Univerzita Hradec Králové
Fakulta informatiky a managementu
Katedra informatiky a kvantitativních metod

Modul sociálních sítí v systému podpory výuky

Bakalářská práce

Autor: Tomáš Pohořský
Studijní obor: Aplikovaná informatika

Vedoucí práce: doc. Mgr. Tomáš Kozel, Ph.D.

Hradec Králové 8 2019

Prohlášení:

Prohlašuji, že jsem tuto bakalářskou práci zpracoval pod vedením vedoucího práce samostatně a s použitím uvedených pramenů a literatury.

vlastnoruční podpis

V Hradci Králové dne 11.8.2019

Poděkování:

Tímto bych chtěl poděkovat svému vedoucímu bakalářské práce doc. Mgr. Tomáši Kozlovi, Ph.D. za metodické vedení, odborné připomínky a cenné rady ke zpracování této práce.

Anotace

Tato bakalářská práce má za cíl analyzovat, navrhnout, popsat a implementovat propojení sociálních sítí Facebook a Twitter s mobilní aplikací UHK Helper psanou v programovacím jazyce Java. Propojení aplikace a sociálních sítí je realizováno skrze programová rozhraní vybraných sociálních sítí: Facebook API a Twitter API. Význam práce spočívá v praktickém použití vybraných rozhraní a je přínosem pro začínající vývojáře mobilních aplikací, jakožto potencionální zdroj informací. Jako výsledek práce byl analyzován, navrhnut a implementován samostatně fungující modul splňující cíle práce, který bude následně vložen do samotné aplikace UHK Helper.

Annotation

Title: Social Networks Module for System of Education Support

The goal of this bachelor's thesis is to analyse, describe and implement the connection between social networks Facebook and Twitter and mobile application UHK Helper written in Java programming language. The connection between the application and the selected social networks is realized over the official interfaces of the selected social networks Facebook API and Twitter API. The importance of this work lies in the practical use of the selected interfaces and is a benefit for starting developers of mobile applications as a source of information. A separately functioning module that meets the goals of the work was analyzed, designed and implemented as the result of this bachelor's thesis and it will be subsequently inserted into the UHK Helper application.

Obsah

1 Úvod ... 1
2 Cíl práce ... 3
3 Programová rozhraní vybraných sociálních sítí ... 4
3.1 Programové rozhraní Facebook Graph API ... 4
3.1.1 Jak začít s Facebook Graph API ... 4
3.1.2 Princip fungování Facebook Graph API ... 6
3.1.3 FQL ... 6
3.1.4 FBML a FBJS ... 6
3.2 Programové rozhraní Twitter API ... 7
3.2.1 Registrace aplikace ... 7
3.2.2 Knihovna Twitter4J ... 8
4 Analýza a návrh modulu ... 9
4.1 Popis aplikace UHK Helper ... 9
4.1.1 Funkce aplikace UHK Helper ... 9
4.2 Analýza modulu ... 10
4.2.1 Zadání a představa o modulu ... 10
4.2.2 Funkcionality modulu ... 11
4.2.3 Součásti modulu ... 12
4.2.4 Use Case Diagram ... 12
4.3 Návrh modulu ... 13
5 Výběr a stručný popis technologií ... 16
5.1 Java a vývojové prostředí Android Studio ... 16
5.1.1 Jazyk Java ... 16
5.1.2 Android Studio ... 17
5.2 Operační systém Android ... 18
5.2.1 Historie a vývoj Androidu ... 18
5.2.2 Emulátor ... 18
6 Popis vybraných aspektů implementace ... 20
6.1 Základní principy tvorby Android aplikace v Android Studiu ... 20
6.1.1 Activity ... 21
6.1.2 AndroidManifest ... 21
6.2 Implementace přihlášení k Facebook účtu ... 22
6.3 Získávání statusů z vybraného facebookového účtu ... 24
6.4 Získávání událostí z vybraného facebookového účtu ... 26

6.5 Získávání dat z vybraného Twitter účtu 28
6.6 Implementace výběru zasílání dat 31
6.7 Ukládání získaných dat 32
7 Shrnutí výsledků 34
8 Závěr a doporučení 36
9 Citovaná literatura 37

Seznam obrázků

Obr. 1: Nastavení aplikace a generování klíčů (vlastní zpracování) 8
Obr. 2: Use Case Diagram modulu sociálních sítí (vlastní zpracování) 12
Obr. 3: Návrh layoutu hlavní stránky (vlastní zpracování) 14
Obr. 4: Návrh layoutu nastavení (vlastní zpracování) 15
Obr. 5: Vývojové prostředí Android Studio (vlastní zpracování) 17
Obr. 6: OS Android v emulátoru Andy Android 46 (vlastní zpracování) 19
Obr. 7: Android SDK emulátor (vlastní zpracování) 19
Obr. 8: Zakládání nového projektu (vlastní zpracování) 20
Obr. 9: Nastavení zasílacího filtru (vlastní zpracování) 35
Obr. 10: Vyhledání nových tweetů z vybraných účtů (vlastní zpracování) 35

Seznam ukázek zdrojových kódů

Ukázka 1: Ukázka souboru AndroidManifest.xml (vlastní zpracování) 21
Ukázka 2: Přihlášení k facebookovému účtu skrze modul 1 (15) 22
Ukázka 3: Přihlášení k facebookovému účtu skrze modul 2 (vlastní zpracování) 23
Ukázka 4: Získání statusů z vybraného facebookového účtu (vlastní zpracování) 25
Ukázka 5: Získání konkrétní události z Facebooku (16) 26
Ukázka 6: Získání události ze zadaného facebookového účtu (17) 27
Ukázka 7: Získání nejnovějšího tweetu ze zadaného účtu (vlastní zpracování) 28
Ukázka 8: Použití komponenty ListView (vlastní zpracování) 29
Ukázka 9: Otevření twitterového účtu dle vybraného tweetu (vlastní zpracování) 30
Ukázka 10: Implementace výběru zasílání dat (vlastní zpracování) 31
Ukázka 11: Implementace podmínky k získání statusů (vlastní zpracování) 31
Ukázka 12: Způsob uložení ArrayListu do sdílených preferencí (vlastní zpracování) 33
Ukázka 13: Opětovné získání ArrayListu ze sdílených preferencí (vlastní zpracování) 33

1 Úvod

V dnešní době se stále více využívá mobilních zařízení typu tablet, či smartphone neboli „chytrý telefon“. Mnoho lidí si již bez chytrého telefonu neumí představit svůj každodenní život. Budík, fotoaparát a GPS dnes patří mezi standartní prvky výbavy dnešních smartphonů. I přístup k internetu je též velmi hojně využíván skrze chytré telefony. Chytré telefony se tak staly součástí našeho každodenního života. Zatímco ještě před pár lety byl zájem v oblasti vývoje aplikací a jiného softwaru obecně orientován spíše na desktopové prostředí a notebooky, dnes se vývoj aplikací mohutně rozrůstá právě na mobilní platformu. Operační systém Android, který zaujímá největší procento mezi operačními systémy určených pro mobilní platformu (1), nabízí tak širokou škálu možností vývoje aplikací, že skoro každý, kdo má základní znalosti o vývoji aplikací, má možnost naprogramovat svou vlastní. Dnes již existuje nespočet aplikací, které lze zdarma, či za určitou sumu stáhnout a nainstalovat do chytrého mobilního telefonu a rychlým tempem vznikají další. Aplikace na dnešních mobilních telefonech mohou využívat gyroskopy a zjišťovat aktuální polohu pomocí GPS. Výkon hardwaru dnešních chytrých telefonu již lze srovnávat se stolními počítači a notebooky. Toto všechno vede k jejich masovému rozšíření a oblíbenosti. Přes to jsou dále objevovány nové nápady a koncepty na nové využití mobilních telefonů. Jedním z případů je například použití aplikace mobilního telefonu za účelem podpory výuky, nebo k přístupu na sociální sítě.

Že člověk obvykle ke spokojenosti potřebuje kontakt s jinými lidmi, je věc veřejně známá. Proto se v dnešní době, kdy došlo k masovému rozšíření chytrých mobilních zařízení a jejich připojení k internetu těší vysoké oblibě i sociální sítě, které umožňují lidem být v téměř každém okamžiku v kontaktu se svými přáteli. Sociální sítě jsou dle definice (2) definovány jako „*systém složený ze souboru sociálních aktérů, individuálně nazývaných uzly a sbírky sociálních vztahů, nazývaných vazby nebo odkazy*“. Jinými slovy, je to systém nejčastěji přístupný přes webové stránky, kam se po registraci přihlašují uživatelé a skrze které probíhá komunikace, navazování vztahů, vyjadřování názorů a mnoho dalších aktivit. Nejrozšířenější sociální sítí je dnes zcela jistě Facebook (3).

Facebook se stal masově používaný zejména díky několika funkcím, které nabízí. Jedná se především o sdílení obsahu, zasílání zpráv, stavbu osobního profilu, plánování událostí a mnoho dalšího. Přístup k Facebooku dnes nabízí i chytrý mobilní telefon skrze aplikaci, jejíž výhody jsou: Rychlejší přístup, menší datový přenos a vyšší přehlednost.

Další velice rozšířenou a oblíbenou sociální sítí je Twitter. Twitter funguje na principu tvorby a sdílení krátkých textových zpráv tzv. tweetů, které mohou ostatní sledující hodnotit, komentovat a sdílet. Je oblíbený díky možnosti rychle komentovat například nastalou situaci a je tak populární i u čelních představitelů různých zemí, čímž se stává mocným nástrojem při šíření informací. Za zmínku stojí i sociální síť Instagram. Instagram je sociální síť dostupná skrze mobilní aplikace a je určená pro sdílení a komentování fotografií zachycených na chytrý mobilní telefon, či již sdílené na jiných sociálních sítích. Instagram tedy na rozdíl od Facebooku a Twitteru přímo vyžaduje vlastnictví chytrého mobilního telefonu.

Všechny tyto uvedené sociální sítě mají společné to, že je využívá velké množství lidí a obsah sdílený skrze tyto sociální sítě se může virálně rozšířit mezi velké množství uživatelů během relativně krátké doby. To je jeden z důvodů, proč sociální sítě využívají i společnosti k propagování svých nabídek, či naopak společnosti, které chtějí ze sociálních sítí zjistit postoj zkoumané skupiny lidí k danému tématu. Tudíž i vývojáři aplikací mohou mít zájem využívat výhod a sílu sociálních sítí ve svých aplikacích. Často se v aplikaci, určené nejen pro mobilní platformu, objevuje tlačítko „Sdílet na Facebooku“ či „Tweetnout“. Vývojáři aplikace tím získají bezplatnou reklamu, jelikož u sdíleného obsahu se může objevit hláška „Sdíleno skrze aplikaci XY“, nebo „Sdíleno s aplikací XY“.

Aplikace využívající sociální sítě mohou být také zaměřeny k podpoře výuky a studia. Takovou aplikací je například UHK Helper (4), který byl vytvořen za účelem pomoci studentům Univerzity Hradec Králové a jehož propojení se sociálními sítěmi bude zajišťovat právě modul vyvíjený v rámci této bakalářské práce. Jak ale toto propojení aplikace se sociálnímu sítěmi funguje? Právě tato otázka je předmětem zkoumání v této bakalářské práci.

