacích obzvláště je toto široce využíváno a to především za účelem zachování responzivity aplikace[14].

4.8.4 MPAndroidChart

Tato knihovna umožňuje vykreslování grafů v aplikacích pro systém Android. Podporuje všechny běžné typy grafů a umožňuje jejich uzpůsobení konkrétnímu případu užití. Dokonce existuje i verze této knihovny pro iOS, což v budoucnu usnadní tvorbu případného portu[15].

4.8.5 Material-Calendar-View

Obsahem této knihovny je kalendář, který umožňuje zobrazování událostí, což není poskytováno základním *CalendarView* v Android knihovnách. Tato knihovna nabízí i vylepšený widget pro výběr data, ale pro účely aplikace postačuje widget přítomný v základních knihovnách systému Android. Tento kalendář je využit v aktivitě *CalendarActivity*[16].

Kapitola 5

Backend aplikace

Jedním z klíčových bodů této práce je využití REST API ke komunikaci s již existujícím backendem. Tento backend byl vyvinut již dříve ve společnosti *Skills Fighters* pro interní využití v rámci zaučování nového člena. Souběžně nebyla vyvinuta žádná frontend aplikace, která by umožňovala interakci s backendem a tedy jeho využití. Toto má nyní poskytnout aplikace vyvíjená v rámci této práce.

5.1 Stav backendu

Backend je napsaný na základě aplikačního rámce *Spring Framework*. Toto řešení bylo zvoleno primárně z důvodů praktických. Společnost *Skills Fighters* se z velké míry zabývá vývojem backendů pro své zákazníky a pro tyto účely je právě výše zmíněná technologie využívaná.

Rozhraní REST, jak je popsané v druhém bodě zadání této práce, je v backend aplikaci kompletně implementované. Pro aktivity a skupiny jsou implementované samostatné controllery v nichž jednotlivé metody představují endpointy, se kterými následně komunikují frontend aplikace, jako je právě tato Android aplikace.

Ověřování pomocí služby Firebase je již v backend aplikaci zcela vyřešené. Každý příchozí request prochází filtry, které ověřují přítomnost hlavičky obsahující ověřovací token vygenerovaný pro uživatele přihlášeného ve frontend aplikaci.

Komunikace s databází, v tomto případě *MySQL databází*, je zajištěna pomocí frameworku *Hibernate*. V databázi aplikace ukládá uživatele, aktivity a skupiny. Uživatelé jsou registrování při první přijatém requestu od daného uživatele z frontend aplikace, není třeba posílat zvlášť request o registraci. Za účelem zjednodušení vývoje byl využit nástroj *Docker* pro spouštění *MySQL* databáze. Toto řešení zjednodušuje případné testování jiných druhů databází, usnadňuje přenos vývoje mezi různými zařízeními a při opětovném spuštění poskytuje čisté prostředí pro spouštění testů.

Celý backend je důkladně otestovaný sadou unit, integračních a akceptačních testů. Funkčnost jeho implementovaných částí je tedy již z velké části ověřena a během psaní frontend aplikace se díky tomu žádné potíže s funkcionalitou backendu nevyskytovaly.

Backend aplikace ovšem nebyla na začátku vývoje Android aplikace trvale běžící na žádném serveru. Bylo tedy nutné nejprve zprovoznit backend aplikaci lokálně a eventuálně vyřešit nasazení backend aplikace na server, který umožní provoz aplikace v produkci.

5.2 Rozšíření

Ačkoliv byla backend aplikace z velké míry funkční, nepodporovala celou funkcionalitu Android aplikace, tak jak byla funkcionalita stanovena zadáním. V rámci rozšíření byl kladen velký důraz na minimální zásahy do již dříve vytvořených rozhraní. Backend byl tedy měněn primárně formou přidávání další funkcionality s důrazem na minimální zásahy do již funkčních částí. K vývoji bylo použito vývojové prostředí *IntelliJ IDEA* od společnosti *Jetbrains*.

