Sem vložte zadání Vaší práce.

České vysoké učení technické v Praze
Fakulta informačních technologií
Katedra softwarového inženýrství

Bakalářská práce

# Klient portálu najdi-lékárnu.cz na platformě Android

***Daniel Pína***

Vedoucí práce: Ing. Zdeněk Troníček, Ph.D.

5. května 2015

# Poděkování

Rád bych poděkoval panu Ing. Zdeňku Troníčkovi, Ph.D. za jeho vstřícný přístup a odborné vedení při zpracování této bakalářské práce.

# 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 5. května 2015 .....

### Odkaz na tuto práci

Pína, Daniel. *Klient portálu najdi-lékárnu.cz na platformě Android.* Bakalářská práce. Praha: České vysoké učení technické v Praze, Fakulta informačních technologií, 2015.

# Abstrakt

Cílem této bakalářské práce je návrh a implementace mobilního klienta pro operační sytém Android, který bude přebírat funkcionalitu webového rozhraní „najdi-lékárnu.cz“. Hlavním cílem je vytvořit aplikaci, která bude vyhledávat lékárny v okolí a informace o léčivých přípravcích. V této práci je popsána analýza, návrh, implementace a testování této aplikace.

**Klíčová slova** lék, lékárna, Android, mobilní aplikace, substitut léku, interakce s lékem, vyhledávání lékárny, informace o léku, Java, SQLite

# Abstract

The goal of the thesis is design and implementation of mobile client for Android operating system which will take over the functionality of web interface "najdi-lékárnu.cz". The main aim is to create an application that will search nearby drug stores and information about medicaments. This thesis describes analysis, design, implementation and testing of the application.

**Keywords** medicine, pharmacy, Android, mobile application, drug substitute, drug interaction, searching pharmacies, information about the drug, Java, SQLite

# Obsah

**Úvod** **1**

**1 Analýza** **3**
1.1 Současný stav řešení problematiky . . . . . 3
1.2 Podobné aplikace . . . . . 3
1.3 Analýza požadavků . . . . . 7
1.4 Analýza případů užití . . . . . 10

**2 Návrh** **11**
2.1 Architektura OS Android . . . . . 11
2.2 Vývoj aplikací pro OS Android . . . . . 12
2.3 Verze OS Android . . . . . 13
2.4 Architektura aplikace pro OS Android . . . . . 13
2.5 Návrh uživatelského rozhraní . . . . . 15
2.6 Databáze . . . . . 19
2.7 Synchronizace . . . . . 22

**3 Implementace** **25**
3.1 Skenování čárových kódů . . . . . 25
3.2 Ukládání dat . . . . . 26
3.3 Práce na pozadí . . . . . 28
3.4 Propojení se serverem . . . . . 29
3.5 Lokalizace . . . . . 31
3.6 Přístup k funkcím telefonu . . . . . 31
3.7 Využití Google Maps . . . . . 33
3.8 Uživatelské rozhraní . . . . . 33
3.9 Vyhledávání . . . . . 35

**4 Testování** **39**
4.1 Testování autorem . . . . . 39

4.2 Usability test . . . . . 39

**Závěr** **45**

Pokračování v práci . . . . . 45

**Literatura** **47**

**A Seznam použitých zkratek** **51**

# Seznam obrázků

1.1 Diagram případů užití . . . . . 10

2.1 Poměr OS v prodaných zařízeních . . . . . 14
2.2 Poměr verzí OS Android . . . . . 14
2.3 Typy zobrazení obsahu . . . . . 17
2.4 Záložka lékárny . . . . . 18
2.5 Záložka léky . . . . . 18
2.6 Záložka oblíbené . . . . . 18
2.7 Detail lékárny . . . . . 20
2.8 Detail léku . . . . . 20
2.9 Doménový model . . . . . 21
2.10 Diagram aktivit: Aktualizace databáze . . . . . 23

3.1 Vyhledávání lékáren . . . . . 37

# Seznam tabulek

2.1 Poměr verzí OS Android . . . . . 13

3.1 Rychlost databází: 5000krát volání SELECT . . . . . 26

4.1 Testovací zařízení . . . . . 40

# Úvod

Téměř každý z nás už někdy kupoval léky, někdo víckrát, někdo třeba jen jednou. Lidé, kteří léky nakupují pravidelně, se jistě již setkali s problémem, kde a jaké léky koupit, protože ne v každé lékárně je konkrétní lék stejně drahý.

V dnešní době je velmi těžké se zorientovat na farmaceutickém trhu a vybrat tu správnou lékárnu, která by vyhovovala našim zákaznickým potřebám. Ke dni 18.10.2014 bylo v České republice registrováno 2835 lékáren [1], což je patrné ve větších městech, kde se lékárny objevují na každém rohu.

Někteří možná nalezli lékárnu, která jim vyhovuje, ale ani netuší, že konkrétní lék mohou nakupovat levněji, protože mohou koupit substitut léku se stejnou účinnou látkou, který je výrazně levnější. (V České republice bylo k 18.10.2014 registrováno 56720 léčivých přípravků [2]). Může se také stát, že lék, který má pacient předepsán od lékaře mu nevyhovuje a rád by užíval jiný lék, ale se stejným účinkem. Pacienti by mohli k nalezení léku se stejnou účinnou látkou využít internet, zeptat se lékaře nebo se poradit přímo v lékárně. Ale to může být jednak velmi časově náročné, jednak i neobjektivní, protože někteří lékaři a lékárníci předepisují a doporučují léky, které nabízí farmaceutická firma, se kterou mají dohodu. Díky tomu mají lékaři či lékárníci provize od konkrétní farmaceutické firmy za to, že podporují prodej léků, které určitá firma vyrobila. Podporují prodej takových léků i přesto, že existuje jiný substitut, který je mnohem levnější a stejně účinný. Zákazníci si proto raději hledají informace o lécích a substitutech sami. To je však někdy velmi technicky i časově náročné, a proto častěji spoléhají na lékaře či lékárníky. Ti mohou být, a ve většině případů i jsou ovlivněni firmami, aby předepisovali nebo nabízeli jejich léky.

Může také nastat situace, kdy někdo potřebuje nějaký určitý lék co nejrychleji, například kvůli akutní bolesti nebo léku, který si zapomeneme doma. V této situaci potřebuje najít co nejrychleji lékárnu, ale pokud dané okolí nezná, může to být problém. Málo kdo s sebou nosí osobní počítač s připojením na internet, aby tuto informaci mohl jednoduše najít. Proto nejvhodnější bude

hledat tyto informace na mobilním zařízení, jelikož většina těchto zařízení má prakticky stálý přístup k internetu a to ulehčuje vyhledávání potřebných informací.

Proto se moje bakalářská práce zabývá vývojem klientské aplikace pro internetový portál „najdi-lékárnu.cz“ pro mobilní operační systém Android, která by měla být kvalitní, uživatelsky přívětivá a měla by usnadnit rozhodování mnoha uživatelům. Hlavní funkcí aplikace je vyhledávání lékáren, léků a nabízení informací o konkrétním léku, včetně nalezení substitutů a interakcí.

Práce je rozdělena do několika částí. První část se zabývá analýzou problému. V další tvořím návrh, následně ho implementuji a v poslední části vytvořenou aplikaci testuji. Aplikace by měla být přínosná pro uživatele, kteří chtějí mít přehled o lékárnách a lécích a chtějí mít tyto informace vždy po ruce.

Kapitola 1

# Analýza

## 1.1 Současný stav řešení problematiky

V současné době je plně funkční webové rozhraní portálu „najdi-lékárnu.cz“[3], ke kterému již byly vytvořeny aplikace na operační systémy Windows Phone a iOS, ty přebírají funkcionalitu webového rozhraní. Na operační systém Android existuje taktéž aplikace spojená s daným portálem. Tato aplikace pro Android se však vytvářela v roce 2013 a od té doby se API portálu „najdi-lékárnu.cz“ několikrát změnilo. To zapříčinilo zastaralost této aplikace. Její funkčnost není úplná a je tedy nedostačující k plnému nasazení do ostrého provozu.

Na operační systém Android existují různé, dříve zmíněné, aplikace s podobnou funkcionalitou. Tyto aplikace mají své nevýhody, které bych chtěl ve své práci vylepšit.

## 1.2 Podobné aplikace

### 1.2.1 Android

Aplikace pro rešeršní zpracování, které by měly podobné využití, jsem hledal na Google Play[4], jelikož Google Play je oficiální online distribuční služba pro Android aplikace. Porovnával jsem pouze bezplatné aplikace, které mají společné či podobné prvky s moji aplikací. Nalezl jsem 4 aplikace („Lékárny v ČR“, „HealthKartPlus“, „Medscape“ a „Drugs.com“). Většina aplikací však má pouze částečnou funkčnost oproti aplikaci „Najdi lékárnu“, kterou jsem vytvořil a pouze „Lékárny v ČR“ jsou v českém jazyce. Ostatní jsou v anglickém jazyce, což je pro uživatele z České republiky značná nevýhoda. Zahraniční aplikace často nemají informace o českých lécích a některé léky mají různé názvy v různých zemích. Takové aplikace jsou pro Čechy nepoužitelné. Také proto si myslím, že moje aplikace „Najdi lékárnu“ bude pro české uživatele velkým přínosem. To ale neznamená, že jsem se následujícími apli-

kacemi neinspiroval, ba naopak. V mé aplikaci „Najdi lékárnu“ jsem se snažil poučit z nedostatků těchto aplikací, které jsem porovnával a naopak jsem se nechal inspirovat mnohými dobrými nápady.