2 Cíl práce

Cílem této bakalářské práce je analyzovat, navrhnout a implementovat modul pro komunikaci aplikace UHK Helper s vybranými kanály sociálních sítí. Přínos modulu sociálních sítí spočívá ve zlepšení distribuce užitečných informací. Práce může též sloužit i jako zdroj informací pro začínající vývojáře mobilních aplikací pro platformu Android.

3 Programová rozhraní vybraných sociálních sítí

Co jsou to sociální sítě je již vysvětleno v úvodu této práce. Nyní je třeba vysvětlit co znamená pojem „programové rozhraní“ a k čemu se využívá. V angličtině se pro tento termín využívá zkratka API. API je zkratka z anglického *Application Programming Interface*, což se do českého jazyka překládá právě jako „programové rozhraní pro tvorbu aplikací“. API tedy může umožnit vývojářům třetích stran jakýsi návod či možnost, respektive způsob, jak přistupovat k datům a službám nějakého systému. Daniel Jacobson, Dan Woods a Greg Brail ve své knize APIs: A Strategy Guide (5) popisují, že API je v podstatě jakýsi kontrakt. Vývojáři využívají dané API, protože se mohou spolehnout na jeho správnost a funkčnost a zavazují se, že budou dodržovat podmínky používání daného API. (5) Obecně se dá říci, že API je soubor knihoven a procedur, které může vývojář aplikací použít při vývoji své aplikace. Například Facebook Graph API je tedy rozhraní, které nabízí knihovny a metody pro vývojáře, kteří chtějí svou aplikaci propojit či komunikovat s Facebookem.

3.1 Programové rozhraní Facebook Graph API

Obecná definice API byla vysvětlena výše, tato část se bude věnovat již konkrétnímu Facebook Graph API. API mohou být buďto komerční, kde je obvykle nutno zaplatit poplatek za jeho využití, anebo mohou být zdarma, což je právě případ Facebook Graph API. Toto konkrétní API tedy může zdarma využít kdokoliv bude chtít. Musí ale dodržet určité podmínky jeho použití, které jsou popsány na oficiálních stránkách Facebooku pro vývojáře. (6)

3.1.1 Jak začít s Facebook Graph API

Ze všeho nejdříve je nutné založit si „developer account“, neboli účet pro vývojáře. K tomu slouží webové stránky Facebooku pojmenované „Facebook for developers“. To se provede zcela jednoduše kliknutím na výrazné tlačítko „Get started“ a dále se řídit dle pokynů formuláře. Nutná bude i potvrzující SMS zaslaná na zadané telefonní číslo s obsahem verifikačního kódu. Ten je nutno zadat do připraveného textového pole ve formuláři. Nyní by měl, po úspěšném provedení všech kroků, existovat nový účet pro vývojáře. Celý postup je zcela intuitivní a přehledný.

Po vytvoření nového vývojářského účtu je nutné vyvíjenou aplikaci zaregistrovat a získat tak její ID, pod kterým ji bude Facebook evidovat, a které se při vývoji aplikace zadá do příslušné části kódu. To se provede pouhým stisknutím tlačítka „Create app“ a vyplněním formuláře. Po tomto kroku je aplikace zaregistrována.

V případě, že má vývojář v úmyslu z Facebooku tímto způsobem získávat data, je nutné získat příslušná oprávnění. Například pro účely modulu vyvíjeného v rámci této práce bylo bezpodmínečně nutné získat oprávnění nazvané „user-events“, které povoluje získávat data o událostech daného uživatele. Další druh takového oprávnění je například „user-posts“, dovolující v aplikaci zobrazit statusy daného uživatele. Dále například „user-photos“, „user-videos“ a mnoho dalších. Za zmínku ještě stojí „Page Public Content Access“ a „Instagram Public Content Access“. První jmenovaný povoluje číst data z uživatelského profilu a je nutnou součástí například pro čtení uživatelských statusů společně s oprávněním „user-posts“. Druhý jmenovaný dovoluje získávat data ze sociální sítě Instagram, o které bylo napsáno pár slov v úvodu práce.

O oprávnění lze požádat na stránkách Facebooku pro vývojáře. K tomu je nutná, kromě ukázky vyvíjené aplikace a popisu jejího použití, i verifikace uživatele. Většina oprávnění si vystačí s tzv. Individual verifikací, která se získá zasláním kopie občanského průkazu a vyplněním formuláře obsahující osobní informace o vývojáři. Některá oprávnění ovšem vyžaduje i tzv. Business verifikaci, která je určena pro podnikatele a k jejímu získání je nutné podat informace o firmě vývojáře.

Mezi další možnosti, které vývojářský účet nabízí, je například analýza počtu uživatelů aplikace, počtu volání (requestů), či počet chyb. Vše doplněno o přehledné grafy. Další výhodnou možností je přidání dalších vývojářů, nebo testovacích účtů.

3.1.2 Princip fungování Facebook Graph API

V zásadě se používají dva hlavní přístupy. A to protokol REST a SOAP. V obou případech se jedná o webové služby. Většina vývojářů preferuje REST díky jeho jednoduchosti. Metoda REST (7) (Representational State Transfer) vznikla v roce 2000 a funguje na principu odeslání požadavku (requestu) na danou adresu URL. Odpověď (response) je ve formátu XML, nebo JSON. XML neboli eXtensible Markup Language je značkovací jazyk sloužící k serializaci dat. JSON, JavaScript Object Notation slouží ke stejnému účelu a je tak jakýsi konkurent XML. SOAP (Simple Object Access Protocol) podobně jako REST získává data skrze HTTP protokol, ale zatímco REST zprostředkovává přístup k datům, SOAP zprostředkovává přístup ke službám. SOAP vznikl v roce 1998 za podpory Microsoftu.

3.1.3 FQL

Facebook Querry Language byl databázový dotazovací jazyk, který využíval Facebook Graph API do verze 2.1. Syntaxí se velice podobal klasickému dotazovacímu jazyku SQL. Sloužil k formulování dotazů pro databázi a vracel data ve formátu JSON. Od 8. srpna 2016 už jazyk není podporován. Facebook přestal FQL podporovat s přechodem na novou verzi API 2.1. Místo dotazování se pomocí FQL se dnes používá přímé volání requestů pomocí Facebook Graph API, které může vracet odpovědi například v JSON formátu. Tento způsob bude dále popsán v praxi v implementaci modulu.

3.1.4 FBML a FBJS

Jazyk FBML (Facebook Markup Language) je jakýmsi rozšířením jazyka HTML (Hyper Text Markup Language) pro potřeby Facebooku a může sloužit pro tvorbu aplikací pro Facebook. Jeho použití však není nutné. Jakožto produkt společnosti Facebook nabízí mnoho užitečných funkcionalit a dokáže zefektivnit vývoj aplikace. (8)

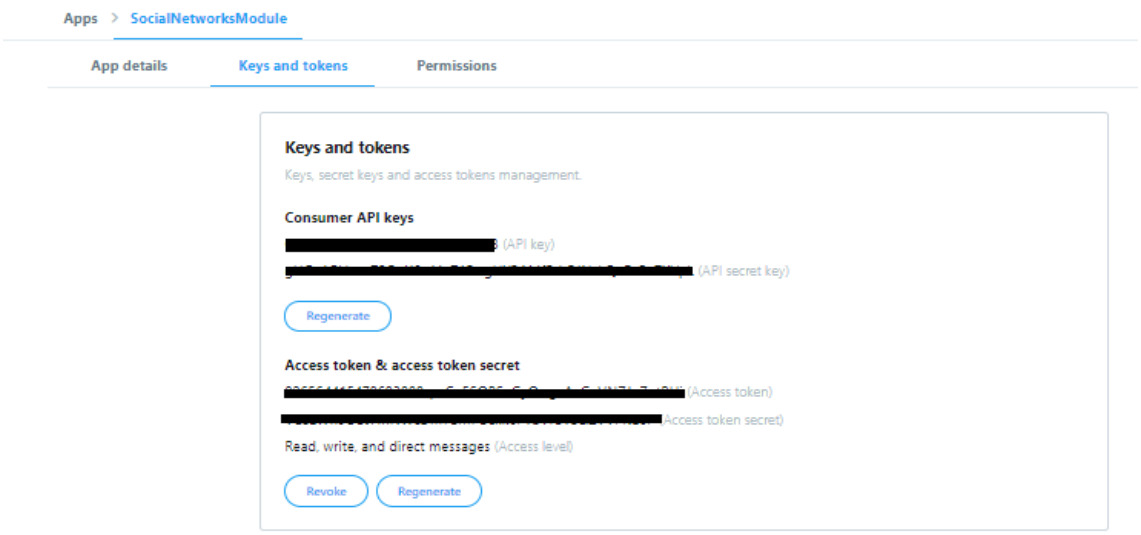
Zatímco FBML je rozšířením standartního HTML, FBJS (Facebook JavaScript) je mírně modifikovaný JavaScript, který byl upraven přímo pro použití při tvorbě Facebook aplikací a pro spolupráci s FBML.

3.2 Programové rozhraní Twitter API

V této části bude stručně pojednáno o programovém rozhraní Twitter API, registraci a nastavení nové aplikace a zmínku o knihovně twitter4j.

3.2.1 Registrace aplikace

Pro vývoj aplikací využívající data či služby Twitteru je vyžadován registrovaný účet. U Twitteru není třeba upgradovat běžný účet na vývojářský, jako je tomu například u Facebook Graph API. Zde stačí jen se přihlásit na Twitterových stránkách pro vývojáře pod stejným loginem a založit zde novou aplikaci, což je nutnou podmínkou dalšího vývoje. Po založení aplikace, které obsahuje například základní informace o vývojáři, k čemu by měla aplikace sloužit, nebo co by měla využívat, bude aplikaci přiřazeno unikátní ID a pod záložkou „Keys and tokens“ lze vygenerovat potřebné klíče. Jedná se o Consumer API Key, tajný Secret API Key a dva tokeny: Access token a Access secret token. Tyto čtyři řetězce budou třeba zejména k OAuth autorizaci, což je protokol sloužící k autentizaci a autorizaci uživatele fungující na mnoha platformách. Dále v náhledu aplikace pod záložkou „Permissions“ je nutné zkontrolovat, popřípadě nastavit, zda bude vyvíjená aplikace data z Twitteru pouze číst, nebo i zapisovat. Což by v praxi znamenalo zejména přidávání nových tweetů, či způsoby reakcí na ně. Důležité je, po provedených změnách, vygenerovat nové klíče a tokeny! Již z jejich použití a názvů vyplývá, že klíče i tokeny by neměly být sdělovány nepovolaným osobám.



Obr. 1: Nastavení aplikace a generování klíčů (vlastní zpracování)

3.2.2 Knihovna Twitter4J

Twitter4J je neoficiální Java knihovna určená pro komunikaci s Twitter API. (9) Knihovna je zdarma ke stažení a použití. Poslední stabilní verze je verze 4.0.7. Knihovna poskytuje mnoho funkcionalit, včetně již zmíněné OAuth autorizace. Pro využití v této bakalářské práci bylo nutné importovat soubor twitter4j-core. Což je základní komponenta této knihovny obsahující například podporu pro protokol REST, nebo vyhledávání dat na Twitteru. Praktické použití je zobrazeno v části implementace modulu.

