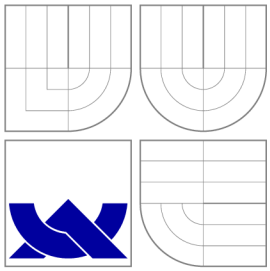
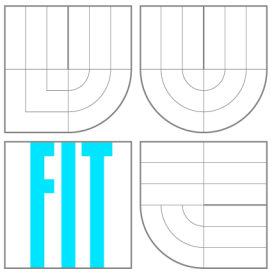


VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

# AGREGACE A DOPORUČOVÁNÍ UDÁLOSTÍ A MÍST Z FACEBOOKU

EVENTS AND PLACES AGREGATION AND SUGGESTIONS FROM FACEBOOK

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE
AUTHOR

MATEJ DUBEŇ

VEDOUCÍ PRÁCE
SUPERVISOR

Ing. IGOR SZŐKE, Ph.D.

BRNO 2015

# Abstrakt

Cílem této práce je přiblížit návrh a implementaci Android aplikace Let's Go Out, která dokáže uživateli doporučit události a místa ze sociální sítě Facebook. Doporučení je vykonáváno za pomoci hybridního doporučujícího systému, který spojuje princip kolaborativního filtrování a obsahového doporučení, sleduje uživatelovi interakce s aplikací a na základě zaznamenaných dat přizpůsobuje proces doporučení. Práce taky pojednává o testování aplikace pomocí porovnávání s doporučujícími systémy konkurenčních aplikací a dosažených výsledcích.



# Abstract

The aim of this bachelor thesis is to explain the design and implementation of an Android applications Let's Go Out, which can recommend Facebook events and places to the user. The recommendation is carried out by using the hybrid recommending system approach that links together the collaborative filtering and a content-based recommendation approach, tracks the user's interaction with the application and, based on recorded data, adapts to the recommendation process. This thesis also describes the testing process that compares the recommender systems of competitive applications and points out achievements.



# Klíčová slova

Kolaborativní filtrování, Obsahové doporučení, Agregace událostí, Agregace míst, Android, Kosinová míra podobnosti, TF-IDF

# Keywords

Collaborative filtering, Content-based recommendation, Events aggregation, Places aggregation, Android, Cosine similarity Measure, TF-IDF


# Citace

Matej Dubeň: Agregace a doporučování událostí a míst z Facebooku, bakalářská práce, Brno, FIT VUT v Brně, 2015

# Agregace a doporučování událostí a míst z Facebooku

## Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana Ing. Igora Szőkeho, Ph.D.

.....
Matej Dubeň
18. května 2015

## Poděkování

Ďakujem svojmu vedúcemu bakalárskej práce, Ing. Igorovi Szőkemu, Ph.D., za cenné rady, pripomienky a usmerňovanie počas celého procesu tvorby práce. Taktiež by som rád poďakoval pánovi Jaromírovi Kotalovi, ktorý mi poskytol všetky možné prostriedky pre dokončenie tejto práce, Renému Klačanovi, ktorý ma pri tejto práci inšpiroval a mamine, sestre, Lucke, Michalovi a Jakubovi za ich podporu a pomoc.

# Obsah

# Kapitola 1

# Úvod

Smartfóny sa stali neoddeliteľnou súčasťou každodenného života, a to už každej generácie. Vzhľadom na rozsiahlu škálu funkcií, ktoré tieto telefóny ponúkajú, sú na bočnú koľaj odsúvané, kedysi dennodenne používané, pomôcky ako diáre, kalendáre, poznámkové bloky, audio rekordéry, atď. Podnikatelia a firmy sa teda logicky uchyľujú ku oslovovaniu a získavaniu nových zákazníkov prostredníctvom aplikácií pre smartfóny.

Obdobne celosvetovo populárnou je aj sociálna sieť Facebook. Milióny ľudí na svete medzi sebou komunikujú prostredníctvom tejto siete. Firmy a podniky taktiež využívajú túto sieť pre propagáciu svojich služieb a udalostí, ktoré sa konajú pod ich záštitou. Užívatelia, majú možnosť odoberať novinky zo svojich obľúbených miest a dostávajú sa im tak pravidelné informácie o udalostiach a akciách od daného usporiadateľa.

Avšak nie každý užívateľ má vytvorený účet na sociálnej sieti Facebook, alebo je prihlásený ku odoberaniu noviniek. Táto práca sa zaoberá vývojom aplikácie, ktorá prináša informácie k užívateľovi bez ohľadu na to, či má Facebook účet, alebo či je prihlásený k odberu noviniek. Údaje sú zbierané na základe užívateľovej polohy, prípadne ním zvolenej oblasti a sérií parametrov, ktoré si užívateľ sám zvolí. Vyhodnocovanie zozbieraných údajov následne určí ich relevantnosť voči užívateľovi na základe výsledkov sledovania jeho dlhodobého chovania a používania aplikácie. Extrahované dáta sú vizualizované užívateľovi v užívateľsky priateľskej forme. Dáva tým užívateľovi prehľad o mieste a čase konania udalostí, prípadne o polohe podnikov, ktoré boli na základe parametrov vyhľadané alebo mu boli odporučené aplikáciou.

# Kapitola 2

# Návrh aplikácie

Cieľom tejto práce je vytvoriť aplikáciu pre mobilné telefóny so systémom Android, ktorá bude schopná vyhľadávať, agregovať a odporúčať udalosti a miesta užívateľovi. Odporúčanie by malo byť prispôsobené užívateľovi a reflektovať jeho históriu používania aplikácie do odporúčacieho procesu. Hlavnou časťou práce je odporúčací systém, ktorý sa nachádza na strane serveru. Server má na starosti prijímanie a vybavovanie požiadaviek od užívateľov, získavanie údajov zo sociálnej siete Facebook, počítanie odporúčacích algoritmov a samotné odporúčanie. Proces vybavovania požiadaviek by mal prebiehať v reálnom čase, vzhľadom na to, že sa jedná o mobilnú aplikáciu, kde užívateľ pri interakcií očakáva spätnú väzbu. Po ukončení implementácie je aplikácia zverejnená širokej verejnosti, testovaná na skupine užívateľov a upravovaná na základe spätnej väzby od užívateľov.

V tejto kapitole sú priblížené aspekty návrhu aplikácie, počnúc prieskumom existujúcich aplikácií, návrhom grafického rozhrania a nakoniec samotného odporúčacieho systému.

## 2.1 Prieskum podobných aplikácií

Obchod s aplikáciami pre Android zariadenia (Google Play) obsahuje množstvo aplikácií, ktoré sa venujú problematike zbierania údajov zo sociálnej siete Facebook. Na trhu je teda pomerne veľa konkurenčných aplikácií. Využívajú najmä údaje z Facebook profilu užívateľa, čo znamená, že užívateľ je nútený sa do aplikácie prihlásiť pomocou svojho Facebook účtu aby mu boli pripomenuté alebo odporučené udalosti z jeho obľúbených stránok. Využívaním údajov z profilu užívateľa je možné docieliť vysokú správnosť odporučenia, vzhľadom na to, že užívateľovi sú predkladané dáta, ktoré korešpondujú s jeho obľúbenými stránkami a aktivitami. Avšak užívateľove používanie aplikácie nemá žiadny vplyv na presnosť odporúčania a odporučených položiek.

Tu ale nastáva problém v prípade, že užívateľ Facebook účet nemá. Na neho sa potom odporúčania nevzťahujú.

Nasledujúca tabuľka demonštruje výhody a nevýhody existujúcich riešení voči aplikácií *Let's Go Out!*:

V tabuľke 2.1 je vidieť oblasti, v ktorých aplikácia *Let's Go Out!* vyniká oproti ostatným aplikáciám, rovnako ako aj oblasti, v ktorých oproti ostatným aplikáciám zaostáva. Vzhľadom na silu konkurencie musí teda byť riešenie správne navrhnuté a vysoko optimalizované aby mala aplikácia šancu na trhu.

Tabulka 2.1: Výhody a nevýhody existujúcich riešení oproti aplikácií *Let's Go Out!*

| *Aplikácia* | *Výhody* | *Nevýhody* |
|---|---|---|
| **All Events in City** | Vlastný webový portál<br>Široká komunita<br>Interný zdroj udalostí | Necielené odporúčanie<br>Agregácia podľa popularity<br>Neodporúča miesta<br>Chýba vizualizácia na mape |
| **Nearify** | Vlastný webový portál | Necielené odporúčanie<br>Agregácia podľa popularity<br>Chýba vizualizácia na mape<br>Neodporúča miesta |
| **Vamos** | Vlastný webový portál<br>Viac externých zdrojov | Necielené odporúčanie<br>Agregácia podľa popularity<br>Neodporúča miesta<br>Nepokrytie malých miest |
| **Eventbride** | Vlastný webový portál<br>Správa vstupeniek | Žiadne odporúčanie<br>Neodporúča miesta<br>Nevyužíva Facebook údaje<br>Chýba vizualizácia na mape |


Obrázek 2.1: Prvky navigácie smerom späť - tlačidlo Domov (vľavo), tlačidlo Späť (vpravo)

## 2.2 Užívateľské rozhranie

Pod pojmom užívateľské rozhranie sú zahrnuté akékoľvek komponenty, ktoré užívateľ vidí a interaktuje s nimi. Android ponúka sadu základných komponent, pomocou ktorých je možné vytvoriť takmer akúkoľvek aplikáciu (ktorá nevyžaduje pokročilé grafické prvky). Je tak zaistená jednotnosť medzi aplikáciami.

Pri návrhu grafického rozhrania sa kladie veľký dôraz na jednoduchosť a intuitívnosť ovládania. Je teda dôležité správne navrhnúť navigáciu v aplikácií. Navigácia dopredu je zvyčajne veľmi intuitívna a užívateľ vie, čo mu bude zobrazené, pretože si to sám vybral. V navigácií smerom vzad rozlišujeme 2 druhy akcií - *Systémové tlačidlo Späť* a *aplikačné tlačidlo Domov*, predstavované ikonou aplikácie v ľavom hornom rohu (viz. obrázok 2.1).

**Tlačidlo Späť** slúži na navigáciu späť ku predchádzajúcej aktivite (rodičovská aktivita ku aktuálnej).

**Tlačidlo Domov** slúži k navigácií na začiatok logického celku (napr. pri prechode cez vyhľadávanie až ku detailu položky, tlačidlo Domov navráti stav aplikácie ku aktivite pre kritéria vyhľadávania).

Za použitia uvedených prvkov navigácie bola navrhnutá mapa navigácie v aplikácií (viz. obrázok 2.2).

Aplikácia pozostáva zo 6 základných obrazoviek, ktoré ponúkajú možnosti vyhľadávania za pomoci zadaných parametrov, zobrazenia výsledkov vyhľadávanie, zobrazenia detailného popisu a vizualizácie polohy udalosti alebo miesta na mape.

## 2.3 Architektúra serverovej časti

Serverová časť aplikácie sa stará o získavanie odporučiteľných dát, vyhľadávanie v Graph API strome (viac ku Graph API v kapitole 4), počítanie odporúčacích algoritmov a prijímanie a spracovávanie požiadaviek REST služieb. Ako úložisko dát používa databázu *MySQL*.

Pri získavaní údajov z Graph API sú dáta filtrované od miest, ktorými sa aplikácia nezaoberá (námestie, cintorín, obchody, firmy, atď.). Po filtrovaní sú údaje uložené do MySQL databázy. Uložené údaje, potrebné pre odporúčací proces, sú rozdelené do jednotlivých databázových tabuliek. Štruktúra databázy je názorne vyobrazená v ER (Entity Relationship) diagrame 2.3. Nad uloženými údajmi prebiehajú výpočty váh jednotlivých položiek. Tieto váhy sú následne používané pri hľadaní odporúčaní.