#### 1.2.1.1 Lékárny v ČR

Dosud je to jedna z mála českých aplikací zabývající se lékárnictvím v České republice. Je to velmi jednoduchá aplikace, která má pouze jednu funkci pro vyhledávání lékáren. Aplikace umí vyhledávat lékárny v okolí a zobrazovat je na mapě. Tedy např. vyhledá všechny lékárny na Praze 6. Dále umí vyhledávat lékárny podle názvu, konkrétní adresy nebo vyhledá lékárny, kde mají zrovna otevřeno. Zde však považuji za nedostatek, že ne u všech lékáren je uvedena otevírací doba, tudíž vyhledávání podle toho, jaká lékárna je zrovna otevřená není úplné.

Po rozkliknutí položky s konkrétní lékárnou najdeme mimo názvu a adresy také kontakty (email nebo telefon), otevírací dobu a seznam lékárníků. Aplikace má ale i rozšíření jako například to, že v aplikaci můžeme vyhledat centra odvykání kouření. Pro uživatele je určitě sympatické, že lze aplikaci velmi jednoduše ovládat, čímž jsem se v mé aplikaci „Najdi lékárnu“ inspiroval.

#### 1.2.1.2 HealthKartPlus

„HealthKartPlus“ je aplikace na zařízení s operačním systémem Android nebo iOS. „HealthKartPlus“ je velmi oblíbená aplikace, získala mnoho ocenění a má více než půl milionu stažení.

Základní funkcí je vyhledávání a objednávání léků online, dokonce umí zjistit dostupnost a cenu léku v okolí místa, kde se zrovna nacházíme. Myslím si, že další rozšíření ocení většina uživatelů a to, že aplikace dokáže najít alternativní lék podle účinné látky a také, že aplikace dává velmi podrobné informace o lécích jako např. výrobce, vedlejší účinky, interakce s jinými léky, alternativy ke zvolenému léku, atd.

Velká databáze je sice velké plus, ale jak jsem již zmínil, pro většinu uživatelů z České republiky je nepoužitelná, protože aplikace „HealthKartPlus“ je v anglickém jazyce. K tomuto nedostatku se přidává i fakt, že aplikace je určena pro uživatele z Indie a pro Čechy tedy ztrácí význam, jelikož v Indii používají jiné názvy léků apod.

#### 1.2.1.3 Medscape

„Medscape“ je komplexní lékařská aplikace, kterou využívá jak mnoho pacientů, tak i lékařů. Najdete zde nejenom vyhledávání léků, jako v aplikacích viz výše, ale i nejnovější lékařské novinky. Za velkou výhodu považuji, že umí upozornit a rozpoznat jaké léky není vhodné užívat současně. Ale především nám pomáhá určit diagnózu a případné léčení nemoci. Tzn. napíšeme jakou máme nemoc, nebo jaké máme příznaky a program to vyhodnotí a podle toho

nám poradí, co máme dělat. Zda máme vyhledat lékaře, či jaké léky nám doporučuje použít apod.

Aplikace umí téměř vše, co pacient potřebuje. Vyhledá nám interakce s jinými léky, alternativy ke zvolenému léku, vedlejší účinky, atd. Ovšem, co považuji za největší nedostatek je fakt, že se každý uživatel musí nejprve registrovat před použitím, což uživatele velmi zdržuje a otravuje. Myslím si, že to i většinu uživatelů odradí od používání „Medscape“.

#### 1.2.1.4 Drugs.com

„Drugs.com“ je poslední aplikace pro OS Android, kterou jsem vyhledal a porovnával. Jedná se také o program v anglickém jazyce, tudíž je určen širšímu spektru uživatelů, ne konkrétně uživatelům z České republiky.

Základní funkcí je opět vyhledávání léků, ale zdaleka to není funkce jediná. Tato aplikace překypuje množstvím funkcí. Jednou z nich je například vyhledávání vedlejších účinků léků, či dávkování jednotlivých léků. Napíše nám, jak jednotlivé léky spolu reagují, zda je bezpečné a vhodné je dohromady užívat apod. Rád bych však speciálně upozornil na jiné funkce, které mě zaujaly. Jedná se o vyhledávání léků podle ceny, kde nám program vyhledá, kde lék prodávají v takovém cenovém rozmezí, které zadáme. Myslím si, že toto uvítá valná většina uživatelů. A dále na rozpoznávání léků podle barvy, typu apod.

Ovšem za velký nedostatek považuji podivné vyhledávání, kdy nám program dovolí napsat pouze počáteční písmeno názvu léku a dále už musíme lék dohledat v seznamu, který nám program nabídne. To uživatele zbytečně zdržuje.

### 1.2.2 Windows Phone

Pro vyhledání aplikací k rešeršnímu zpracování produktů pro Windows Phone jsem použil oficiální obchod Windows Phone Store[5]. Porovnával jsem opět bezplatné aplikace, které mají podobný účel jako aplikace „Najdi Lékárnu“, kterou jsem vytvořil. Tentokrát jsem porovnával dva programy „Zdravotní průvodce“ a „myHealtbox“.

#### 1.2.2.1 Zdravotní průvodce

„Zdravotní průvodce“ je aplikace využívající databázi ze Zlatých stránek, která obsahuje seznam lékařů a jejich kontakty (telefon, email, internetové stránky, adresu ordinace, ordinační hodiny, ale také i hodnocení jednotlivých lékařů). Dále můžeme v aplikaci nalézt seznam nemocnic, zdravotních středisek, veterinářů a v neposlední řadě lékáren. Samozřejmě i s jejich veškerými kontakty. Program umožňuje vyhledávání podle jména nebo polohy. Vyhledá nám konkrétního lékaře, např. Petra Nováka, či zubaře na Praze 9 apod.

Tato aplikace ale nemusí sloužit pouze uživatelům z České republiky, protože existuje verze i pro Slovensko, což hodnotím jako výhodu, jelikož je pro-

gram určený pro větší spektrum uživatelů. Co však považuji za nedostatek, je nutnost připojení k internetu, aby bylo umožněno požadovaná data vyhledat. Tento fakt omezuje četnost používání této aplikace, což je rozhodně škoda.

#### 1.2.2.2 myHealthbox

„MyHealthbox“ je aplikace sloužící jako největší zdroj informací o léčivých přípravcích a zdravotnických výrobcích v Evropě. Obsahuje informace o farmaceutických výrobcích v 16 různých jazycích, ale překlad do českého jazyka je automaticky generovaný a moc dobře nefunguje, což ztrácí na ojedinělosti této aplikace. Obsahuje mapu, kde je zobrazeno přes 120 000 lékáren v 10ti evropských zemích. Také umožňuje vyhledat i kontakty jednotlivých lékáren (název, adresu a telefon).

Jako nevýhodu však považuji nejen špatný překlad do českého jazyka, jak jsem již zmínil, ale také opět nutnost připojení k internetu stejně jako u aplikace „Zdravotní průvodce“, ale navíc v tomto případě i nutnost registrace, což ještě umocňuje skutečnost, že aplikace není uživateli tolik využívaná jako aplikace bez nutnosti registrace a připojení k internetu. Tímto faktem jsem se v mé aplikaci poučil, proto je program, který jsem vytvořil „Najdi lékárnu“ schopný vyhledávat informace bez připojení k internetu a nutné registrace.

### 1.2.3 iOS

Jako poslední jsem zpracovával rešerši produktů pro iOS.

#### 1.2.3.1 Czech Pharmacies

Tato aplikace je dostupná přes App Store[6] zdarma. Velikost aplikace je velice malá, asi 1,4 MB a dá se tedy očekávat, že aplikaci nebudou uživatelé mazat kvůli vysoké velikosti při nedostatku paměti.

Aplikace využívá GPS čip, aby určil polohu zařízení a zobrazí přehledný seznam lékáren v okolí. Ten je seřazen podle vzdálenosti poboček lékáren od nejbližší po nejvzdálenější. U každé lékárny lze zjistit další informace včetně kontaktů. Například telefonní číslo, adresu, e-mail, odkaz na e-shop nebo otevírací dobu. Ne všechny lékárny však nabízejí všechny tyto údaje, u některých menších je pouze telefonní číslo a adresa.

Dále aplikace nabízí zobrazení mapy s lékárnami v okolí s použitím Google Maps, díky kterým je ovládání velmi snadné. Aplikace jako taková neumí k vybrané lékárně přímo navigovat a je nutné adresu zadat do navigační aplikace v telefonu nebo do navigace třetí strany.

Přínosnou funkci pro uživatele představuje také možnost filtrování. Vyhledávat lze podle názvu, adresy nebo vzdálenosti.