4 Analýza a návrh modulu

Pojem analýza je podle (10) definován jako: „*Činnost, která rozkládá složitý systém tak, aby byl lépe pochopitelný.*“ Jedná se tedy rozložení nějakého problému na menší, srozumitelnější části. (10) Výsledkem analýzy může být například model. Model je jakési zobecnění modelovaného objektu, který umožňuje jeho lepší poznání. Model nesmí být příliš složitý, musí být snadný na pochopení, ale na druhou stranu ani moc jednoduchý, který by neplnil svou funkci. Musí být vhodně zvolen, aby zobrazoval žádanou skutečnost a důležité informace. Analýzou a návrhem informačních systému se v dnešní době zabývá celé odvětví softwarového inženýrství.

4.1 Popis aplikace UHK Helper

UHK Helper je aplikace určená pro mobilní platformu, pro operační systémy Android a iOS. Již z názvu aplikace je patrné, že se jedná o software spojený s Univerzitou Hradec Králové. Aplikace má za cíl informovat studenty všech fakult UHK o změnách v rozvrhu, o akcích univerzity, pomáhat studentům se studiem a zlepšit orientaci v jednotlivých budovách. Aby mohla aplikace vykonávat svou informační funkci, je nutná její spolupráce a komunikace s univerzitním systémem IS STAG. Aplikace má ambice v budoucnu omezit přihlašování do systému IS STAG a postupem času převzít část funkcionalit IS STAG. (4)

4.1.1 Funkce aplikace UHK Helper

Základní funkcí je po přihlášení se pod studentským účtem, zobrazování aktuálního rozvrhu předmětů a změn v rozvrhu. Aplikace též umožňuje zobrazit seznam učeben na všech budovách UHK, zobrazit aktuální obsazenost a předměty v učebnách a to včetně časových údajů. Další funkcionalita, kterou UHK Helper nabízí pro přihlášené uživatele, je zobrazení známek, zápočtů, termínů zkoušek a harmonogram akademického roku. Dále také aplikace umožňuje získávat upozornění a notifikace o událostech univerzity dle výběru, který daného studenta zajímá. Nutno zmínit i možnost změny hesla po vypršení jeho doby platnosti.

4.2 Analýza modulu

Nyní k analýze samotného modulu. Nejprve je nutné analyzovat jaké funkcionality by měl modul sociálních sítí nabízet, k tomu lze využít Use Case Diagram. Use Case Diagram, neboli česky Diagram případů užití, zkráceně UCD, je součást jazyka UML. UML, neboli Unified Modeling Language je jazyk, který se využívá v softwarovém inženýrství, jakožto způsob jakým lze analyzovat informační systém. UCD umožní graficky znázornit, co by měl modul nabízet a je snadno pochopitelný i pro uživatele neznající jazyk UML.

4.2.1 Zadání a představa o modulu

Cílem práce je analyzovat, navrhnout a implementovat modul pro komunikaci aplikace UHK Helper s vybranými kanály sociálních sítí. Problém, který by měla implementace modulu vyřešit je skutečnost, že UHK Helper zasílá uživatelům upozornění o univerzitních akcích nezávisle na sociálních sítích. Pokud tedy skupina studentů nemá nainstalovanou aplikaci UHK Helper, přichází tato skupina o toto upozornění. Tato situace se momentálně řeší duplikováním oznámení na sociálních sítí typu Facebook, či Twitter. Nastává tedy vysoká potřeba po onom modulu, který by oznámení a události vyvěšené na sociální sítě přenesl i do aplikace UHK Helper. Právě zde je nutné zdůraznit část cíle práce „s vybranými kanály sociálních sítí“. Modul musí umět filtrovat jen stránky univerzity a její příspěvky. Není žádoucí, aby modul upozorňoval i na akce mimo Univerzitu Hradec Králové, což by mohlo vést k jistému obtěžování uživatele aplikace množstvím notifikací nesouvisejících s univerzitou.

Představa o modulu je tedy jasná: Modul by měl umožnit aplikaci UHK Helper získávat oznámení o novinkách a událostech z konkrétních účtů univerzity na sociálních sítích a zasílat jej jakožto klasické notifikace uživatelům aplikace UHK Helper. Je nutné se zamyslet i nad možností dát uživateli aplikace jistá práva a možnosti k vlastní úpravě zasílacího filtru. Je pravděpodobné, že části uživatelům se může zdát přijímání notifikací obtěžující. To by mohlo vést k tomu, že si aplikaci UHK Helper odinstalují. To je ovšem situace, které je nutné se vyhnout. A to lze provést implementací uživatelské volby. Například vypnout upozornění ze sociálních sítí. Další možností uživatelské volby úpravy filtru je například podle

fakult. Je zbytečné, aby například student studující na fakultě informatiky a managementu dostával informace o interních akcích Filosofické Fakulty. Stejně tak je možné implementovat možnost výběru konkrétní sociální sítě. Zda chce uživatel získávat notifikace například jen z Facebooku, a nikoliv z Twitteru. Co se druhů oznámení týče, může se jednat buď o obyčejné statusy z univerzitních účtů, nebo akce, respektive facebookové události pořádané univerzitou, či jednotlivými fakultami. Právě facebookové události by měly být primární funkcionalitou řešenou v modulu sociálních sítí.

4.2.2 Funkcionality modulu

Základní funkcionality, které by měl modul do aplikace UHK Helper přidat lze vypsat bodově:

- Přijímat novinky a zejména facebookové události z vybraných kanálů sociálních sítí a zobrazovat je v uživatelsky přívětivé formě uživatelům aplikace UHK Helper.
- Možnost uživatelům aplikace nastavit jaké konkrétní sociální sítě sledovat.
- Možnost reagovat na zasílané notifikace.

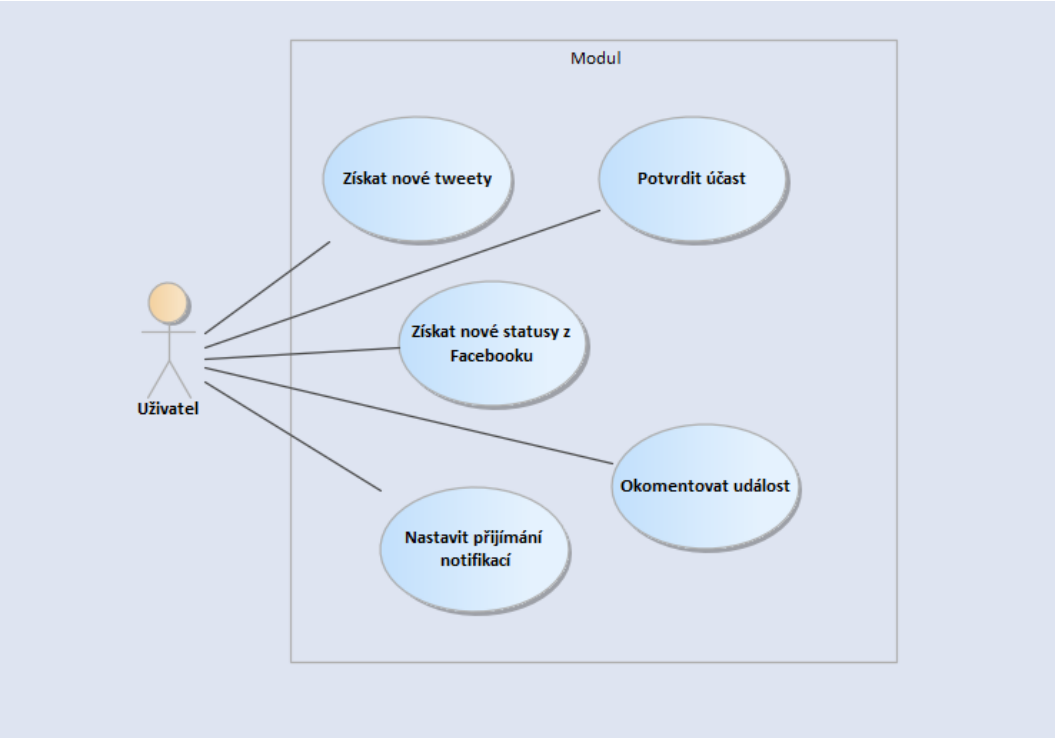
První dva body jsou již vysvětleny výše a není třeba je zde dále popisovat. Co se třetí funkcionality týče, možností jak reagovat na zaslanou notifikaci se značně liší v závislosti, z jaké sociální sítě oznámení pochází, jakého je typu a jakou možnost reakce by uživatel uvítal. Základní a nejpoužívanější reakcí na sociálních sítích, zejména na Facebooku je jistě „Like button“, neboli vyjádření, že se uživateli daný příspěvek líbí. Další možností reakce je přidání komentáře. Důležité je vzít v úvahu i jiné druhy oznámení než obyčejný textový příspěvek. V případě, že je například na Facebooku příspěvek typu anketa, či například událost, na kterou má možnost uživatel Facebooku reagovat přihlášením se, bylo by na místě, kdyby se tato možnost objevila i v aplikaci UHK Helper.

4.2.3 Součásti modulu

Samotný modul musí být rozdělen do několika částí. A to již z důvodu rozdílných programových rozhraní (API) jednotlivých sociálních sítí. Musí být implementována část komunikující s Facebookem a další s Twitterem. Tyto dvě části by měly fungovat nezávisle na sobě a musí být spojeny do jednoho celku – modulu, který bude nabízet i již zmíněné uživatelské nastavení. Ten bude implementován do samotné aplikace UHK Helper.

4.2.4 Use Case Diagram

Co je diagram typových úloh již bylo vysvětleno. Nyní je nutné popsat samotný diagram popisující případy užití modulu sociálních sítí, který je zde nutno analyzovat.



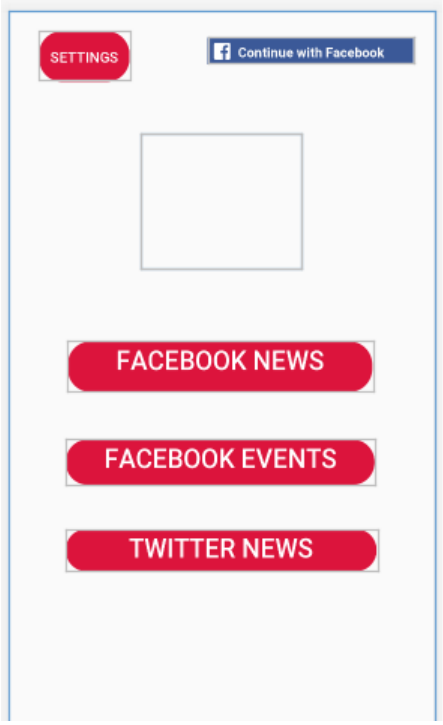
Obr. 2: Use Case Diagram modulu sociálních sítí (vlastní zpracování)

Diagram typových úloh popisuje případy užití jednoto aktéra – klasického uživatele aplikace UHK Helper. Uživateli aplikace má tedy modul nabízet především upozornění a oznámení na univerzitou konané akce a změny. Dále by mu měl modul nabízet možnost nějaké zpětné reakce. To především komentováním a potvrzením účasti na akce. V neposlední řadě by mu měl modul umožnit získávání notifikací ze sociálních sítí nastavit, či zcela zakázat.

Organizátor akce, jenž může být jakýkoliv zaměstnanec univerzity, a který má za cíl přidat oznámení na sociální sítě, musí zajistit, aby se oznámení dostalo mezi co nejvíce lidí. Organizátor akce může být samozřejmě i uživatel aplikace.

