FAKULTA
INFORMAČNÍCH
TECHNOLOGIÍ
ČVUT V PRAZE

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

Název:	Android aplikace na hledání přátel na oběd
Student:	Michal Húževka
Vedoucí:	Ing. Miroslav Balík, Ph.D.
Studijní program:	Informatika
Studijní obor:	Softwarové inženýrství
Katedra:	Katedra softwarového inženýrství
Platnost zadání:	Do konce zimního semestru 2019/20

Pokyny pro vypracování

Navrhněte a implementujte aplikaci pro operační systém Android, která umožní hledání přátel na oběd. Prozkoumejte stávající aplikace na platformě Android s podobným účelem a určete požadavky na aplikaci. Při návrhu a implementaci použijte techniky softwarového inženýrství.

Aplikace bude umožňovat vytvářet uživatelské profily a po vyplnění času, místa a kritérií k vyhledání přítele nalezne vhodného partnera na společný oběd.
Dále bude možné procházet seznam lidí a pozvat některého na oběd. Implementujte komunikaci a ukládání dat na server pomocí webové služby využívající REST API. Proveďte testování aplikace včetně UI a napište seznam možných vylepšení.

Seznam odborné literatury

Dodá vedoucí práce.

Ing. Michal Valenta, Ph.D.
vedoucí katedry

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

V Praze dne 19. března 2018

FAKULTA
INFORMAČNÍCH
TECHNOLOGIÍ
ČVUT V PRAZE

Bakalářská práce

Android aplikace na hledání přátel na oběd

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

14. května 2018

Poděkování

Chtěl bych poděkovat vedoucímu mé práce Ing. Miroslavu Balíkovi Ph.D. za vedení práce, předávání zkušeností a cenné rady. Také bych rád poděkoval své rodině za pomoc a podporu při studiu.

Prohlášení

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

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

V Praze dne 14. května 2018 .....

Odkaz na tuto práci

Húževka, Michal. *Android aplikace na hledání přátel na oběd.* Bakalářská práce. Praha: České vysoké učení technické v Praze, Fakulta informačních technologií, 2018.

Abstrakt

Tato práce se zabývá analýzou, návrhem a implementací mobilní aplikace pro OS Android. Aplikace slouží ke vzájemnému hledání přátel, se kterými umožňuje naplánovat společný oběd. Umožňuje vytvářet událost oběda a zasílat pozvánky. Data jsou ukládána do databáze přes webový server s rozhraním REST.

Práce dopodrobna objasňuje implementační postupy použité při programování aplikace a popisuje vývoj pro platformu Android. Text je doplněn snímky obrazovek aplikace. Dokument zahrnuje analýzu a návrh pomocí metod softwarového inženýrství, včetně analýzy funkčních a nefunkčních požadavků, návrhu wireframe a modelování případů užití. Analyzuje také již existující aplikace s podobným účelem.

Klíčová slova hledání přátel, OS Android, Java, REST API, Google Places

Abstract

This study deals with the analysis, design and implementation of a mobile application for Android OS. Application serves for searching friends with whom it provides an opportunity to schedule a common lunch. Allows to create a

lunch event and send invitations. Data is stored in a database via a REST web server.

This work explains in detail the implementation procedures used in programming of this application and describes development for the Android platform. The text is complemented by screenshots of the application. This document includes analysis and design using software engineering methods, including analysis of functional and non-functional requirements, wireframe design, and case modeling. It also analyzes existing applications with a similar purpose.

Keywords friend search, OS Android, Java, REST API, Google Places

Obsah

Seznam obrázků

Úvod

V dnešní digitální době lidé často místo společnosti druhých dávají přednost displeji mobilního telefonu. Přitom by stačilo jen někoho oslovit. Někoho, kdo se pravděpodobně také jen stydí, nebo si namlouvá, že nemá čas. K oslovení lidí má sloužit právě tato aplikace. Uživatel zadá kam jde na oběd, kdy tam jde a základní parametry s kým by chtěl jít, jako věk, pohlaví nebo záliby. Aplikace potom najde vhodného člověka pro společný oběd.

Pro implementaci jsem zvolil stále populárnější mobilní platformu OS Android. Do své práce jsem zahrnul analýzu, návrh a implementaci podle vzorů softwarového inženýrství. Následuje testování aplikace včetně testování uživatelského rozhraní.

Tato práce pokračuje v následující struktuře: v druhé části se nejdříve věnuji analýze již existujících aplikací a analýze požadavků, v části tři následuje návrh případů užití a návrh uživatelského rozhraní v podobě wireframe. Ve čtvrté části popisuji reálnou implementaci, která se bude zabývat i popisem vývojové platformy pro OS Android a v poslední, páté části píši o průběhu a výsledcích testování.

Kapitola 1

Cíl práce

Cílem praktické části práce je vytvořit aplikaci, která pomůže uživateli nalézt partnera pro společný oběd. Aplikace bude fungovat na mobilních telefonech s operačním systémem Android.

Umožní přihlášení uživatele pomocí Google nebo Facebook účtu, nebo přihlášení na webový server, kde si uživatel předtím vytvořil svůj účet. Webový server bude sloužit pro komunikaci s databází Oracle, na kterou se budou ukládat všechna uživatelská data. Celá aplikace by měla být implementována tak, aby nevytvářela zbytečné „memory leaky“, tedy úniky paměti, kdy část dat zůstane v paměti a nelze ji uvolnit. Tento problém je u Android aplikací poměrně běžný, lze mu však předejít používáním správných implementačních postupů.

Cílem rešeršní části je popis vývojové platformy Android a vysvětlení implementačních postupů a návrhových vzorů použitých v aplikaci, včetně vysvětlení principu fungování, případně uvedení důvodů pro použití zvoleného postupu. Text této práce je vhodným studijním materiálem pro kohokoli, kdo plánuje naprogramovat Android aplikaci podobného rozsahu. Obsahem je také analýza a návrh pomocí technik softwarového inženýrství. Na závěr jsem zhodnotil výsledek mé práce a navrhl další možná vylepšení.

Kapitola 2

Analýza

Nejprve je nutné najít a zanalyzovat již existující aplikace, které mají podobný účel, aby se neopakovala implementace toho, co už někdo naprogramoval přede mnou. Poté se zabývám analýzou funkčních a nefunkčních požadavků, důležitého bodu v rámci vývoje aplikace podle metod softwarového inženýrství.

2.1 Analýza existujících aplikací

Aplikace lze stahovat z oficiálního trhu aplikací Google Play, kde jsou dostupné všechny publikované programy schválené Googlem. Z jiných zdrojů lze aplikace instalovat pouze po povolení instalace z neznámých zdrojů v nastavení telefonu. Tato volba může být značně riskantní kvůli možné nákaze systému malwarem, proto jsem se rozhodl analyzovat aplikace pouze z oficiálního úložiště Google Play. Zde jsou k nalezení další 4 aplikace, které mají také za cíl sdružování lidí k jídlu nebo schůzkám.

2.1.1 Analýza aplikací

1 Lunch Buddy

První nalezenou aplikací je Lunch Buddy. Umožňuje vytváření eventů - událostí, které představují oběd, večeři apod. Při vytváření události lze zadat název restaurace, datum, popis a lze pozvat přátele, které si předtím uživatel přidal pomocí vyhledávání, nebo ze seznamu uživatelů v blízkosti „People nearby“. Nejbližší eventy pak lze zobrazit a přihlásit se na ně. Aplikace vyžaduje LinkedIn účet pro přihlášení.

V aplikaci vše funguje jak má a je dobře graficky a uživatelsky zpracovaná, někomu by ovšem mohla vadit nutnost účtu LinkedIn.

2 amHappy

Ve druhé zkoumané aplikaci bohužel nefunguje vytváření účtů ani přihlašování pomocí Facebook/Google účtu, tudíž nebylo možné ji plně vyzkoušet. Přihlašování bylo testováno na dvou mobilních telefonech s rozdílnými verzemi OS Android a na virtuálním zařízení v emulátoru. Přesto aplikace nefungovala na žádném z těchto zařízení. V jejím popisu na Google Play se píše, že umožňuje založit událost a přizvat lidi, hlasovat o místě a čase události a vyhledávat události blízko současné pozici. Neumožňuje však uživateli hledat a poznávat nové lidi.

3 Lets Go Lunch

Zde je situace podobná jako v předchozí aplikaci. Po zapnutí se objeví prázdná obrazovka a aplikaci není možné ovládat. Na Google Play vývojář uvádí v popisu, že aplikace zobrazuje přehled kam jdou na oběd kolegové a uživatel má možnost přidat se. Lze také založit vlastní událost. Opět ale nelze přes tuto aplikaci hledat nové lidi.

4 FreeForLunch

Poslední nalezenou aplikací je FreeForLunch. Vyžaduje přihlášení pomocí účtu Facebook/Twitter/Vkontakte nebo pomocí čísla mobilního telefonu. Po pokusu o přihlášení na telefonu připojeném k internetu se objeví hláška „Couldn't establish connection to server“. Přihlašování bylo zkoušeno několikrát po sobě v různých dnech, aby byla vyloučena možnost krátkodobého výpadku služby. Aplikace tedy má s největší pravděpodobností nefunkční serverovou stranu. V popisu na Google Play se uvádí, že aplikace má fungovat podobně jako aplikace LunchFriends implementovaná v této práci - uživatel vybere čas a místo a na mapě se zobrazí ostatní uživatelé, které lze zvát na oběd a poznávat tak nové přátele.

2.1.2 Souhrn

Prozkoumání dostupných aplikací k plánování schůzek na oběd přineslo důležité poznatky. Nefunkčnost některých programů je pravděpodobně způsobená tím, že autor přestal produkt udržovat kvůli jeho nepopularitě. Pokud bychom tedy chtěli publikovat aplikaci vyvíjenou v rámci této práce na trh Google Play, museli bychom naše řešení propagovat a investovat do reklamy, abychom aplikaci zpopularizovali.

Výhodou oproti již naprogramovaným řešením je, že tato aplikace umožní uživateli přihlášení nejen pomocí Google a Facebook účtů, ale i pomocí účtu v rámci aplikace, který je možné vytvořit na úvodní stránce tlačítkem pro registraci „Register“. Nebude tak pro uživatele nutnost vlastnit jakýkoliv účet mimo aplikaci. Další výhodou oproti analyzovaným programům je možnost

hledat skrze aplikaci nové přátele. Pro české uživatele je výhodou kompletní překlad do češtiny. Nefunkčnost tří ze čtyř aplikací navíc vyřazuje značnou část konkurence.