#### 1.2.3.2 Lékárny

V AppStore je také zdarma, jako všechny aplikace, které jsem porovnával, ke stažení aplikace „Lékárny“, která zobrazuje přehledně setříděný seznam lékáren ve vašem okolí bez nutnosti jakéhokoliv psaní. Aplikace poskytuje informace jako email, telefonní číslo nebo otevírací dobu. Otevírací doba umožňuje aplikaci vyhledávat právě otevřené lékárny. Aplikace nabízí velmi podrobné vyhledávání a jako předešlá aplikace „Czech Pharmacies“ možnost filtrování. To znamená, že můžeme zadat, že chceme vyhledat například jen nonstop lékárny a aplikace nám vyhledá pouze tyto lékárny.

Za jednu z výhod považuji skutečnost, že na aplikaci se dále pracuje. Nyní je ke stažení již verze 1.2, která odstraňuje nedostatky z předchozích verzí. Aplikace obsahuje asi 2 400 ověřených lékáren z celkového počtu více jak 2 800 lékáren. Z čehož vyplývá, že aplikace neobsahuje všechny lékárny. Je příjemné, že verze 1.2 je lokalizována do českého a anglického jazyka a přepínání probíhá automaticky podle systémového nastavení iPhonu.

## 1.3 Analýza požadavků

### 1.3.1 Funkční požadavky

**F1** Vyhledávání lékáren podle názvu nebo adresy

**F2** Vyhledávání lékáren v okolí

**F3** Zobrazení nalezené lékárny na mapě

**F4** Zobrazení informací o lékárnách

**F5** Vyhledávání léků podle názvu

**F6** Vyhledávání léků podle čárového kódu

**F7** Zobrazení informací o léku včetně zobrazení substitutů a interakcí

**F8** Zjištění ceny léku v okolí

**F1 Vyhledávání lékáren podle názvu nebo adresy:** Uživatel může vyhledávat lékárny podle jména nebo adresy dané lékárny. Lékárny se zobrazí podle vzdálenosti od místa, kde se uživatel nachází. Uživatel tak může hledat například i lékárny v jiném městě, než se právě nachází.

**F2 Vyhledávání lékáren v okolí:** Aplikace uživateli zobrazí lékárny, které se nacházejí do vzdálenosti, kterou si sám určí. Seznam je seřazen od nejmenší po největší vzdálenost od místa, kde se uživatel nachází. Poloha se zjišťuje pomocí služeb, které mobilní zařízení nabízí. Například

pomocí informací z mobilních stanic nebo GPS. Pokud má uživatel zablokováno zjišťování polohy, aplikace ho na to upozorní a dá mu možnost toto nastavení změnit. Pokud jsou informace o poloze k dispozici, aplikace vypočítá vzdálenost a zobrazí lékárny, které jsou do nastavené vzdálenosti.

**F3 Nalezené lékárny zobrazí na mapě:** Pro lepší orientaci si může uživatel zobrazit nalezené lékárny na mapě. Uživateli jsou zobrazeny názvy a adresy lékáren, které jsou zobrazeny na mapě. K zobrazení lékáren na mapě jsou použity Google Mapy, které jsou podporovány OS Android a poskytují mnoho užitečných sad pro zobrazení míst, bodu i celých map. Google Mapy jsou navíc velmi přesné a kvalitně zpracované.

**F4 Zobrazení informací o lékárnách:** Uživatel si může zobrazit informace o jednotlivých nalezených lékárnách. Všechny tyto informace jsou uloženy v lokální databázi telefonu, a tedy nebude potřeba přístup k internetu. Informace obsahují název lékárny, adresu lékárny, kontaktní údaje, otevírací dobu a další doplňující informace jako možnost platit kartou, možnost využití e-receptu a jiné. Uživatel si také může zobrazit lékárnu na mapě nebo zavolat rovnou do lékárny, aniž by musel opouštět aplikaci.

**F5 Vyhledávání léků podle názvu:** Stejně jako u lékáren si může uživatel vyhledat léky podle názvu. Aplikace v nápovědě nabízí názvy léků, které odpovídají již napsanému slovu.

**F6 Vyhledávání léků podle čárového kódu:** Pro jednodušší a rychlejší práci s léky je v aplikaci funkce na skenování čárových kódů. Po naskenování se uživateli zobrazí informace o léku, stejně jako by ho hledal podle názvu. Nemusí tedy lék zadávat do vyhledávání ručně, ale stačí, když vyfotí jeho čárový kód a získá všechny informace o daném léku během pár vteřin.

**F7 Zobrazení informací o léku včetně zobrazení substitutů a interakcí:** Po vybrání léku nebo naskenování kódu se uživateli zobrazí informace o daném léku. Mezi informacemi je například cesta podání, léková forma nebo síla léku. Uživatel si může tyto i další informace stáhnout do svého zařízení. Například příbalový leták, souhrnné informace nebo obal daného léku.

Aplikace k tomuto léku umí najít jeho substituty a interakce. Substituty mají stejnou účinnou látku, cestu podání a lékovou formu. Naopak interakce jsou léky, které by se neměly užívat současně s daným lékem.

**F8 Zjištění ceny léku v okolí:** Uživatel si může zobrazit cenu léku, buď v okolních lékárnách, nebo podle měst. Tato funkce nenabízí online nákup, pouze informaci o ceně v jednotlivých lékárnách.

### 1.3.2 Nefunkční požadavky

**N1** Data uložena v aplikaci

**N2** Synchronizace dat se serverem „najdi-lékárnu.cz“

**N3** Intuitivní ovládání

**N4** Přehlednost zobrazených informací

**N5** Uživatelsky přívětivé rozhraní

**N1 Data uložena v aplikaci:** Všechny potřebná data by měla být uložena v aplikaci, aby byla aplikace funkční, i když zařízení není připojeno k internetu. V dnešní době se současnými výkony mobilních zařízení bude aplikace pracovat rychleji, než kdyby musela každý požadavek posílat na server a čekat na odpověď.

V databázi jsou uložena data o lékárnách, lécích, substitutech a interakcích. Tato data se automaticky aktualizují, pokud jsou nějaké změny na serveru, jinak jsou tato data pouze pro čtení. Uživatel je může používat pouze prostřednictvím této aplikace.

**N2 Synchronizace dat se serverem „najdi-lékárnu.cz“:** Při zapnutí aplikace se zjistí, zda je zařízení připojeno k internetu a případně aplikace pošle dotaz na server, zda existuje nová verze dat. Pokud ne, nic se neděje, pokud ano, aplikace pošle dotaz na změny a ty pak uloží do databáze. Toto vše se děje na pozadí aplikace, tedy uživatele to nijak neovlivní při práci s aplikací.

**N3 Intuitivní ovládání:** Uživatel by se neměl v aplikaci ztratit a vždy by měl vědět na co kliknout v průchodu aplikací. Tomu pomáhá pojmenování jednotlivých obrazovek a zpětná navigace v ActionBaru.

**N4 Přehlednost zobrazených informací:** Aplikace zobrazuje mnoho informací, a proto je důležité, aby se v nich uživatel dobře vyznal. V mé aplikaci jsem se o to snažil, a proto uživateli nabízím pouze hlavní informace a nezahrnuji ho informacemi pro něj nedůležitými.

**N5 Uživatelsky přívětivé rozhraní:** Uživatelské rozhraní se snaží dodržovat zásady a doporučení stanovené na oficiálních stránkách pro vývoj Android aplikací. Na stránkách se popisují konstrukční zásady, užití barev, stylů a rozměrů.

## 1.4 Analýza případů užití

Výše uvedené funkční požadavky se přímo promítají do případů užití. Tento diagram slouží pro návrh UI i pro tvorbu scénářů pro usability testing, který ověří, zda je daný návrh pro uživatele dostatečně jednoduchý, srozumitelný a zda je uživatel schopen v aplikaci dosáhnout všech daných cílů. Dále slouží ke zpřesnění odhadů pracnosti implementace.

Model případů užití se skládá ze seznamu účastníků a diagramů případů užití. V seznamu účastníků je jen jedna osoba, a to uživatel.