Údaje zo siete Facebook sú získavané len za účelom odporúčania a selektujú sa len polia dôležité pre odporúčanie. Zvyšné zobrazované údaje, ktoré sú potrebné v Android aplikácií sú sťahované jednorazovo a to na žiadosť od klienta. Databáza teda nepresahuje viac, než je potrebné. Taktiež zasielané údaje prostredníctvom RESTful služieb sú tým minimalizované a teda nezaťažujú užívateľove internetové pripojenie.

Serverová aplikácia komunikuje s klientskymi zariadeniami na základe definovanej sady RESTful služieb (viac o definovaných RESTful v kapitole 5).

## 2.4 Odporúčací systém

Základnou požiadavkou na odporúčací systém je odporúčanie v reálnom čase. Vzhľadom na to, že získavanie váh jednotlivých položiek je výpočetne náročný proces, je snaha paralelizovať operácie do čo najvyššej úrovne a generalizovať výpočty tak, aby jednotlivé časti boli znovu použiteľné ako pre odporúčanie miest tak aj pre odporúčanie udalostí. Odporúčací systém je teda možné rozdeliť na 2 kategórie - odporúčanie miest a odporúčanie udalostí.

Pri návrhu **odporúčania udalostí** sa vyskytol problém s hodnotením budúcich udalostí. Užívatelia nemajú na základe čoho ohodnotiť udalosti, ktoré sa ešte nestali. Ponúka sa teda otázka, ako odporúčať udalosti, pokiaľ hodnotenie nie je dostupné? Z dvoch prístupov odporúčania (kolaboratívne a obsahové) teda ostáva obsahové odporúčanie. Namiesto hodnotenia samotnej udalostí sa vyžaduje od užívateľa hodnotenie relevantnosti danej udalosti voči jeho záujmom. Udalosti s pozitívnom hodnotením relevantnosti sú teda považované za referenčné a na základe ich obsahu sú odporúčané iné udalosti užívateľovi.

**Odporúčanie miest** spĺňa predpoklady pre kombinovanie odporúčacích systémov - z pohľadu návrhu, je získanie údajov o hodnotení jednoduché a teda môže byť spracované pri odporúčacom procese. Údaje o miestach taktiež obsahujú, či už krátke alebo dlhé popisy, čo umožňuje aj obsahové odporúčanie.

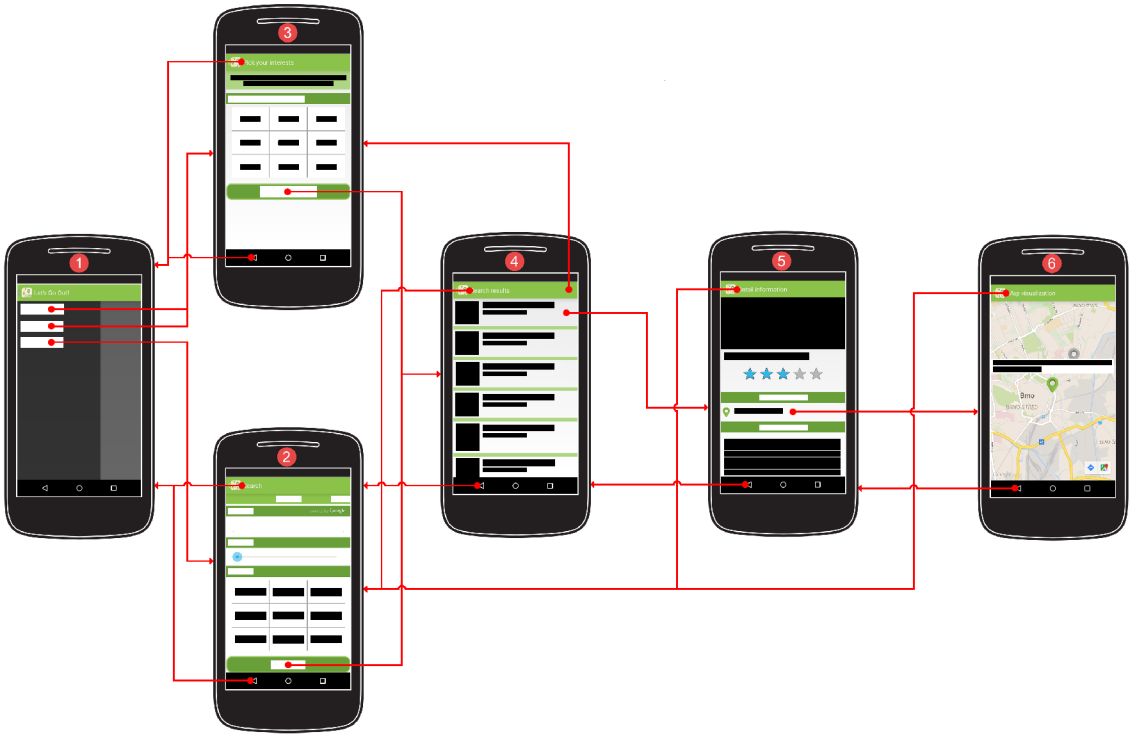
Po definovaní odporúčacích prístupov je potrebné rozhodnúť, ako budú odporúčacie systémy spolupracovať. Keďže typ prístupu odporúčania pre udalosti je obmedzený na obsahový, zostáva otázka spolupráce prístupov pri odporúčaní miest.

Po krátkom prieskume zameranom na počet udalostí a miest v určitej oblasti (skúmaná bola oblasť Brno a okolie), bolo nájdených prostredníctvom Graph API viac ako 200

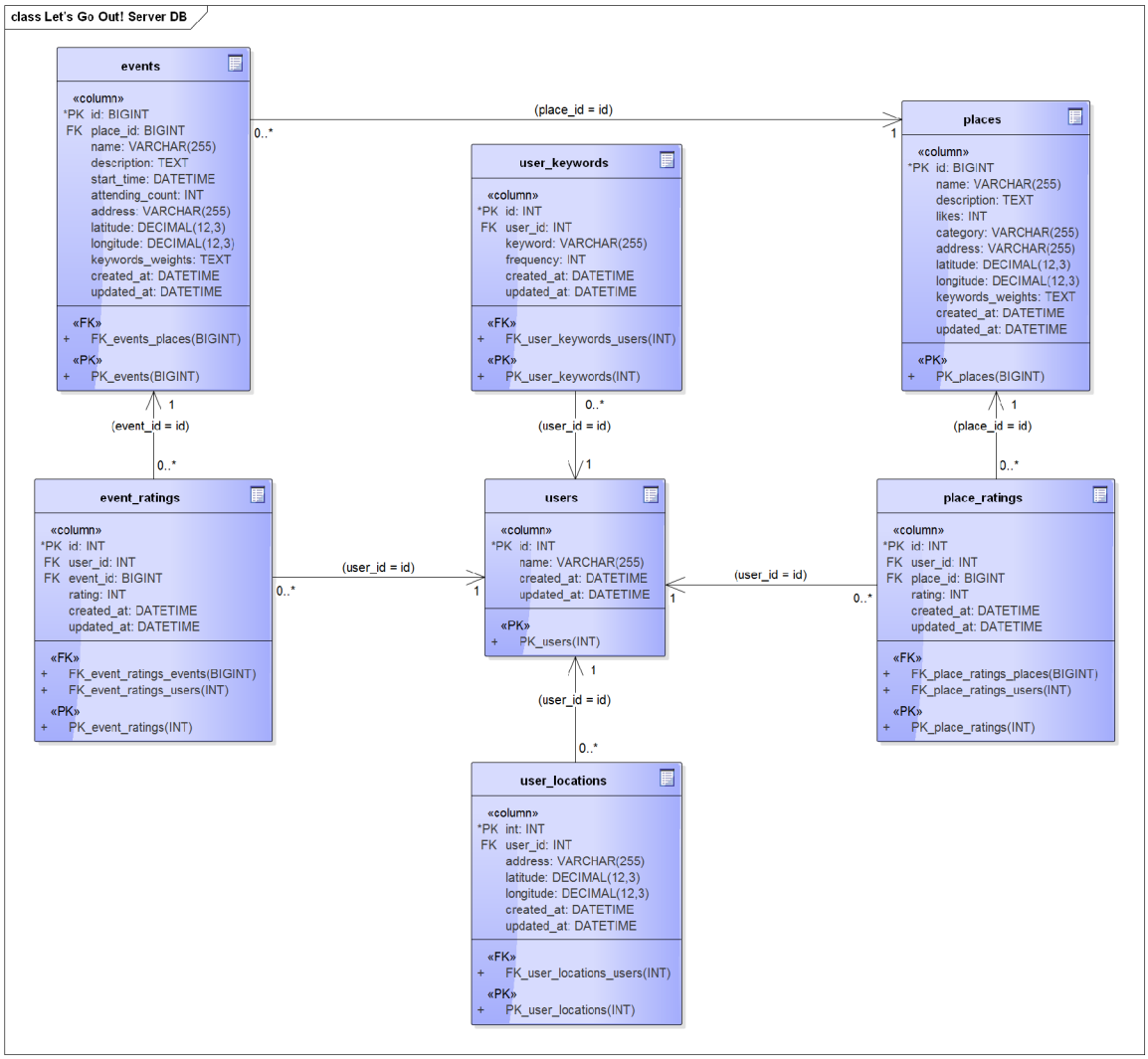
ako miest, tak aj udalostí. Filtrovanie pomocou kľúčových slov, kategórií a odporúčacích algoritmov v sérií by tento dátový základ mohol zúžiť až na prázdnu množinu a novým užívateľom by teda neboli odporúčania nájdené. Vybraným spôsobom spolupráce je teda paralelné vykonávanie algoritmov a následné zjednotenie množím.

## 2.5 Zhrnutie

Ako odporúčací systém bol teda vybratý koncept hybridného systému, ktorý zahŕňa ako odporúčanie na základe obsahu (udalosti, miesta), tak aj kolaboratívne odporúčanie (miesta), ktoré berie do úvahy jednotlivé hodnotenia miest a hľadá tak najbližšieho suseda ku aktuálnemu užívateľovi. Tento koncept je obohatený taktiež prvkami strojového učenia. Získaním užívateľových záujmov a sledovaním jeho správania si systém vybuduje bázu kľúčových slov a parametrov, ktoré potom ovplyvnia celkový výber odporučených položiek. Keďže užívateľa najviac osloví vizuálna stránka aplikácie, je teda jednoduchý a intuitívny grafický design nemenej dôležitý.




Obrázek 2.2: Mapa navigácie v aplikácií: Počiatočnou aktivitou je MainActivity (1), ktorá prostredníctvom menu spúšťa 2 aktivity a 1 fragment. Pri voľbe Vyhľadávania sa pripojí fragment SearchFragment (2), z ktorého je možné sa pomocou tlačidla Search dostať na aktivitu SearchResultActivity (4). Pri voľbe akéhokoľvek z odporúčaní a za podmienky, že užívateľ ešte nezadal svoje preferencie pre odporúčanie, sa spustí aktivita InterestsActivity (3). Po vyplnení a potvrdení preferencií je užívateľ presmerovaný na SearchResultActivity (4). Pokiaľ ale užívateľ svoje preferencie už vyplnil, tak z MainActivity (1) sa vybraním akéhokoľvek odporučenia dostane priamo na SearchResultActivity (4), z kade je potom možné dostať sa na Detail Activity (5). Z DetailActivity (5) vedie jedna cesta dopredu, a to na MapActivity (6). Pri použití tlačidla Home (ľavý horný roh s ikonou aplikácie) sa akákoľvek aktivita presmeruje priamo na MainActivity (1), kde je automaticky zobrazený SearchFragment (2). Pri návrate pomocou systémového tlačidla Späť, je užívateľ presmerovaný vždy o 1 aktivitu vzad cestou, ktorou sa k aktuálnej aktivite dostal.