2.2 Analýza požadavků

2.2.1 Funkční požadavky

1. Aplikace bude umožňovat registraci nových uživatelů.
2. Aplikace umožní přihlášení pomocí účtů Google a Facebook nebo pomocí registrovaného účtu.
3. Po zadání kritérií: věkové rozpětí, pohlaví, záliby, rozpětí času a místo oběda program najde seznam lidí splňujících zadaná kritéria.
4. Program umožní procházet seznam osob vyhovujících kritériím a některého vybrat.
5. Po výběru člověka ze seznamu přijde dotyčnému skrze aplikaci pozvánka na oběd spolu s informacemi o zasilateli a s možností přijmout nebo odmítnout.
6. Uživateli se ukládá historie lidí, s kterými již šel na oběd.
7. Aplikace bude ukládat všechna uživatelská data na databázový server.

2.2.2 Nefunkční požadavky

1. Aplikace bude fungovat na OS Android ve verzi 4.2 nebo vyšší.
2. Jako úložiště dat bude aplikace využívat databázový server Oracle.
3. Jazykem implementace bude Java a grafické layouty budou vytvořeny pomocí textových XML souborů a pomocí návrhového prostředí v programu Android studio.
4. Aplikace bude využívat programovacích technik zabraňujících memory leakům.
5. Pro komunikaci se serverem bude aplikace používat jeho REST webové rozhraní.

Kapitola 3

Návrh

V této kapitole se zabývám modelováním případů užití (use-case, UC), abych dokázal vystihnout a správně pochopit způsoby interakce uživatele s touto aplikací. Následuje návrh obrazovek aplikace (Wireframe), který slouží k rozvržení jednotlivých prvků na obrazovkách aplikace a umožňuje tak lepší představu, jak aplikace bude vypadat po implementaci a jak se mezi sebou budou jednotlivá okna přepínat.

3.1 Modelování případů užití

1 UC1 Vyhledat lidi

1. Tento případ užití začíná, když uživatel ťukne na tlačítko „Search people“.
2. Na následující obrazovce uživatel zadá věk, pohlaví a koníčky (hobbies) požadované osoby.
3. Na další obrazovce zadá místo a čas, kam a kdy by chtěl jít na oběd.
4. Uživatel zadá maximální vzdálenost od místa, kam by byl ochotný jít na oběd a rovněž maximální časový rozdíl.
5. Po odeslání dat server odpoví seznamem lidí vyhovujících vyhledávacím kritériím.

2 UC2 Pozvat nalezeného člověka na oběd

1. Tento případ užití začíná, když uživatel vybere někoho ze seznamu lidí nalezených vyhledáváním.
2. Cílový uživatel obdrží pozvánku s informacemi o uživateli, který ho pozval.

3. Pozvaný uživatel souhlasí s pozvánkou. Pokud nesouhlasí, obdrží o tom druhý uživatel zprávu a tento případ užití končí.

4. Uživatelé na sebe navzájem obdrží kontaktní informace, aby případně mohli dohodnout jiný čas nebo místo oběda.

3 UC3 Vybrat člověka ze seznamu uživatelů

1. Tento případ užití začíná, když uživatel ťukne na tlačítko „Browse all people“.

2. Uživatel vybere někoho ze seznamu všech uživatelů, kteří jdou na oběd.

3. Cílový uživatel obdrží pozvánku s jeho místem a časem oběda a s osobními údaji uživatele, který ho pozval.

4. Pozvaný uživatel souhlasí s pozvánkou. Pokud nesouhlasí, obdrží o tom druhý uživatel zprávu a tento případ užití končí.

5. Uživatelé na sebe navzájem obdrží kontaktní informace, aby případně mohli dohodnout jiný čas nebo místo oběda.

4 UC4 Procházet historii obědů

1. Tento případ užití začíná, když uživatel ťukne na tlačítko „Lunch history“.

2. Program zobrazí seznam všech obědů, na kterých uživatel byl v minulosti.

3. Uživatel vybere jednu položku ze seznamu.

4. Zobrazí se detailní informace o daném obědu, včetně informací o druhé osobě.

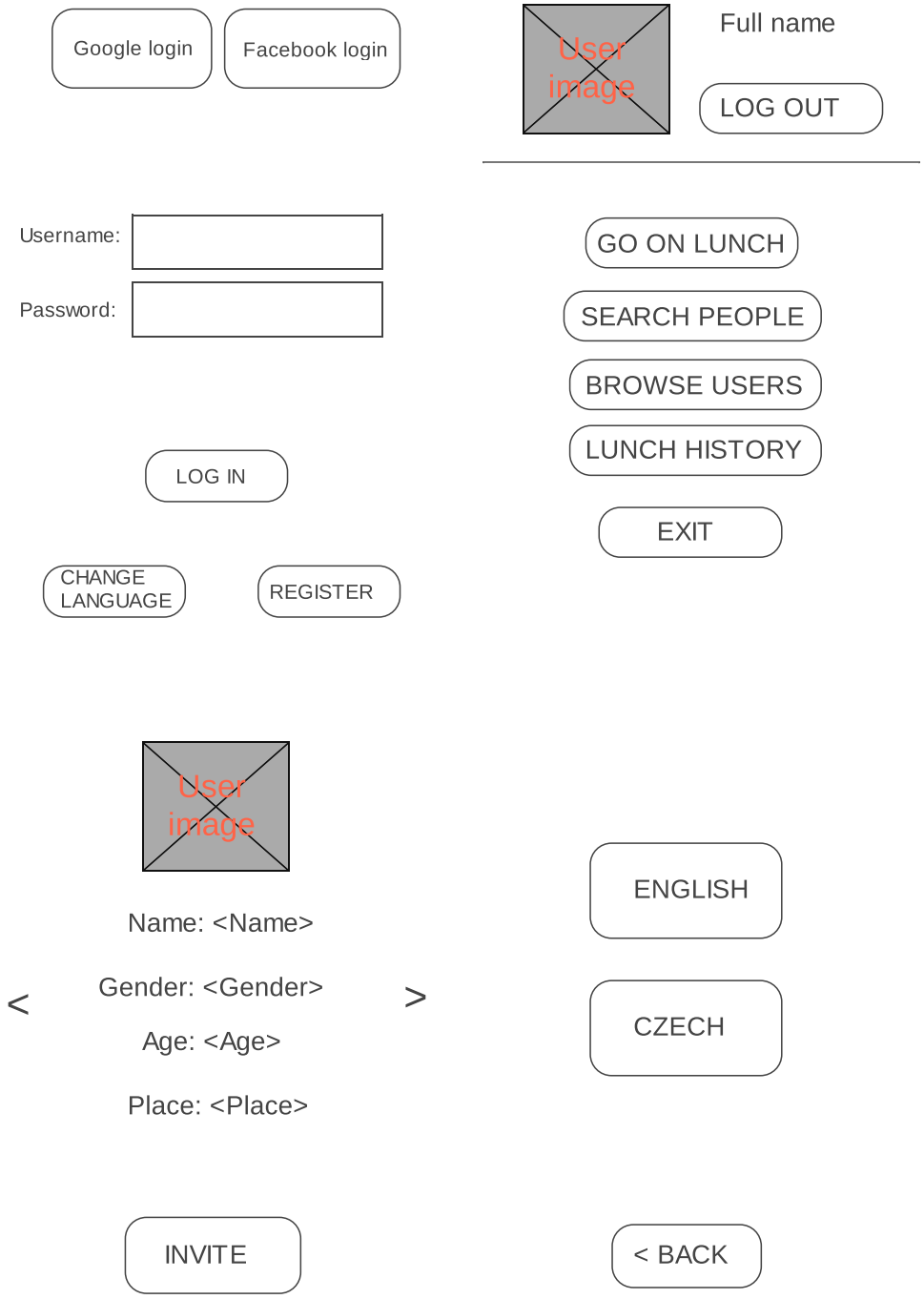
3.2 Wireframe

Wireframe, v překladu drátěný model, je návrh všech obrazovek aplikace, na kterém je vyobrazené rozmístění všech funkčních elementů. Slouží k vyobrazení prvků na obrazovce a jejich funkcí. Nejde naopak o grafický návrh - neobsahuje obrázky, pouze čáry a text [1]. V mém návrhu jsou vyobrazená rozložení čtyř aktivit (viz 1) aplikace. Výsledek lze vidět na obrázku 3.1, kde jsou (postupně zleva doprava, seshora dolů) aktivity:

- `MainActivity` - přihlašovací aktivita, spouští se při startu aplikace.
- `HomeActivity` - domovská stránka aplikace, kde je vidět jméno aktuálně přihlášeného uživatele, jeho profilový obrázek, tlačítko „Log out“ pro odhlášení a hlavní menu.
- `FriendResultsActivity` - aktivita s vyobrazením jednoho člověka, spouští se jako výsledek vyhledávání tlačítkem „Search people“ nebo „Browse all people“ z domovské stránky.
- `ChangeLanguageActivity` - obrazovka pro změnu jazyka, na výběr je angličtina a čestina.

Schéma přepínání mezi aktivitami je následující:
Po ťuknutí na tlačítko „Log in“ v přihlašovací aktivitě (nebo po přihlášení pomocí Google/Facebook účtu) se ověří přihlašovací údaje a pokud vše souhlasí, přepne se aplikace do domovské stránky `HomeActivity`. Do třetí aktivity `FriendResultsActivity` se lze dostat tak, že uživatel vyhledá přátele pomocí tlačítka „Search people“ z domovské aktivity (mezi těmito budou obrazovky pro zadávání vyhledávacích kritérií), nebo pomocí tlačítka „Browse all people“ rovněž z domovské aktivity (procházení všech uživatelů bez filtrování). Do aktivity `ChangeLanguageActivity` se lze přepnout tlačítkem „Change language“ z přihlašovací obrazovky.

Finální reálný vzhled se povedlo ve výsledku rozložit přibližně stejně jako v tomto návrhu.



Obrázek 3.1: Wireframe aplikace

Kapitola 4

Implementace

Tato kapitola je v úvodu věnována operačnímu systému Android a vývoji pro platformu Android. Popisuji důležité součásti mobilních aplikací a postup při vypuštění aplikace na trh. Poté se věnuji popisu implementace aplikace vyvíjené v rámci této práce. Zde informuji o použitých metodách, o správných a špatných postupech a variantách implementace.

4.1 Operační systém Android

Android je nejrozšířenější operační systém pro mobilní telefony. Je založený na jádře Linuxu a je dostupný jako otevřený software (open source). Jeho vývoj vede firma Google pod hlavičkou konsorcia firem Open Handset Alliance. První oficiální verze 1.0 byla vydaná v roce 2008. Verze Androidu se kromě názvů také číslují kladnými čísly od 1, tzv. API level. Zatím nejvyšší verzí je Android 8.1 Oreo (API level 27), vydaná 5. prosince 2017. Nyní se již připravuje nová verze 9 Android P která už má vydaný první vývojářský náhled. OS Android neběží pouze na mobilních telefonech, využívají ho i některé chytré hodinky (v podobě Wear OS) nebo televize (Android TV). [2]

4.1.1 Vývoj pro OS Android

Hlavním jazykem pro vývoj Android aplikací je Java obohacená o Android SDK (Software Development Kit). Nedávno byl jako sekundární oficiální jazyk představen Kotlin, který je podobný Javě, ale má některé zlepšující funkce a lze ho používat přímo současně s Javou (volat v něm metody z Javy a naopak). Kód místo Java Virtual Machine běží na Dalvik Virtual Machine, který je postupně nahrazován jeho následníkem Android Runtime.