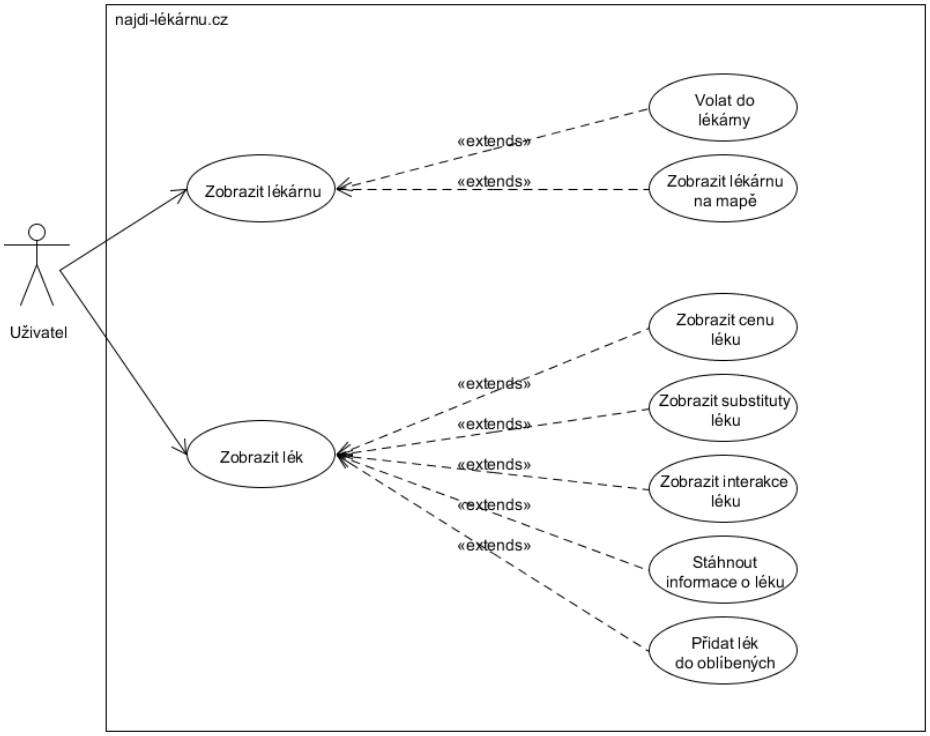
Uživatel se může pomocí záložek dostat do tří hlavních kategorií:

**Lékárny:** zobrazuje seznamy hledaných nebo nejbližších lékáren.

**Léky:** zobrazuje hledané léky.

**Oblíbené:** zobrazuje léky, které si uživatel označí jako oblíbené, aby k nim měl rychlý přístup.

Při vytváření tohoto diagramu jsem kladl velký důraz na nejčastější činnosti, které lze od uživatele očekávat.




Obrázek 1.1: Diagram případů užití

Kapitola 2

# Návrh

Tato kapitola se zabývá návrhem aplikace. Do návrhu aplikace patří návrh architektury, uživatelského rozhraní, databáze, synchronizace a výběr nástrojů pro vývoj.

## 2.1 Architektura OS Android

Architektura OS Android je rozdělena do pěti vrstev: jádro, knihovny, aplikační framework, běhové prostředí a aplikace. Každá vrstva má svoji funkci. Vrstvy zároveň nemusí být přímo odděleny a mohou se částečně prolínat s ostatními vrstvami.

Nejnižší vrstvou je vlastní jádro OS, které propojuje používaný hardware se zbytkem softwaru ve vyšších vrstvách. Jádro OS Androidu je postaveno na Linuxu a využívá mnoho jeho vlastností, jako například správu paměti, správu sítí nebo správu procesů, která se používá, pokud běží více aplikací najednou. Výběr systému Linux jako jádro OS Android byl odůvodněn tím, že se Linux snadno sestaví na různých zařízeních a je tedy zaručena přenositelnost mezi zařízeními.

Další vrstvou jsou knihovny. Tyto knihovny jsou psány v jazyce C a C++. Funkce těchto knihoven jsou poskytovány pomocí Android Application Framework.

Třetí vrstvou je takzvaná Android Runtime vrstva, která obsahuje aplikační virtuální stroj. Dalvik Virtual Machine (DVM) je registrově orientovaná architektura, která využívá výše uvedených vlastností linuxového jádra. DVM také pomáhá při překladu aplikace, která probíhá zkompilováním zdrojového kódu v jazyce Java do Java byte kódu a následně se Java byte kód pomocí Dalvik kompilátoru znova překompiluje a výsledný Dalvik byte kód je spuštěn na DVM.

Aplikační vrstva poskytuje přístup ke službám, které mohou být následně použity přímo v aplikacích. Mezi nejdůležitější služby patří například Sada prvků View, které se používají pro sestavení uživatelského rozhraní. Další

službou je např. Content providers, která umožňuje přístup k datům jiných aplikací a mnohé další užitečné služby.

Nejvyšší vrstvou je samotná aplikace, která je používána běžným uživatelem. Aplikace na OS Android se stahují z oficiálního obchodu Google Play Store. Některé aplikace, jako například webový prohlížeč, kalendář nebo kontakty jsou již v mobilu předinstalovány. Tyto základní aplikace jsou doplněny o aplikace samotných výrobců zařízení, tedy každý mobilní telefon s OS Android nemusí mít ve výchozím nastavení stejné aplikace. [7]

## 2.2 Vývoj aplikací pro OS Android

Aplikace pro OS Android jsou psány v jazyce Java. Pro vývoj se používají 2 hlavní vývojové prostředí, a to Android Studio nebo Eclipse. Já jsem vybral Android Studio, jelikož je to oficiální vývojové prostředí od firmy Google[8]. Android Studio je rychlé, má menší nároky na výkon a obsahuje lepší našeptávač.

Java je objektově orientovaný programovací jazyk od firmy Sun Microsystems z roku 1995. Jeho výhodou je přenositelnost mezi různými systémy, tedy i na mobilní zařízení.[9]

Dále je potřeba k vývoji Android aplikací Android Software Development Kit (SDK) a Java Development Kit (JDK).

SDK obsahuje nástroje pro vývoj aplikací pro OS Android, jako jsou například USB ovladače, emulátory, dokumentace a hlavně samotný Android. JDK je soubor základních nástrojů a knihoven pro vývoj aplikací. Součástí JDK je Java Runtime Enviroment (JRE), které slouží pro spouštění aplikací a vývojových nástrojů. Dále JDK obsahuje překladač, debugger a další. V aplikacích pro OS Android jsou tři hlavní typy souborů, a to sources, resources a soubor Manifest.

Sources jsou zdrojové kódy s příponou .java, kde se píše programová logika, stejně jako při programování v Javě.

Resources jsou prostředky, které podporují aplikaci. Patří mezi ně například obrázky, definice barev a další prostředky, které představují, jak bude aplikace vypadat. Hlavní součástí zdrojových kódů jsou takzvané layout soubory, které se píší jazykem XML a představují, jak budou vizuelně vypadat jednotlivé obrazovky aplikace.

Manifest je hlavní soubor, který musí mít každá Android aplikace. Soubor je uložen v kořenovém adresáři, je psaný stejně jako layout pomocí XML a dává systému základní informace o aplikaci, které musí systém mít před jejím spuštěním. Jsou zde definované všechny použité prvky a oprávnění, která aplikace potřebuje.[10]

## 2.3 Verze OS Android

V říjnu roku 2008 vyšla první verze OS Android. Od té doby každý rok přibyla minimálně jedna verze. Nové verze opravují chyby předchozích verzí a zároveň přidávají nové funkce. Na obrázku 2.1[11] je vidět, že procentuální poměr užívaných mobilních zařízení s OS Andorid se stále zvyšuje. Koncem roku 2014 lehce stoupl prodej zařízení s iOS, což bylo pravděpodobně způsobeno vydáním nového iPhonu 6. To mělo určitý vliv na prodej zařízení s OS Android, ale z grafu je vidět, že tyto výchylky se stávaly s každým vydáním nového iPhonu. V současné době má OS Android asi 80% uživatelů mobilních zařízení. Od 3. verze se každá nová verze pojmenovává podle některé sladkosti začínající na následující písmeno z abecedy oproti předchozí verzi. Tedy od nejstarší po nejnovější se verze jmenují: Cupcake, Donut, Eclair, Froyo, Gingerbread, Honeycomb, Ice Cream Sandwich, Jelly Bean, KitKat a Lollipop. [12]

Tolik verzí samozřejmě způsobilo, že vývojáři aplikací pro OS Android musí své aplikace přizpůsobovat jak novým, tak starým verzím, a to bývá občas trochu problém. Toto částečně řeší Google, který vytvořil support knihovny, které převádějí nové funkce do starších verzí. Na obrázku 2.2[13] a v tabulce 2.1[13] je vidět v jakém procentuálním množství jsou jednotlivé verze zastoupeny. Android podporuje dopřednou kompatibilitu, což znamená, že aplikace bude fungovat i na verzích, které teprve vyjdou. [14]

Moje aplikace bude podporovat většinu stále používaných verzí, tedy od verze Froyo, a díky dopředné kompatibilitě bude fungovat i na všech novějších verzích.