Obrázek 2.3: Entity Relationship diagram, databázových tabuliek. Tabuľky *places* a *events* obsahujú informácie zo siete Facebook. Tabuľka *users* je s nimi vo vzťahu M:N prostredníctvom väzobných tabuliek *place_ratings* a *event_ratings*. Tabuľky *user_locations* a *user_keywords* obsahujú doplňujúce údaje o užívateľovi.

# Kapitola 3

# Odporúčacie systémy

Odporúčacie systémy vznikli za jediným účelom, a to pomôcť všetkým užívateľom nájsť vhodné a hlavne pre užívateľa relevantné informácie. Môžeme teda povedať, že odporúčacie systémy sú srdcom aplikácie, ktorá komunikuje priamo s užívateľom, spracováva spätnú väzbu, monitoruje užívateľove interakcie s aplikáciou a reflektuje tieto informácie do presnejších predpovedí.

Odporúčacie systémy môžeme rozdeliť, na základe zdroju informácií, podľa ktorého dochádza ku odporúčaniu, na 2 skupiny:

**Obsahové odporúčacie systémy** Dochádza ku hľadaniu najrelevantnejších informácií, na základe ich obsahu a kľúčových prvkov.

**Kolaboratívne odporúčacie systémy** Ku odporúčaniu využíva hodnotenia celej skupiny ľudí. Odporúčanie je presnejšie a špecifickejšie pre užívateľa, avšak nastáva problém nazývaný "cold start".

**Hybridné odporúčacie systémy** Spájajú viacej druhov odporúčacích systémov do jedného celku.

V tejto kapitole je načrtnutý historický vývoj odporúčacích systémov, 3 základné prístupy ku odporúčaniu, algoritmy všeobecne považované za zlatý štandard pri vývoji takýchto systémov a problémy, s ktorými je nutné sa vysporiadať. Hlavným zdrojom informácií v tejto kapitole je [3].

## 3.1 História

Myšlienka odporúčania a prispôsobovania obsahu užívateľovi sa objavila prvýkrát v roku 1979 v systéme Grundy [5]. Tento systém zastupoval knihovníka a odporúčal knihy na základe užívateľského profilu. Užívatelia boli rozdeľovaní do hierarchických stereotypov podľa profilu, ktorý mali v systéme vytvorený. So vzrastajúcim počtom užívateľov a požiadaviek na odporúčanie ale vzrastali aj nároky na dátové úložiská a systémové zdroje, pretože odporúčací systém vyžadoval podrobné informácie o jednotlivých položkách.

V záujme zlepšovania odporúčania a automatizácie celého procesu vzišla myšlienka *kolaboratívneho filtrovania* položiek. Tento princíp sa objavil prvýkrát v roku 1992 v projekte *Tapestry* [2], ktorý bol vyvinutý spoločnosťou Xerox PARC (Palo Alto Research Center). Avšak skutočná automatizácia bola zavedená až v projekte GroupLen [4] v roku 1994, ktorý odporúčal užívateľom články na internete na základe porovnávania podobností užívateľov.

Na kolaboratívnom filtrovaní bolo od vtedy postavených mnoho odporúčacích systémov, ktoré ale majú stále spoločný princíp - **vyhľadávane najbližších susedov** (angl. *k-nearest neighbor / kNN*).

## 3.2 Vyhľadávanie najbližších susedov

Pod týmto pojmom rozumieme získavanie podobných položiek na základe existujúcich podobných vlastností s referenčnou položkou. V oblasti odporúčacích systémov sa jedná najčastejšie o hodnotenie položiek (kolaboratívne filtrovanie) a váhu podobnosti obsahu položiek (obsahové odporúčanie).

Prvým používaným prístupom pre vyhľadávane najbližších susedov je **Pearsonov koeficient korelácie** [3], ktorý je definovaný nasledujúcim vzťahom:

$$sim(u_1, u_2) = \frac{\sum_{p \in P}(r_{u_1,p} - \overline{r_{u_1}})(r_{u_2,p} - \overline{r_{u_2}})}{\sqrt{\sum_{p \in P}(r_{u_1,p} - \overline{r_{u_1}})^2}\sqrt{\sum_{p \in P}(r_{u_2,p} - \overline{r_{u_2}})^2}} \tag{3.1}$$

Parametre $u_1$ a $u_2$ označujú dvoch porovnávaných užívateľov. Premenná $r_{u,p}$ predstavuje hodnotenie užívateľa $u$ (v našom prípade $u_1$ alebo $u_2$) spracovávanej položky $p$. $\overline{r_u}$ je priemer hodnotení pre užívateľa $u$ (opäť v našom prípade $u_1$ alebo $u_2$) . *Pearsonov koeficient korelácie* naberá hodnôt v rozsahu -1 (malá podobnosť) až 1 (silná podobnosť).

*Pearsonov koeficient korelácie* je považovaný za štandard pri vyhľadávaní najbližšieho suseda podľa užívateľov. V tomto prístupe dochádza ku porovnaniu hodnotení 2 užívateľov, ako je to ukázané vo vzťahu 3.1.

Druhým kolaboratívnym prístupom je vyhľadávanie najbližšieho suseda podľa hodnotenia položiek, teda vyhľadávanie najbližšieho suseda medzi položkami. Štandardom v tomto smere je algoritmus **Kosínusovej miery podobnosti** (angl. *Cosine Similarity Measure*) [3]. Základom výpočtu je hľadanie podobnosti medzi dvomi $n$-dimenzionálnymi vektormi reprezentujúcimi 2 porovnávané položky. Tento prístup je často využívaný v oblasti klasifikácie textu a data miningu pre zisťovanie podobnosti dvoch textových dokumentov. V neposlednej rade je taktiež aplikovaný pri obsahovom odporúčaní.

*Kosínusovú mieru podobnosti* je možné vyjadriť nasledujúcim vzťahom:

$$sim(\vec{i_1}, \vec{i_2}) = \frac{\vec{i_1} \cdot \vec{i_2}}{|\vec{i_1}| * |\vec{i_2}|} \tag{3.2}$$

Parametre $i_1$ a $i_2$ označujú 2 porovnávané položky v systéme. Čitateľ vzťahu predstavuje skalárny súčin vektorov dvoch spracovávaných položiek. Menovateľom vzťahu je súčin 2 Euklidovských priestorov vektorov (Euklidový priestor vektoru je definovaný ako $\sqrt{\vec{i} \cdot \vec{i}}$, teda odmocnina zo skalárneho súčinu vektoru samého so sebou). Vzťah naberá hodnôt v rozmedzí 0 (malá podobnosť) až 1 (silná podobnosť).

Využitie spomenutých vzťahov je rozobrané v nasledujúcich podkapitolách.

## 3.3 Obsahové odporúčanie

Myšlienkou obsahového odporúčania je nasmerovať užívateľa na položky v systéme, ktoré spadajú do jeho okruhov záujmu. Každý užívateľ teda musí mať v systéme profil, ktorý obsahuje nasledujúce informácie:

- Zaznamenané údaje o užívateľovom používaní aplikácie
- Explicitne vyžiadané informácie o užívateľovi

**Údaje o užívateľovom používaní aplikácie** môžu zahŕňať vyhľadávané kľúčové slová a ich frekvenciu alebo typy položiek, ktoré užívateľ vyhľadával atď.

Mnoho aplikácií taktiež využíva **explicitné zisťovanie záujmov**. Užívateľ je požiadaný o vyplnenie, alebo výber záujmov zo škály, ktorú aplikácia ponúka. Tento spôsob je značne jednoduchší na vývoj, avšak jeho slabou stránkou je neautomatizovaná škálovateľnosť, teda neprispôsobuje aplikáciu vývinu užívateľových záujmov, ktoré sa môžu často meniť.

Pri *odporúčaní na základe obsahu* je predpokladom získať podrobné informácie o spracovávaných položkách. *Príklad:* Relevantné informácie pre knihy môžu byť *autor*, *žáner*, *kľúčové slová*, atď. Získavanie takýchto údajov môže byť ale veľmi pracné, a často aj problémové, pokiaľ dané informácie nie sú verejne prístupné. Veľký dôraz sa teda kladie na automatizáciu procesu získavania údajov. Systém taktiež musí tieto údaje zachovávať vo svojej databáze, čo pri veľkom počte položiek môže spôsobiť vysoké pamäťové nároky. Na druhú stranu, pre správnu selekciu odporúčaných položiek nie je nutné zvažovať širokú škálu užívateľov a ich hodnotenia jednotlivých odporúčaných prvkov v systéme.

**Predpovedanie možného hodnotenia** užívateľa pre položky v systéme je následne počítané za použitia textového obsahu jednotlivých položiek. Pre spracovanie textového obsahu sa využívajú princípy klasifikácie textu. V tejto práci si priblížime **2 princípy**, a to:

1. Frekvencia slov v texte
2. TF-IDF kódovanie textu

Princíp **frekvencie slov v texte** sa spolieha na rozdelenie textu na slová a následné porovnávanie frekvencie výskytu jednotlivých slov s ostatnými frekvenciami v iných položkách (dokumenty, články, atď.). Jedná sa o veľmi triviálny princíp, ktorý má ale nevýhodu v tom, že položky s dlhším textovým obsahom budú mať vždy prednosť, pred položkami s krátkym textom.

Tento problém je vyriešený za použitia **TF-IDF** kódovania. Text, kódovaný formátom TF-IDF, je reprezentovaný ako vektor jednotlivých váh pre vybrané kľúčové slová v texte. Na výpočet váhy kľúčového slova sa používa nasledujúci vzťah:

$$TF\text{-}IDF(k,d) = TF(k,d) * IDF(k) \tag{3.3}$$

Koeficient **TF** predstavuje frekvenciu kľúčového slova v spracovávanom texte (angl. *term frequency*). Vyššie spomínaný problém s prioritou položiek s dlhším obsahom je vyriešený za pomoci využitia maximálneho počtu výskytov iného, než spracovávaného, kľúčového slova [1]:

$$TF(k,d) = \frac{freq(k,d)}{maxOthers(k,d)} \tag{3.4}$$

Výsledkom je teda podiel frekvencie slova $k$ v dokumente $d$ a maximálnej hodnoty výskytu akéhokoľvek iného slova v texte.

V spracovávaných informáciách sa často vyskytujú slová, ktoré majú vysoký počet výskytov vo všetkých spracovávaných položkách, avšak nedajú sa radiť medzi kľúčové slová

(napr. spojky, predložky, atď.). Koeficient **IDF** (angl. *inverse document frequency*) slúži na eliminovanie vysokej váhy slov tohoto typu.

Nech $N$ je počet všetkých možných dokumentov ku odporučeniu a *n(k)* je počet všetkých dokumentov, v ktorých sa kľúčové slovo $k$ vyskytuje. Potom koeficient IDF pre kľúčové slovo $k$ je počítaný nasledujúcim vzťahom.

$$IDF(k) = \log \frac{N}{n(k)} \tag{3.5}$$

Za pomoci vyššie uvedených vzťahov je možné vytvoriť vektor váh jednotlivých slov pre každý spracovávaný dokument. Následne sú vektory váh použité pre získavanie najbližšieho suseda - teda podobných dokumentov. V prípade obsahového odporúčania sa štandardne využíva, ako už bolo spomínané, vzťah pre kosínusovú mieru podobnosti 3.2. Na základe výslednej hodnoty podobnosti je teda možné predpokladať reakciu užívateľa na danú položku v závislosti na jeho reakcií na referenčnú položku.