Ačkoliv lze Android aplikace vyvíjet i v jiných vývojových prostředích, je programování v nativním IDE Android studio na platformě IntelliJ přirozeně nejvhodnější a uživatelsky nejpřívětivější k vytváření Android aplikací. Nabízí klasické funkcionality moderních IDE jako např. šikovné klávesové zkratky,

podrobné a rychlé hledání řetězců nebo regulárních výrazů, generování metod, refaktoring v podobě přejmenování souborů nebo názvů tříd a proměnných včetně všech jejich výskytů v projektu, vytváření souborů podle šablon, debugger, profiler, spouštění testů, historie změn souborů, integrovaná podpora VCS (Subversion, Git, Mercurial, CVS), ovládání sestavovacího nástroje Gradle nebo např. inspekce kódu na možné bugy. Kromě toho nabízí dodatečné funkce určené speciálně pro Android platformu zahrnující emulátor (virtuální zařízení s operačním systémem Android), na kterém lze zkoušet aplikace bez potřeby reálného zařízení, zobrazování struktury Android projektu (manifest, Java kód, zdroje - obrázky, XML soubory, konstanty), grafický i textový editor XML layoutů a další.

4.1.2 Základní prvky aplikace

Hlavní stavební prvky aplikace jsou tzv. komponenty - tvoří obrazovky a určují chování aplikace. Mají své životní cykly a specifické způsoby programování, které je potřeba si před implementací prostudovat.

1 Aktivity a layout soubory

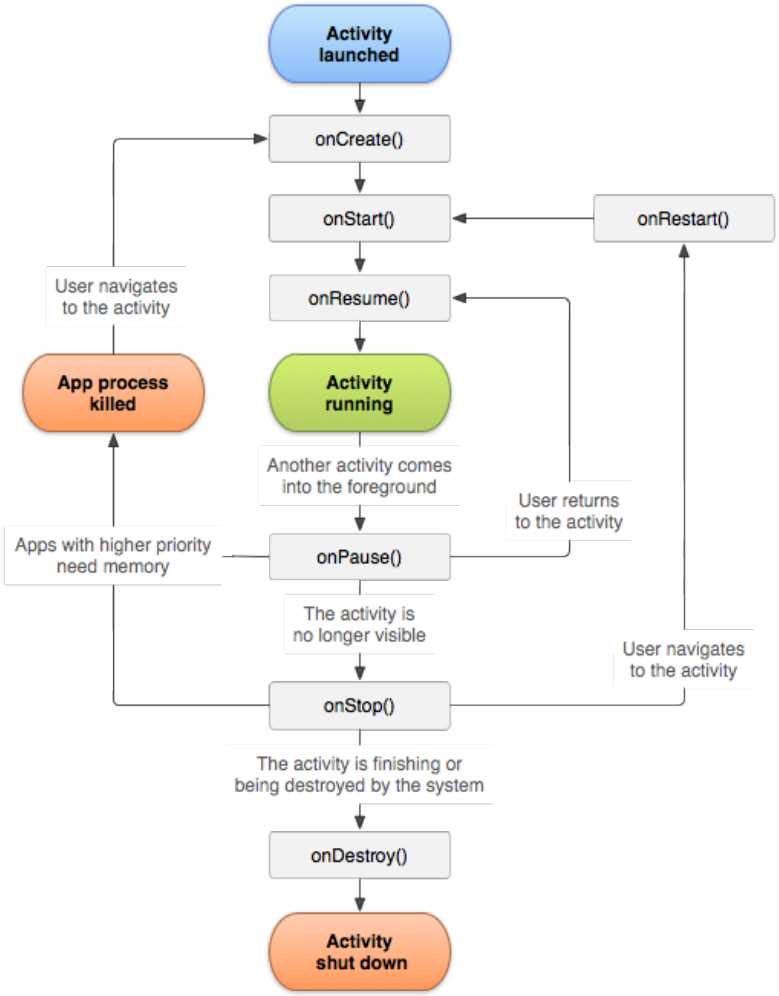
Aktivita (`Activity`) je hlavní prvek aplikace. Jde o jednu obrazovku, na které jsou umístěné elementy UI, tzv. widgety - tlačítka, texty, obrázky, přepínače. Každá aktivita má svůj vzhled a obsah zapsaný v XML souboru, kde jsou všechny widgety na této aktivitě umístěné do kontejneru `Layout` (RelativeLayout, ConstraintLayout, LinearLayout). Životní cyklus aktivity lze vidět na obrázku 4.1.

Widgety a layouty jsou souhrnně nazývány `Views`. Jejich vzhled, umístění a chování lze měnit pomocí atributů. Vlastnosti těchto `Views` lze určovat i za běhu aplikace přímo v kódu. K tomu slouží metoda `findViewById(int id)` která vrátí objekt `View`, ten je možné přetypovat a dál s ním pracovat, například u tlačítka `Button` lze měnit jeho popis pomocí `button.setText(String s)`. Často se používá metoda `view.setOnClickListener(OnClickListener l)`, kde se zaregistruje nový objekt `OnClickListener` s přepsanou metodou `onClick`, která se vykoná, když na daný `View` uživatel ťukne.

Dále mohou být v aktivitě umístěné fragmenty, které se chovají jako část aktivity a umožňují například lépe zobrazit obsah, když je k dispozici zařízení s větší obrazovkou, jako třeba tablet. Fragmenty se pak zobrazují např. najednou vedle sebe, místo nad sebou s nutností posouvat pohled dolů na druhý fragment.

2 Service

`Service` je služba, tj. komponenta aplikace, která běží na pozadí a nejčastěji vykonává nějaké dlouhotrvající činnosti, jako například přehrávání hudby nebo



Obrázek 4.1: Životní cyklus aktivity

stahování dat. Služby lze spouštět a zastavovat z jiných komponent, např. aktivit.

3 Content provider

`ContentProvider` je třída, která zprostředkovává přístup k datům, nejčastěji uloženým v databázi. Lze ho využívat i v jiných aplikacích a získávat tak potřebná data. Existují platformní content providery Calendar a Contacts provider, které poskytují události uživatele z kalendáře a kontakty v mobilním telefonu. Data lze načítat přes `ContentResolver`, nebo do seznamů (`ListView`) pomocí `CursorAdapter`, nebo asynchronně na pozadí pomocí `CursorLoader`.

4 Broadcast receiver

`BroadcastReceiver` je další důležitá komponenta aplikace, která umožňuje přijímat zprávy (broadcasty). Tyto zprávy jsou posílány, když se stane nějaká událost, např. systém se nabootoval, zařízení se začalo nabíjet. Aplikace také mohou posílat své vlastní broadcasty. Systém je rozesílá všem aplikacím, které se zaregistrovaly k jejich přijetí. To lze udělat v kódu nebo v Android manifestu pomocí elementu `<receiver>`.

5 Android manifest

`AndroidManifest.xml` je soubor v projektu, který slouží k základní konfiguraci aplikace. Jsou v něm zapsané všechny pravomoce (permissions) aplikace - například vypnutí/zapnutí Wifi nebo používání čtečky otisků prstů. Dále jsou v tomto souboru vyjmenované všechny aktivity aplikace, služby, content providery, broadcast receivery, také název Java balíčku a různá metadata.

6 Resources

Kromě již zmíněných layout souborů aktivit patří ke zdrojům (resources) i obrázky, ikony aplikace (které mají speciální verze podle rozlišení displeje), pojmenování HTML barev v `colors.xml`, styly v `styles.xml` a řetězcové konstanty v `strings.xml`.

7 build.gradle

`build.gradle` je konfigurační soubor pro sestavovací systém `Gradle`. Projekt obsahuje typicky 2 soubory, jeden pro celý projekt (všechny moduly) a potom jeden pro každý modul (který je typicky jen jeden). V souboru je seznam repozitářů a „dependencies“ - závislosti projektu na knihovnách, které jsou staženy z repozitářů a použity při sestavování. Projektový modul obsahuje v bloku buildscript závislosti pro sestavení androidích aplikací obecně - „android build tools“ a v bloku allprojects obsahuje závislosti společné pro všechny moduly. `build.gradle` pro každý modul pak obsahuje závislosti pro daný modul a kromě toho také blok android a v něm několik důležitých vlastností:

- `minSdkVersion` - určuje API level - minimální verzi OS Android na zařízení nutnou pro běh této aplikace. Nižší verze nebudou podporovány.
- `targetSdkVersion` - cílový API level - označuje vůči které verzi SDK byla aplikace testována a funguje bez problémů. Pokud ji spustíme na novějším androidu, mohou se spustit některá opatření pro zpětnou kompatibilitu a aplikace se bude chovat jako kdyby běžela na zařízení s API levelem rovným `targetSdkVersion`, aby se zajistilo, že bude fungovat tak, jak se očekává.

- `compileSdkVersion` - specifikuje, jaký API level použije gradle pro kompilaci aplikace. Aplikace potom může využívat funkce z tohoto API nebo z nižších, použitím funkce z vyššího API dojde k chybě při synchronizaci projektu pomocí gradle.
- `applicationId` - unikátní identifikátor aplikace pro publikování (jednoznačně identifikuje aplikaci v obchodě Google Play), většinou se používá název Java balíčku.
- `versionCode` - kladné celé číslo, které specifikuje verzi aplikace. Používá se k odlišení verzí této aplikace na Google Play. Při vytvoření nové verze se vždy používá následující celé číslo.
- `versionName` - definuje název verze, například `3.0`.

8 Intent

Intent (záměr) je způsob jak doručit zprávu a požadovat akci po jiné komponentě. Dělí se na 2 typy: explicitní a implicitní [3]. U explicitních se konkrétně specifikuje komponenta, která má tento záměr obdržet, zatímco u implicitních se pouze napíše akce, která se má provést a systém najde všechny aplikace v zařízení, které mají v Android manifestu deklarovaný filtr pro daný záměr (`intent filter`). Pokud záměr souhlasí s filtrem, systém spustí komponentu vyhledané aplikace a doručí jí tento záměr. Pokud je k dispozici více aplikací se stejným filtrem, systém zobrazí dialog a nechá uživatele si vybrat.

4.1.3 Publikování aplikace a obchod Google Play

Ve chvíli, kdy je aplikace hotová a chceme ji publikovat na Google Play, je nutné ji sestavit, zabalit do archivu s příponou .apk a elektronicky podepsat. Je tedy potřeba vygenerovat si veřejný a privátní klíč pro autentizaci. Vše je dopodrobna popsané na oficiálních stránkách androidu v nápovědě pro vývojáře: `https://developer.android.com/studio/publish/app-signing.html`.