5.2.1 Nasazení na cloud

Nejzásadnějším předmětem změn bylo nasazení backend aplikace na server. Jelikož *Skills Fighters* je malá společnost, nemá k dispozici vlastní server, na němž by backend mohl trvale běžet. Má ovšem k dispozici účet ve službě *Amazon Web Services*. Jedná se o subjekt ve vlastnictví společností Amazon, který poskytuje cloudové platformy pro nasazení webových aplikací.

Pro nasazení aplikace byla zvolena služba *Elastic Beanstalk*, která umožňuje nasazení aplikace zabalené ve formátu war. Tato služba automaticky vytvoří dle nastavení virtuální počítač a databázi pro nasazenou aplikaci. Po nahrání aplikace zabalené v již výše zmíněném formátu *war* ji tato automaticky služba nasadí. Následně služba poskytne endpoint, který mohou využít frontend aplikace k připojení k běžící backend aplikaci. *Elastic Beanstalk* poskytuje i možnost automatického škálování při vysokém vytížení spuštěné aplikace, to ovšem vzhledem k momentální míře využití nehraje roli.

5.2.2 Firebase Messaging na backendu

Dále bylo třeba uzpůsobit backend aplikaci k využití služby Firebase Messaging. Tato služba je v aplikaci používaná k zasílání notifikací všem zařízením uživatele o tom, že došlo ke změně dat a v reakci na tuto notifikaci si všechna zařízení uživatele znovu načtou data z backendu.

Firebase Messaging funguje tak, že backend posílá requesty o zaslání notifikace službě Firebase. Teprve Firebase tyto notifikace posílá samotným zařízením. V těchto requestech je definováno, jak má notifikace vypadat a

jakým zařízením se má zaslat. Není ovšem možné posílat zprávy přímo uživatelům na základě jejich identifikačních tokenů.

Zařízení jsou rozlišována pomocí registračních tokenů unikátních pro jednotlivá zařízení uživatelů. Tyto registrační tokeny je tedy nutné v backendu ukládat. V backend aplikaci byla vytvořena nová entita, která reprezentuje tyto registrační tokeny a jejich vazbu ke konkrétnímu uživateli v databázi.

Tyto registrační tokeny posílá Android aplikace. Byl tedy pro jejich zpracování vytvořen nový controller, který kontroluje přítomnost přijatého tokenu v databázi a pokud se tam ještě nenachází, token do databáze zapíše k uživateli, který mu daný token zaslal.

Na závěr bylo třeba upravit controllery pro aktivity a skupiny. Do všech metod, které jakýmkoliv způsobem mění v databázi uložené aktivity a skupiny, bylo přidáno posílání Firebase zprávy všem zařízením uživatele, za účelem provedení obnovy právě načtených dat v aplikaci.

5.2.3 Zbývající změny

Bylo doplněno zobrazování skupin na základě parentId dané skupiny. Předtím backend podporoval pouze zobrazení všech skupin daného uživatele a bylo tedy třeba skupiny filtrovat až v Android aplikaci. Toto zvětšovalo prodlevu mezi zvolením skupiny v aplikaci a zobrazením jejího obsahu.