Základné otázky, ktoré systémy s obsahovým odporúčaním musia riešiť, sú nasledovné:

1. Ako zistiť oblasti užívateľových záujmov?
2. Ako zistiť podobnosť dvoch položiek?
3. Ako prispôsobiť odporúčanie vývinu užívateľových záujmov?
4. Ako získať podrobné údaje o položkách v systéme?

## 3.4 Kolaboratívne filtrovanie

Na rozdiel od obsahových odporúčacích systémov, ktoré pre potreby výpočtu odporúčania používajú podrobné údaje o prvkoch, sa **kolaboratívne systémy** spoliehajú na užívateľov v komunite, ktorá aplikáciu používa. Pre potreby odporúčania je potrebná explicitná spätná väzba v podobe hodnotenia. Tento fakt môže spôsobovať problémy, pretože nie každý užívateľ je ochotný ohodnotiť svoju skúsenosť, čo znamená že nárast databázy hodnotení je v začiatkoch veľmi pomalý.

Princípy kolaboratívneho odporúčania môžeme rozdeliť do dvoch skupín:

1. Odporúčanie na báze užívateľov
2. Odporúčanie na báze položiek

Cieľom **odporúčania na báze užívateľov** je nájsť užívateľa, ktorý je podobný v minulých hodnoteniach s aktuálnym užívateľom aplikácie, pre ktorého sú aktuálne počítané položky ku odporučeniu a cielene navrhnúť položky, ktoré aktuálny užívateľ ešte nevidel ("podobný"užívateľ ich v minulosti ohodnotil).

Pre získavanie najbližších susedov sa štandardne používa *Pearsonov koeficient korelácie* 3.1. Za použitia tohoto vzťahu je možné predpokladať hodnotenie užívateľa. Pre tento predpoklad sa v tomto princípe používa nasledujúci vzťah.

Nech $N$ je množina všetkých užívateľov, ktorý spadajú do skupiny najbližších susedov užívateľa $a$, $r$ je hodnotenie položky od daného užívateľa a $\vec{r}$ je priemerné hodnotenie daného užívateľa. Potom za použitia vzťahu *sim(a,p)*, ktorý predstavuje vzťah 3.1, vypočítame predpokladané hodnotenie položky $p$ užívateľom $a$ ako:

$$pred(a,p) = \vec{r_a} + \frac{\sum_{b \in N} sim(a,b) * (r_{b,p} - \vec{r_b})}{\sum_{b \in N} sim(a,b)} \tag{3.6}$$

Výsledkom funkcie je predpokladané hodnotenie, na základe ktorého je možné odporučiť položku.

Naopak **odporúčania na báze položiek** sa zaoberá porovnávaním vektorov hodnotení dvoch položiek, čo znamená, že cieľom je vyhľadať najbližších susedov pre položky, ktoré užívateľ už hodnotil.

Štandardom pre získavanie najbližších susedov, v tomto princípe, je vzťah *Kosínusovej miery podobnosti* 3.2. Predpokladané hodnotenie užívateľa je následne počítané za pomoci nasledujúceho vzťahu.

Nech $M$ je množina všetkých položiek, ktoré užívateľ $u$ ohodnotil a $r$ je hodnotenie položky od užívateľa. Potom za použitia vzťahu *sim(u,p)*, ktorý predstavuje vzťah 3.2, vypočítame predpokladané hodnotenie ako:

$$pred(u,p) = \frac{\sum_{i \in M} sim(i,p) * r_{u,i}}{\sum_{i \in M} sim(i,p)} \tag{3.7}$$

Výsledkom funkcie je predpokladané hodnotenie, na základe ktorého je možné odporučiť položku.

*Kolaboratívne odporúčacie systémy* taktiež čelia problému **cold start**, čo znamená že systém je závislý na databáze hodnotení, ktorá ale pri jeho spustení systému neexistuje.

Základné otázky, ktoré kolaboratívne odporúčacie systémy musia riešiť, sú nasledovné:

1. Ako nájsť podobných užívateľov alebo podobné položky?
2. Ako vypočítať podobnosť?
3. Ako sa má systém správať v prípade nových užívateľov?
4. Ako riešiť *cold-start* problém?

## 3.5 Hybridné prístupy

Hybridné odporúčacie systémy sú vytvorené kombináciou 2 predchádzajúcich skupín. Je tak docielené odstránenie nevýhod oboch skupín za cenu mierneho nárastu systémových nárokov. odporúčania sú teda presnejšie ako pre nových užívateľov, tak aj pre stálu komunitu.

Pri hybridných odporúčacích systémoch môžeme hovoriť o **troch konštrukciách**, a to:

**Paralelný prístup** ako už názov napovedá, spočíva v spojení dvoch systémov, ktoré nezávisle na sebe vytvoria sadu odporúčaných položiek. 2 vytvorené sady sú následne spojené do jednej a predložené užívateľovi.

**Sériový prístup** používa výstup z jedného odporúčacieho systému ako vstup do druhého. Výber odporúčaných položiek je teda užší a tým pádom aj špecifickejší.

**Monolitický prístup** sa, na rozdiel od predchádzajúcich prístupov, neskladá z dvoch komponent ale len z jednej. Tento hybridný systém kombinuje princípy viacerých odporúčacích systémov do jednej komponenty.

Základné otázky, ktoré hybridné odporúčacie systémy musia riešiť (okrem otázok jednotlivých častí systému), sú nasledovné:

1. Ako skombinovať odporúčacie prístupy do jedného celku?
2. Ako zjednotiť váhu položiek z dvoch rozličných odporúčacích systémov?

## 3.6 Využívané prístupy

Väčšina zo spomínaných vzťahov je v aplikácií *Let's Go Out!* využívaná. Keďže aplikácia využíva ako kolaboratívny prístup tak aj obsahový, najpodstatnejším vzťahom je *Kosínusová miera podobnosti* 3.2. Tá sa využíva v obidvoch odporúčacích prístupoch. Pri udalostiach dochádza ku počítaniu podobnosti na základe váh *TF-IDF* 3.3. Odporúčanie miest využíva *Kosínusovú mieru podobnosti* ako pri obsahovom tak aj pri kolaboratívnom odporúčaní. Obsahové odporúčanie miest využíva, podobne ako pri udalostiach, *TF-IDF* váhu jednotlivých popisov miest. Kolaboratívne odporúčanie aplikuje tento vzťah na vektory hodnotení dvoch porovnávaných položiek. Jedná sa teda o prístup *odporúčania na báze položiek*.

# Kapitola 4

# Vývojové prostriedky

Počas vývoja bolo nutné naštudovať a pochopiť niekoľko technológií, ktoré sú nevyhnuteľné pre vypracovanie tejto práce. Do týchto technológií patria primárne:

**Android OS** Operačný systém klientskej časti aplikácie

**Ruby on Rails** Technológia na pozadí serverovej časti aplikácie

**Graph API** Aplikačné rozhranie, zverejnené spoločnosťou Facebook, pre prístup k údajom sociálnej siete

V nasledujúcich podkapitolách sú tieto technológie krátko priblížené.

## 4.1 Android OS

Android je aktuálne najpoužívanejším mobilným operačným systémom s podielom na trhu viac ako 76,6% [1]. Je vyvíjaný spoločnosťou Google a klasifikuje sa ako *Open Source* systém, čo znamená, že zdrojový kód je voľne prístupný a stiahnuteľný. Avšak na trhu chytrých telefónov, len malé percento všetkých zariadení disponuje s Androidom v originálnej podobe. Výrobcovia mobilných zariadení často modifikujú zdrojový kód a obohacujú ho o vlastné aplikácie a firmware.

Užívateľské rozhranie operačného systému Android je založené na priamej manipulácií s dátami pomocou dotykových senzorov na displeji zariadenia. Dosahuje tým vysokú úroveň intuitívnosti, čo robí ovládanie zariadení, ako tablety a chytré telefóny, jednoduchým.

V historickom vývoji verzií systému Android došlo ku niekoľkým principiálnym zmenám, ktoré zmenili programátorský prístup ku vývoju mobilných aplikácií.

Verzie Androidu, pre chytré telefóny, sa dajú rozdeliť do 2 skupín:

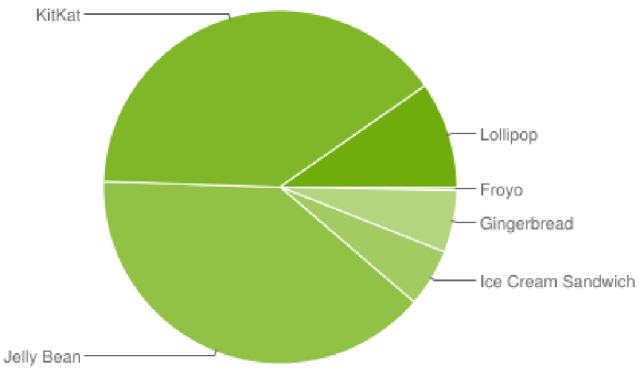
**Android 2.3 a nižšie** Aplikácie pozostávali zo sérií aktivít. Aktivity majú nízku znovu použiteľnosť čo viedlo ku zbytočnému narastaniu zdrojového kódu, a pri veľkých aplikáciách ku neprehľadnosti. v dnešnej dobe je na trhu približne 6% zariadení, ktoré disponujú systémom Android s verziou Android 2.3 a nižšie.

**Android 4.0 a vyššie** Aplikácie boli obohatené o fragmenty, ktoré značne zvýšili znovu použiteľnosť a prehľadosť aplikácií. Fragmenty boli predstavené vo verzií Android 3.0,

[1] http://www.idc.com/prodserv/smartphone-os-market-share.jsp

ktorá bola ale nasadená len na tablety a chytré telefóny obišla. Prístup s použitím aktivít bol firmou Google označený ako zastaraný. Aktuálne zastúpenie tabletov s verziou systému Android 4.0 a vyššie je približne 94%.

| Version | Codename | API | Distribution |
|---|---|---|---|
| 2.2 | Froyo | 8 | 0.3% |
| 2.3.3 - 2.3.7 | Gingerbread | 10 | 5.7% |
| 4.0.3 - 4.0.4 | Ice Cream Sandwich | 15 | 5.3% |
| 4.1.x | Jelly Bean | 16 | 15.6% |
| 4.2.x | | 17 | 18.1% |
| 4.3 | | 18 | 5.5% |
| 4.4 | KitKat | 19 | 39.8% |
| 5.0 | Lollipop | 21 | 9.0% |
| 5.1 | | 22 | 0.7% |




Obrázek 4.1: Zastúpenie verzií Android OS na trhu mobilných zariadení - prevzaté z [2]

Z obrázku 4.1 je badateľné, ako zariadenia s verziami Android OS 4 a vyššie dominujú trhu. A práve vzhľadom na tento fakt bola ako minimálna podporovaná verzia Androidu, pre túto prácu, vybraná verzia Android OS 4.0.

### 4.1.1 Android SDK

Pre vývoj aplikácií na Android zariadenia bola firmou Google vydaná súprava nástrojov *Android Software Development Kit* (skr. Android SDK). Táto súprava ponúka vývojárom potrebné knižnice pre vývoj aplikácií (základné komponenty každej Android aplikácie), emulátor systému Android, vzorky zdrojových kódov pre pochopenie princípu, základné návody a dokumentáciu. Spolu s vývojom operačného systému Android sa vyvíja aj Android SDK, o čo sa stará nástroj *Android SDK Manager*.