Finální sestavení se provádí pomocí gradle buildu s typem `release`. Vývojář si musí založit u Googlu svůj vývojářský účet s poplatkem $25, jeho šifrovací klíč je potom svázaný s tímto účtem. Podpisem se zajistí, že aplikace v Google Play opravdu pochází od správného vývojáře a že nikdo jiný nemůže vydat falešnou verzi aplikace, protože podpis jeho privátním klíčem by nesouhlasil s veřejným klíčem a instalace takové aplikace by skončila s chybou.

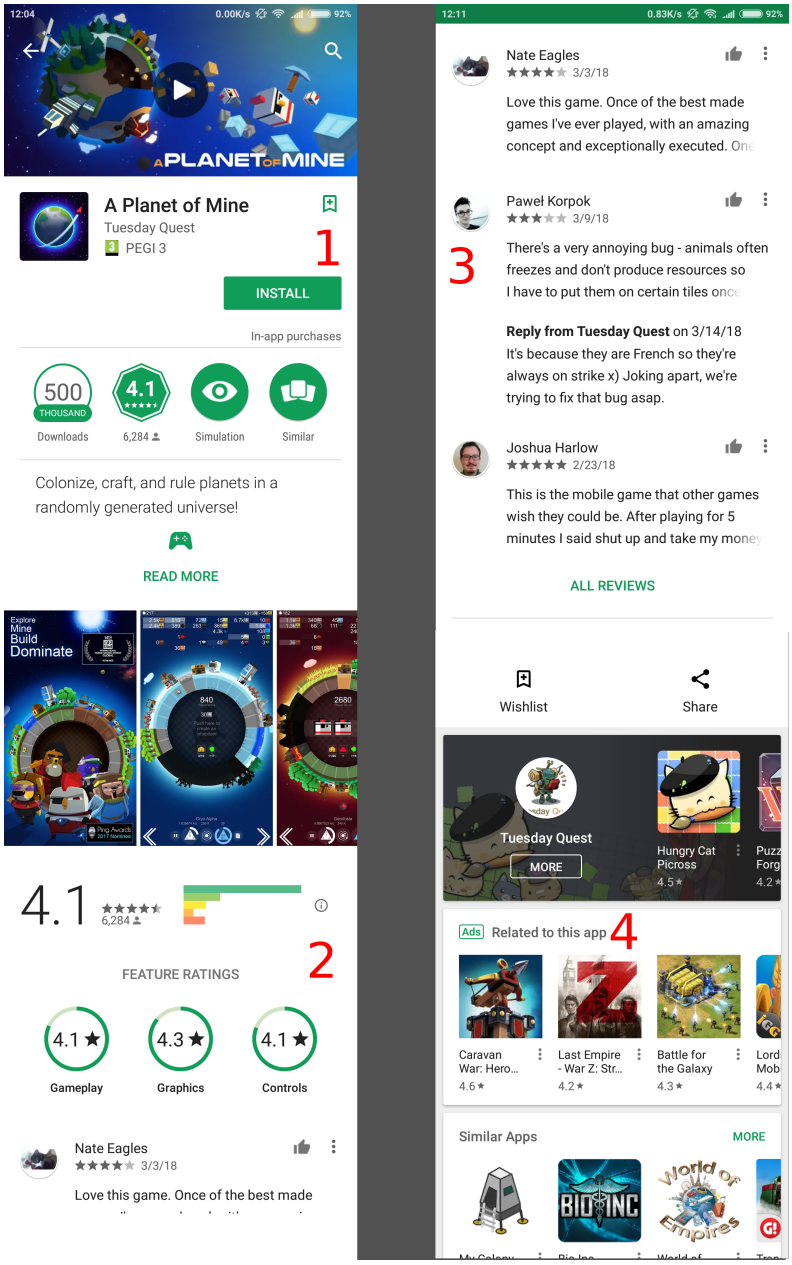
Jakmile je aplikace nahraná do Google Play, je volně přístupná ke stažení každému, kdo má zařízení s OS Android a splňuje určité požadavky. To jsou:

1. minimální verze OS Android (`minSdkVersion`),
2. technické parametry zařízení jako velikost obrazovky, výkon, paměť,

3. lokalizace - vývojář může omezit stažení aplikace jen na určité země.

Stažení aplikace lze zpoplatnit. Výši si určuje sám vývojář, avšak měla by být odpovídající produktu, jinak by se uživatelé přikláněli ke stažení nějaké alternativní aplikace. Výše poplatku může být různá, například aplikace pro GPS navigaci mohou stát až dva tisíce korun, mobilní hry stojí většinou jen kolem 50,- Kč a některé lepší do 200,- Kč. Druhým možným způsobem zpoplatnění jsou tzv. In-app purchases, neboli nakupování produktů v aplikaci. Příkladem může být virtuální měna v mobilní hře nebo přídavné mapy do GPS navigace. Další možností výdělku jsou reklamy v aplikaci (Google AdMob), nebo přístup k obsahu přes pravidelné placené odběry (subscriptions). Častou praktikou jsou reklamy v aplikaci s možností zaplatit si za placenou verzi aplikace, která už je bez reklam. Z vydělaných peněz si Google účtuje 30% poplatek, vývojář tedy nakonec dostane pouze 70%.

Při publikování aplikace na Google Play je vhodné přidat k aplikaci poutavé obrázky, screenshoty nebo videa, které zaujmou a přilákají potenciální uživatele. Na obrázku 4.2 lze vidět stránku aplikace `A Planet of Mine`. Stránka obsahuje ukázkové video (nahoře na obrazovce 1), tlačítko pro instalaci `INSTALL`, informaci o tom jestli aplikace obsahuje In-app nákupy, informace o počtu stažení, o hodnocení aplikace a typ aplikace. Na obrazovce 2 jsou další důležitou součástí stránky popis aplikace a screenshoty (příp. obrázky). Následuje hodnocení a komentáře uživatelů, kteří si tuto aplikaci stáhli a chtěli ji ohodnotit (obrazovka 3). A konečně, na obrazovce 4, seznam dalších aplikací od tohoto vývojáře a seznam podobných aplikací.



Obrázek 4.2: Google Play stránka aplikace

4.2 Implementace aplikace LunchFriends

V této sekci se věnuji samotné implementaci aplikace. Hlavním obsahem je popis použitých programovacích vzorů a tříd. Zabývám se alternativními možnostmi implementace a argumentuji pro použité metody.

4.2.1 Způsob implementace

Aplikaci jsem programoval ve vývojovém prostředí Android studio a zkoušel na reálném zařízení Xiaomi Redmi Note 4 s verzí OS Android 7.0. Ke zpřístupnění možnosti vyzkoušet aplikaci na reálném zařízení se musí povolit volba USB debugging v nastavení telefonu. Dále jsem aplikaci zkoušel spouštět i na jiných zařízeních, abych vyzkoušel, jak je aplikace kompatibilní. To byly Samsung Galaxy J5 s verzí Android 6.0 a virtuální zařízení v emulátoru s verzí Android 7.1. Ve finálním stádiu aplikace fungovala dobře na všech zařízeních, bylo nutno opravit několik chyb týkajících se rozmístění prvků v Layout souborech.

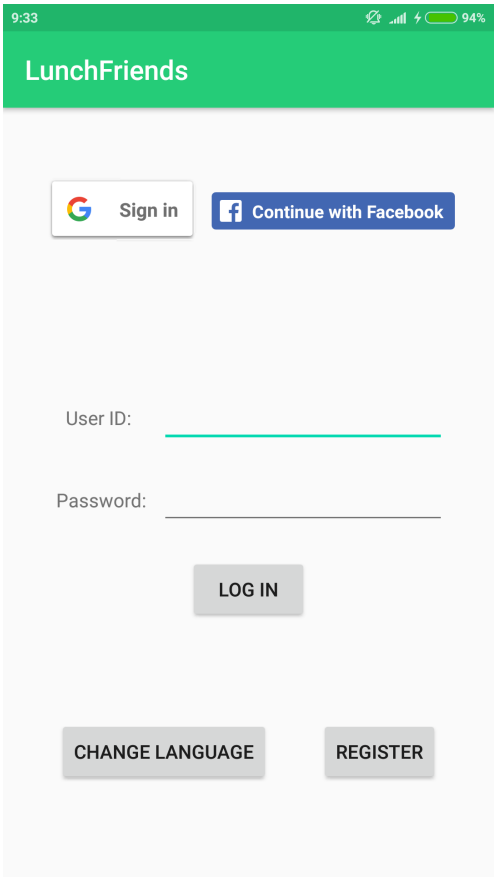
Při programování jsem se snažil o co největší čistotu kódu - dodržování principu DRY (Don't repeat yourself), používání komentářů a vzhledovou úpravu v podobě formátování, správného používání mezer a odsazování. Snažil jsem se dosahnout co nejmenší paměťové a časové složitosti, délku kódu a zjednodušování podmíněných výrazů. Je také vhodné projít si před publikací aplikace všechna varování (warning), která zobrazuje Android studio. Poměrně často se stává, že je potřeba upravit kód, protože některá varování jsou důležitá například z hlediska kompatibility aplikace s různými verzemi systému Android.

4.2.2 Aktivity

V této sekci popisuji nejdůležitější součásti aplikace - aktivity. Vystihnu jejich obsah a účel. Text doplním snímky obrazovek s popisem funkcí jednotlivých elementů.

1 MainActivity

Tato aktivita (na obrázku 4.3) je první obrazovka, která se uživateli zobrazí, když aplikaci zapne. Obsahuje tlačítka pro přihlašování pomocí Google nebo Facebook účtu, které bude popsáno v další sekci. Dále se zde nachází políčka (`EditText`) pro zadávání uživatelského ID a hesla pro přihlášení k účtu, který je uložený na serveru. Políčko pro heslo má nastavený atribut `inputType` na hodnotu `textPassword`, která způsobuje, že zadané řetězce se nezobrazují přímo, ale jsou nahrazeny tečkami, což je vhodné kvůli zabezpečení. Tlačítkem „Log in“ se pak zahájí přihlašovací proces - získá se zadané přihlašovací ID a heslo a ověří se, jestli se tyto údaje shodují s údaji na serveru. Po úspěšném přihlášení se aplikace přepne do aktivity `HomeActivity`. Pokud je přihlášení neúspěšné, zobrazí se uživateli vyskakovací zpráva `Toast` s důvodem, proč



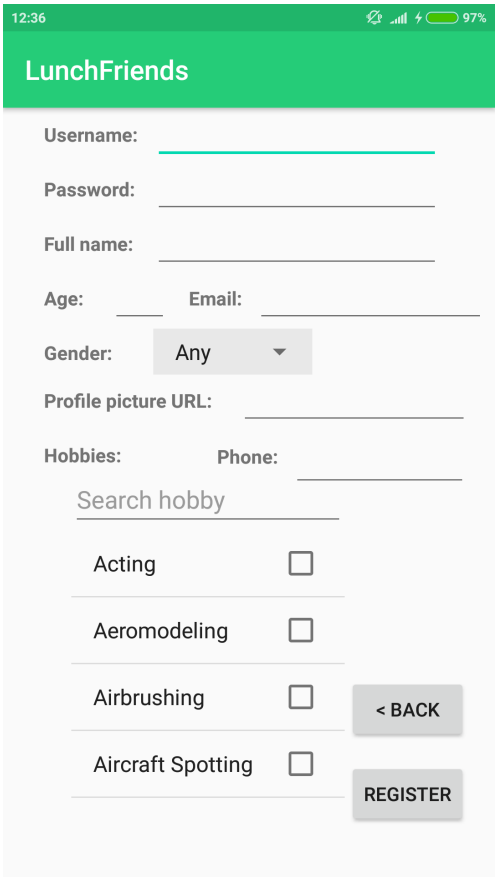
Obrázek 4.3: Aktivita MainActivity

přihlášení selhalo. To může být nedostupnost internetu, neexistující uživatelské ID nebo pokud nesouhlasí hesla. Při zapnutí této aktivity se také mažou všechny obědové události a pozvánky, které mají prošlý čas, tj. týkají se oběda v minulosti.