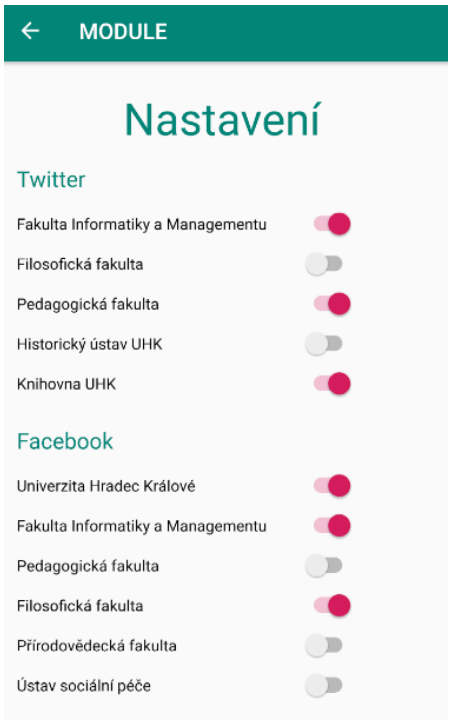
4.3 Návrh modulu

Modul sociálních sítí bude vyvíjen samostatně jako nezávislá spustitelná aplikace pro android. Teprve po jejím dokončení, ověření funkčnosti a testování bude připojen do samotného UHK Helperu. V kapitole o analýze modulu byla popsána jistá představa, co by měl modul obsahovat a umět. V této části bude modul popsán podrobněji a s ukázkami komponent, které budou v implementaci využity. V první fázi modulu je třeba navrhnout layout, neboli rozložení komponent na stránky. Následně bude popsáno jejich použití.



Obr. 3: Návrh layoutu hlavní stránky (vlastní zpracování)

Na obrázku č. 3 je zobrazen prvotní návrh layoutu hlavní stránky. V horní části lze najít tlačítko s nastavením, které uživateli otevře novou stránku, neboli Aktivitu. Vedle něj je přihlašovací tlačítko k Facebooku. Tato komponenta, kterou nabízí k použití Facebook API, zavolá již hotový přihlašovací formulář. Uprostřed je patrná komponenta ImageView pro zobrazení profilového obrázku přihlášeného uživatele. Níže jsou umístěna tři tlačítka. Každé po stisknutí spustí novou aktivitu, ve které se zobrazí získaná data. Možnost zobrazení dat do komponent TextView je jen jedna z možností, jak prezentovat data. Další je například zasílání notifikací, které využívá UHK Helper. Systém zasílání notifikací by mohl být implementován v modulu po celkovém spojení s UHK Helperem. V rámci návrhu a prvních fází implementace budou využity komponenty TextView a ListView kvůli jejich jednoduchosti a přehlednosti.



Obr. 4: Návrh layoutu nastavení (vlastní zpracování)

Na obrázku č. 4 je zobrazen návrh nastavení z jakých účtů chce uživatel získávat novinky. Modul zcela jistě nebude uživateli poskytovat možnost vlastního zadávání sledovaných účtů, jelikož by bylo přinejmenším velmi složité zjišťovat, zda se jedná o validní univerzitní účty. Univerzitní účty tedy budou v modulu zadány jako konstanty v kódu. Použitou komponentou pro nastavení je Switch, kterou nabízí Android SDK a jejíchž použití je nenáročné a spolehlivé.

5 Výběr a stručný popis technologií

Implementace modulu sociálních sítí využívá množství technologií. Modul se musí napsat pomocí vhodného programovacího jazyka. Pro praktickou implementaci byl vybrán známý a celosvětově rozšířený, objektově orientovaný jazyk Java. Pro výběr právě tohoto jazyka hovoří zejména jeho jednoduchost a robustnost. Jakožto vývojové prostředí pro vývoj modulu bylo vybráno Android Studio. Mezi dalšími alternativami lze uvést například vývojová prostředí Eclipse, NetBeans, či IntelliJ Idea. Volba Android Studia byla opodstatněná. Je zde sice kladen důraz na svižnost a přehlednost, což splňují i výše zmíněná vývojová prostředí, ovšem zásadní výhodou Android Studia je pro potřeby vývoje modulu množství doplňků, pluginů a zejména nástrojů, které Android Studio nabízí přímo pro vývoj aplikací pro Android. Dále je nutné alespoň stručně popsat operační systém Android, pro který je celá aplikace určena.

5.1 Java a vývojové prostředí Android Studio

Programovací jazyk Java je v této práci popsán jen stručně, jelikož se jedná o jeden z nejrozšířenějších a nejpoužívanějších programovacích jazyků. Poté následuje stručný popis vývojového prostředí Android Studio.

5.1.1 Jazyk Java

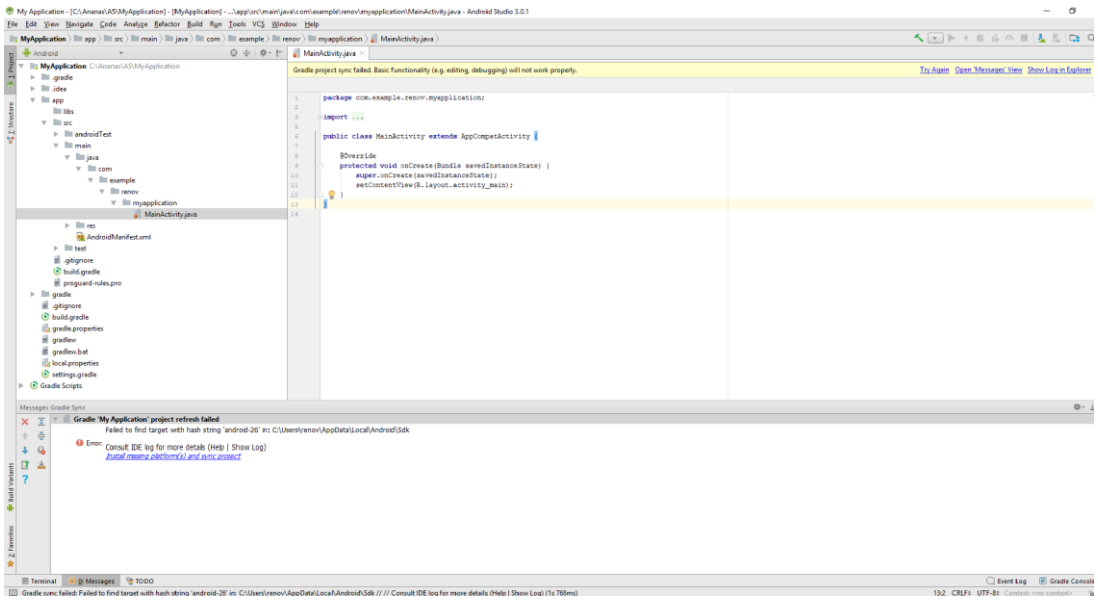
Java je objektově orientovaný programovací jazyk vytvořený v roce 1995. James Gosling, hlavní tvůrce a vývojář Javy se rozhodl držet se při vývoji několika principů (11). Java má jednoduchou a snadno pochopitelnou syntaxi, která je založena na C/C++. Je objektově orientovaná, klade vysoký důraz na robustnost a bezpečnost. Pojem robustnost znamená, že se program napsaný v robustním jazyce bude snažit fungovat dál i přes chybu, která může nastat. Dále je Java nezávislá na architektuře a přenositelná. Důraz je kladen na rychlost a výkonnost. Je interpretovaná a podporuje více vláknové aplikace.

Java se tedy díky těmto principům stává jedním z nejvíce používaných programovacích jazyků jak na klasických stolních počítačích, kde se uplatňuje Java SE (Standard Edition) tak i na webu – Java Enterprise Edition.

5.1.2 Android Studio

K programování aplikace je nutné vývojové prostředí. Důvody, proč padl výběr zrovna na Android Studio, jsou již vysvětleny na začátku této kapitoly. Moderní vývojová prostředí umožňují nespočet funkcí, které samotné psaní zdrojového kódu zpříjemňují a zvyšují tak komfort a rychlost vývoje. Jedná se například o funkce generování částí kódu jako jsou getry, setry či konstruktory. Dále se jedná o našeptávače, zvýraznění syntaxe, upozornění na syntaktické chyby, překladač, compiler anebo debugger. Toto všechno Android Studio nabízí. Aplikace psané v Android Studiu jsou ale určeny především jako mobilní aplikace, což přesně splňuje požadavky této práce.

Android Studio vzniklo v roce 2013 a je založené na vývojovém prostředí InteliJ IDEA. Jeho hlavní vývojář je společnost JetBrains. Podrobnosti a návody na práci s Android Studiem jsou popsány v uživatelské příručce (12). Android Studio se na první pohled příliš neliší od jiných vývojových prostředí. Částečně je to dáno tím, že vychází z prostředí IntelliJ Idea. V levé části je tradičně správce balíčků a souborů, uprostřed samotné místo pro psaní kódu a ve spodní části konzole. Vše je patrné na obrázku číslo 5 níže.



Obr. 5: Vývojové prostředí Android Studio (vlastní zpracování)

5.2 Operační systém Android

Operační systém Android je dnes zcela jistě nejrozšířenějším operačním systémem na chytrých telefonech (13). Proto zde bude stručně popsána jeho historie, vývoj a dnešní situace. K testování implementace modulu bude použit emulátor umožňující běh mobilní aplikace na desktopové platformě.

5.2.1 Historie a vývoj Androidu

První verze Androidu byla vydána v roce 2008 s příchodem chytrých telefonů (14). Tím vytlačil z prvních příček do té doby používaný Symbian OS a tlačítkové telefony. Ty se ovšem i dnes těší velké oblibě především pro svou jednoduchost, odolnost a nízkou cenu. Dnes vyvíjí Android společnost Google, který systém postavil na jádře GNU/Linuxu (14). Android je vydáván po verzích. Nejnovější verze je Android 9.0 Pie, který byl uveden 6. srpna 2018 (14).