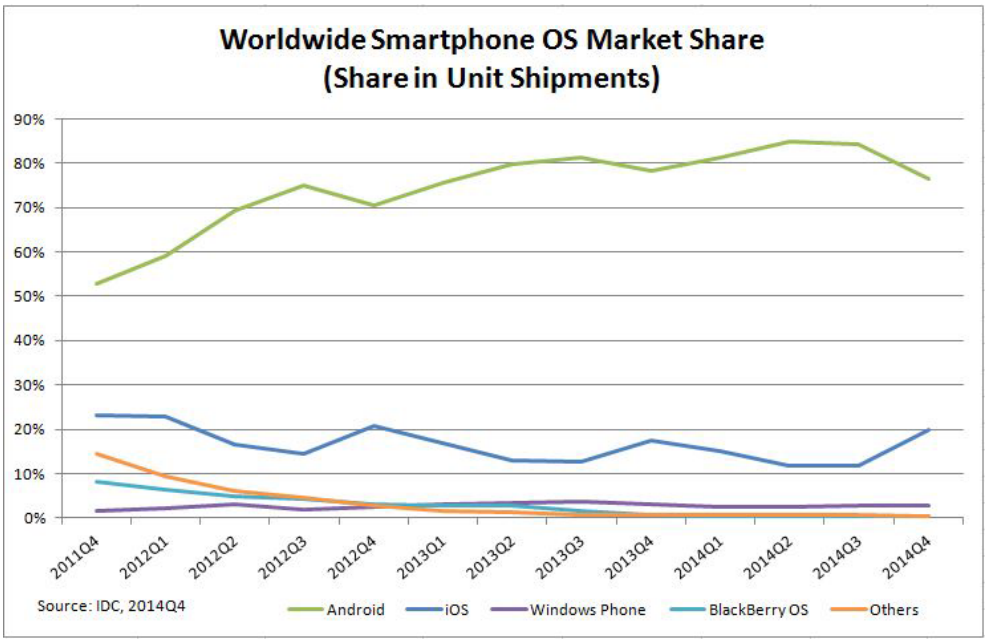
| **Verze** | **Název** | **API** | **Distribuce** |
|---|---|---|---|
| 2.2 | Froyo | 8 | 0.4% |
| 2.3 | Gingerbread | 10 | 6.4% |
| 4.0 | Ice Cream Sandwich | 15 | 5.7% |
| 4.1 - 4.3 | Jelly Bean | 16 - 18 | 40.7% |
| 4.4 | KitKat | 19 | 41.4% |
| 5.0 - 5.1 | Lollipop | 20 - 22 | 5.4% |

Tabulka 2.1: Poměr verzí OS Android

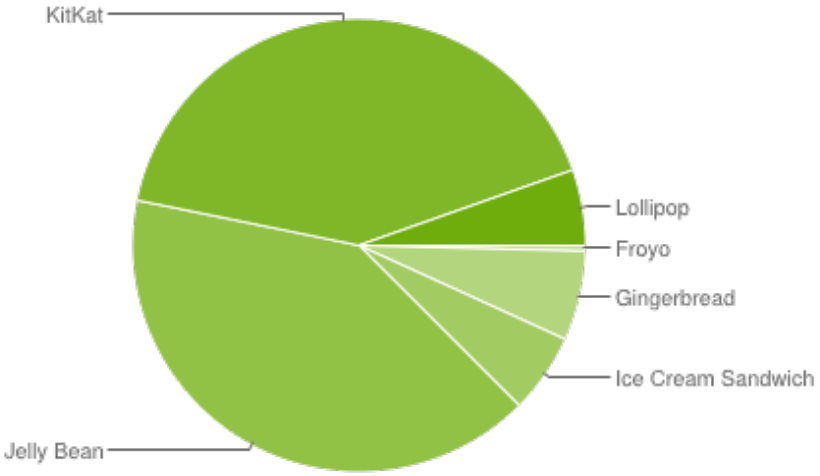
## 2.4 Architektura aplikace pro OS Android

### 2.4.1 Aktivity

Aktivita je jedna obrazovka, kterou uživatel vidí. Obsahuje grafické uživatelské rozhraní pro interakci s uživatelem. Aplikace obsahuje více aktivit a uživatel si svým průchodem aplikací jednotlivé aktivity zobrazuje. Aktivity si mohou mezi sebou předávat informace a tím spolu komunikovat a spolupracovat. Každá aktivita má svůj životní cyklus, který můžeme rozdělit do stavů[15]:




Obrázek 2.1: Poměr OS v prodaných zařízeních




Obrázek 2.2: Poměr verzí OS Android

**Activity starts:**
Počátek, aktivita je inicializována.

**Activity is running:**
Aktivita je již zobrazena a uživatel s ní může komunikovat. V jediném okamžiku může být právě jedna aktivita v tomto stavu.

**Process is killed:**
Activity Manager zrušil aktivitu z důvodu nedostatku paměti.

**Activity is shut down:**
Activity Managerem ukončil aktivitu a ta již nevyužívá žádnou paměť.

### 2.4.2 Service

Komponenta service představuje proces, který běží na pozadí. Používá se k vykonávání úkolů, které zabírají více času nebo přistupují ke zdrojům, které nejsou uložené v zařízení, a tedy není známá doba odezvy např. připojení k serveru. Pokud by se komponenta service nepoužívala a aplikace by čekala na odezvu ze serveru, v tu chvíli by aplikace nebyla schopna reagovat na příkazy uživatele.

### 2.4.3 Content provider

Content provider je rozhraní aplikace, které umí sdílet data, jak mezi aplikacemi, tak i mezi jednotlivými aktivitami. Aplikace uchovává data v souborech nebo v databázích. K těmto souborům pomocí content provideru mohou přistupovat i jiné aplikace, což je velmi výhodné.

### 2.4.4 Broadcast receiver

Broadcast reciever je komponenta sloužící k reagování na oznámení. Podle určení na ně reaguje, například výpisem na stavový řádek nebo spuštěním jiné komponenty. Příklad použití broadcast recieveru může být reakce na oznámení o nízkém stavu baterie, o zachycení fotografie, doručení SMS zprávy nebo stažení dat.

## 2.5 Návrh uživatelského rozhraní

Návrh grafického uživatelského rozhraní pro aplikaci vychází především z části „Případy užití“. Během návrhu uživatelského rozhraní jsem se držel rad z webu Android Design[16]. Toto je velmi důležitá část celé práce, protože i sebelepší aplikace, která se špatně ovládá, by nemusela mít žádný úspěch.

Pro základní návrh jsem použil nástroj Fluid UI[17]. Je to nástroj, který umožňuje rychlé vytváření prototypů uživatelského rozhraní. Výhodou Fluid UI je to, že má webové rozhraní a může tedy lépe a rychleji probíhat diskuze nad již vytvořeným řešením a jeho následnými úpravami. Prototyp nenese žádnou funkcionalitu, ale lze jím procházet jako klasickou hotovou aplikací. Změny v grafickém rozhraní lze provádět velmi snadno, proto bylo rozhodnuto navrhnout přímo přibližnou podobu finálního vzhledu.

Mezi nefunkčními požadavky jsou i 3 požadavky spadající do uživatelského rozhraní, a jsou to: „Intuitivní ovládání“, „Přehlednost zobrazených

informací“ a „Uživatelsky přívětivé rozhraní“. Během tvorby návrhu uživatelského rozhraní jsem se inspiroval u jiných aplikací OS Android. OS Android má mnoho mobilních zařízení, jak malé mobilní telefony, tak velké tablety. Proto jsem se rozhodl vytvořit uživatelské rozhraní pro jakoukoliv velikost obrazovky. Pro malé obrazovky se budou zobrazovat seznamy a detaily samostatně a pro velké obrazovky, například pro tablety, které se většinou používají v orientaci naležato, se budou vedle seznamu zobrazovat i detaily lékáren či léků.

Jsou dva základní typy vzhledů aplikací pro zobrazování informací. První typ zobrazí uživateli nabídku možností, ze kterých si uživatel vybere viz obrázek 2.3, kde se tento typ nachází na levé straně. Výhodou tohoto typu je, že po vybrání jedné možnosti se s ostatními již nemusí zabývat. Nevýhoda je však, že se nemůže jedním kliknutím dostat do jiné možnosti. Nejprve se musí vrátit a poté vybrat požadovanou možnost.

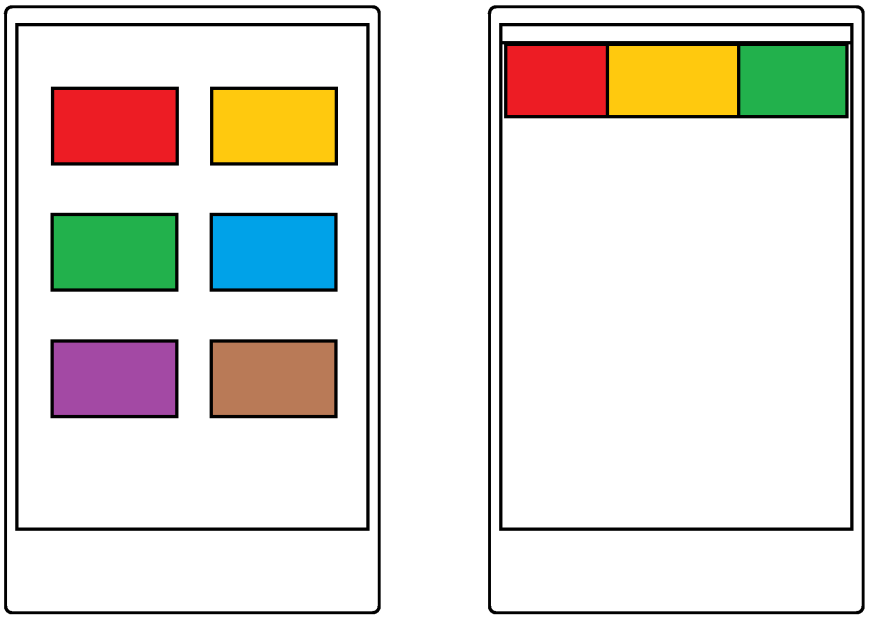
Tento problém řeší druhá varianta, která obsahuje záložky, mezi kterými si může uživatel jednoduše vybírat viz obrázek 2.3, kde se tento typ nachází na pravé straně. Mezi novější technologie patří i takzvané SwipeTab, kde je možno mezi záložkami přecházet pomocí tahu, buď zprava doleva pro přejití na další záložku, nebo naopak pro přejití na předchozí záložku. U tohoto modelu je naopak nevýhodou, že záložky zabírají určitou část obrazovky.