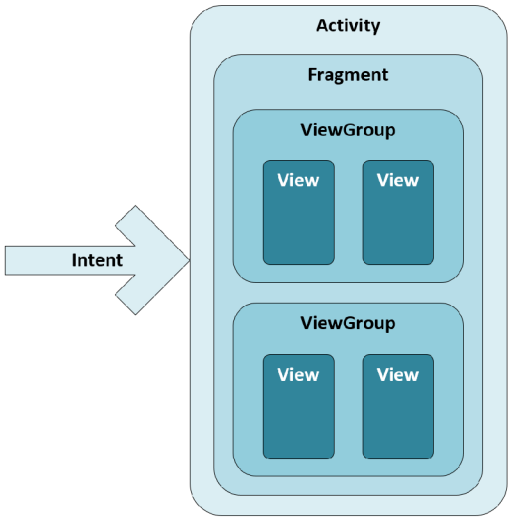
V dnešnej dobe je už Android SDK integrovaný do oficiálneho vývojového prostredia - **Android Studio** [3]. Android Studio je postavené na báze existujúceho vývojového prostredia IntelliJ IDEA, vyvíjané firmou JetBrains.

### 4.1.2 Architektúra Android aplikácií

Každá Android aplikácia pozostáva so sérií komponent, ktoré tvoria užívateľské rozhranie. Tieto komponenty sú hierarchicky usporiadané v štruktúre, ktorú môžete vidieť na obrázku 4.2.

Jednotlivé komponenty hierarchie sú v nasledujúcich riadkoch popísané (informácie sú prevzaté z Android SDK dokumentácie [4]).

[2] https://developer.android.com/about/dashboards/index.html
[3] https://developer.android.com/sdk/index.html
[4] http://developer.android.com/reference/packages.html




Obrázek 4.2: Hierarchia základných komponent Android aplikácie

**Intent** Objekt predstavujúci správu, ktorou komponenty vyžadujú určitú akciu od druhých komponent aplikácie. Najčastejšie slúži ako prepojenie medzi aktivitami. Pomocou Intentov je možné:

1. Komunikovať s aktivitami - Predávanie parametrov a spúšťanie nových aktivít
2. Zasielať požiadavky na Broadcast prijímače - Spúšťanie systémových volaní, napr. nastavenia, fotoaparát, atď.
3. Komunikovať so službami na pozadí - Spúšťanie jednorazovej operácie na pozadí

**Activity** Základný komponent Android aplikácií, ktorá v sebe nesie nadefinované užívateľské rozhranie s ktorou môže užívateľ interaktovať. Aplikácie vo väčšine prípadov pozostávajú z viacerých aktivít, ktoré sú medzi sebou prepojené prostredníctvom naimplementovanej navigácie.

Pri spúšťaní viacerých aktivít sú jednotlivé aktivity ukladané do pozadia na zásobník aktivít a aktívna ostáva len posledná spustená aktivita. Činnosť predchádzajúcej aktivity je pozastavená. Pri navigácií v spätnom smere sa aktuálna aktivita ukončuje (pokiaľ to nie je inak definované priamo v aktivite) a do popredia sa dostáva posledná pozastavená aktivita (opäť, pokiaľ to nie je inak naimplementované). V priebehu činnosti sa aktivita môže dostať do nasledujúcich stavov - vytvorená, aktívna, pozastavená, ukončená.

**Fragment** Fragmenty vznikli za účelom vytvorenia dynamického užívateľského rozhrania, zvýšenia znovu použiteľnosti komponent a zníženia zložitosti kódu. Sú to teda znovu použiteľné časti užívateľského rozhrania, ktoré sa nachádzajú v aktivitách. Aktivita môže mať viac fragmentov, pričom každý fragment je samostatný a môže byť s ním manipulované v reálnom čase (pridávanie, odoberanie fragmentov v závislosti na užívateľových interakciách).

Životný cyklus fragmentu je úzko spätý so stavom aktivity. Napriek tomu, že aktivita počas svojej činnosti môže fragmenty vytvárať, ničiť a pozastavovať, tak pri pozastavení samotnej aktivity sú pozastavené aj jej fragmenty.

**ViewGroup** Zapuzdrená skupina komponent, ktoré vytvárajú užívateľské rozhranie. Taktiež je to koreňový element každého rozloženia, ktoré je definované pre aktivitu.

**View** Samotné komponenty, s ktorými užívateľ interaktuje. View komponenty musia byť umiestnené vo ViewGroup komponente.

## 4.2 Ruby on Rails

Ruby on Rails (ďalej len RoR) je nadstavba programovacieho jazyka Ruby, ktorá je určená pre vývoj objektovo orientovaných webových aplikácií. Abstrahuje zložité a opakujúce sa úlohy do jednoduchých príkazov, čím značne urýchľuje vývoj.

Programovanie v RoR sa vyznačuje nasledujúcimi charakteristikami:

**Jednotnosť kódu** Pri akomkoľvek vývoji je nutné definovať a dodržiavať štruktúru a konvencie, aby nedochádzalo ku nejednotnosti. RoR má pevne definovanú štruktúru aplikácií a taktiež konvencie pomenovávania súborov. Z dôvodu zachovania jednotnosti v kóde, ponúka RoR sadu príkazov, ktoré generujú súbory zdrojového kódu vo fixnej štruktúre a pomenováva ich v rámci RoR konvencií. Súbory sú hierarchicky zoradené do priečinkov, ktoré zodpovedajú ich funkcionalite.

**Rozšíriteľnosť** RoR patrí ku tzv. *Open Source* jazykom, teda jeho zdrojové kódy sú voľne prístupné a stiahnuteľné na internete. Veľa ľudí na celom svete sa angažuje v rámci RoR komunity pri rozširovaní funkcionality jazyka. Vyvíjajú tzv. *gemy* (alebo *RubyGems*), ktoré pridávajú nové možnosti pri vývoji aplikácií v RoR. Gemy je možné stiahnuť manuálne z internetu. RoR taktiež ponúka sadu príkazov pre prácu s RubyGems, ktorá automatizujú proces inštalácie jednotlivých rozšírení.

**Podpora REST** Representational State Transfer (ďalej len REST) je architektúra, ktorá určuje jednotný spôsob komunikácie klientov so serverom (viac ku architektúre REST nájdete v sekcií 5.2.3. RoR aplikácie majú natívnu podporu službám REST. Rovnako ako smerovanie požiadavkou pri bežných webových aplikáciách, funguje aj smerovanie REST požiadavkou. Rozdielom je len vizualizácia spracovaných dát, kde pri REST službách sa používa reprezentácia dát pomocou *XML* alebo *JSON*.

V tejto práci plní RoR úlohu serverovej časti, ktorá extrahuje dáta zo sociálnej siete Facebook. Za týmto účelom je používaná Open Source knižnica *Koala*, dostupná aj ako gem. Knižnica Koala abstrahuje komunikáciu s oficiálnym aplikačným rozhraním Facebooku - Graph API.

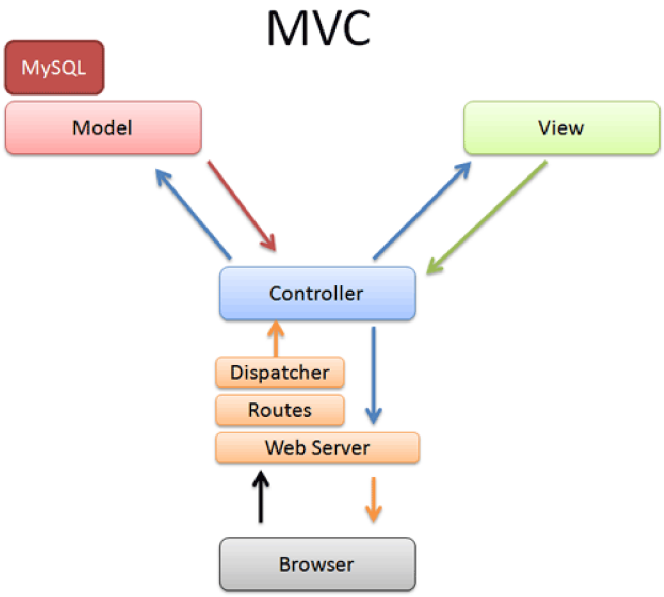
### 4.2.1 MVC Architektúra

Základom Ruby on Rails je *architektúra Model-View-Controller* (MVC). Jedná sa o 3-vrstvovú architektúru, ktorá v jednom cykle údaje prijme, spracuje a zase podá užívateľovi (viz. obrázok 4.3).

Jednotlivé vrstvy plnia nasledujúce funkcie:

**Model** Predstavuje zapuzdrenie dát, ktorá sa nachádzajú v aplikačnej databáze. Priamo s týmito dátami pracuje a modifikuje ich.

[5] http://betterexplained.com/wp-content/uploads/rails/mvc-rails.png




Obrázek 4.3: Architektúra Model-View-Controller v Rails aplikáciách - prevzaté z [5]

**View** Vizualizuje spracovávané údaje do užívateľsky prijateľnej formy. Používa sa jazyk HTML.

**Controller** Stará sa o prijímanie a vybavovanie všetkých užívateľských požiadaviek.

V aplikácií *Let's Go Out!* vrstva *View* nie je používaná, keďže sa nejedná o webovú aplikáciu, dostupnú cez prehliadače. Komunikácia s klientom prebieha na *Controller* vrstve, ktorá komunikuje priamo s *Model* vrstvou a jej výstup prekladá do formátu JSON, ktorý je následne zaslaný do užívateľovho zariadenia

## 4.3 Graph API

Sociálna sieť Facebook zverejnila aplikačné rozhranie, pomenované *Graph API*[6], pomocou ktorého je možné prehľadávať a získavať údaje o stránkach, udalostiach ba dokonca aj užívateľoch Facebooku. Graph API dáva taktiež možnosť zapisovať do sociálneho grafu Facebooku. Údaje majú medzi sebou vzťahy, ktoré tvoria celkovú formu sociálneho grafu.

Graph API, ako už napovedá názov, reprezentuje dáta v podobe grafu. Jednotlivé záznamy sú rozdelené na polia, uzly a hrany medzi nimi.

**Pole** Súčasť uzla, ktorá v sebe nesie údaje o danom uzle (meno, popis, lokáciu, atď.)

**Uzol** Samotná reprezentácia dát. Delia sa na koreňové a nekoreňové uzly. *Koreňové uzly* predstavujú miesta pripojenia a dotazovania do sociálneho grafu Facebooku. Na tieto uzly je možné dotazovať sa priamo. *Nekoreňové uzly* predstavujú údaje, na ktoré nie je možné dotazovať sa priamo. Je nutné prejsť grafom z koreňového uzlu až po žiadaný nekoreňový pre získanie požadovaného údaju.

[6] https://developers.facebook.com/docs/graph-api/overview/

**Hrana** Prepojenia, ktoré definujú jednoznačný vzťah medzi uzlami. Oddeľujú takto koreňové a nekoreňové uzly.

Údaje zo sociálneho grafu sú prezentované vo formáte **JSON**. Príkladom výstupných údajov je nasledujúca štruktúra, so sériou vybraných polí.

```
{
  "id": "172997189478225",
  "name": "Fakulta␣informačních␣technologií␣VUT␣v␣Brně",
  "category": "University",
  "checkins": 878,
  "founded": "1.␣1.␣2002",
  "location": {
    "city": "Brno",
    "country": "Czech␣Republic",
    "latitude": 49.226307672434,
    "longitude": 16.596902622675,
    "street": "Božetěchova␣1/2",
    "zip": "612␣66"
  }
}
```

Na všetky uzly a hrany je možné dotázať sa pomocou HTTP GET požiadavku. v prípade, že ide o akciu, ktorá zahŕňa zapisovanie do grafu, je nutné použiť HTTP POST požiadavku. Aplikácia *Let's Go Out!* využíva len HTTP GET požiadavky, keďže do grafu nepotrebuje zapisovať.

# Kapitola 5

# Implementácia

Implementácia aplikácie bola rozdelená do 2 fáz, obdobne ako tomu je aj v architektúre aplikácie. v prvej fáze bola implementovaná serverová časť a po nej prišla na rad implementácia Android klienta.