Dalším prvkem je tlačítko „Change language“, které přepíná na obrazovku pro změnu jazyka aplikace (`ChangeLanguageActivity`), kde má uživatel na výběr mezi češtinou a angličtinou. Důležité je tlačítko „Register“, které aplikaci přepne do aktivity `RegisterActivity`. Tato aktivita bude podrobněji popsána v následující sekci.

2 RegisterActivity

Obrazovka umožňující registraci uživatele. Obsahuje všechna políčka, která vystihují informace o uživateli. Všechny hodnoty jsou povinné. Nachází se zde i seznam koníčků, ze kterého lze vybrat požadované zájmy. Po stisknutí tlačítka „Register“ se zkontroluje, jestli jsou všechny položky neprázdné. Pokud nejsou,



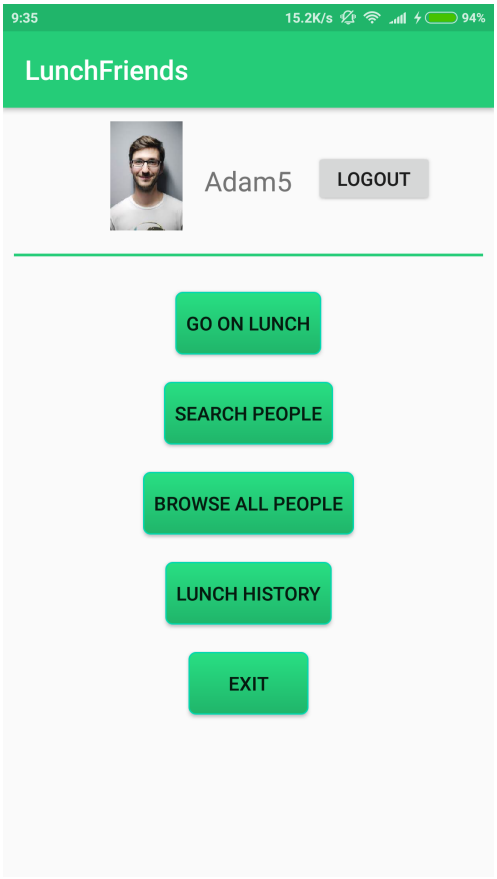
Obrázek 4.4: Aktivita RegisterActivity

zobrazí se červený text „All fields are required“. Pokud je vše správně, odešlou se data na server v podobě třídy `Person` transformované do formátu XML. Server potom uloží tato data do databáze.

3 HomeActivity

Další důležitou aktivitou je `HomeActivity`. To je domovská stránka aplikace, kde se nachází hlavní menu. V horní části lze vidět informace o aktuálně přihlášeném uživateli - jeho profilový obrázek a celé jméno. Napravo je tlačítko pro odhlášení, které rozliší, jakým způsobem byl uživatel přihlášen a odhlásí ho. Při spuštění této aktivity se zároveň spustí načítání pozvánek na oběd pro uživatele, který je momentálně přihlášen. Ťuknutím na položky v menu lze vyvolat následující akce:

- `Go on lunch` - spuštění aktivity `GoOnLunchActivity` pro založení události oběda.

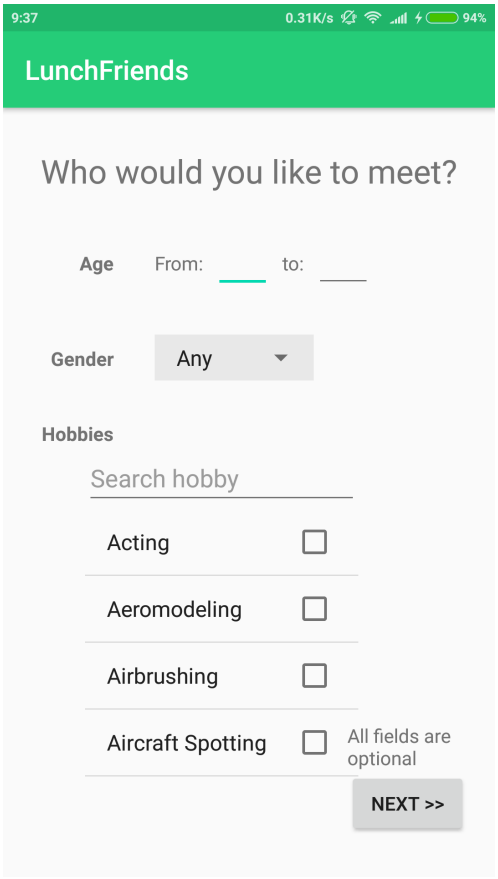


Obrázek 4.5: Aktivita HomeActivity

- `Search people` - spuštění `SearchFriends1Activity` pro zadání vyhledávacích kritérií.
- `Browse all people` - spuštění `FriendResultsActivity` se seznamem všech uživatelů, kteří chtějí jít na oběd.
- `Lunch history` - spuštění `LunchHistoryActivity`, kde je seznam všech obědů tohoto uživatele, které proběhly v minulosti.
- `Exit` - ukončí aplikaci.

4 SearchFriends1Activity

Zde lze vidět aktivitu pro zadávání kritérií ke hledání přátel na oběd. Podmínkou pro nalezení nějakého člověka je, že si tento člověk předtím vytvořil událost oběda pomocí tlačítka „Go on lunch“ z hlavního menu. Pokud by uživatel chtěl sám zvolit místo a čas oběda, musí založit vlastní událost a počkat,



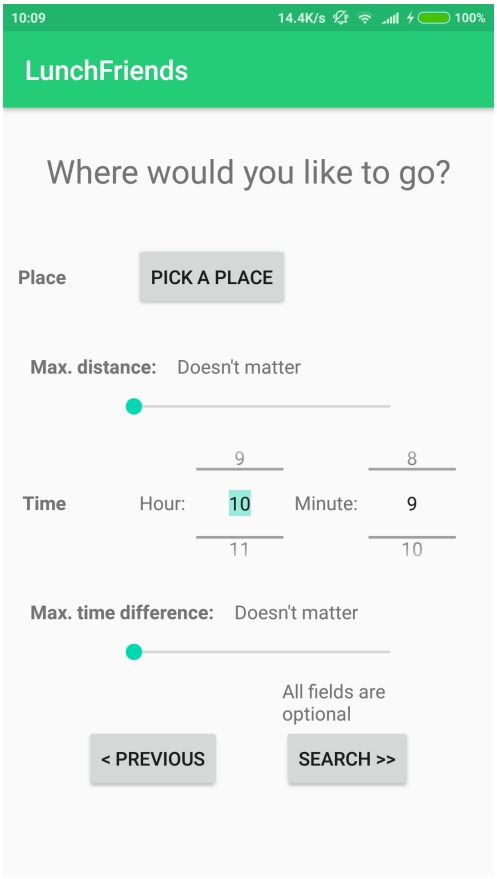
Obrázek 4.6: Aktivita SearchFriends1Activity

až ho někdo pozve. Druhou možností je někoho pozvat a následně se s ním domluvit na jiném místě nebo čase.

Kritéria zahrnují věkové rozpětí, pohlaví a seznam zálib. Člověk vyhovuje vyhledávání, pokud jeho záliby obsahují alespoň jednu zálibu ze všech, které jsou vybrané ze seznamu (tedy pokud tyto dva seznamy mají neprázdný průnik).

5 SearchFriends2Activity

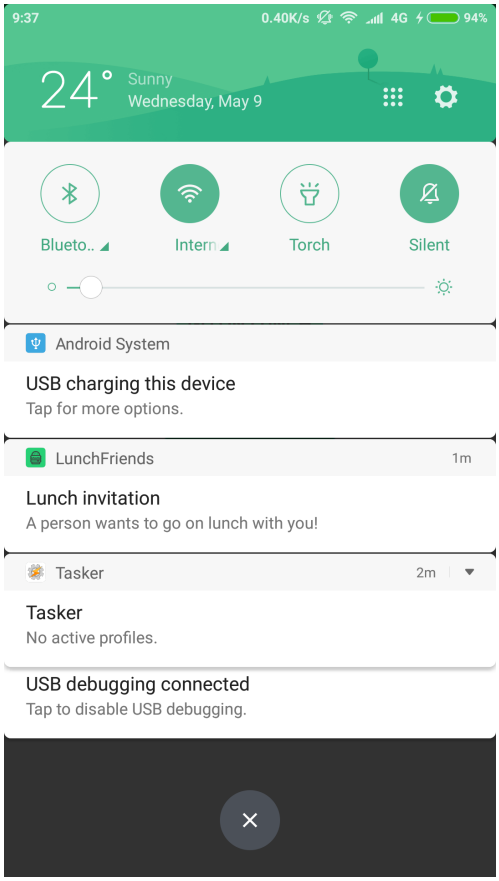
Druhá část zadávání vyhledávacích kritérií, spouští se po ťuknutí na tlačítko „Next“ v aktivitě `SearchFriends1Activity`. Umožňuje zvolit kritéria času a místa oběda, spolu s rozpětím vzdálenosti a časové odlišnosti. Lze ji vidět na obrázku 4.7.



Obrázek 4.7: Aktivita SearchFriends2Activity

6 Notifikace

Pozvánky na oběd se načítají každých 30 sekund ve vlákně na pozadí. Když právě přihlášeného uživatele někdo pozval, obdrží pozvaný uživatel notifikaci v systémové notifikační liště (druhá notifikace ze seznamu na obrázku 4.8). Když tento uživatel na notifikaci ťukne, spustí se aktivita s názvem `NotificationActivity` se základními informacemi o uživateli, který ho pozval. Uživatel zde má možnost pozvánku přijmout nebo odmítnout. Po libovolné z těchto dvou akcí se smaže tato pozvánka z databáze. Pokud uživatel nic nezvolí, obdrží notifikaci znovu za 10 minut. Po přijmutí se spustí aktivita `UserInfoActivity` s kontaktními informacemi na člověka, který ho pozval a uživatel tak má možnost ho kontaktovat a dále se s ním domluvit. Pokud odmítne, aktivita končí a nic dalšího se neděje.