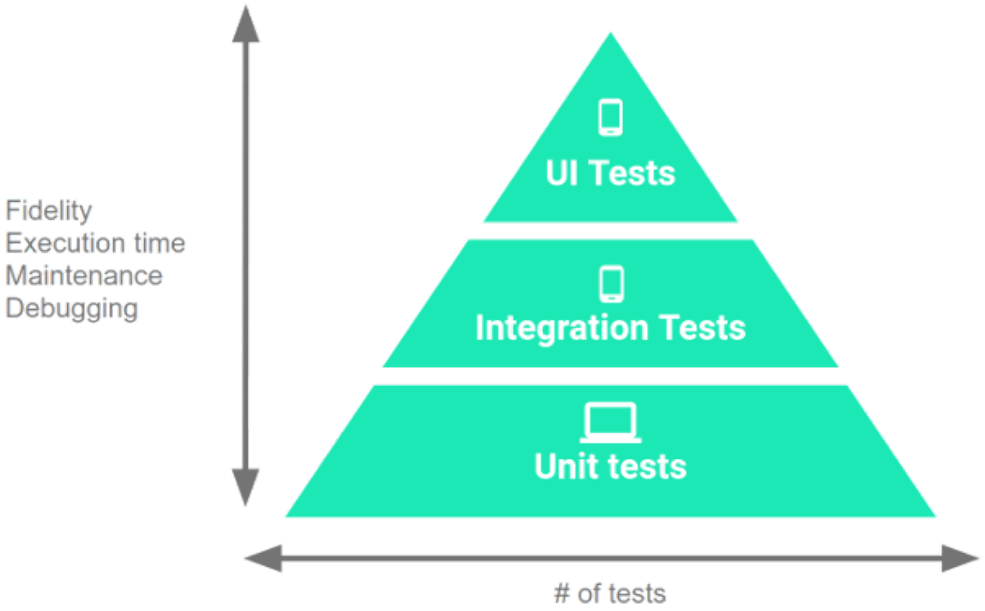
Pro rozšíření funkcionality aplikace bylo přidáno počítání aktivit ve skupinách. Backend tedy umožňuje pro zvolenou skupinu spočítat všechny aktivity, které pod ni spadají.

Na backend byla doplněna entita *UIOptions*, která má vazbu právě s jednou skupinou a obsahuje informace o jejím vzhledu. Pro komunikaci s Android aplikací v souvislosti s touto entitou byl vytvořen controller, který tuto komunikaci zajišťuje.

Kapitola 6

Testování

V rámci vývoje softwaru je důležité software správně testovat. Psaním testů během vývoje je zajištěna spolehlivost a stabilita kódu. Toto je důležité zejména při pokračujícím vývoji, abychom se ujistili, že nová funkcionalita a změny v kódu nenarušují již dříve hotovou funkcionalitu.



Obrázek 6.1: Graf zobrazující vztah počtu testů jednotlivých typů ku jejich věrnosti realitě, délce spouštění, náročnosti údržby a debugování.

Testů je použito několik druhů, které se liší nejen množstvím testovaného kódu a funkcionality, ale i způsobem, jakým je kód testován. Ačkoli unit testy, které testují pouze jednotlivé metody, jsou velice rychlé a je snazší jejich údržba, oproti rozsáhlejším testům netestují aplikaci jako celek a tedy plně netestují výslednou funkčnost. Jak napovídá obrázek výše, cílem je mít většinu

testů v podobě unit testů. Dokumentace pro vývojáře k systému Android doporučuje mít 70% unit testů, 20% integračních testů a 10% testů uživatelského rozhraní[17].

Všechny tyto testy jsou spouštěny prostřednictvím vývojového prostředí Android Studio.

6.1 Unit testy

Unit testy mají za cíl testovat nejmenší jednotku funkcionality, tedy v případě Android aplikací typicky jednotlivé metody. Vždy si vytváříme pouze třídu, do níž metoda spadá a na místo závislostí, které třída vyžaduje, vytváříme „Mock" objekty. Tyto objekty neimplementují žádnou funkcionalitu a tedy pokud některou z jejich metod voláme, v rámci testu je jí nastavena návratová hodnota. Tímto je dosaženo maximální izolace testované metody, potažmo třídy a je tedy testována pokud možno pouze jedna konkrétní funkcionalita[18].

Na systému Android rozlišujeme dva typy unit testů podle způsobu jakým jsou spouštěny. Jedná se o:

- Lokální testy jsou spouštěny přímo ve vývojovém prostředí a nevyžadují tedy ke svému spuštění běžící Android zařízení či emulátor. Jejich výhodou je vysoká rychlost, ale nejsou vhodné pro testování objektů se složitějšími závislostmi;
- Instrumentované testy narozdíl od lokálních testů vyžadují buď běžící Android zařízení nebo emulátor. Díky tomu je možné v aplikaci testovat i funkce, které vyžadují závislosti v systému v Android.

V této aplikaci jsou využity testy lokální, ale s využitím oficiální knihovny *Robolectric*, která umožňuje použití kontextu aplikace tím, že prostředí systému Android simuluje. Díky tomu není nutné spouštět testy v emulátoru nebo na reálném zařízení, což zvyšuje jejich rychlost[19].