V tejto kapitole sú priblížené detaily implementácie ako klientskej strany, tak aj serverovej strany, popis načítavania údajov, výpočtu váhy položky podľa obsahu, a ďalších netriviálnych častí kódu.

## 5.1 Štruktúra serverovej časti

Ako už bolo spomínané, serverovú časť tvorí webová aplikácia na platforme Ruby on Rails (ďalej len RoR). RoR aplikácie majú fixnú štruktúru, ktorá je daná konvenciami jazyka. RoR abstrahuje vytváranie projektu do nasledujúceho príkazu, pomocou ktorého je vygenerovaná celá priečinková štruktúra webovej aplikácie.

```
rails new <meno_projektu>
```

Hierarchia priečinkov v RoR aplikáciách je rozsiahla a obsahuje aj priečinky, ktoré aktuálne nie sú v aplikácií používané, avšak z dôvodu RoR konvencií, musia byť v adresári projektu prítomné. Popíšeme si teda priečinky a súbory, ktoré sú pre serverovú časť potrebné:

**app/controllers/** *Controller* triedy, potrebné pre spracovávanie požiadavkou od klienta, sa nachádzajú v tomto priečinku. Jedná o súbory o 3 súbory. Požiadavky týkajúce sa miest sú riešené v kontrolery `pages_controller.rb`. Požiadavky týkajúce sa udalostí rieši kontroler `event_controller.rb`. `application_controller.rb` je posledný kontroler, ktorý obsahuje metódu starajúcu sa o zaznamenanie nových užívateľov a prihlásenie známych užívateľov.

**app/models/** *Model* triedy, ktoré zapuzdrujú objekty v databáze a akcie s nimi spojené, sa nachádzajú v tomto priečinku. Okrem tried pre miesta a udalosti, ktoré sa starajú o ukladanie údajov do databázy, sa tu nachádza aj trieda, ktorej funkciou je komunikácia s Graph API prostredníctvom knižnice Koala (`facebook.rb`).

**app/workers** Obsahuje triedy, ktoré sa využívajú pre asynchrónne spracovanie. Jedná sa hlavne o hromadné získavanie údajov zo siete Facebook a o počítanie váh položiek na základe obsahu.

**config/** Obsahuje konfiguračné súbory aplikácie. Najdôležitejšie z nich sú `database.yml`, v ktorom sú definované nastavenia aplikačnej databázy, prípadne databáz, v prípade zmeny prostredia (prostredia pre vývoj, testovanie alebo produkciu) a súbor `routes.rb`, v ktorom sú definované smerovania REST služieb.

**config/initializers** Zložka obsahuje všetky nastavenia pre potrebné časti aplikácie. Tieto nastavenia sa spúšťajú pri štarte serverovej aplikácie.

**db/** Obsahuje súbor `scheme.rb`, v ktorom sú uložené schémy databázových tabuliek, používaných aplikáciou.

**db/migrate/** Táto zložka obsahuje ruby skripty, kde sú definované tabuľky databázy. Pomocou týchto skriptov a RoR nástroja `rake` je možné vytváranie databázy automatizovať. Migrácie taktiež slúžia na verzovanie databázy.

**lib/** Pomocné moduly sa nachádzajú v tejto zložke. Moduly v aplikácií *Let's Go Out!* počítajú Kosínusovú mieru podobnosti a koeficient TF-IDF.

**Gemfile** Konfiguračný súbor, kde sú definované potrebné gemy, pre správnu funkčnosť aplikácie. Používa sa pri inštalácií aplikácie. Pomocou tohoto súboru sú automaticky nainštalované všetky potrebné gemy pri inštalácií aplikácie. Súbor je uložený v koreňovom adresári.

Pri svojej činnosti využíva serverová časť MySQL databázu, kde ukladá údaje získané zo sociálneho grafu siete Facebook. Štruktúra databázy bola načrtnutá pomocou ER diagramu v kapitole 2.

## 5.2 Získavanie údajov z Facebooku

Neoddeliteľnou súčasťou celej práce sú údaje zo siete Facebook, bez ktorých by práca stratila zmysel. Ich získavanie prebieha v triede `facebook.rb`, ktorá spadá do vrstvy *Model*. Pri dotazovaní sa do siete Facebook používa gem **Koala on Rails** [1] (skr. Koala).

Pomocou knižnice Koala dokáže serverová časť aplikácie zasielať požiadavky na Graph API. Pred samotným používaním musí byť ale aplikácia zaregistrovaná na developerskej stránke Facebooku, kde sú jej vygenerované údaje (`app_id`, `app_secret`). Za použitia vyžiadaného prístupového tokenu môže aplikácia pristupovať ku sociálnemu grafu siete Facebook. Získavanie autorizácie prebieha nasledovne:

```
@oauth = Koala::Facebook::OAuth.new(app_id, app_secret)
@access_token = @oauth.get_app_access_token
@graph = Koala::Facebook::API.new(@access_token)
```

Pomocou premennej `@graph` je následne možné dotazovať sa do sociálneho grafu. Dotazovanie prebieha pomocou funkcie `get_object(string)`, ktorej je potrebné predať ako parameter požiadavku formovanú v tvare `/<uzol>/<hrana>`, kde `<uzol>` je povinné pole, bez ktorého nie je možné navrátiť hodnoty. Návratová hodnota funkcie `get_object(string)` je požadovaný objekt, či pole objektov, vo formáte JSON.

Údaje sú zo siete Facebook získavané dvomi spôsobmi - **automaticky** alebo **na vyžiadanie**. V nasledujúcich podkapitolách sú podrobnejšie popísané obidva spôsoby.

[1] https://github.com/arsduo/koala/wiki/Koala-on-Rails

### 5.2.1 Automatické načítavanie údajov do databázy

Facebook údaje sú v aplikácií automaticky načítavané a ukladané do databázy. Údaje z databázy sú následne použité pri počítaní váh položiek pre odporúčací proces. Ku automatickému načítaniu údajov dochádza v dvoch prípadoch, a to pri počiatočnom načítaní alebo pri pravidelnom dennom načítaní.

Pokiaľ užívateľ spustí aplikáciu po prvýkrát, jeho identifikácia je zaznamenaná do tabuľky `users` a je mu pridelený jednoznačný identifikátor. Vytvorí sa záznam v tabuľke `user_locations` pre jeho aktuálnu lokáciu (pokiaľ je známa) a spustí sa asynchrónne načítanie údajov v užívateľovom okolí (zabezpečenie asynchrónneho behu má na starosti Ruby gem **Sidekiq** [2]). Asynchrónne načítanie je umiestnené v `LoadPlaceEventsWorker` triede. Táto asynchrónna úloha vykonáva 2 činnosti.

1. Zavolanie metódy pre načítanie udalostí a miest podľa zadaných parametrov (geografická šírka, dĺžka a vzdialenosť od daného bodu).

2. Po dokončení predchádzajúceho načítavania sú spustené ďalšie 2 úlohy, opäť asynchrónne, ktoré prepočítavajú váhu jednotlivých slov načítaných udalostí a miest.

Proces samotného načítania údajov prebieha v cykle, pretože s novou verziou Graph API 2.3[3] prišlo aj nové obmedzenie na vrátený počet záznamov na jednu požiadavku. Je teda potrebné v cykle sa opakovane dotazovať na Graph API a prispôsobovať nastavený parameter požiadavky `offset`, ktorá definuje index prvého načítaného prvku z celej sady navrátených záznamov. Nasledujúci skrátený úsek kódu zobrazuje niekoľkonásobné dotazovanie sa na Graph API kvôli spomínanému obmedzeniu:

```
limit = 25
offset = 0
begin
  # From GraphAPI 2.3 - default limit of returned records is set to 25
  # Loop cycles through all the records, continuously querying.
  fb_query_limit = "&limit=#{limit}&offset=#{offset}"
  places = @graph.get_object("search?#{fields}&#{query}#{conditions}#{
      fb_query_limit}")
  ...
  offset += limit
end until places.count < 25
```

Textový reťazec `fb_query_limit` obsahuje parametre, ktoré nastavujú limit a odsadenie sady výsledkov. V prvom cykle má teda tvar `limit=25&offset=0`. Zvyšné parametre vymedzujú užší okruh navrátených hodnôt. Parameter `fields` obsahuje vymedzení polia, ktoré sú pre potreby odporúčania dôležité, aby bola veľkosť navrátených dát čo najmenšia. Udalosti je možné načítať spolu s miestami, v ktorých sa konajú, takže nie je nutné explicitne vyžadovať údaje zvlášť o udalostiach a miestach. Polia, ktoré sú potrebné pre odporúčací systém, sú vypísané v tabuľke 5.1.

Parameter `query` v sebe nesie textový reťazec, ktorý vymedzuje získané položky podľa ich názvu. Tento parameter je ale voliteľný a nie vždy využitý. Posledným parametrom je `condition`, v ktorom sú uložené informácie o type hľadanej položky a o polohe, ktorá má byť prehľadávaná. Obsah tohoto parametru vyzerá nasledovne:

```
type=place&center=<lat>,<lon>&distance=<dist>
```

[2] http://sidekiq.org/
[3] https://developers.facebook.com/docs/apps/changelog

Tabulka 5.1: Načítané detailné informácie, dôležité pre potreby odporúčania

| *Typ položky* | *Vybrané polia* |
|---|---|
| **Miesto** | názov, krátky popis, detailný popis, zoznam kategórií, lokácia, počet označení za obľúbené |
| **Udalosť** | názov, krátky popis, detailný popis, lokácia, čas začiatku udalosti, počet ľudí, ktorý sa zúčastnia |

Po načítaní sú údaje filtrované pomocou definovanej sady kľúčových kategórií, ktoré boli vybraté na základe ich relevantnosti ku aplikácií (vybrané kľúčové slová sú výsledkom filtrovania všetkých nájdených kategórií za pomoci nástroju Graph API Explorer[4]). Výsledné kategórie spolu s ich kľúčovými slovami je možné vidieť v tabuľke 5.2.

Tabulka 5.2: Kategorizácia kľúčových kategórií miest

| *Kategória* | *Kľúčové slová* |
|---|---|
| **Bar** | wine, vineyard, tea, bar, pub, karaoke, cafe, hookah, coffee, cafeteria, beer |
| **Library** | library |
| **Museum** | museum, history, gallery |
| **Music** | club, nightlife, karaoke, dance, concert |
| **Outdoors** | zoo, aquarium, wildlife, park, sightseeing, square |
| **Restaurant** | restaurant, steak, salad, pizza, food, donuts, bagels, cupcake, chicken, wings, grill, burger |
| **Sport** | yoga, pilates, tennis, swimming, pool, sport, golf, skating, gym, archery, gun, range, recreation, fitness |
| **Theatre** | theatre, orchestra, movie |
| **Wellness** | health, salon, spa, beauty, care, massage, acupuncture |

Vyfiltrované miesta sú následne ukladané do databázy, pričom sa spracovávajú aj udalosti, ktoré k nim patria. Vzniká tak vzťah medzi tabuľkou udalostí a miest. Po načítaní neexistuje udalosť, ktorá nemá záznam v tabuľke `places`.

### 5.2.2 Načítavanie údajov na vyžiadanie

Trieda `Facebook` obsahuje niekoľko ďalších metód, ktorými sa dotazuje do sociálneho grafu pomocou Graph API. Medzi tieto metódy patrí získanie podrobných údajov o udalosti alebo mieste, či vyhľadávanie.