Obrázek 4.8: Notifikace

4.2.3 Google a Facebook login

První implementovanou funkcionalitou bylo přihlašování pomocí účtů Google a Facebook. Návody lze nalézt na stránkách obou společností v sekci pro vývojáře. Oba způsoby přihlášení mají rozdílný způsob implementace.

1 Google login

K začlenění možnosti přihlašování pomocí Google účtu do aplikace je potřeba přidat do souboru `build.gradle` (na modulové úrovni) knihovnu nazvanou `play-services-auth` jako závislost. V této aplikaci byla použita verze knihovny `11.8.0`. Také je nutné vytvořit stránku aplikace (pokud ještě neexistuje) v online portálu pro vývojáře Google Developers Console (`https://console.developers.google.com`) a nechat si vygenerovat klíč pro identifikaci aplikace při komunikaci se servery společnosti Google, nazvaný „OAuth 2.0 client ID“. Tento klíč se uloží jako řetězcová konstanta do souboru `strings.xml` ve složce `res/values` v projektu. Poté se přidá přihlašovací tlačítko do lay-

out souboru aktivity (prvek `SignInButton` poskytovaný ve zmíněné knihovně v balíčku `com.google.android.gms.common`) a může se začít implementovat kód. V odpovídající aktivitě se vytvoří objekt `GoogleSignInOptions`, kde se specifikuje typ přihlášení, informace, které chceme získat o uživateli po přihlášení (email, profil) a povolení k různým akcím, které jsou spojené s účtem (ukládání do cloudu apod.). Potom je třeba vytvořit objekt `GoogleApiClient`, pomocí kterého se spravuje samotné přihlašování. Přihlášení probíhá pomocí metody `startActivityForResult`, kde se vloží intent s odkazem na `GoogleApiClient` a spustí se přihlašovací okno.

Když je dostupný výsledek přihlášení, v metodě `onActivityResult` se získá výsledek v podobě objektu `GoogleSignInAccount`, pokud se přihlášení povedlo, nebo výjimky `ApiException`, pokud se nepovedlo. Z objektu `GoogleSignInAccount` je možné získat informace o přihlášeném uživateli, jako např. email, URL profilového obrázku apod. a dále s nimi pracovat. Instanci `GoogleApiClient` si v projektu ukládám do třídy `GoogleApiClientSingle`, kde je uložena po celý běh aplikace a lze ji tak získat i v druhé aktivitě a její pomocí odhlásit uživatele.

2 Facebook login

Podobně jako u Google přihlašování, je před integrací přihlašování pomocí Facebook účtu třeba vygenerovat hash šifrovacího klíče pro naše vývojové prostředí a spojit ho s danou aplikací na stránkách pro Facebook vývojáře (`https://developers.facebook.com/`). To slouží pro ověřování požadavků na přihlašování pomocí této aplikace. Hash lze vygenerovat pomocí nástroje `keytool` z JDK. Dále je potřeba přidat informace do souboru `AndroidManifest.xml` - zaregistrovat dvě přihlašovací aktivity a metadata `ApplicationId`, které se používá pro identifikaci aplikace.

Poté se přidá tlačítko do layout souboru aktivity (prvek `LoginButton` z balíčku `com.facebook.login.widget`). Poté už je vše připraveno a přichází implementační část. Naprogramuje se chování přihlašovacího tlačítka v aktivitě. Najde se toto tlačítko klasicky pomocí metody `findViewById` a na něj se zaregistruje objekt `FacebookCallback<LoginResult>`, který zpracovává výsledek přihlášení. Informace o přihlášeném uživateli lze získat z profilu - `Profile.getCurrentProfile()`.

4.2.4 Google Places API

1 Places

Pro získání místa k setkání přátel při obědě je použité programovací rozhraní Google Places, které poskytuje funkce pro práci s místy v mapách Google. Opět je potřeba přidat do modulového souboru `build.gradle` knihovnu, pro tento případ se knihovna jmenuje `play-services-places`, ve verzi `11.8.0`.

Kromě tohoto opatření je také nutné přidat do Android manifestu speciální hodnotu (meta-data) tzv. API key, který spojuje požadavky na Googlu server s touto aplikací a identifikuje ji při použití rozhraní Google Places. Tento klíč si lze nechat vygenerovat na stránkách Google Developers Console, kde se předtím naše aplikace musí zaregistrovat.

Z knihovny Google Places byl konkrétně použit nástroj Place Picker. Po stisknutí tlačítka „Choose place“ v aplikaci se otevře okno s mapou, na které má uživatel možnost vybrat nějakou restauraci nebo prosté souřadnice na mapě. Místo se uloží do objektu typu `Place`, který lze získat v kódu aktivity a dále s ním pracovat. V aplikaci si ukládám jeho `PlaceId` - unikátní identifikátor tohoto místa, abych mohl později snadno znovu získat tento objekt `Place` pomocí metody `getPlaceById` třídy `GeoDataClient`.

2 DistanceMatrix

Pro zjišťování vzdálenosti dvou míst od sebe jsem použil Google nástroj s názvem `DistanceMatrix`, který umožňuje pomocí GET požadavku s identifikátory `PlaceId` dvou míst napsaných v URL požádat o zjištění vzdálenosti těchto míst od sebe. Ke komunikaci s touto službou jsem použil stejný `TaskFragment`, který používám při komunikaci se serverem mé aplikace. Výslednou vzdálenost služba vrací jako XML soubor s informací v metrech, kilometrech a přibližnou dobou cesty z jednoho místa do druhého autem (v minutách a sekundách). Pro použití tohoto nástroje je opět potřeba nechat si vygenerovat klíč API v konzoli pro vývojáře a přidat ho do URL požadavku.

4.2.5 Překlad do češtiny

1 Řetězcové konstanty

Ačkoliv se to nemusí zdát, značnou část času určeného na implementaci zabere překládání aplikace. Výchozím jazykem je angličtina, já jsem se rozhodl jako sekundární jazyk aplikace použít češtinu. Bylo třeba přeložit všechny řetězce textu a popisky na všech obrazovkách, notifikacích apod. To zahrnuje i seznam koníčků (zálib), který má velikost cca 370 položek. K překladu tohoto seznamu byl použit Google překladač, následovala ruční kontrola a opravy. Celkem tato aplikace používá asi 70 řetězcových konstant.

Všechny textové řetězce se ukládají jako konstanty do souboru s názvem `strings.xml` ve složce projektu `res/values`. Překlad je potřeba uložit do souboru ve složce `res/values-b+xx`, kde xx značí dvoupísmenný název jazyka (čeština má označení „cs“, lze nalézt v nápovědě pro vývojáře). Každý řetězec má pojmenování `name`, pomocí kterého se na něj poté lze odkazovat a vkládat ho na potřebná místa.

2 Ukládání jazyka přes vypnutí aplikace

Aby aplikace fungovala správně a byla uživatelský přívětivá, je potřeba, aby se zvolený jazyk uložil i když se aplikace vypne a znovu obnovil až se zapne. K tomu je použitá metoda `saveLocale`, která využívá rozhraní (interface) `SharedPreferences`. Jedná se o rozhraní, které umožňuje ukládat páry klíč-hodnota do XML souborů, které přetrvávají i pokud je aplikace vypnutá, což je přesně to, co potřebujeme. K uložení zvoleného jazyka je vložen pár „Language"-„xx", kde xx je dvoupísmenný řetězec značící název jazyka, tedy buď „en" pro angličtinu, nebo „cs" pro češtinu. Jazyk se potom obnovuje z tohoto XML souboru při každém startu aplikace pomocí metody `loadLocale`.

4.2.6 Barvy a colors.xml

Podobně jako řetězcové konstanty, ukládají se i barvy v souboru s příponou xml. Tento soubor `colors.xml` je ve složce `res/values` v adresáři projektu. Jedna barva má podobu elementu `<color>` s atributem name - pojmenování barvy a mezi otevíracím a zavíracím XML tagem je HTML kód dané barvy. Na tuto barvu se pak lze kdekoliv v aplikaci odkazovat pomocí jejího jména.

4.2.7 Třída Log

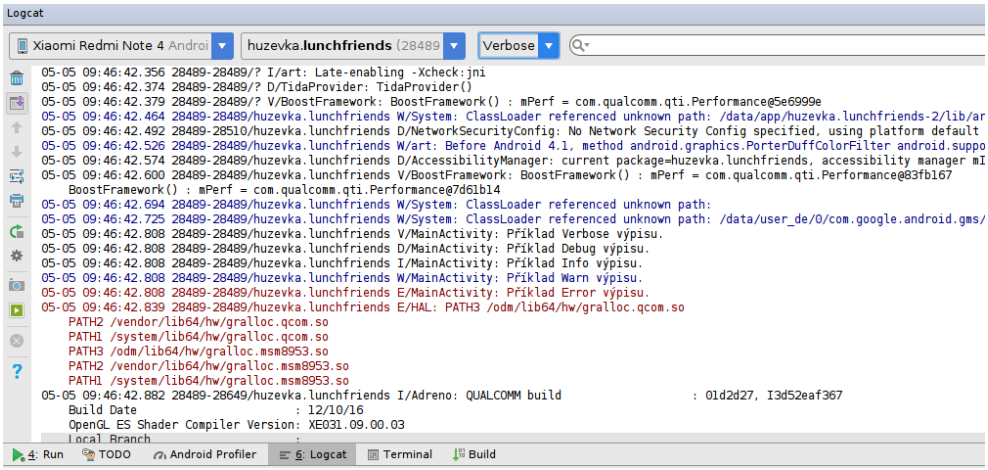
Při programování je často potřeba vypsat si obsah nějaké proměnné nebo vypsat si text, když se vykonávání kódu dostalo na určité místo. K tomu se hodí třída `Log`, která umožňuje vypsat informace do logů. Z logů lze data jednoduše zobrazovat v Android studiu v dolní liště na záložce „Logcat" (klávesová zkratka `Alt+6`). Existuje 6 úrovní zpráv: Verbose, Debug, Info, Warning, Error a Assert. V Android studiu lze zvolit, které se budou zobrazovat a lze tak například vyfiltrovat nepotřebná Verbose oznámení, kterých se většinou při běhu aplikace vypisuje mnoho a překáží, když chceme vidět důležitější zprávy. Ukázkové okno „Logcat" lze vidět na obrázku 4.9.