6.2 Integrační testy

Integrační testy se od unit testů liší tím, že netestují pouze funkce samy o sobě v izolaci, ale už v kombinaci s dalšími částmi kódu. A to ať už uvnitř aplikace, tedy například spolupráci dvou tříd, ale i mimo samotnou aplikaci, tedy například komunikaci se serverem. Test tedy nereflektuje vnitřní obsah metody, tak jako to dělají unit testy, ale testuje až výstup těchto metod[17].

Pomocí integračních testů je v této aplikaci testována zejména komunikace mezi Android aplikací a backend aplikací pomocí REST API. Tyto testy fungují s plně běžící backend aplikací a testují tedy komunikaci tak, jak probíhá při normálním běhu aplikace. Při běhu integračních testů ovšem nelze zajistit přihlášení uživatele. Místo pravého Firebase autentizačního tokenu je

tedy využito napevno zvoleného debuggovacího tokenu. V případě, že backend přijme request s takovým tokenem, aplikace přeskočí autentizaci pomocí služby Firebase a považuje uživatele za správně přihlášeného testovacího uživatele. Tato možnost byla v backendu vytvořena již dříve a to v rámci testování backendu. Tyto testy jsou spouštěny pomocí knihoven *Robolectric* a *JUnit*.

6.3 Akceptační testy

Akceptační testy mají za cíl testovat, zda aplikace splňuje specifikaci, jak je definovaná zadáním. Tyto testy přistupují k aplikaci z pohledu uživatele. Akceptační testy nutně nepracují s vnitřní strukturou aplikace, přistupují k ní jako k černé skříňce[20].

Na Androidu jsou ekvivalentem akceptačních testů takzvané UI testy. Aplikace je spuštěna celá a tyto testy napodobují chování uživatele. V testech nejsou volány samostatné metody jednotlivých tříd. V testech je simulována interakce uživatele s uživatelským rozhraním aplikace. Netestujeme tedy, zda stisk tlačítka zavolá některou z metod aplikace, ale testujeme, jestli se nám po stisknutí tlačítka správně zobrazí menu, či zdali se nám správně zobrazuje další část aplikace[21].

Knihovna pro vytváření těchto testů v systému Android se nazývá *Espresso*. *Espresso* umožňuje přistupovat k jednotlivým komponentám obrazovky, testovat jejich stav a volat nad nimi metody, které napodobují interakci uživatele s aplikací. Tyto testy jsou spouštěny nad běžící aplikací, přičemž tato aplikace může být spuštěna buď v emulátoru nebo na fyzickém zařízení[22].

Pomocí těchto testů je otestováno několik základních funkcionalit, jako je přidání skupiny či správné načtení skupiny po jejím zvolení. Jelikož komunikace s backend aplikací nemusí být nutně okamžitá, jsou do testů přidány časové prodlevy, aby aplikace přijala data a test tedy proběhl správně. Zde je ukázka akceptačního testu, který testuje přidávání nových skupin.

```
@Test
fun addGroup() {
    onView(withId(R.id.refresh))
        .perform(click())

    Thread.sleep(5000)

    val numberOfItems = activityTestRule.activity.
        rv_item_list.adapter!!.itemCount

    onView(withId(R.id.fab_add))
        .perform(click())

    onView(withText(R.string.new_group))
        .check(matches(isDisplayed()))

    onView(withClassName(endsWith("EditText")))
        .perform(replaceText(testGroupName))

    onView(withText(R.string.save))
        .perform(click())

    Thread.sleep(5000)

    onView(withId(R.id.refresh))
        .perform(click())

    Thread.sleep(5000)

    assertEquals(numberOfItems + 1,
     activityTestRule.
        activity.rv_item_list.adapter!!.itemCount
     )
}
```

6.4 Uživatelské testování

Aplikace byla v průběhu vývoje testována mnohokrát různými uživateli. Jednalo se ovšem pouze o drobné testy, nebyly vytvořeny konkrétní testovací scénáře. Spíše se vždy jednalo o vyzkoušení nové funkcionality a využití zpětné vazby k dalšímu vývoji. Tyto připomínky sloužily primárně k dalším úpravám uživatelského rozhraní a k vývoji další funkcionality aplikace.