**Metóda vyhľadávania** sa volá pri spustení vyhľadávania v aplikácií. Načítanie prebieha podobne ako pri automatickom načítaní dát. Do Graph API požiadavku sú aplikované obmedzenie lokality a kľúčové slová ktoré užívateľ zadal. Výstupné údaje sú následne filtrované sadou kľúčových kategórií, v závislosti na užívateľovom výbere. Rovnako ako pri automatickom načítaní dát sú navrátené polia filtrované len na polia potrebné ku zobrazeniu výsledkov vyhľadávania, avšak sú obohatené o URL profilového obrázku, ktorý je vo výsledkovom liste tiež zobrazený.

[4]Odkaz na Graph API Explorer sa nachádza na stránke https://developers.facebook.com/tools-and-support/.

Po užívateľovom výbere požadovanej nájdenej položky v Android aplikácií sa odosiela požiadavka na server pre **získanie detailných informácií**. Táto metóda následne získa potrebné polia ku zobrazeniu detailných informácií a odošle ich do užívateľovho zariadenia, kde sú informácie vizualizované.

Spomínané metódy načítania dát neukladajú záznamy do databázy, pretože vyhľadávanie sa môže často meniť v závislosti na lokácií, teda nemusia byť relevantným cieľom odporúčania. Avšak **pri ohodnotení sa položka stáva relevantnou**, pretože je považovaná za referenčnú položku pri odporúčaní. Pri prijatí hodnotenia od užívateľa, server použije identifikáciu položky pre získanie potrebných informácií a uloží túto položku do databázy aplikácie. Po úspešnom ukladaní je následne spustený proces počítania váh kľúčových slov, aby položka mohla byť zaradená medzi referenčné položky a použitá v odporúčacom procese.

### 5.2.3 Representational state transfer (REST) služby

Ako už bolo spomenuté, Android aplikácia komunikuje so serverom na báze RESTful služieb. RESTful služby sú natívne podporované v RoR aplikáciách. Povolené prístupové adresy sú definované v smerovacom súbore `routes.rb`, ktorý sa nachádza v priečinku `config` v adresári projektu. V tomto súbore je taktiež možné definovať obmedzenia (napr. povoliť len HTTP GET požiadavky pre danú adresu a pod.). Nasledujúci príklad ukazuje definíciu smerovacieho záznamu REST služby aplikácie *Let's Go Out!*:

```
match 'places/nearby' => 'places#nearby', via: :all
```

Tento záznam definuje REST službu na adrese `places/nearby`. Patričnou akciou, pri prijatí požiadavku na tejto adrese je spustenie metódy `nearby()`, ktorá sa nachádza v controller triede `places_controller.rb`. Tento príklad dokazuje chytrosť RoR pri rozpoznávaní tried na základe pomenovacích konvencii. Nastavenie `via: :all` nastavuje adresu prístupu obidvom HTTP požiadavkám - GET, POST.

Spôsob, ako rozlíšiť smerovanie vo webovej aplikácií na zobrazenie novej stránky a zaslanie REST odpovede, je v tom, čo bude patričná trieda kontroler renderovať. Pri obyčajnej webovej aplikácií, kontroler generuje šablónu, ktorá je následne zobrazená užívateľovi. Pri REST službách, šablóny nie sú renderované, namiesto toho je renderovaný JSON objekt z údajov, ktoré majú byť zaslané.

Definované RESTful služby, pomocou ktorých klientská aplikácia komunikuje so serverom, sú uvedené v tabuľke 5.3.

Tabulka 5.3: Definované RESTful služby pre komunikáciu klienta a serveru

| *Služba* | *Kontroler* | *Popis* |
|---|---|---|
| **search** | places, events | Vyhľadávanie v sociálnom grafe siete Facebook |
| **get** | places, events | Získavanie detailných informácií o udalosti alebo mieste |
| **recommend** | places, events | Získanie odporučených položiek pre daného užívateľa |
| **rate** | places, events | Zaslanie hodnotenia udalosti alebo miesta |
| **login** | application | Vytvorenie alebo prihlásenie užívateľa. Navracia priradenú identifikáciu. |

## 5.3 Obsahové odporúčanie

Ako už bolo spomenuté v podkapitole 5.2.1, po načítaní údajov zo siete Facebook, sú nájdené položky na základe ich obsahu **kódované do formátu TF-IDF**. Toto kódovanie pozostáva z poľa kľúčových slov a ich váhovom ohodnotení. Aby sa vyhlo prepočítavaniu váh v reálnom čase užívateľa, je tento proces vykonávaný asynchrónne. Modul, ktorý sa o počítanie váh stará, sa nazýva *ContentEncoder* a nachádza sa v súbore `lib/content_encoder.rb`.

Pri analýze obsahu položiek nastal **problém podobnosti slov**. Slová sa často líšia ale ich výpovedná hodnota je rovnaká (napr. *konferencia* a *konferencie*). Aplikácia je postavená tak, aby podporovala väčšinu krajín na svete, čo ale znamená obrovské nároky na lexikálne spracovávanie textu. Bol teda vybratý jednoduchší prístup a to hľadanie syntaktickej podobnosti slov. K dosiahnutiu tohoto cieľa je použitý Ruby gem **similar-text** [5], ktorý ponúka jednoduchú metódu `similar(String)`, ktorá vráti percentuálnu zhodu dvoch slov zo syntaktického hladiska.

Proces spracovávania textu prebieha v niekoľkých krokoch. Popis položky je v prvom rade potrebné zbaviť interpunkcie, kde je použitá RoR metóda `transliterate`. Výsledok tejto metódy je názorne zobrazený na nasledujúcom príklad.

```
irb(main):003:0> ActiveSupport::Inflector.transliterate "ľščťžýáíéřôä"
=> "lsctzyaieroa"
```

Následne je celý text prevedený do malých písmen a všetky znaky, okrem písmen a medzier, sú odstránené. Text je rozdelený na slová pomocou medzier medzi nimi, čím sa vytvorí reťazec slov, použitých v texte, ktorý je ale ešte filtrovaný. Odstránia sa všetky slová s dĺžkou nižšou ako 3, kde tieto slová často nemajú v texte kontextuálny význam (tzv. *stop-words*). Taktiež sú ignorované všetky URL, ktoré po spomínaných úpravách vytvoria dlhé slovo bez zmyslu. Algoritmus potom cyklí cez všetky kľúčové slová a počíta ich váhu za použitia vzťahu 3.3.

Výstupom z predchádzajúcej funkcie je pole kľúčových slov s priradenými váhami. Toto pole je na úrovni databáze serializované a ukladané do tabuliek *events* a *places* ako hodnota `keywords_weights`. Ukážka hodnoty tohoto stĺpca vyzerá nasledovne:

```
...
- keyword: kavarny
  weight: 3.5263605246161616
- keyword: centru
  weight: 2.0794415416798357
- keyword: klidna
  weight: 5.484796933490655
- keyword: historicka
  weight: 4.787491742782046
- keyword: party
...
```

Samotné obsahové odporúčanie potom cyklí medzi referenčnými položkami (položky, ktoré užívateľ ohodnotil, či už v rámci obľúbenosti alebo relevantnosti voči jeho záujmom) a porovnáva váhy kľúčových slov s položkami, ktoré užívateľ ešte neohodnotil. Opäť je tu počítaná percentuálna podobnosť slov. Pri zhode viac ako 80% je váha, ako pre referenčné tak aj pre porovnávané kľúčové slovo, vložená do polí.

Pokiaľ polia hodnotení majú po skončení porovnávaní 15 a viac prvkov (spoločných kľúčových slov), je počítaná kosínusová miera pravdepodobnosti za použitia modulu *Co-*

[5] https://github.com/valcker/similar_text-ruby

*sineSimilarityMeasure*, ktorý sa nachádza v súbore `lib/cosine_similarity_measure.rb`. Na základe vzťahu 3.2 sú vypočítané váhy podobnosti jednotlivých položiek. Tieto váhy sú upravené v závislosti na počte spoločných slov, tzn. váha položky je vynásobená počtom spoločných slov, pretože pri minimálnom počte spoločných slov môže nastať vyššia podobnosť, než pri vyššom počte. Výsledné pole položiek je zoradené podľa miery podobnosti a navrátené užívateľovi prostredníctvom RESTful služby, kde sú potom užívateľovi vizualizované.

## 5.4 Kolaboratívne odporúčanie

Odporúčanie na základe hodnotenia prebieha obdobne ako odporúčanie na základe obsahu. Podobnosti hodnotení dvoch položiek sú získavané pomocou kosínusovej miery podobnosti 3.2. Avšak namiesto váh kľúčových slov, sa ako parametre predávajú vektory hodnotení obidvoch položiek od spoločných užívateľov.

Výsledná podobnosť je použitá pri výpočte predpokladaného užívateľovho hodnotenia danej položky, za použitia vzťahu 3.7. Pokiaľ je vypočítaná hodnota vyššia alebo rovná 3 (hodnota 3 je na škále 1 až 5 neutrálna, kde hodnota 2 označuje mierne negatívne hodnotenie, takže nemôže byť do odporúčania zahrnutá), je táto položka zaradená medzi odporučené. Výsledné pole je potom zoradené podľa vypočítaného predpokladu hodnotenia a zaslané na užívateľské zariadenie, kde je patrične vizualizované.

## 5.5 Štruktúra Android aplikácie

Klientská Android aplikácia sa skladá zo série 5 aktivít, ktoré medzi sebou komunikujú a predávajú si kompetencie v závislosti na vybranej činnosti. Komunikácia medzi aktivitami je vyobrazená na obrázku 2.2. Jednotlivé aktivity sú v krátkosti popísané v nasledujúcom prehľade.

**MainActivity.java** Základná aktivita, ktorá začína celú činnosť aplikácie. Jej hlavnými dvomi súčasťami je základné menu a kontajner, ktorý obsahuje `SearchFragment` Android fragment. Tento fragment sa skladá zo série komponent, kde užívateľ zadáva kritéria pre vyhľadávanie. Hlavné menu aplikácie sa plynulou animáciou objavuje z ľavej strany obrazovky. Pri štarte aplikácie sa táto aktivita taktiež stará o získanej aktuálnej polohy užívateľa a jeho prihlásení do systému.

**InterestsActivity.java** Spúšťa sa vždy pri prvom požiadavku o odporučenie udalostí alebo miest, alebo môže byť manuálne spustená zo zoznamu odporučených položiek (kontextové menu aktivity). Táto aktivita obsahuje užívateľské nastavenia pre odporúčania, ktoré sú uložené v objekte `SharedPreferences`. Tento perzistentný objekt je využívaný ako úložisko, ktoré pretrváva aj mimo životný cyklus aplikácie. Pre potreby aplikácie *Let's Go Out* uchováva informácie o identifikácií užívateľa, aktuálnej polohe užívateľa a preferencie, ktoré si užívateľ zvolil aby boli použité v odporúčacom procese

**SearchResultActivity.java** Výsledky, navrátené či už z procesu hľadania alebo odporúčania, sú vizualizované pomocou tejto aktivity. Výsledky zobrazuje vo forme zoznamu so základnými informáciami o položke - obrázok, názov, kategória (miesta), prípadne čas začiatku (udalosti). Aktivita je napísaná univerzálne ako pre vyhľadávanie, tak pre odporúčanie. Na základe akcie, ktorá ju spustila, dynamicky prispôsobuje svoj názov a kontextové menu.

**DetailActivity.java** Obsahuje detailné informácie o aplikácií. Pomocou hviezdicového hodnotenia môže užívateľ ohodnotiť kvalitu miesta, prípadne relevantnosť udalosti k jeho záujmom.

**MapActivity.java** Vizualizácia polohy na mape je vykonávaná v tejto aktivite. Pri spúšťaní očakáva sadu položiek obsahujúce súradnice a názov miesta. Pomocou týchto údajov sú potom vytvorené označenia na mape, spolu s informačným oknom.

Pre vykonávanie svojej činnosti Android aplikácia taktiež využíva externé aplikačné rozhrania od firmy Google, a to konkrétne *Google Places API Web Service* a *Google Maps Android API v2*.