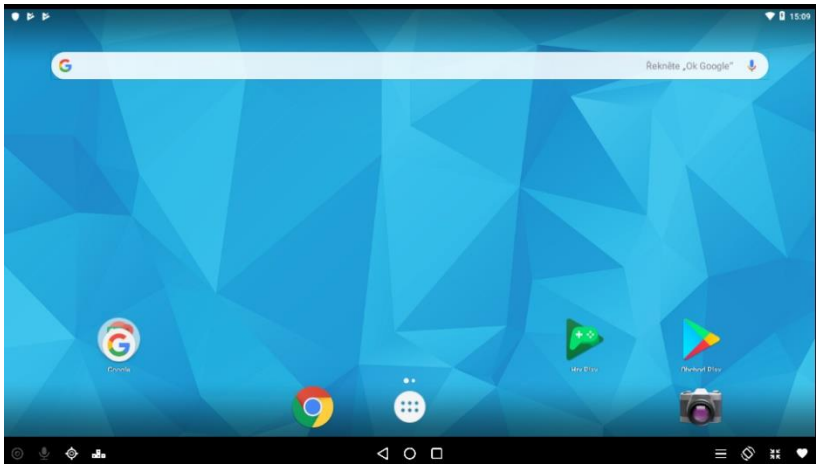
5.2.2 Emulátor

Emulátorem rozumíme typ softwaru, který umožňuje na dané platformě spouštět aplikace určené pro jinou platformu. V praktickém použití to znamená například situaci: Na fyzickém PC je nainstalován operační systém Windows 10. V tomto systému se spustí emulátor, který vytvoří virtuální prostředí operačního systému Android. Uživatel tedy může na tomto virtuálním zařízení spouštět a používat software určený na Android. Pro potřeby této bakalářské práce byla využita právě výše uvedená situace, zejména k testování modulu při implementaci. Důvodů, proč bylo k tomuto účelu použito emulátoru namísto fyzického zařízení bylo několik:

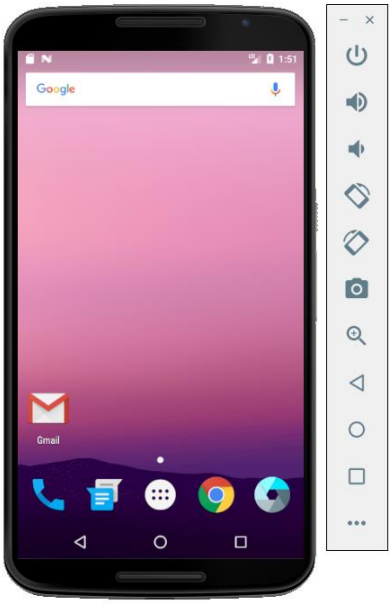
- Jedná se o jednoduché a bezpečné řešení
- Snadné získání informací z logu, nebo konzole
- Možnost testování na několika verzích Androidu

Emulátorů je na trhu mnoho, freeware i placené verze. Pro příklad lze uvést např. Andy Android 46, který je dostupný zdarma. Pro potřeby testování modulu v průběhu implementace byl ale použit Emulátor Android SDK, který je již integrovaný v samotném Android Studiu, a který dovoluje vybrat parametry, jako

úhlopříčku displeje, typ zařízení (smartphone, tablet…) a verzi Androidu. Emulátor byl při vývoji použit v začátcích a v průběhu implementace modulu. Ve finální fázi testování bylo nutné testovat funkčnost modulu i na fyzickém zařízení.



Obr. 6: OS Android v emulátoru Andy Android 46 (vlastní zpracování)



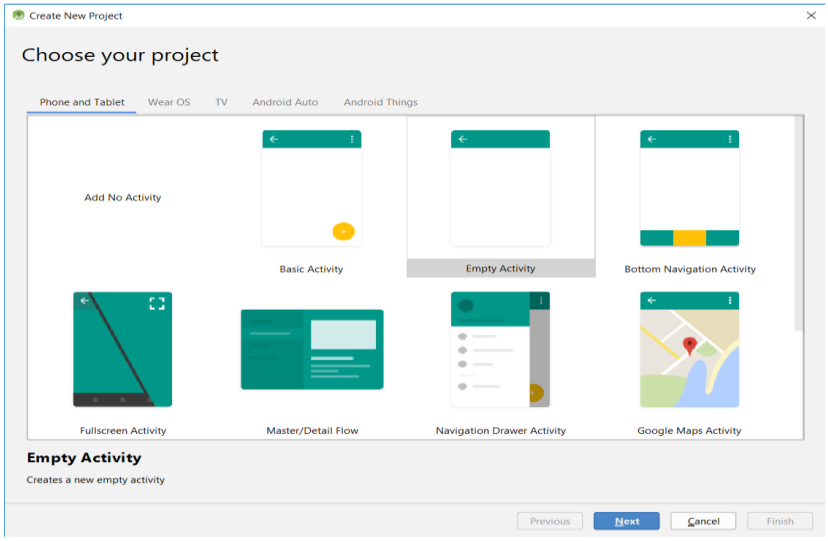
Obr. 7: Android SDK emulátor (vlastní zpracování)

6 Popis vybraných aspektů implementace

V této části práce budou popsány vybrané části implementace modulu společně s ukázkami zdrojových kódů. Nejdříve bude prakticky popsána základní orientace ve vývojovém prostředí a základní entity. Poté budou popsány a vysvětleny vybrané části implementace modulu.

6.1 Základní principy tvorby Android aplikace v Android Studiu

Po spuštění Android Studia vyzve uživatele nabídka k založení nového projektu. Součástí procesu zakládání nového projektu je i výběr typu Aktivity na různý hardware. Tento krok je zobrazen na obrázku č. 8.



Obr. 8: Zakládání nového projektu (vlastní zpracování)

Pro účely implementace modulu sociálních sítí byla vybrána možnost *Empty Activity* v sekci *Phone and Tablet*. Tímto se vytvoří nová aktivita zvaná MainActivity, která bude typu *Empty Activity*. MainActivity je první aktivita, která se zobrazí po spuštění aplikace. V dalším kroku se nastaví jméno aplikace, programovací jazyk (Java nebo Kotlin) a nejnižší číslo verze Android API. Čím nižší číslo API (čím starší), tím větší je jistota, že aplikaci budou podporovat všechna mobilní zařízení. Android Studio umí vypočítat přibližně kolik procent zařízení bude aplikaci podporovat. Čím novější API tím více vývojářských možností API poskytne, ale je riziko, že na smartphonech se starší verzí Androidu nebude aplikace plně funkční.

6.1.1 Activity

Activity, česky aktivita je základní komponenta grafické reprezentace obsahu aplikace. Po vytvoření aktivity se vytvoří dva soubory v oddělených balíčcích. Jsou to například již zmíněná *MainActivity.java,* se kterou lze pracovat jako s běžnou Java třídou, a která dědí ze třídy *AppCompatActivity*. A XML soubor *activity_main.xml*, který se nachází v balíčku *layout*. A právě do tohoto souboru se pomocí XML přidávají různé grafické komponenty jako je například Button, TextView, Switch, dále různé widgety a kontejnery které Android SDK nabízí. Pro usnadnění a urychlení vývoje nabízí Android Studio velice důležitou možnost nejen textového náhledu, ale i jakési grafické rozhraní, ve kterém lze tažením myši přidávat komponenty a zadávat jim parametry bez použití XML. Mezi oběma náhledy však lze jednoduše přepínat tlačítkem v dolní části náhledu a lze při vývoji oba náhledy kombinovat.

6.1.2 AndroidManifest

AndroidManifest je XML soubor nacházející se v kořenovém adresáři projektu, který obsahuje základní informace o aplikaci a všech jejích částech. Pro potřeby implementace modulu sociálních sítí tento dokument obsahuje například informace o dalších aktivitách, anebo informaci, jakou je například ID Aplikace pro komunikace s Facebookem, respektive s Facebook Graph API. Obě situace znázorňuje následující kód na ukázce č. 1.

```
<manifest>
<application>
        <activity android:name=".Tweet">
            <meta-data
            android:name="android.support.PARENT_ACTIVITY"
            android:value="com.example.module.Tweet" />
        </activity>
        <activity android:name=".Settings">
           <meta-data
            android:name="android.support.PARENT_ACTIVITY"
            android:value="com.example.module.Settings" />
        </activity>
        <meta-data
        android:name="com.facebook.sdk.ApplicationId"
        ndroid:value="FacebookID" />
    </application>
</manifest>
```

Ukázka 1: Ukázka souboru AndroidManifest.xml (vlastní zpracování)

Syntaxe jazyka XML zde nebude popisována, předpokládá její znalost. V ukázce č. 1 jsou zapsány dvě aktivity, a to aktivita Tweet a aktivita Settings. Obě obsahují záznam o jejich rodičovské aktivitě. Tento záznam je důležitý při implementaci tlačítka pro návrat k původní aktivitě. Dále je v ukázce záznam ohledně facebookového ID aplikace, které však v textu není z bezpečnostních důvodů zapsáno. O tomto ID již byla řeč v předchozí kapitole. Pro připomenutí, jedná se o řetězec vygenerovaný při registraci aplikace v rozhraní Facebooku pro vývojáře a slouží k identifikaci dané aplikace.

6.2 Implementace přihlášení k Facebook účtu

Pro získání informací o událostech a novinkách z facebookových kanálů pomocí Graph API je nutné se skrze aplikaci přihlásit k facebookovému účtu. Proto i tento aspekt je v modulu sociálních sítí implementován. Níže je ukázka zdrojového kódu, který přihlášení obstarává. Po přihlášení se k facebookovému účtu aplikace v tomto případě získá profilový obrázek uživatele a uloží jej do připravené datové struktury v *MainActivity*. Zdrojový kód, který zajistí přihlášení a získání profilového obrázku uživatele je k dispozici v ukázkách č. 2 a 3.

```
protected void onCreate(Bundle savedInstanceState) {
    private CallbackManager callbackManager = CallbackManager.Factory.create();
    private ImageView imgAvatar = = (ImageView) findViewById(R.id.avatar);
    private LoginButton loginButton = (LoginButton)findViewById(R.id.login);

    loginButton.setReadPermissions(Arrays.asList("public_profile", "email"));
    LoginManager.getInstance().logInWithReadPermissions(
                                        this, Arrays.asList("public_profile"));

    loginButton.registerCallback(callbackManager, new FacebookCallback<LoginResult>(){

       @Override
       public void onSuccess(LoginResult loginResult) {
          //Následující kód z ukázky č. 3
       }

       @Override
       public void onCancel() {
          Log.i("Cancel", "Login was cancelled!");
       }

       @Override
       public void onError(FacebookException error) {
          Log.i("Error", "Error in Facebook login process!");
       }
    });
}
```

Ukázka 2: Přihlášení k facebookovému účtu skrze modul 1 (15)

Ukázka č. 2 obsahuje metodu *onCreate()*, která se volá automaticky po vytvoření aktivity, ve které je tato metoda napsána. V této metodě bude probíhat celý proces přihlášení. Prvně se deklaruje objekt *CallbackManager*, který má za úkol zajišťovat zpětné volání z této aktivity zpět do Facebook SDK. Následně dojde k vytvoření datové struktury pro uložení profilového obrázku pomocí komponenty *ImageView,* které se přiřadí komponenta z XML souboru této aktivity skrze id komponenty. Ve stejném principu se deklaruje *LoginButton*, což je předpřipravená komponenta určená právě pro tento případ použití integrovaná v Graph API. Dále se požádá o oprávnění číst z veřejného facebookového profilu pomocí metody *setReadPermission()*, která přijímá jako první parametr informaci, že se jedná o veřejný profil a druhý parametr, který specifikuje, co bude po přihlášení vyžadováno. V tomto ukázkovém případě je to email.

Dále se vytvoří již hotový předpřipravený přihlašovací formulář, do kterého uživatel zadá své uživatelské jméno a heslo. Formulář je stejně jako přihlašovací tlačítko součástí Graph API. Následují tři předdefinované metody. Jsou to metody *onSuccess(), onCancel() a onError().* Metoda *onSuccess()* se zavolá, pokud přihlášení proběhne v pořádku. Metoda *onCancel()* se zavolá, pokud se z nějakého důvodu uzavře přihlašovací okno a metoda *onError()* jen pokud dojde k neočekávané chybě, kterou pomocí funkce Log.i() vypíše do konzole.