Kapitola 7

Nahrání na Google Play

Součástí zadání je i nahrání aplikace na Google Play. Pro nahrávání aplikací na Google Play je nezbytný vývojářský Google účet. Pro aktivaci vývojářských možností na již existujícím Google účtu stačí pouze zaplatit poplatek ve výši 25 $ společnosti Google[23].

Aplikace lze na Google Play nahrávat ve dvou podobách:

- **APK**, celým názvem *Android Package* je standardní formát pro distribuci aplikací na operačním systému Android. Každý APK soubor obsahuje všechna data, která jsou potřebná pro spuštění aplikace[1];

- **Android App Bundle** je novější formát pro nahrávání aplikací na Google Play. Tento formát zahrnuje zkompilovaný kód a všechny další součástí aplikace, ale samotnou tvorbu instalačních souborů **APK** ponechává na službě Google Play. Služba Google Play tak může poskytovat různým uživatelům různé podoby aplikace, v závislosti na jimi používaném zařízení, což může zmenšit velikost stahované aplikace[24].

Pro nahrání aplikace byl zvolen formát *Android App Bundle*, jelikož je tento formát novější a je v oficiální dokumentaci doporučený pro nahrávání aplikací na Google Play.

Pro samotné nahrání aplikace na Google Play je třeba získat podepsanou podobu aplikace. Toto lze vytvořit pomocí vývojového prostředí Android Studio s využitím klíče, který si vygenerujeme. Tento klíč slouží pouze k podepsání aplikace, když ji nahráváme, aplikace je před distribucí uživatelům podepsána ještě jedním klíčem, který je už ovšem vygenerovaný a spravovaný společností Google. Existuje možnost používat jeden klíč, jak k nahrávání aplikace, tak i k její distribuci uživatelům, ale toto řešení má nevýhody. V případě ztráty klíče není možné nahrávat nové verze aplikace jinak než jako novou aplikaci. Je nutno brát v potaz i riziko odcizení klíče. Proto bylo v rámci vývoje zvoleno řešení první[25].

Při nahrávání aplikace na Google Play si vývojář volí, jakým způsobem je aplikace distribuována. Aplikaci lze distribuovat pouze vývojářem určené skupině uživatelů, v tomto případě se jedná o *Interní test* nebo *Uzavřenou verzi*. Aplikaci lze dále nahrát jako *Otevřenou* verzi, kdy již bude distribuována mezi širokou veřejnost. Takto nahrané aplikace jsou ovšem ve službě Google Play zobrazovány odděleně od ostatních aplikací. V tomto režimu lze omezit maximální počet uživatelů. Posledním způsobem vydání aplikace je vydání do *Produkce*. Aplikace je v tomto režimu dostupná všem uživatelům[26].

Vyvíjená aplikace byla postupně nasazená v *Interním testu*, následně v *Uzavřené* verzi, v *Otevřené* verzi a na závěr byla aplikace nasazená i do *Produkce*. Aplikace byla nahrána pod názvem „Activity Tracker“, který byl zvolen zadavatelem. K únoru 2020 není na Google Play dostupná nejnovější verze Android aplikace v důsledku zrušení původně použitého vývojářského účtu.

Kapitola 8

Další rozvoj

Jelikož aplikace momentálně neumožňuje ukládání dat v zařízení a vždy je stahuje z backendu, je zcela závislá na Internetovém připojení. Jednou z možností, jak aplikaci rozšířit, je implementace ukládání dat offline s jejich pozdějším nahráním na backend.

Do budoucna je plánováno také vytvořit porty této aplikace i pro systém iOS a také webovou aplikaci. Toto podstatně rozšíří potenciální uživatelskou základnu. Obzvláště webová aplikace umožní implementaci pokročilejší funkcionality.