V mé aplikaci jsem se po zvážení výhod a nevýhod rozhodl pro druhou variantu, která bude pro tuto aplikaci výhodnější a také proto, že druhá varianta lépe vyhovuje velkým obrazovkám používaným na tabletech. Každá záložka bude mít své označení pro lepší orientaci uživatele a aktuální záložka bude zvýrazněna, aby se uživatel neztratil. Při zapnutí aplikace se automaticky zobrazí první záložka. Záložky jsou rozděleny na Lékárny, Léky a Oblíbené.

### 2.5.1 Lékárny

Obrazovka s lékárnami se zobrazí při zapnutí aplikace. Je to první záložka a první věc, kterou uživatel uvidí, proto je velmi důležité, aby byla dobře navržena a uživatele upoutala. V seznamu lékáren se automaticky zobrazí odkazy na lékárny, které jsou v dané vzdálenosti. Každý odkaz na lékárnu obsahuje její název, adresu a barevně označenou vzdálenost od uživatele. U vzdálenosti můžeme vidět 3 druhy barev, zelenou pro vzdálenost do jednoho kilometru, oranžovou pro vzdálenost mezi jedním a třemi kilometry a červenou, která je u lékáren, které jsou dál než tři kilometry. U odkazů na lékárny jsou zobrazeny pouze hlavní informace, aby byl seznam pro uživatele přehledný. Nad seznamem se nachází tlačítko simulující  mapy. Pod tímto tlačítkem se nachází obrazovka, která je tvořena pomocí Google Maps a jsou na ní zobrazeny lékárny ze seznamu.

V horní části na ActionBar jsou tři tlačítka, první pro vyhledávání, druhé pro znovuzobrazení lékáren v okolí a poslední je nastavení. Tato obrazovka je zobrazena na obrázku 2.4.



Obrázek 2.3: Typy zobrazení obsahu

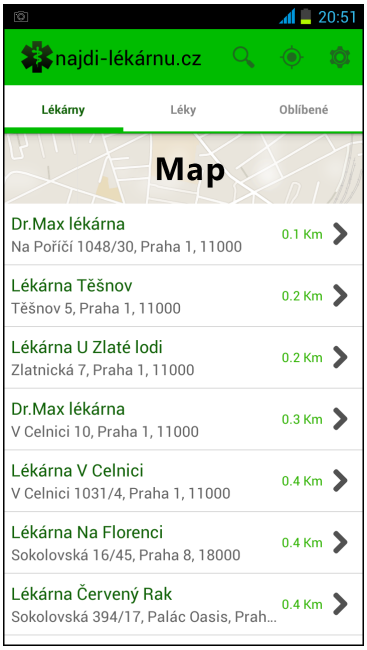
### 2.5.2 Léky

Obrazovka s léky se nachází napravo od lékáren, tedy jako druhá v pořadí v seznamu záložek. Je velmi jednoduchá, nachází se zde pouze vyhledávání. Pokud však zkusíme něco hledat, tak se nám zobrazí léky odpovídající zadanému výrazu. Léky jsou v seznamu pod sebou a jsou řazeny podle abecedy. V každé položce seznamu je název léku a pod ním jeho forma. Na pravé straně jsou základní informace jako síla, počet léku v balení nebo jestli je lék na recept nebo volně prodejný.

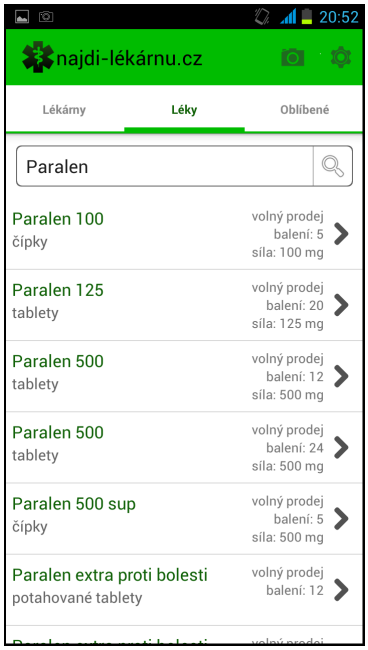
V horní části je ikona zobrazující fotoaparát, přes kterou přistupujeme ke skenování čárových kódů. Tato obrazovka je zobrazena na obrázku 2.5.

### 2.5.3 Oblíbené

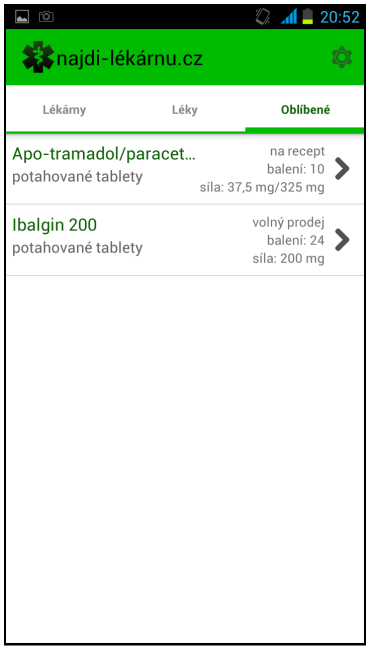
V této třetí a zároveň poslední záložce se nám zobrazují léky, které jsme přidali do oblíbených. Zobrazují se nám jako seznam léků pod sebou a řádky mají stejnou strukturu jako řádky v záložce „Léky“. Tato obrazovka je zobrazena na obrázku 2.6.




Obrázek 2.4: Záložka lékárny




Obrázek 2.5: Záložka léky




Obrázek 2.6: Záložka oblíbené

### 2.5.4 Detail lékárny

Detail lékárny bude jako samostatná aktivita, na kterou se aplikace bude odkazovat ze seznamu lékáren v záložce „Lékárny“. Jako hlavní prioritu jsem si stanovil, aby zobrazovaná obrazovka byla pro uživatele přehledná. Aby informace zobrazované na obrazovce byly stručné, ale abych zároveň uvedl vše podstatné. Proto u toho, co jsem navrhl, jsem se držel známého rčení, méně je více. Detail lékárny jsem navrhl velmi jednoduše, je zobrazen pouze pomocí dvou barev, které pomáhají uživateli se strukturováním obsahu. Tento obsah jsem rozdělil na čtyři části a to základní informace, kontakty, otevírací dobu a obecné informace. Mezi základní informace patří název a adresa lékárny, u kontaktů zobrazuji telefon a emailovou adresu a v obecných informacích jsou zahrnuty informace o bezbariérovém přístupu, možnosti platit kartou, specializaci lékárny a další. Navíc jsem do této stránky zabudoval možnost přímého volání do lékárny a zobrazení lékárny na mapě. Tyto informace a funkce jsou na obrazovce rozmístěny tak, aby se v nich uživatel snadno orientoval. Tato obrazovka je zobrazena na obrázku 2.7.

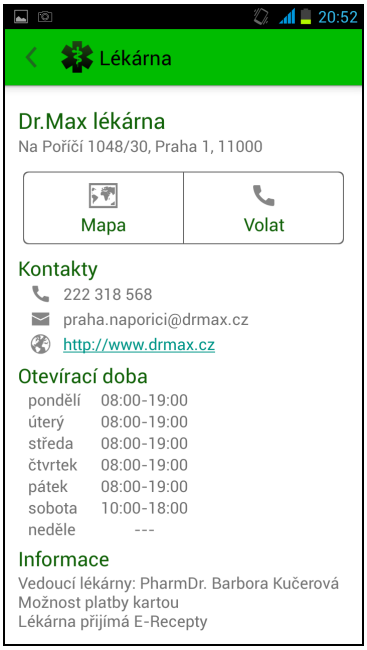
### 2.5.5 Detail léku

Na tuto obrazovku se uživatel dostane výběrem konkrétního léku ze seznamu hledaných nebo oblíbených léků. Tato obrazovka má stejné stylování jako detail lékárny, a tím se udržuje konzistentnost vzhledu aplikace. Uživatel se zde dozví mimo názvu léku také cestu podání, lékovou formu, sílu, velikost balení a další důležité informace, které jsou přehledně zobrazeny ve strukturovaném seznamu. Tato obrazovka má funkce jako hledání substitutů nebo interakcí, dále hledání cen daného léku a stahování informací ve formě .pdf. Všechny tlačítka, která vyvolávají tyto funkce, mají stejný styl, který by měl být pro uživatele přívětivý a hlavně přehledný. Nachází se zde také funkce, která přidá tento lék do seznamu oblíbených léků. Toto tlačítko je značeno hvězdou, která je pro přidávání objektu do seznamu oblíbených obvyklá a uživatelé jsou na toto značení zvyklí. Tato obrazovka je zobrazena na obrázku 2.8.