Další postup je umístěn v metodě onSuccess() a je k vidění na ukázce č. 3.

```
public void onSuccess(LoginResult loginResult) {
   progressDialog = new ProgressDialog(MainActivity.this);
   progressDialog.setMessage("Collecting data...");
   progressDialog.show();
   GraphRequest request = GraphRequest.newMeRequest(loginResult.getAccessToken()
                                    , new GraphRequest.GraphJSONObjectCallback() {
      @Override
      public void onCompleted(JSONObject object, GraphResponse response) {
         progressDialog.dismiss();
         URL profile_picture = new URL("https://graph.facebook.com/"
                                        + obj.getString("id")
                                        + "/picture?width=150&height=150");
         Picasso.with(this).load(profile_picture.toString()).into(imgAvatar);
      }
   });
   Bundle parameters = new Bundle();
   parameters.putString("fields", "id, email");
   request.setParameters(parameters);
   request.executeAsync();
}
```

Ukázka 3: Přihlášení k facebookovému účtu skrze modul 2 (vlastní zpracování)

Protože komunikace, přihlášení a získání dat trvá nezanedbatelný čas, i několik málo sekund, nebylo by rozumné ničím uživatele neinformovat. Proto se okamžitě po zavolání metody *onSuccess()* vytvoří objekt progressDialog, což je opět komponenta patřící do Graph API a která informuje uživatele o pokračování úlohy. Nyní přichází na řadu vytvoření requestu, neboli požadavku. Právě zasílání požadavku na Graph API je stěžejní funkcionalita, o které je pojednáno níže u získávání samotných dat z Facebooku. Požadavek vrací odpověď, která je ve většině případech, stejně jako v tomto případě ve formátu JSON. Co je formát JSON je vysvětleno v kapitole o principu fungování Graph API.

Po úspěšném vyřízení požadavku se automaticky volá metoda onComplete(), která jako parametr přijímá onu odpověď – response ve formátu JSON. Uvnitř metody se jako první provede uzavření nyní již zbytného informačního progressDialogu a z nastavené URL adresy se získá profilový obrázek a následně jej uloží do připravené datové struktury ImageView. Objekt Bundle specifikuje, po čem se má dotaz ptát. V tomto případě je to id facebookového profilu a email. Jako poslední se specifikuje, že se má request provést asynchronně.

6.3 Získávání statusů z vybraného facebookového účtu

V předchozí části byl popsán způsob implementace přihlášení a získání profilového obrázku přihlašovaného uživatele. V této části bude vysvětlen postup získání statusů z vybraného facebookového účtu tak, jak jej doporučuje oficiální dokumentace Facebook Graph API. Modul sociálních sítí má získávat z vybraných (univerzitních) facebookových účtů primárně události. Ale i nové statusy by mohli být uživateli přínosné už z hlediska užitečných informací. Aby však tento způsob fungoval, musí být uživatel přes aplikaci přihlášen k Facebooku. A to tak, jak je to popsáno v části výše. K odeslání nového požadavku je nutné mít přidělený AccessToken, což je řetězec, který se získá při přihlášení a který je nutné zadat jako parametr nového požadavku. Další nutnou podmínkou pro úspěšné získání statusů z vybraného účtu je mít povolení „Page Public Content Access“ a „user-posts“, o kterých je pojednáno v části Jak začít s Graph API. K získání těchto povolení je

ovšem nutné mít i „Buisness verification“, která již byla také zmíněná v části Jak začít s Graph API. Získání statusů je zobrazeno na ukázce č. 4.

```
private void getFacebookData(String id){
   Bundle params = new Bundle();
   params.putString("fields", "message");
   params.putString("limit", "10");
   GraphRequest request = new GraphRequest(AccessToken.getCurrentAccessToken(),
                                    "/ "+ id + "/posts", params, HttpMethod.GET,
   new GraphRequest.Callback() {
      @Override
      public void onCompleted(GraphResponse response) {
         try {
            JSONObject jsonResponse = new JSONObject(response.getJSONObject().toString());
            Log.i("Response", jsonResponse.toString());
         }
         catch (Exception e) {
            e.printStackTrace();
         }
      }
   });
   request.setParameters(params);
   request.executeAsync();
}
```

Ukázka 4: Získání statusů z vybraného facebookového účtu (vlastní zpracování)

Na ukázce č. 4 je metoda *getFacebookData(),* která jako parametr získává řetězec znaků reprezentující id účtu, ze kterého má aplikace získat data (statusy). Takové unikátní ID má každá facebooková stránka a dá se vyčíst například ze zdrojového kódu stránky jako „page-id“. Podobně tak má i každý uživatel své unikátní „user-id“, které lze získat například pomocí několika webových stránek po zadání URL adresy facebookového účtu do textového pole. Nejen stránka či uživatel, ale i událost má své unikátní ID. O událostech ale bude pojednáno dále. Nyní k samotnému kódu.

Po deklaraci hlavičky metody se vytvoří objekt Bundle. Ten specifikuje parametry, které má aplikace žádat a popřípadě i jejich počet, který je v ukázce nastaven klíčovým slovem „limit“ s hodnotou 10. V tomto případě se tedy jedná o posledních deset statusů. Následně se vytvoří samotný request podobný tomu jako v ukázce kódu pro přihlašovaní. Rozdíl je ale v jeho parametrech. Jako první parametr se sice opět dosadí požadovaný přístupový token, který je již od přihlášení uložen v objektu AccessToken a získán pomocí jeho metody *getCurrentAccessToken().* Dále je v parametru uvedeno ID uživatele, od kterého má request získat data. Jak jej získat je již popsáno výše. Část */posts* na konci řetězce značí, že se mají získat jen statusy či události, které zvolený uživatel sám zveřejnil.

Existuje i možnost */tagged*, která získá jen statusy, ve kterých je uživatel pouze označen jiným uživatelem. A jako poslední parametr jsou ony specifikující parametry deklarované na začátku metody. Přepsaná metoda *onCompleted()* se automaticky zavolá po úspěšném vytvoření dotazu a opět získává response-odpověď. Pro odpověď používá Graph API formát GraphResponse. Odpověď v tomto formátu je pro jednodušší zpracování následně převedena do JSON. Metoda *Log.i()* slouží v Android Studiu jen k vypsání odpovědi do konzole. Jelikož tato metoda umí vypisovat do konzole jen text, je nutné zavolat metodu *toString()*, která převede objekt JSON na řetězec znaků. Stěžejní je v této ukázce proces vytváření requestu, což je hlavní oficiální možnost jak získávat různá data z facebookového profilu nebo stránky. V další části bude popsán postup získání událostí z daného facebookového profilu, který má být v modulu sociálních sítí stěžejní funkcionalitou a který funguje opět na principu zasílání requestu.

6.4 Získávání událostí z vybraného facebookového účtu

Zrcadlení událostí z facebookových stránek univerzity je jeden z klíčových funkcionalit modulu sociálních sítí. Získání událostí je ale skrze Graph API řešeno velice podobně jako získání statusů. Lišit se bude jednak v parametrech požadavku, tak i v potřebných povoleních. Ukázka zdrojového kódu včetně samotného požadavku je k dispozici na ukázce č. 5. Co se povolení týče, k získání informací o událostech není na rozdíl od statusů potřeba „Page Public Content Access“ ani Business verifikaci. K získání událostí stačí jen povolení „user-event“ a individuální verifikaci.

```
public void getEvents(String id){
   GraphRequest request = new GraphPathRequest(AccessToken.getCurrentAccessToken(),"/"+id,
   new GraphRequest.Callback() {
      @Override
      public void onCompleted(GraphResponse response) {
         try {
            JSONObject jsonResponse = new JSONObject(response.getJSONObject().toString());
            Log.i("Response", jsonResponse.toString());
         }
         catch (Exception e) {
            e.printStackTrace();
         }
      }
   });
   request.executeAsync();
}
```

Ukázka 5: Získání konkrétní události z Facebooku (16)

Již na první pohled je zřejmé, že se kód na ukázce č. 5 od předchozí ukázky č. 4 příliš neliší. Jen se zde místo ID facebookové stránky zadává přímo ID události. Toto řešení je ovšem pro potřeby modulu nevyhovující, jelikož nutně vyžaduje znalost ID konkrétní události. I když zjištění onoho ID události je podobně složité jako zjištění ID uživatele, Graph API nabízí modifikované řešení, které bude pro modul sociálních sítí vhodnější. Dojde jen ke změně parametrů požadavku. Oba případy jsou názorně ukázány v oficiální dokumentaci ke Graph API. Druhá možnost získání událostí je na ukázce č. 6.

```
public void getEvents(String id){
   GraphRequest request =  new GraphRequest(AccessToken.getCurrentAccessToken(),
                                            "/" + id + "/events",null,HttpMethod.GET,
   new GraphRequest.Callback() {
      public void onCompleted(GraphResponse response) {
         try {
            JSONObject jsonResponse = new JSONObject(response.getJSONObject().toString());
            Log.i("Response", jsonResponse.toString());
         }
         catch (Exception e) {
            e.printStackTrace();
         }
      }
   });
   request.executeAsync();
}
```

Ukázka 6: Získání události ze zadaného facebookového účtu (17)

Kód se tedy liší jen tím, že tentokrát se za ID, které již neznačí ID události nýbrž ID facebookové stránky, ze které se má událost získat, přidá se přípisek „/events“ a specifikuje se, že se má použít metoda GET a nikoliv metoda POST. Tato forma již pro použití v modulu sociálních sítí vyhovuje.

6.5 Získávání dat z vybraného Twitter účtu

Modul sociálních sítí má za cíl získávat nejnovější příspěvky nejen z vybraných účtů Facebooku, ale i z Twitteru. Jednotlivé fakulty Univerzity Hradec Králové mají své Twitter účty, kde se také mohou objevit relevantní informace. Proto je pokládáno za nutné zde popsat implementaci jejich získávání a zobrazení v mobilní aplikaci. Pro potřeby modulu sociálních sítí byla použita neoficiální knihovna twitter4J (5), která komunikuje s Twitter API skrze autentizační metodu OAuth, která je popsána v kódu dále. Ještě před začátkem psaní zdrojového kódu je bezpodmínečně nutné registrovat budoucí aplikaci na oficiálních webových stránkách Twitteru pro vývojáře. Po úspěšné registraci lze vygenerovat klíče a přístupové tokeny nutné pro vytvoření komunikace. Současně je nutné v nastavení aplikace po registraci udělit aplikaci povolení alespoň pro čtení. Přístupový klíč, tajný klíč a oba tokeny budou využity při OAuth autentizaci.

```
private void findTweets(){
    ConfigurationBuilder cb = new ConfigurationBuilder();

    cb.setDebugEnabled(true);
    cb.setOAuthConsumerKey("ACCESSKEY");
    cb.setOAuthConsumerSecret("SECRETKEY");
    cb.setOAuthAccessToken("ACCESSTOKEN");
    cb.setOAuthAccessTokenSecret("SECRETTOKEN");

    TwitterFactory tf = new TwitterFactory(cb.build());
    Twitter = tf.getInstance();
    String fim = "@fimuhk";

    List<Status> statuses = twitter.getUserTimeline(fim);
    Status newest = statuses.get(0);
    TextView twPlace1 = (TextView)findViewById(R.id.twPlace1);
    twPlace1.setText(newest.getUser().getName() + ":    " +  newest.getText());

}
```

Ukázka 7: Získání nejnovějšího tweetu ze zadaného účtu (vlastní zpracování)

Zdrojový kód v ukázce č.7 popisuje právě situaci získání nejnovějších tweetů z daného účtu. Tento kód tvoří metoda *findTweets()*, která zařizuje jak navázání komunikace, tak i získání dat. V metodě je nejprve vytvořena instance třídy ConfigurationBuilder. Což je třída, která slouží k základní konfiguraci připojení, a nad kterou je provedena OAuth autorizace pomocí správně zadaných přístupových tokenů. Dále je vytvořena instance třídy TwitterFactory, což je jakási tovární třída, která přejímá jako parametr konfigurační nastavení z třídy ConfigurationBuilder a vytváří samotné propojení s Twitterem. Když je propojení

uskutečněno, metoda *getUserTimeLine()*, která přijímá za parametr jméno twitterového účtu vrátí posledních 20 tweetů daného uživatele, proto je návratová hodnota uložena do datové struktury List<Status>, která se naplní 20 posledními statusy. Pro potřeby modulu sociálních sítí stačí zobrazit jen ten nejnovější tweet, proto se dále bude pracovat jen se Statusem *newest,* do kterého je uložen první prvek z listu – nejnovější tweet. Ten se již známým způsobem uloží do připraveného TextView jako řetězec znaků.