4.2.8 Techniky zabraňující únikům paměti

Častým problémem Android aplikací jsou úniky paměti (memory leaky), kdy v paměti zůstanou data, která nelze uvolnit a zbytečně zabírají místo. Místa přitom není mnoho, starší mobilní zařízení trvale trpí nedostatkem paměti a jak se aplikace zlepšují a zvětšují se hardwarové nároky, situace se stále zhoršuje. Použitím technik zamezujícím memory leakům lze snadno ušetřit paměť.

1 Single instance a Singleton

Pro uchování nějakého objektu, ke kterému potřebujeme přistupovat na více místech v aplikaci, je vhodné použít návrhový vzor Singleton. Jde o třídu,



Obrázek 4.9: Okno Logcat

která obsahuje statickou třídní proměnnou, ve které je uložen potřebný objekt. Obvykle se pro tento účel používá zmíněný vzor Singleton, ten však při vývoji pro Android není bezpečný právě kvůli možným únikům paměti. Kromě toho také není bezpečný pro multi-threading a má více dalších problémů.

Lepší je použití vzoru Single instance, kdy tato třída dědí od třídy s názvem `Application` a operační systém má o ní lepší přehled. Vytvoří ji vždy, když se spustí aplikace a zavolá její metodu `onCreate` podobně jako u aktivity. Postará se i o její řádné ukončení. V mé implementaci jsem tento vzor použil pro uchovávání objektu `GoogleApiClient` ve třídě `GoogleApiClientSingle` a ve třídě `LoginSingle` pro uchování informací o právě přihlášeném uživateli.

2 AsyncTask, TaskFragment a vlákna

`AsyncTask`, v plném znění asynchronous task, neboli paralelní úkol, je třída, kterou je potřeba podědit, když chceme vykonat akci na pozadí (mimo hlavní vlákno, tzv. UI thread). Vytvoříme třídu (například `LoadImageTask`), která dědí od třídy `AsyncTask` a přepisuje metody:

- `onPreExecute` - provede se před vykonáním akce na pozadí.
- `doInBackground` - obsah této metody se provede na pozadí.
- `onPostExecute` - vykoná se po skončení akce na pozadí.

Celý proces se spouští z hlavního vlákna pomocí metody `execute`.

Třída `AsyncTask` má tři šablonové generické parametry. První značí třídu, jejíž objekt (nebo více těchto objektů, protože parametr je v podobě `Object... obj`, připouští tedy proměnný počet parametrů) se bude předávat do metody `execute`, která ho dále předá metodě `doInBackground` a v ní je tak možnost

s tímto objektem pracovat. Druhý šablonový parametr značí třídu, pomocí které se bude v metodě `doInBackground` předávat postup vykonané operace pomocí `publishProgress` (například kolik procent je hotových). V metodě `onProgressUpdate` tohoto objektu `AsyncTask` pak lze například vypsat tento postup na obrazovku a dát tak uživateli informaci, jak přibližně dlouho ještě bude muset čekat. Poslední, třetí parametr značí třídu, jejíž objekt bude vracet metoda `doInBackground` jako výstup celé operace a tento objekt se předá do metody `onPostExecute`. Protože po zavolání metody `execute` se vytvoří nové vlákno a vykonávání kódu na hlavním vlákně se přesune na další operace, musí se implementovat zvláštní předávání výstupního objektu v metodě `onPostExecute`. To probíhá tak, že se v naší třídě `AsyncTask` vytvoří rozhraní (interface) `TaskCallbacks` s metodami `onPreExecute` a `onPostExecute`, které se zavolají v odpovídajících metodách objektu `AsyncTask`. Tento interface se naimplementuje v aktivitě, z které voláme úkol a takto získáme možnost něco provést na hlavním vlákně před vykonáním a po vykonání operace na pozadí a kromě toho taky možnost použít výstupní objekt celé operace v `doInBackground`.

Způsob použití async tasku je velmi důležitý z hlediska úniků paměti. Nabízí se nejjednodušší možnost vytvořit potomka třídy `AsyncTask` jako vnitřní třídu v aktivitě. Zde ale dojde k úniku paměti, když aktivita skončí ještě předtím, než doběhne úkol. Ten totiž, jako každá vnitřní třída, drží referenci na vnější objekt, tedy aktivitu a ta proto nebude uvolněna když skončí, protože úkol s její referencí stále existuje a běží na pozadí.

Jednou ze správných metod použití je tzv. `TaskFragment`. Jde o klasický fragment, který nemá žádné UI. Připojí se k aktivitě přes třídu nazvanou `FragmentManager` a umístí se do něj daný `AsyncTask`. Tento fragment má nastavenou hodnotu `setRetainInstance(true)` a proto se zachovává i při změně konfigurace (typicky skončení aktivity). I tento fragment obsahuje referenci na aktivitu, avšak to je uvolněna (proměnná je nastavena na hodnotu `null`) v metodě `onDetach` fragmentu, když tato aktivita končí. K žádnému úniku paměti tak v tomto případě nedojde a aplikace s použitím tohoto postupu funguje, jak má. Hodí se také, že při vypnutí a opětovném spuštění aktivity neztratíme její referenci, protože se sama s fragmentem opět propojí v metodě `onAttach`.

Další možností, jak vytvořit paralelně běžící část kódu jsou klasická vlákna (třída `Thread`). Lze je používat stejně jako v desktopových Java programech. Pro účely této aplikace jsem však považoval za vhodnější použití zmíněné třídy `AsyncTask`, kvůli mnohem lepší přehlednosti použití a čistotě kódu.

4.2.9 Server a databáze

1 Server

Pro komunikaci s databází a ukládání dat byl použit webový server s rozhraním REST API. Jazykem implementace je JavaScript, konkrétně serverové prostředí Node.js. Zdrojový kód je napsaný ve dvou souborech:

1. `server.js` - obsahuje hlavní kód pro spuštění serveru, spuštění obsluhy požadavků a ukončování běhu po zachycení ukončovacích signálů SIGTERM a SIGINT.

2. `handleRequest.js` - obsahuje funkci `handleRequest`, která se používá pro zpracování všech HTTP požadavků na server (GET pro získání dat a POST pro uložení dat a získání odpovědi).

Zpracování požadavku vypadá tak, že se nejdříve rozliší jeho typ pomocí URL, které bylo použito pro odeslání. Pokud to bylo metodou POST, server získá a zpracuje zaslaná data a převede je z formátu XML do objektů. Potom s nimi provede požadovanou akci nad databází a odešle odpověď s informací, jestli data byla úspěšně zpracována. Pokud požadavek byl typu GET, odešle server SELECT příkaz s požadovanými parametry dle URL na databázi a jako odpověď odešle získaná data ve formátu XML.

Server používá několik knihoven, které lze instalovat pomocí nástroje `npm` příkazem `npm install <jméno knihovny>`. Těmito knihovnami jsou:

1. `oracledb` - knihovna pro komunikaci s databází. Používá tzv. connection pooling - udržuje několik připojení k databázi najednou a ty tak lze rychle za sebou použít bez nutnosti čekání na opětovné připojení.

2. `http` - slouží ke správě HTTP požadavků.

3. `object-to-xml` - pro převádění objektů na reprezentaci ve formátu XML.

4. `libxmljs` - pro převádění objektů z XML do JavaScriptových proměnných.

Server lze spustit po instalaci potřebných knihoven a programu Node.js v minimální verzi 8. Spouští se příkazem `nodejs server.js` v adresáři se zdrojovými soubory. Po správném startu vypíše na konzoli „Listening on port 3000“.

2 Komunikace se serverem v aplikaci

Pro komunikaci se serverem jsem v Android aplikaci použil knihovnu jménem `spring-android-rest-template` ve verzi `1.0.1.RELEASE` z odnože frameworku Spring pro Android. Umožňuje jednoduchou komunikaci přes REST

rozhraní. Data se odesílají a přijímají ve formátu XML, proto je potřeba použít knihovnu pro transformaci mezi formátem XML a objekty. Pro tento účel byla použita knihovna `simple-xml` ve verzi `2.7.1` z frameworku `simpleframework`. Data jsou převáděna automaticky, stačí do REST šablony (`RestTemplate`) přidat parametr `new SimpleXmlHttpMessageConverter()` a vytvořit potřebné modelové třídy, do kterých budou data transformována. Tyto třídy jsem v projektu umístil do balíčku s názvem „model“. V modelových třídách je potřeba použít anotaci `Root` s parametrem `name` nad deklarací třídy a anotaci `Element`, též s tímto parametrem, nad každou proměnnou třídy.

4.2.10 Logo a úprava obrázků

K vytvoření loga aplikace byl použit open source nástroj na úpravu obrázků GIMP (GNU Image Manipulation Program), který je známý jako bezplatná alternativa Adobe Photoshopu a navíc funguje i na operačních systémech s jádrem Linux. Poskytuje podobné funkce jako zmiňovaný Photoshop. K vytvoření verzí loga pro zařízení s nižším rozlišením displeje byl použit nástroj Asset Studio přímo v Android studiu pod volbou `New -> Image asset`.

GIMP byl mimo jiné použit i na ořezávání a doupravení screenshotů a obrázků vložených do textu této práce.

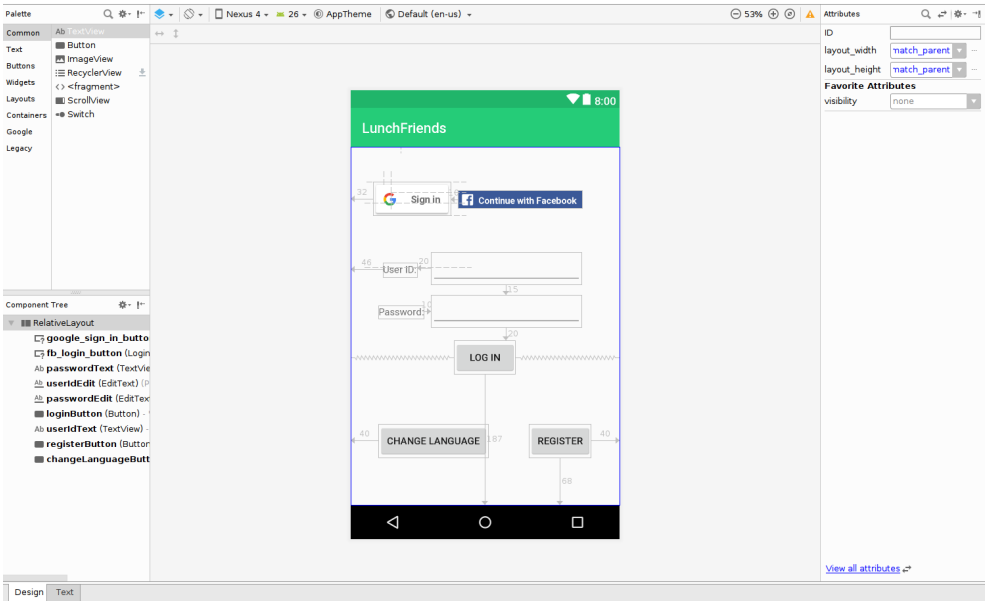
4.2.11 Layout

Jako layout ve všech aktivitách jsem použil `RelativeLayout`, protože je vhodný na umístění elementů bez celkového rozložení (jako např. `LinearLayout`) ale se závislostí umístění každého prvku na ostatních. Specifikuje se rozmístění v závislosti na jiném prvku nebo závislosti na okraji obrazovky. Nešel použít `LinearLayout`, protože prvky na obrazovce jsou různě veliké a celkově by nesedělo rozložení. Druhou možností byl `TableLayout`, avšak díky lepší kontrole umístění prvků jsem se nakonec rozhodl pro `RelativeLayout`. Rozložení obrazovky `MainActivity` (přihlašovací obrazovka) v layout editoru Android studia spolu s rozmístěním a vztahy jednotlivých prvků lze vidět na obrázku 4.10.