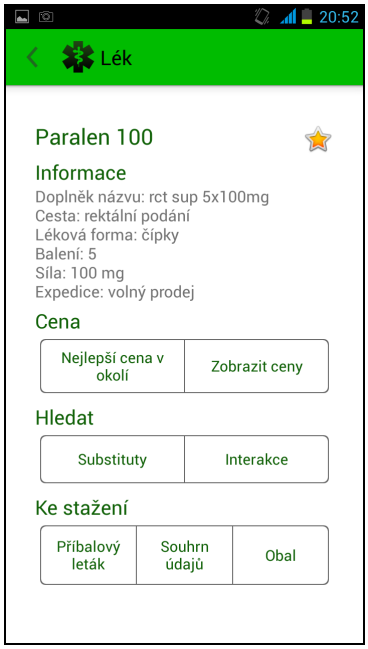
## 2.6 Databáze

Jak již bylo uvedeno v požadavcích, aplikace musí mít uložena všechny data ve vlastní databázi. Data musejí být aktuální, tedy být stejná jako na serveru. Samotná data nesmí mít uživatel možnost měnit, pouze je číst. Jediná data, která může uživatel měnit jsou, jestli je lék v seznamu oblíbených či nikoliv.

Je mnoho způsobů, jak uložit data do paměti telefonu, ale jelikož je potřeba uložit desetitisíce záznamů, je nejvýhodnější z hlediska rychlosti použít relační databázi. V knihovnách je k dispozici relační databáze SQLite[18], kterou jsem se rozhodl použít. Pro navrhovanou aplikaci je potřeba vytvořit




Obrázek 2.7: Detail lékárny




Obrázek 2.8: Detail léku

novou databázi. Během návrhu jsem zjistil, že některá data, která přicházejí ze serveru, jsou zbytečná a nebylo tedy nutné je zahrnovat do databáze.

**Datové typy v SQLite[19]**

**NULL:** Značí prázdnou hodnotu, položka ve sloupci nebyla vyplněna.

**INTEGER:** Kladná i záporná celá čísla, rozsah je až 8 B.

**REAL:** Číslo s plovoucí desetinnou čárkou, rozsah je 8 B.

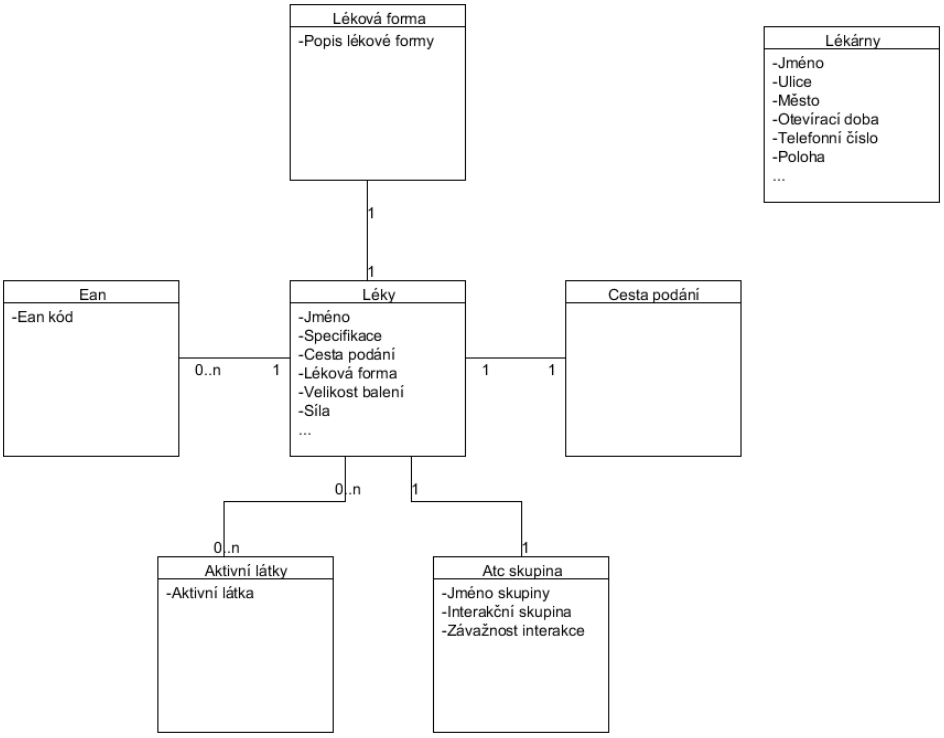
**TEXT:** Reprezentuje text, jedná se o řadu znaků zakončených hodnotou nula. Výchozí nastavení maximální délky je jedna miliarda bytů.

**BLOB:** Není určitý datový typ, uloží se přesně to, co bylo dáno na vstup. Limity pro velikost jsou stejné jako u typu TEXT.

### 2.6.1 Tabulky

Tabulky databáze jsem přizpůsoboval příchozím datům, která přijímám ze serveru.

**activeSubstance:** Tabulka activeSubstance uchovává data o účinných látkách jednotlivých léků. Tato tabulka je používaná například k vyhledávání substitutů.




Obrázek 2.9: Doménový model

**atc:** Jméno tabulky atc je zkrácený název pro Anatomicko-terapeuticko-chemickou skupinu. ATC je mezinárodním systémem pro třídění léčiv, kde jsou jednotlivá léčiva tříděna podle účinků na jednotlivé orgány. V aplikaci používám tyto data pro hledání interakcí mezi jednotlivými léky.

**ean:** Tato tabulka obsahuje seznam čárových kódů jednotlivých léků.

**form:** Form představuje formu léku (např. jestli se jedná o gel nebo oční roztok a jiné). Tato tabulka pomáhá při vyhledávání substitutů k léku.

**medicine:** Medicine je jedna ze dvou hlavních tabulek. Uchovává většinu informací o jednotlivých lécích. Kromě dat, která přijímá aplikace ze serveru obsahuje i sloupec pro označení, jestli je lék zařazen do oblíbených či nikoliv.

**pharmacies:** Toto je druhá hlavní a zároveň nejrozsáhlejší tabulka, která má v sobě všechny potřebné údaje o jednotlivých lékárnách.

**route:** Tabulka route obsahuje jednotlivé způsoby podání léku.

### 2.6.2 Dotazy

V OS Android vytváříme a posíláme dotazy na databázi pomocí SQLite adaptéru. Tento adaptér může vytvořit, změnit nebo smazat jednotlivé tabulky. Má také metody, které spouštějí SQL příkazy a provádějí další úkoly správy databáze. S databází se v projektu bude komunikovat výhradně přes tento adaptér.

## 2.7 Synchronizace

Pro chod celého systému je důležité implementovat synchronizaci mezi aplikací a serverem. Pokud by aplikace zobrazovala neplatná nebo zastaralá data, mohlo by to vést ke ztrátě zájmu uživatelů o tuto aplikaci.

Nabízí se zde otázka, zda uživateli oznámit, že je potřeba aktualizovat databáze, nebo tuto akci provést automaticky. První možností je při zapnutí aplikace zjistit, zda jsou na serveru nějaká nová data a pokud ano, dát uživateli vědět. Ten se bude moci rozhodnout, zda databázi aktualizovat nebo ne.

Druhý způsob opět kontroluje při zapnutí aplikace, zda neproběhly nějaké změny na serveru a pokud ano, tak by byly staženy a zaznamenány do databáze. Výhodou tohoto způsobu je, že nezatěžuje uživatele dalšími upozorněními. Nevýhodou je, že aplikace může začít stahovat data ve chvíli, kdy je uživatel připojen na pomalé síti nebo může aplikaci v průběhu aktualizace vypnout a tím se aktualizace nedokončí.