Je nutné zde zmínit i alternativní možnost implementace této části. Další možností je vytvoření a zaslání požadavku na URL adresu poskytnutou na oficiálních stránkách Twitteru pro vývojáře, která musí obsahovat parametr specifikující zdrojového uživatele, což lze zařídit parametry *screen_name*, nebo *user_id*. Do této zdrojové URL adresy je možné přidat další nepovinné parametry, například parametr nastavující počet zaslaných tweetů. Odpověď je poskytnuta ve formátu JSON. Není tak v tomto případě nutné použití knihovny twitter4j, ale autorizace za pomocí přístupových klíčů a tokenů je stále vyžadována.

Ukládání tweetů to obyčejného TextView je nejen uživatelsky nepřívětivé, ale i velice nepraktické. Při navýšení počtu tweetů by se musel manuálně vytvořit další TextView, což vyžaduje zásah i do layoutu aktivity. Komponenta TextView byla použitelná v prvotních fázích testování. V průběhu implementace modulu však bylo nutno využít vhodnějších komponent. Nabízí se několik možností z nichž byla vybrána komponenta ListView pro svou jednoduchost, přehlednost a možnost snadné implementace funkcionality, která zajišťuje otevření twitterového učtu dle vybraného tweetu. Tato funkcionalita bude popsána dále. V ukázce č. 8 je zobrazena konfigurace komponenty ListView.

```
private ListView = (ListView)findViewById(R.id.listView);
private ArrayList<String> arrayList = new ArrayList<>();

/* Získání tweetů */

arrayList.add(newest.getUser().getName() + ":    " + newest.getText());

ArrayAdapter adapter;
adapter = new ArrayAdapter(this,android.R.layout.simple_list_item_1,arrayList);

listView.setAdapter(adapter);
listView.setClickable(true);
```

Ukázka 8: Použití komponenty ListView (vlastní zpracování)

Použití komponenty ListView je vcelku velice snadné. Celý princip spočívá v použití arrayListu, do kterého se ukládají jednotlivé tweety. Z něj se poté s pomocí třídy ArrayAdapter stává ListView. Jako poslední je nad objektem listView zavolána metoda *setClickable()* s parametrem *true*, která umožňuje právě onu funkcionalitu, která byla zmíněna výše. Po kliknutí na vybraný tweet v komponentě ListView se otevře prohlížeč, či aplikace (závisí na volbě uživatele) a zobrazí twitterový účet, ze kterého byl tweet publikován. Zdrojový kód, který zobrazuje tuto situaci je popsán v ukázce č. 9.

```
listView.setOnItemClickListener(new AdapterView.OnItemClickListener() {

   public void onItemClick(AdapterView<?> parent, View view,int position, long id) {
      String clicked = arrayList.get(position).substring(0,3);
        if(clicked.equals("FIM")){
           String url = "https://twitter.com/fimuhk";
           Intent i = new Intent(Intent.ACTION_VIEW);
           i.setData(Uri.parse(url));
           startActivity(i);
        }
        if(clicked.equals("PdF")){
           String url = "https://twitter.com/PdF_UHK";
           Intent i = new Intent(Intent.ACTION_VIEW);
           i.setData(Uri.parse(url));
           startActivity(i);
        }
   }
});
```

Ukázka 9: Otevření twitterového účtu dle vybraného tweetu (vlastní zpracování)

Jedná se o použití interface OnItemClickListener, který reaguje na kliknutí myší na jakýkoliv prvek ListView. Dle prvních tří písmen názvu účtu modul pozná jakou stránku má otevřít. I přes to, že by se obecně nejednalo o příliš bezpečné řešení, pro potřeby modulu plně postačuje, jelikož sledovaných univerzitních účtů je omezené množství a nepředpokládá se změna názvů jednotlivých fakult či názvů jejich twitterových účtů.

6.6 Implementace výběru zasílání dat

Jak bylo řečeno v návrhu modulu, uživatel by měl mít možnost výběru z jakých zdrojů chce získávat novinky. V následující ukázce kódu je tento problém řešen pomocí komponenty Switch, kterou nabízí Android SDK a jehož stav se ukládá do sdílených uživatelských preferencí. Data jsou z daného Twitterového účtu získána a zobrazena jen v případě, pokud je to u daného účtu povoleno.

```
public Switch twFim = (Switch) findViewById(R.id.twFim);
public SharedPreferences settings = getSharedPreferences("myPrefs", 0);
public boolean onOff = settings.getBoolean("switchkey", false);

   twFim.setChecked(onOff);
   twFim.setOnCheckedChangeListener(new CompoundButton.OnCheckedChangeListener() {

      @Override
      public void onCheckedChanged(CompoundButton buttonView, boolean isChecked) {
        SharedPreferences settings = getSharedPreferences("myPrefs", 0);
        SharedPreferences.Editor editor = settings.edit();
        editor.putBoolean("switchkey", isChecked);
        editor.commit();
      }
   });
```

Ukázka 10: Implementace výběru zasílání dat (vlastní zpracování)

Nyní k vysvětlení výše uvedené ukázky č. 10. Po deklaraci a vytvoření komponenty Switch je vytvořena proměnná *settings,* což je sdílená preference, která umožňuje získávat a ukládat stav komponenty Switch i po uzavření aplikace. Boolean proměnná *onOff* je naplněna hodnotou uloženou v *settings* a podle které je následně nastaven i switch. Dále je vytvořen OnCheckedChangeListener, který po každé změně stavu switche aktualizuje jeho hodnotu ve sdílených preferencích a uloží jej. Následně stačí před samotným získáváním dat v jednotlivých třídách vytvořit podmínku, která říká, že se data získají a zobrazí jen pokud je patřičný switch nastavený na hodnotu *true*, neboli je získávání dat z tohoto kanálu povoleno. Tato situace je zobrazena v ukázce č. 11.

```
SharedPreferences mPrefs = getSharedPreferences("myPrefs",0);
boolean str = mPrefs.getBoolean("switchkey", false);
   if (str) {
      List<Status> statuses = twitter.getUserTimeline(fim);
      Status newest = statuses.get(0);
      TextView twPlace1 = (TextView)findViewById(R.id.twPlace1);
      twPlace1.setText(newest.getUser().getName() + ":    " + newest.getText());
   }
```

Ukázka 11: Implementace podmínky k získání statusů (vlastní zpracování)

Tato ukázka zdrojového kódu je známá část zdrojového kódu části o získávání tweetů z Twitteru. Ovšem opatřena onou podmínkou, která dovolí získání dat jen pokud je hodnota proměnné *str* získaná ze sdílených preferencí nastavená na *true*, což značí, že daný switch je zapnut neboli nastaven na stav On.

6.7 Ukládání získaných dat

Jelikož mobilní telefon ve většině případů není připojen k internetu vždy, je nutné získaná data udržet dostupná i v tomto případě. Po opětovném připojení k síti a stažení nových dat se uložené informace aktualizují. Jeden druh ukládání informací již byl v modulu použit a je popsán i v této práci v části o implementaci výběru zasílání dat. Jedná se o sdílené preference, které umožňují pohodlně a rychle ukládat zejména primitivní datové typy. Další možností je ukládání na interní úložiště telefonu. Tato možnost lze použít pro ukládání téměř čehokoliv do vytvářeného souboru. Nutné je ale brát v potaz výjimku, která nastane pokud v mobilním telefonu již nebude volná paměť při vytváření souboru, či výjimku pokud se soubor nepodaří nalézt při čtení. Ukládání do externího úložiště je poslední možnost a probíhalo by velice podobně, ale pro potřeby modulu je tato možnost nevyhovující, jelikož externí úložiště nemusí být vždy k dispozici. Z těchto tří možností byla vybrána možnost ukládání s pomocí sdílených preferencí a to zejména díky jednoduchosti a absence nutnosti řešení problémy s vytvářením, nebo čtením ze souboru. Celý proces bude probíhat stejně jako bez této funkcionality. Uživatel načte nové tweety, které se mu zobrazí do připravené ListView komponenty. Pokud vše půjde takto bez chyby, tweety se zároveň bez vědomí uživatele uloží i do sdílených preferencí. Pokud ale nebude připojení k síti k dispozici, tweety se nezískají z Twitteru, nýbrž z těchto sdílených preferencí. Po každém získání nových dat z Twitteru se tak zároveň automaticky aktualizují i informace ve sdílených preferencích.

```
SharedPreferences tweets = getSharedPreferences("tweets", MODE_PRIVATE);
SharedPreferences.Editor editor = tweets.edit();

Gson = new Gson();
String json = gson.toJson(backList);
editor.putString("list", json);
editor.apply();
```

Ukázka 12: Způsob uložení ArrayListu do sdílených preferencí (vlastní zpracování)

Ukázka kódu č. 12 obsahuje uložení ArrayListu do sdílených preferencí, který obsahuje získané tweety z vybraných kanálů sociální sítě Twitter. Nejdříve je nutné vytvořit instance samotných sdílených preferencí , kde se opět do parametrů uvede název preferencí a jejich mód. Ten značí pro koho bude tato sdílená preference viditelná. V ukázce kódu je nastavena jako privátní. Tudíž jiná třída než ta, ve které je sdílená preference vytvořena ji neuvidí. Následně je vytvořen objekt Gson. Gson, neboli také Google Gson je Java knihovna, která se používá k přesunu a serializaci dat z formátu JSON. Odkaz na tuto knihovnu je nutno zadat do souboru AndroidManifest. Skrze tuto třídu, respektive její metodu *toJson()* je možné uložit ArrayList jako obyčejný řetězec textu. Tento řetězec, ač není brán jako primitivní datový typ, lze uložit do sdílených preferencí.