Další možností vylepšení aplikace je implementace komunity. Uživatelé by mohli mít možnost sdílet své skupiny s ostatními uživateli. Aplikace by poté mohla být použita i v jiném kontextu než jen sledování vlastních aktivit, aplikaci by bylo možné s patřičnými úpravami použít například ke sledování příchodů do zaměstnání jako digitální variantu docházkového systému.

Závěr

Dle zadání práce byla navržena a vytvořena Android aplikace, která umožňuje uživatelům sledování a následné seskupování svých aktivit. Zásadní částí aplikace je ukládání těchto aktivit na již existující backend běžící na vzdáleném serveru, k čemuž je nutná autentizace uživatelů. Ta je zajištěna službou Google Firebase, jak je vyžadováno zadáním práce. Krom autentizace je služba Google Firebase využita k zasílání zpráv z backend aplikace upozorňujících na změnu uložených dat. Samotná Android aplikace pak komunikuje s backend aplikací prostřednictvím rozhraní REST. Dále bylo vyžadováno nahrání vytvořené aplikace na obchod Google Play a její případná publikace, což bylo dle zadání provedeno. Posledním bodem zadání je vypracování testů k Android aplikaci a to unit testů, integračních testů a akceptačních testů.

V rámci vývoje aplikace bylo dosaženo všech cílů vytyčených v zadání. Aplikace umožňuje vytvářet vlastní skupiny, do nichž lze správně přidávat záznamy. Přihlášení s využitím služby Firebase funguje a to jak pomocí emailu, tak i pomocí poskytovatelů přihlášení Google a Facebook. Aplikace je nahrána na Google Play a je tam i publikována. Firebase Messaging, který je vyžadovaný v zadání bakalářské práce, je použit pouze pro posílání zpráv z backend aplikace o změně dat, na něž aplikace následně reaguje obnovením dat. Aplikace byla v průběhu vývoje testována a byly pro ni vypracovány testy dle zadání. Všechny zadáním specifikované druhy testů byly v aplikaci implementovány.

Největším problémem během vývoje bylo nasazení Android aplikací na Google Play. Ačkoliv samotný proces nahrání je poměrně jednoduchý, jelikož je aplikace závislá na běžícím backendu, bylo nutné nejprve tento backend nasadit na cloud. Přestože byla backend aplikace z velké části hotová a funkční, nebyla na nasazení ještě zcela připravena. Bylo tedy třeba backend aplikaci rozšířit a dokončit. Zadavatelská firma poskytla přístup k firemnímu účtu na Amazon Web Services, které poskytují jak virtuální prostředí pro spouštění aplikací, tak i potřebné databáze a to vše již s platnými certifikáty pro bezproblémovou internetovou komunikaci.

Následně vývoj pokračoval rozšířením funkcionality aplikace nad rámec

vymezený zadáním. Do aplikace bylo doplněno vykreslování základních grafů podle dat zapsaných v aplikaci. Dále pak bylo přidáno zobrazování záznamů v kalendáři. Navíc bylo umožněno označování jednotlivých skupin pomocí barev pro zvýšení přehlednosti aplikace.

Do budoucna je možnost rozšířit aplikaci o další funkcionality, jako je sdílení dat s ostatními uživateli či ukládání dat v aplikaci offline. Další cestou následného vývoje je vytvoření webové aplikace nebo portu aplikace pro systém iOS.

Literatura

[1] Google Inc.: *Application Fundamentals*. [cit. 2019-04-22]. Dostupné z: https://developer.android.com/guide/components/fundamentals

[2] Křížek, O.: Proč použít Kotlin místo Javy. October 2018, [cit. 2019-04-22]. Dostupné z: https://blog.seznam.cz/2018/10/proc-pouzit-kotlin-misto-javy/

[3] Google Inc.: *Distribution dashboard*. [cit. 2019-11-25]. Dostupné z: https://developer.android.com/about/dashboards

[4] Google Inc.: *Google Firebase*. [cit. 2019-04-22]. Dostupné z: https://firebase.google.com/

[5] Google Inc.: *Firebase Authentication*. [cit. 2019-04-22]. Dostupné z: https://firebase.google.com/docs/auth/