4.2.12 Třída LunchFriendsTools

1 Metody třídy

Třída s názvem `LunchFriendsTools` z balíčku `tools` je pomocná třída, která poskytuje statické metody, které jsou zde strategicky umístěné kvůli přehlednosti a neopakování kódu. Jde například o metody na převod mezi „dp“ (density independent pixel, míra velikosti nezávislá na rozlišení displeje) a pixely podle rozlišení obrazovky aktuálního zařízení, zjištěného za běhu aplikace. Další metodou je například `isNetworkAvailable` pro zjištění, jestli je dostupné internetové připojení. Tato metoda se používá před přihlašováním



Obrázek 4.10: Přihlašovací obrazovka v layout editoru

uživatele. Třída obsahuje i funkce pro práci s obrázky - `setImageLoading` a `scaleBitmap`.

2 Ověřování identity při přihlašování

Důležitou metodou je `bytesToHex` pro převod dat z formátu pole bytů do řetězcové hexadecimální reprezentace (používá se po výpočtu hash kódu hesel uživatelů). Hesla jsou hashována, aby se neukládala rovnou v textové reprezentaci. Potenciální útočník, který by se dostal k datům v databázi, by tak musel hledat kolizi hashovací funkce, aby zjistil původní heslo a mohl se přihlásit na účet dané osoby. Při přihlašování uživatele se kontroluje pouze shoda hash kódu zadaného hesla s hash kódem v databázi.

4.2.13 ListView se zájmy

Pro seznam zájmů uživatele je použit speciální seznam `ListView`, ve kterém lze zaškrtávat položky a navíc v něm lze vyhledávat pomocí políčka pro zadávání textu. Položky zaškrtnuté po vyhledávání je potřeba vyhledat dle indexu v tabulce a jejich textovou reprezentaci uložit do množiny `TreeSet`, protože po smazání textu z vyhledávacího políčka by neseděly indexy zaškrtnutých položek. Pro ukládání do databáze se zájmy opět převedou na indexy v tabulce a uloží se jako text (databázový sloupec typu `VARCHAR2 (1500 CHAR)`) ve formátu „<mezera> { index <mezera> }“, kde prvek ve složených závorkách se může libovolněkrát opakovat.

Kapitola 5

Testování

Testování softwaru je vhodné provádět k ověření správného fungování programu. U větších projektů dokáže testování započaté už v raných fázích vývoje ušetřit zdroje určené na údržbu softwaru. Psaní testů navíc umožňuje hlouběji pochopit požadavky a charakteristiku řešeného problému. V následujících sekcích budou popsány různé druhy testů a jejich použití v aplikaci LunchFriends.

5.1 Unit testy

Unit (jednotkové) testy jsou pravděpodobně nejčastěji implementovaným druhem testů. Každý test má za úkol ověřit funkčnost nějaké izolované komponenty, typicky aktivity. Test by měl být libovolně krát opakovatelný. To znamená, že pokud například měnil data v databázi, je potřeba je po testu vrátit opět do původního stavu.

Na platformě Android se pro automatické jednotkové testy používá framework JUnit. Každý test je zapsaný v metodě označené anotací `@Test`. Tyto metody jsou umístěny ve třídě, která je potomkem třídy `TestCase`. V této třídě máme také možnost přepsat metodu `setUp`, která je zavolána před každým testem, a metodu `tearDown`, která se volá po každém testu. To je vhodným místem pro uvedení všech dat do původního stavu.

Existují dva typy testů:

- Lokální testy - spouštěné pouze na lokálním stroji (počítači), běží na Java Virtual Machine (JVM).
- Instrumented (přístrojové) testy - běží přímo na Android zařízení nebo na emulátoru.

V aplikaci jsem použil přístrojové unit testy ke kontrole správnosti metod pro převod mezi DP a pixely ve třídě `LunchFriendsInstrumentedTest`

a druhý test ve třídě `MainActivityTest` s použitím knihovny Espresso pro ovládání UI, kde se ověřovalo přihlašování.

5.2 Integrační testy

Integrační testy, na rozdíl od jednotkových testů, mají za úkol ověřovat spolupráci testovaných komponent. V Android testing frameworku existují specializované testovací třídy, které zajišťují vytváření těchto testů pro hlavní komponenty aplikace (aktivity, služby apod.).

5.3 Testování uživatelského rozhraní

Pomocí testování uživatelského rozhraní je možné určit, jestli aplikace neobsahuje chyby v návrhu grafického rozhraní a jestli je dobře ovladatelná. K tomuto účelu jsem oslovil 10 osob ve věku 18-60 let, které měly za úkol projít několik testovacích scénářů:

1. Registrace - osoba měla za úkol vytvořit nový účet.
2. Vyhledání uživatelů - úkolem bylo vyhledat uživatele pomocí kritérií.
3. Založení oběda - cílem bylo založit novou událost oběda.

Při testování byla objevena chyba v kódu aplikace - při registraci se nekontrolovalo, jestli je zadané URL profilového obrázku. Pokud nebylo, aplikace se rovnou přepnula do další aktivity a zobrazila chybu. Nedala tak uživateli možnost tento údaj dopsat. Druhou objevenou chybou byla nemožnost označit zálibu pomocí zaškrtávacího čtverečku (checkbox), protože se čtverečky vyskytovaly příliš blízko k okraji seznamu. Třetí chybou byly příliš dlouhé názvy zálib, které přetékaly přes řádek a druhá část jejich názvu nebyla vidět.

První chybu jsem opravil tak, že se kontroluje, jestli jsou všechna políčka vyplněna, a pokud nejsou, zobrazí se hláška „All fields are required“. Druhou chybu jsem vyřešil tak, že jsem zvětšil seznam zálib do šířky a třetí, poslední chyba byla opravena přejmenováním příliš dlouhých zálib na alternativní jména. Zároveň jsem zkontroloval, jestli se v aplikaci nevyskytují podobné chyby.

5.4 Statická analýza kódu

Kód aplikace byl též otestován statickou analýzou kódu, konkrétně programem PMD. Tato analýza nenašla výrazné problémy, jediným varováním byla možnost spojit dohromady dvě vnořené `if` podmínky.

Závěr

Cílem práce bylo vytvořit aplikaci, která umožňuje hledat přátele ke společným obědovým schůzkám. Aplikace měla využívat postupy zabraňující memory leakům a REST server pro komunikaci s databází, do které se ukládají data.

Implementace aplikace proběhla bez větších problémů, aplikace byla otestována a funguje jak má. K uvedení aplikace do reálného provozu by bylo potřeba pouze najít vhodný hosting databázového serveru a použít postup pro publikování aplikace napsaný v sekci 4.1.3. Následně by probíhalo vylepšování aplikace skrze nové verze, případně opravování dosud nenalezených chyb.

Na závěr uvádím seznam dalších možných vylepšení. Bylo by vhodné dát uživateli možnost obnovy zapomenutého hesla a používat sůl při ukládání hesla kvůli lepšímu zabezpečení. Dalším vylepšením by bylo rozšíření data obědů na dny - v současnosti se rozlišuje pouze čas v hodinách a minutách. Posledním možným zlepšením by mohlo být přidání možnosti změny profilových informací uživatele.

Literatura

[1] Wireframe - Wikipedie [online]. Wikipedia contributors. 2018, [cit. 27.4.2018]. Dostupné z: https://cs.wikipedia.org/wiki/Wireframe

[2] Android (operační systém) - Wikipedie [online]. Wikipedia contributors. 2018, [cit. 11.4.2018]. Dostupné z: https://cs.wikipedia.org/wiki/Android_(opera%C4%8Dní_systém)

[3] Android Developers [online]. Google Inc. 2018, [cit. 11.4.2018]. Dostupné z: https://developer.android.com/index.html

[4] Lee, W.-M.: *Beginning Android 4 application development*. Indianapolis, Ind: Wrox/John Wiley & Sons, 2012, ISBN 978-1-118-19954-1, [cit. 4.5.2018].

[5] Mednieks, Z.: *Programming Android*. Sebastopol, Calif: O'Reilly, 2011, ISBN 978-1-449-38969-7, [cit. 4.5.2018].

[6] Blundell, P.: *Learning Android application testing*. Birmingham, UK: Packt Publishing, 2015, ISBN 978-1-78439-533-9, [cit. 4.5.2018].

[7] Stack Overflow [online]. Stack Exchange Inc. 2018, [cit. 15.4.2018]. Dostupné z: https://stackoverflow.com/

[8] A Planet of Mine [online]. 2018, [cit. 22.4.2018]. Dostupné z: https://play.google.com/store/apps/details?id=com.tuesdayquest.myplanet

[9] Lunch Buddy [online]. Jiaming Zhang. [cit. 20.4.2018]. Dostupné z: http://lunchbuddy.zcoder.io/

[10] Lets Go Lunch [online]. 2017, [cit. 20.4.2018]. Dostupné z: https://www.letsgolunch.com/

[11] List of Hobbies [online]. [cit. 26.4.2018]. Dostupné z: http://www.notsoboringlife.com/list-of-hobbies/

[12] Singletons in Android [online]. [cit. 5.5.2018]. Dostupné z: https://medium.com/@programmerr47/singletons-in-android-63ddf972a7e7

[13] Wireframe.cc - minimal wireframing tool [online]. [cit. 11.5.2018]. Dostupné z: https://wireframe.cc/

Příloha A

Seznam použitých zkratek

XML Extensible markup language

VCS Version control system

OS Operating system (Operační systém)

UI User interface

API Application programming interface

IDE Integrated Development Environment

REST Representational state transfer

HTML Hypertext markup language

GPS Global positioning system

JDK Java development kit

Příloha **B**

Obsah přiloženého flash disku

- `readme.txt` ..... stručný popis obsahu flash disku
- `exe` ..... adresář se spustitelnou formou implementace
- `src`
  - `impl` ..... zdrojové kódy implementace
  - `thesis` ..... zdrojová forma práce ve formátu LaTeX
- `text` ..... text práce
  - `BP_Húževka_Michal_2017.pdf` ..... text práce ve formátu PDF