**Google Places API** sa využíva pri zadávaní polohy vo vyhľadávaní. Každý napísaný text je zaslaný na aplikačné rozhranie, ktoré vráti zoznam zodpovedajúcich miest podľa mena. Tieto nájdené položky sú následne zobrazené v menu, ktoré sa automaticky ukáže. Prijaté objekty obsahujú aj jednoznačnú identifikáciu miesta. Táto identifikácia je uložená pri vybratí daného miesta. Na základe identifikácie objektu je po spustení hľadania zaslaná ďalšia požiadavka na toto aplikačné rozhranie, ktoré na základe identifikácie vráti geografickú šírku a výšku vybranej polohy. Tieto údaje sú použité pri vyhľadávaní udalostí a miest.

**Google Maps Android API v2** je aplikačné rozhranie, pomocou ktorého je možné zobrazovať mapy v aplikáciách. Je potrebné mať vygenerovaný kľúč, ktorým je požiadavka validovaná pri žiadosti o získanie mapy. Súbor `AndroidManifest.xml` obsahuje tento kľúč v konfigurácií aplikačného rozhrania.

# Kapitola 6

# Testovanie a nasadenie

Oficiálne a voľne prístupné úložisko a predajné miesto Android aplikácií je **Google Play**[1]. Aplikácia *Let's Go Out!* bola na tomto úložisku zverejnená a je pravidelne spravovaná (QR kód nájdete na obrázku 6.1). Taktiež serverová časť bola nasadená a spustená. Testovanie ale prebiehalo vo väčšine prípadov mimo Google Play. Vytvorený APK archív bol rozdistribuovaný medzi rôzne skupiny ľudí v rozličných oblastiach.

Pre potreby testovania bola vytvorená druhá verzia odporúčacieho systému, pri ktorom sa vzali do úvahy prístupy konkurenčných aplikácií. Myšlienkou je vytvoriť jednoduchý odporúčací systém, predložiť ho užívateľom na testovanie a zozbierať spätnú väzbu od nich. Následne im predložiť pokročilý odporúčací systém, vyvinutý v rámci tejto práce, opäť zozbierať spätnú väzbu a porovnať výsledky.

## 6.1 Testovacia verzia odporúčacieho systému

Položky v tejto verzií sú odporúčané len na základe všeobecnej obľúbenosti na sociálnej sieti Facebook, čo zahŕňa hodnotu položky `likes` pre miesta a hodnotu položky `attending_count` pre udalosti.

Celkový proces takéhoto odporúčania je rádovo rýchlejší avšak nehľadí na užívateľove preferencie. Cieľom testovania je zistiť zaváženie kvality a rýchlosti u užívateľov.

## 6.2 Výsledky testovania

Užívateľom, ktorý testovali aplikáciu bol predložený dotazník, v ktorom mali ohodnotiť svoje skúsenosti s aplikáciou. Dotazník obsahoval nasledujúce otázky:

1. Ako by ste ohodnotili presnosť odporúčania miest?
2. Ako by ste ohodnotili presnosť odporúčania udalostí?
3. Ktorú funkciu najviac využívate?
4. Priemerne ako často aplikáciu využívate?
5. Čo sa Vám v aplikácií páčilo?
6. Naopak, čo sa Vám v aplikácií nepáčilo?

[1] https://play.google.com/store

Obrázek 6.1: QR kód aplikácie *Let's Go Out!*

Na dotazník bolo prijatých 24 odpovedí. Nasledujúca tabuľka zobrazuje počet odpovedí ku otázkam o presnosti odporúčania pre jednotlivé verzie (**Verzia 1** označuje pokročilý odporúčací systém, ktorý je hlavnou časťou tejto práce, **Verzia 2** označuje jednoduchý systém vyvinutý pre potreby testovania).

Tabulka 6.1: Zastúpenie odpovedí respondentov na otázku: Ako by ste ohodnotili presnosť odporúčania miest?

| *Odpoveď* | *Verzia 1* | *Verzia 2* |
|---|---|---|
| **Presne podľa môjho gusta** | 6 | 2 |
| **Len 1 až 2 miesta mi nesadli** | 12 | 7 |
| **Polovica miest bola mimo môjho záujmu** | 4 | 9 |
| **Len 1 až 2 miesta mi sadli** | 2 | 6 |
| **Ani jedno miesto ma neoslovilo** | 0 | 0 |

Tabulka 6.2: Zastúpenie odpovedí respondentov na otázku: Ako by ste ohodnotili presnosť odporúčania udalostí?

| *Odpoveď* | *Verzia 1* | *Verzia 2* |
|---|---|---|
| **Presne podľa môjho gusta** | 11 | 3 |
| **Len 1 až 2 miesta mi nesadli** | 5 | 4 |
| **Polovica miest bola mimo môjho záujmu** | 6 | 9 |
| **Len 1 až 2 miesta mi sadli** | 1 | 5 |
| **Ani jedno miesto ma neoslovilo** | 1 | 3 |

Z tabuliek je zrejmé, že nasadenie pokročilej verzie systému malo vplyv na užívateľskú skúsenosť. Zvýšila sa presnosť odporúčania čo sa odrazilo na hodnotení od užívateľov.

Funkcie, ktoré sú v aplikácií využívané boli tiež ovplyvnené. Na rozdiel od *verzie 2*, kde hlavnou využívanou funkciou bolo vyhľadávanie, vo *verzií 1* je sú naopak najvyužívanejšími práve odporúčacie funkcie (viz. tabuľku 6.3).

Nasadenie pokročilého systému nemalo vplyv na frekvenciu používania aplikácie ako celku. 21% užívateľov priznalo používanie aplikácie na dennej báze, zatiaľ čo 58% aplikáciu využíva maximálne 2x do týždňa. Zvyšných 21% užívateľov priznalo veľmi sporadické až žiadne používanie aplikácie.

Tabulka 6.3: Zastúpenie odpovedí respondentov na otázku: Ktorú funkciu najviac využívate?

| *Odpoveď* | *Verzia 1* | *Verzia 2* |
|---|---|---|
| **Vyhľadávanie** | 6 | 10 |
| **Odporúčanie miest** | 9 | 7 |
| **Odporúčanie udalostí** | 7 | 5 |
| **Nevyužívam** | 2 | 2 |

Na otázky, čo sa užívateľom v aplikácií páčilo a nepáčilo, prišli ako pozitívne tak aj negatívne odpovede. Väčšina užívateľov sa zhodla na tom, že veľkým kladom aplikácie je vizualizácia mapy a rýchlosť vyhľadávania. Myšlienku odporúčania hodnotili užívatelia rôznorodo, keďže sa jedná o subjektívnu otázku. Našli sa totiž odpovede, ako napr. *"Presne sa trafila do môjho vkusu, ďakujem."* ale na druhú stranu prišli aj odpovede ako *"Odporúčanie udalostí sa veľmi netrafilo"*.

# Kapitola 7

# Záver

Cieľom tejto práce bolo vyvinúť Android aplikáciu *Let's Go Out!*, ktorej hlavnou úlohou je odporúčať užívateľovi udalosti a miesta zo sociálnej siete Facebook. Na pozadí aplikácie bol vyvinutý hybridný odporúčací systém, ktorý je tvorený spojením kolaboratívneho a obsahového odporúčacieho prístupu. Odporúčanie prebieha za použitia matematických vzťahov (viz. kapitola 3), pomocou ktorých sa počíta podobnosť položiek v systéme. Na základe tejto podobnosti sú potom položky užívateľovi predložené. Aplikácia bola publikovaná na oficiálnej stránke Android aplikácií Google Play a je voľne dostupná.

Pre testovacie potreby bola vyvinutá špeciálna verzia odporúčacieho systému, ktorá je založená na prístupe konkurenčných aplikácií, a to všeobecnej obľúbenosti položiek v sociálnej sieti Facebook. Skupine testovacích užívateľov bola predložená aplikácia s týmto jednoduchým odporúčacím systémom a následne bola zozbieraná spätná väzba prostredníctvom dotazníkov. Ukázalo sa, že len 37,5% užívateľov bolo spokojných s viac ako polovicou odporučených miest a len 29% bolo spokojných s viac ako polovicou odporučených udalostí. Následne bola užívateľom predložená aplikácia, s implementovaným pokročilým odporúčaním a ich spätná väzba bola opäť zozbieraná v podobe dotazníkov. Pri pokročilom odporúčaní, až 75% užívateľov pri miestach a 67% užívateľov pri udalostiach sa stretlo s maximálne dvomi nepresnými odporúčaniami. Zavedenie pokročilého odporúčania taktiež zvýšilo využívanie odporúčacej funkcionality aplikácie.

Vzhľadom na to, že oblasť odporúčacích systémov sa neustále vyvíja, aplikácia *Let's Go Out!* nesmie zaostať a musí napredovať spolu s technológiami. Naviac v porovnaní s konkurenčnými aplikáciami má nevýhodu v tom, že pre ňu neexistuje webový portál, ktorý ponúka rovnakú funkcionalitu.

Možné rozšírenia aplikácie teda zahŕňajú vytvorenie webového portálu s podporou rovnakej funkcionality ako mobilná aplikácia. Rozšírilo by to základňu a prilákalo viacej užívateľov (napr. tých, ktorý o mobilnej aplikácií nevedia). Ďalším vylepšením, ktoré by zlepšilo funkcionality a rozšírilo dátovú bázu, je napojenie na externé portály udalostí a miest. Aplikácia by tak nebola naviazaná len na údaje zo siete Facebook. Konkurenčné aplikácia taktiež obsahujú možnosť správy vstupeniek na udalosti. Pri snahe prekonať konkurenciu, musí aplikácia ponúkať minimálne rovnakú funkcionalitu než konkurenčné aplikácie. Predmetom ďalšieho vývoja je teda integrácia manažéra vstupeniek na udalosti, aby užívateľ mohol rezervovať, prípadne zakúpiť vstupenky priamo v aplikácií.

Vytvorená aplikácia *Let's Go Out!* nenadväzuje na nijaké aktuálne bakalárske práce, ani externé projekty.

# Literatura


[1] Chakrabarti, S.: *Mining the Web: Discovering Knowledge from Hypertext Data.* Morgan-Kauffman, 2002, ISBN 1-55860-754-4.

[2] Goldberg, D.; Nichols, D.; Oki, B. M.; aj.: Using Collaborative Filtering to Weave an Information Tapestry. *Communications of the ACM - Special issue on information filtering*, ročník 35, č. 12, 1992: s. 61–70, doi:10.1145/138859.138867.

[3] Jannach, D.; Zanker, M.; Felfernig, A.; aj.: *Recommender Systems: An Introduction.* Cambridge University Press, 2011, ISBN 978-0-521-49336-9.

[4] Resnick, P.; Iacovou, N.; Suchak, M.; aj.: GroupLens: An Open Architecture for Collaborative Filtering of Netnews. In *Proceedings of the 1994 ACM Conference on Computer Supported Cooperative Work*, New York, NY, USA: ACM, 1994, ISBN 0-89791-689-1, s. 175–186, doi:10.1145/192844.192905.

[5] Rich, E.: Readings in Intelligent User Interfaces. kapitola User Modeling via Stereotypes, San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1998, ISBN 1-55860-444-8, s. 329–342.

# Příloha A

# Obsah CD

**/src/**
: Zdrojové súbory aplikácie

**/docs/**
: Dokumentácia ku aplikácií

**/latex/**
: Zdrojové súbory technickej správy

**/letsgoout.apk**
: Vygenerovaný inštalačný balíček obsahujúci Android aplikáciu *Let's Go Out!*

**/plagat.pdf**
: Vytvorený plagát

**/technicka-sprava.pdf**
: Text technickej správy