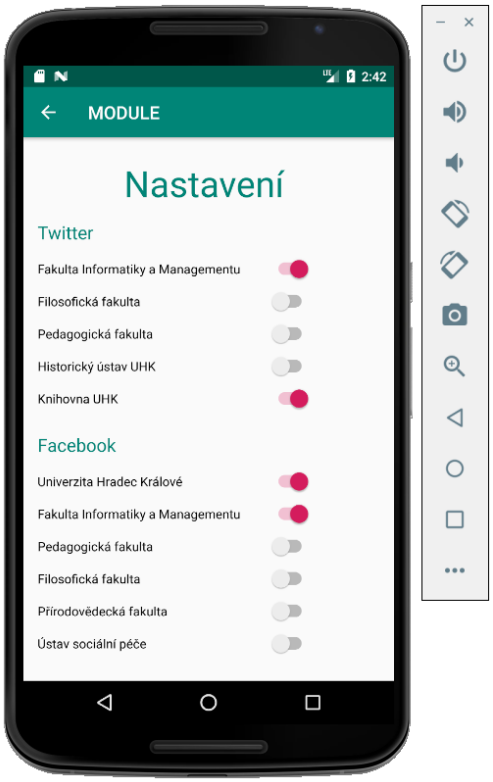
```
private void getBackup(){
    SharedPreferences tweets = getSharedPreferences("tweets", MODE_PRIVATE);
    Gson = new Gson();
    String json = tweets.getString("list", null);
    Type = new TypeToken<ArrayList<String>>(){ }.getType();
    backList = gson.fromJson(json, type);

    ArrayAdapter adapter;
    adapter = new ArrayAdapter(this,android.R.layout.simple_list_item_1,backList);

    listView.setAdapter(adapter);
    listView.setClickable(true);
}
```

Ukázka 13: Opětovné získání ArrayListu ze sdílených preferencí (vlastní zpracování)

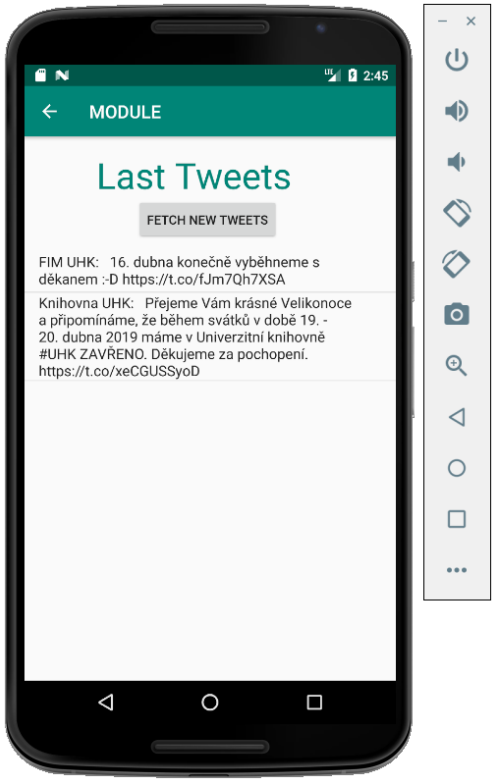
Nyní k ukázce kódu č. 13, na které je ukázán příklad opětovného získání dat ze sdílených preferencí a tyto data jsou následně vložena do komponenty ListView. Celý postup je v podstatě reverzní k minulé ukázce zdrojového kódu. Jen při zpětném převádění řetězce na ArrayList je nutné říci, že řetězec má být přeložen do datové struktury ArrayList<String> a to se provede právě díky třídě Type. Nyní stačí do ArrayAdapteru použitého pro konstrukci ListView zadat místo původního ArrayListu, jehož obsah je získáván přímo ze sociální sítě, ArrayList, který se naplní uloženými daty ze sdílených preferencí.

7 Shrnutí výsledků

Mobilní aplikace UHK Helper, pro kterou je modul sociálních sítí určen je již mezi studenty poměrně rozšířena a aktivně využívána. UHK Helper je ovšem stále ve vývoji a stále pro něj vznikaní nové moduly a vylepšení. Takovým je i modul sociálních sítí popsaný v této bakalářské práci. Modul byl analyzován, navrhnut a implementován jako samostatná nezávislá aplikace, která bude po fázi testování připojena do aplikace UHK Helper. Ve fázi analýzy byl vytvořen koncept, jak by měl modul vypadat, co by měl uživatelům nabízet a jak by se měl chovat. Ve fázi návrhu bylo dohodnuto, jak bude modul vypadat, jakých komponent bude využívat a jak bude získaná data prezentovat. Ve fázi implementace byla vytvořena první verze funkční nezávislé aplikace – modul sociálních sítí, který bude později spojen s UHK Helperem. První verze modulu sociálních sítí je již schopna zajišťovat nejdůležitější případy užití, jako je například získání dat z vybraného facebookového účtu, přihlášení k facebookovému účtu skrze aplikaci, což je nutný krok k získání dat z Facebooku. První verze modulu sociálních sítí umí též získat nejnovější tweety z Twitteru, respektive z vybraných twitterových účtů a měnit uživatelské nastavení filtru z jakých účtů sociálních sítí chce uživatel získávat nové tweety či statusy. Univerzitní twitterové a facebookové účty jsou předem vybrány a uživatel má jen pravomoc zapínat, nebo vypínat zasílání dat. Data jsou v první verzi modulu zobrazována jako řetězce textu v komponentě ListView, či v případě obrázků jako obrázek v ImageView. Po spojení s aplikací UHK Helper se předpokládá použití zasílání anotací.



Obr. 9: Nastavení zasílacího filtru (vlastní zpracování)



Obr. 10: Vyhledání nových tweetů z vybraných účtů (vlastní zpracování)

Na obrázku č. 9 je znázorněno samotné nastavení zasílání novinek z Twitteru a z Facebooku. Jak již bylo zmíněno, jedná se o komponentu Switch, která využívá ukládání hodnot pomocí sdílených preferencí. Obrázek č. 10 představuje výběr nejnovějších tweetů z nastavených univerzitních Twitter účtů. Tweety se zobrazují pomocí komponenty ListView, která umožňuje nenáročné a pro potřeby modulu vhodné řešení.

8 Závěr a doporučení

Cílem této bakalářské práce bylo analyzovat, navrhnout a implementovat modul sociálních sítí, který by umožnil komunikovat aplikaci UHK Helper s vybranými kanály nejpoužívanějších sociálních sítí. V úvodu práce byla nastíněna dnešní situace ohledně chytrých telefonů a sociálních sítí. V analýze modulu byly pomocí Use Case diagramu popsány případy užití modulu, což jsou zejména: Získávání novinek a událostí z univerzitních účtů sociálních sítí, nebo nastavení zasílacího filtru. V návrhu modulu byl vytvořen layout hlavní stránky a nastavení, ukázány použité komponenty pro zobrazení dat, jako například TextView, ListView, či ImageView, komponenty Switch a vysvětlen princip fungování modulu. V další části byla popsána samotná implementace vybraných částí modulu. Například přihlášení k Facebooku skrze modul, nebo získávání dat z vybraných sociálních sítí.

Cíl práce byl tedy splněn. Po analýze a návrhu byla implementována první verze modulu sociálních sítí, jakožto samostatné a nezávislé aplikace, která bude po fázi testování a mírných, zejména grafických úpravách spojena s aplikací UHK Helper.

Do budoucna se předpokládá další vývoj modulu i samotné aplikace UHK Helper. V případě modulu sociálních sítí se může jednat například o implementaci dalších možností zpětné vazby na zaslaná data, nebo jejich vhodnější zobrazení.

9 Citovaná literatura

1. **NetApplications.com.** Operating System Market Share. *Net MarketShare.* [Online] NetApplications.com, 2017. [Citace: 8. Srpen 2019.] https://netmarketshare.com/operating-system-market-share.aspx?id=platformsMobile.

2. **Barnett, George A.** *Encyclopedia of Social Networks.* Thousand Oaks : SAGE Publications, Inc., 2011. 978-1-4129-7911-5.

3. **Statista.** Most popular social networks worldwide as of July 2019, ranked by number of active users. *Statista.com.* [Online] Statista , 2019. [Citace: 11. Srpen 2019.] https://www.statista.com/statistics/272014/global-social-networks-ranked-by-number-of-users/.

4. **Černý, Michal.** O Aplikaci. *UHK Helper.* [Online] 2018. [Citace: 12. Leden 2019.] http://lide.uhk.cz/fim/student/cernymi4/uhk_helper/index.html.

5. **Jacobson, Daniel, Woods, Dan a Brail, Greg.** *APIs: A Strategy Guide.* Sebastopol : O'Reilly Media, 2011. 978-1449-30892-6.

6. **Facebook © 2018.** Facebook Platform Policy. *Facebook for developers.* [Online] Facebook, 7. Květen 2018. [Citace: 25. 12 2018.] https://developers.facebook.com/policy/.

7. **Fielding, Roy Thomas.** CHAPTER 5 Representational State Transfer (REST). *Donald Bren School of Information and Computer Sciences.* [Online] UCI Donald Bren School of Information & Computer Sciences, 2000. [Citace: 11. 8 2019.] https://www.ics.uci.edu/~fielding/pubs/dissertation/rest_arch_style.htm.

8. **Stay, Jesse.** *FBML Essentials.* Sebastopol : O'Reilly Media, 2008. 978-0-596-51918-6.

9.**twitter4j.** *twitter4j.* [Online] Apache, 2007. [Citace: 12. 3 2019.] http://twitter4j.org/en/.

10. Bruckner, Tomáš, a další. *Tvorba informačních systémů.* Praha : Grada Publishing, a.s., 2012. 978-80-247-4153-6.

11. The Java Language Environment. *Oracle.com.* [Online] 1. 1 1999. [Citace: 3. Leden 2019.] https://www.oracle.com/technetwork/java/intro-141325.html.

12. Meet Android Studio. *Android Developers.* [Online] 8. Listopad 2018. [Citace: 5. Leden 2019.] https://developer.android.com/studio/intro/.

13. Android zvýšil podíl na trhu chytrých telefonů na 80 %. *MAM.cz.* [Online] Economia, a.s., 8. Srpen 2013. [Citace: 24. 11 2018.] https://mam.cz/c1-60392420-android-zvysil-podil-na-trhu-chytrych-telefonu-na-80.

14. Callaham, John. The history of Android OS: its name, origin and more. *Android Authority.* [Online] 28. Červen 2019. [Citace: 11. Srpen 2019.] https://www.androidauthority.com/history-android-os-name-789433/.

15. Facebook Login for Android - Quickstart. *Facebook for developers.* [Online] Facebook, 2019. [Citace: 5. Srpen 2019.] https://developers.facebook.com/docs/facebook-login/android.

16. Event. *Facebook for develoepers.* [Online] Facebook, 2019. [Citace: 5. Srpen 2019.] https://developers.facebook.com/docs/graph-api/reference/event/.

17. Page Events. *Facebook for developers.* [Online] Facebook, 2019. [Citace: 5. Srpen 2019.] https://developers.facebook.com/docs/graph-api/reference/page/events/.

Oskenované zadání práce

Univerzita Hradec Králové
Fakulta informatiky a managementu
Akademický rok: 2017/2018

Studijní program: Aplikovaná informatika
Forma: Prezenční
Obor/komb.: Aplikovaná informatika (ai3-p)

Podklad pro zadání BAKALÁŘSKÉ práce studenta

PŘEDKLÁDÁ:	ADRESA	OSOBNÍ ČÍSLO
Pohořský Tomáš	Masarykova 74, Humpolec	I1600593

TÉMA ČESKY:

Modul sociálních sítí v systému podpory výuky

TÉMA ANGLICKY:

Social Networks Module for System of Education Support

VEDOUCÍ PRÁCE:

doc. Mgr. Tomáš Kozel, Ph.D. - KIKM

ZÁSADY PRO VYPRACOVÁNÍ:

\par{Analyzovat, navrhnout a implementovat modul pro komunikaci aplikace UHK Helper s vybranými kanály sociálních sítí.

Osnova:

Úvod
Programová rozhraní vybraných sociálních sítí
Analýza a návrh modulu
Výběr a stručný popis technologií
Popis vybraných aspektů implementace
Výsledky
Závěr

\par}

SEZNAM DOPORUČENÉ LITERATURY:

\par{bude upřesněno\par}

Podpis studenta: ..... **Datum:** .....

Podpis vedoucího práce: ..... **Datum:** .....