[6] Google Inc.: *Firebase Cloud Messaging*. [cit. 2019-04-22]. Dostupné z: https://firebase.google.com/docs/cloud-messaging/

[7] Pichlík, R.: REST. October 2007, [cit. 2019-04-22]. Dostupné z: https://dagblog.cz/a-rest-c5156313d79e

[8] Moon, J. W.: Onda - hobby and activity tracker. August 2017, [cit. 2019-04-22]. Dostupné z: https://play.google.com/store/apps/details?id=com.jungwoomoon.onda

[9] Holdings, A.: Habit Tracker. July 2018, [cit. 2019-04-22]. Dostupné z: https://play.google.com/store/apps/details?id=com.oristats.habitbull

[10] gerosyab: Daily Log. August 2018, [cit. 2019-04-22]. Dostupné z: https://play.google.com/store/apps/details?id=net.gerosyab.dailylog

[11] Idesis, S.: Learn the Model-View-Controller Pattern. June 2019, [cit. 2020-02-05]. Dostupné z: https://openclassrooms.com/en/courses/4661936-develop-your-first-android-application/4679186-learn-the-model-view-controller-pattern

[12] Firebase: *FirebaseUI for Android*. [cit. 2020-02-05]. Dostupné z: https://github.com/firebase/FirebaseUI-Android

[13] Square, Inc.: *Retrofit*. [cit. 2019-11-25]. Dostupné z: https://square.github.io/retrofit/

[14] RxJava: *RxJava: Reactive Extensions for the JVM*. [cit. 2019-11-25]. Dostupné z: https://github.com/ReactiveX/RxJava

[15] PhilJay: *MPAndroidChart*. [cit. 2019-11-25]. Dostupné z: https://square.github.io/retrofit/

[16] Applandeo: *Material-Calendar-View*. [cit. 2020-02-05]. Dostupné z: https://github.com/Applandeo/Material-Calendar-View

[17] Google Inc.: *Firebase Cloud Messaging*. [cit. 2019-04-22]. Dostupné z: https://developer.android.com/training/testing/fundamentals

[18] Nguyen, Q.: Falling in love with Android Testing. May 2017, [cit. 2020-02-05]. Dostupné z: https://medium.com/mindorks/falling-in-love-with-android-testing-dd11ffa6ac3e

[19] Google Inc.: *Firebase Cloud Messaging*. [cit. 2020-02-05]. Dostupné z: https://developer.android.com/training/testing/unit-testing

[20] Ghahrai, A.: Acceptance testing. December 2018, [cit. 2020-02-05]. Dostupné z: https://www.testingexcellence.com/acceptance-testing-agile

[21] Google Inc.: *Firebase Cloud Messaging*. [cit. 2020-02-05]. Dostupné z: https://developer.android.com/training/testing/ui-testing

[22] Google Inc.: *Firebase Cloud Messaging*. [cit. 2020-02-05]. Dostupné z: https://developer.android.com/training/testing/espresso

[23] Google Inc.: *How to use the Play Console*. [cit. 2020-02-05]. Dostupné z: https://support.google.com/googleplay/android-developer/answer/6112435

[24] Google Inc.: *About Android App Bundles*. [cit. 2020-02-05]. Dostupné z: https://developer.android.com/guide/app-bundle

[25] Google Inc.: *Sign your app*. [cit. 2020-02-05]. Dostupné z: https://developer.android.com/studio/publish/app-signing

[26] Google Inc.: *Prepare & roll-out releases*. [cit. 2020-02-05]. Dostupné z: https://support.google.com/googleplay/android-developer/answer/7159011

Příloha A

Seznam použitých zkratek

API Application Programming Interface

CRUD Create, Read, Update, Delete

HTML Hypertext Markup Language

HTTP Hypertext Transfer Protocol

JSON JavaScript Object Notation

MVC Model–view–controller

REST Representational State Transfer

UI User Interface

URL Uniform Resource Locator

URN Uniform Resource Name

XML Extensible Markup Language

Příloha B

Obsah přiloženého CD

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