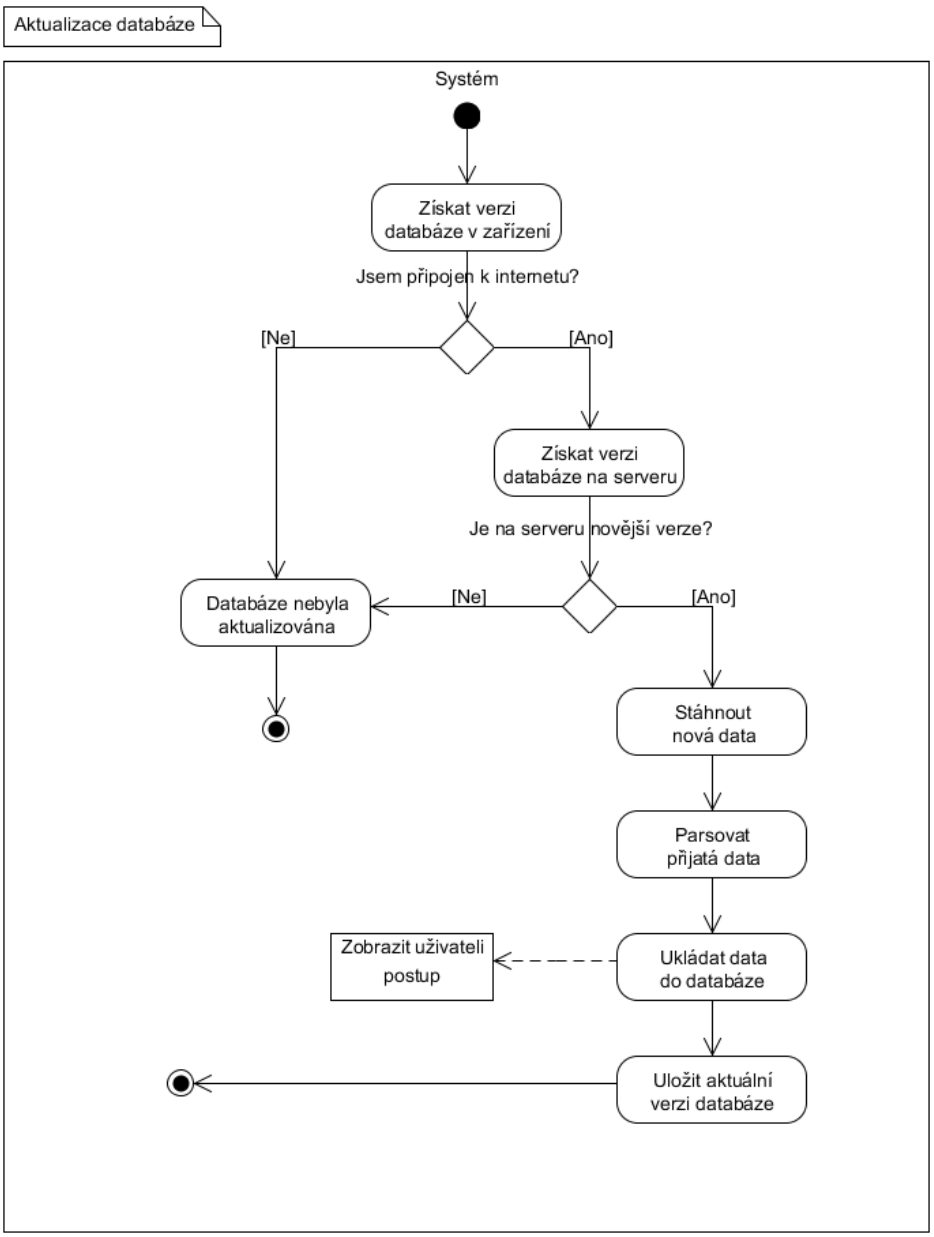
Po zvážení výhod a nevýhod jsem vybral a upravil druhou možnost. Druhou možnost jsem vybral proto, že nechci, aby aplikace zahrnovala uživatele informacemi, které jsou pro něho nedůležité. Musí se ale dbát na jistá omezení s tímto spojená. Přesný průběh synchronizace je uveden v podkapitole „Synchronizace dat“.

S tímto rozhodnutím souvisí i další možnost, jestli se má uživateli zobrazovat pokrok aktualizace nebo ne. Díky předešlému rozhodnutí jsem se rozhodl pokrok aktualizace nezobrazovat. To bude mít za následek, že uživatel nebude rušen informací o aktualizaci a tím se opět usnadní práce s touto aplikací.

### 2.7.1 Synchronizace dat

Synchronizace bude pravidelnou událostí, během které se budou upravovat data v databázi, aby odpovídala aktuálním datům na serveru. Aplikace po svém startu zkontroluje, zda je připojená na internet a pokud ano, zjistí zda je připojená na síť Wi-fi. Z toho se dá usoudit, že se nachází na jednom určitém místě a neměl by nastat výpadek sítě. Připojení k Wi-fi síti se zjišťuje z důvodu, že aktualizovaných dat může být velké množství. Pokud není mobilní zařízení připojeno na Wi-fi síť, aplikace automaticky přeruší synchronizaci a dále pracuje jen s daty, která má uložena v databázi. Pokud je připojení potvrzeno, aplikace zjistí verzi databáze, která je uložená v mobilním zařízení

a porovná ji s verzí na serveru. Pokud se neshodují, uloží se nově přijatá data do databáze.




Obrázek 2.10: Diagram aktivit: Aktualizace databáze

Kapitola 3

# Implementace

## 3.1 Skenování čárových kódů

Využil jsem toho, že tvořím aplikaci na mobilní zařízení, které má jiné vlastnosti než stolní počítač. Hlavním rozdílem je mobilita zařízení. Mobilní telefon můžeme mít vždy při ruce a používat ho kdekoliv na cestách na rozdíl od počítačů, které jsou buď stále na jednom místě, nebo je sice přenášíme jako notebooky, ale manipulace s nimi na cestách není nikterak příjemná. Dále jsem využil toho, že převážná většina telefonů s operačním systémem Android má zabudovaný fotoaparát. Proto jsem se rozhodl uživateli ulehčit práci a přidat funkci pro skenování čárových kódů. Tuto funkci mohou uživatelé využít doma, pokud nechtějí zadávat dlouhé jméno léku nebo například v samotných lékárnách, kde rychlým vyfocením čárového kódu zjistí jeho cenu a porovnají s cenou v okolních lékárnách. Android nemá žádné vestavěné knihovny, které by umožňovaly skenování čárových kódů, proto jsem musel hledat jinou variantu pro umožnění skenování.

První možnost je nejjednodušší, aplikace zjistí pomocí Intentu, jestli existuje v zařízení jiná aplikace, která umí skenovat čárové kódy a požádá ji, aby akci provedla za ni. Výhodou je, že aplikace nemusí obsahovat žádné další komponenty, a tedy zmenšuje celkovou velikost aplikace. Problémem je, že pokud uživatel žádnou takovou aplikaci v zařízení nemá a chce skenovat čárové kódy, musí nejprve aplikaci stáhnout. To může být problém ve chvíli, kdy uživatel nemá přístup na internet nebo potřebuje skenovat okamžitě.

Druhou možností je využít již existující knihovny. Pro skenování čárových kódů je vhodná knihovna například ZXing[20], která je napsaná v jazyce Java a je využívána mnoha jinými programy na skenování čárových a QR kódů. Výhodou tohoto řešení je, že aplikace nepotřebuje jiné aplikace jako předchozí možnost, ušetří uživateli čas, protože skenování kódů je zabudováno přímo v aplikaci a skenování může být prováděno kdykoliv, i když uživatel není připojen na internet. Jedinou nevýhodou je, že by se zvýšila celková velikost aplikace o velikost ZXing knihovny.

Třetí možností je vytvořit vlastní algoritmus na skenování čárových kódů. Tato možnost je nejnáročnější a tento problém je již vyřešen a úspěšně funguje. Vlastní implementaci takového řešení jsem zamítl, protože by to bylo příliš náročné a není to tématem této bakalářské práce.

Rozhodl jsem se pro druhou možnost, tedy využít již existujících knihoven a skenování zapojit přímo do aplikace.

## 3.2 Ukládání dat

Aby mohli uživatelé používat aplikaci i bez připojení k internetu, je potřeba aby byly informace uloženy přímo v aplikaci. K tomu aplikace využívá databázi, která umožňuje uložit velké množství strukturovaných informací. Databáze obsahuje informace o lékárnách a lécích. Je potřeba zajistit, aby uživatel mohl aplikaci používat okamžitě po jejím stažení. Proto se databáze neplní až při prvním spuštění, ale je již předvyplněná daty, která uživateli umožňují okamžité používání aplikace. Data sice nemusejí být aktuální, ale data se po připojení k internetu zaktualizují a uživatel bude mít již plnohodnotnou aktualizovanou databázi.

### 3.2.1 Databáze SQLite

Databáze je ideálním místem pro ukládání dat, které mají být strukturované a dají se opakovaně číst. Pro Android aplikace se používá databáze SQLite.

SQLite je Open Source databáze, která podporuje standardní relační funkce databáze jako jsou například: SQL syntaxe a transakce. Je to malá knihovna, která se připojí k aplikaci a pomocí jednoduchého rozhraní ji lze využívat. Každá databáze má svůj vlastní soubor, který obsahuje jednotlivé tabulky s jednotlivými záznamy. Databáze je velmi malá, průměrně 275kB, a je také velmi rychlá, viz tabulka 3.1 [21]. Jelikož je SQLite vložena do každého zařízení s operačním systémem Android, použití databáze nevyžaduje žádné procedury k nastavení nebo její správu. Musí se pouze definovat SQL příkazy pro tvorbu a aktualizaci databáze. Velikost SQLite databáze je omezena na 2 terabyty, což v této aplikaci nebude ani zdaleka dosaženo a počet řádků v jedné tabulce je omezen na $2^{64}$, ale to je pouze teoretické, protože paměť databáze dojde dříve, než se tabulka dokáže naplnit.

| **Typ databáze** | **Rychlost** |
|---|---|
| PostgreSQL | 4.614 s |
| MySQL | 1.270 s |
| SQLite 2.7.6 | 1.121 s |

Tabulka 3.1: Rychlost databází: 5000krát volání SELECT

### 3.2.2 Práce s databází

Všechny potřebné třídy pro práci s databází jsou uloženy v balíčku „Android.database“. Pro správu veškerých operací týkajících se databáze je definovaná třída „SQLiteOpenHelper“. Tato třída poskytuje abstraktní vrstvu mezi databází a aktivitami, která je schopna vytvořit a aktualizovat databázi.

SQLiteDatabase je základní třída pro práci s SQLite databází v Androidu a poskytuje metody pro otevření, dotazování, aktualizování a zavírání databáze. Konkrétně metody insert(), update() a delete(). Dotazy mohou být vytvořeny pomocí metod rawQuery() a query() nebo přes třídu „SQLiteQueryBuilder“.

Zde získískávám lékárny pomocí metody query().

```
Cursor cursor = db.query(TABLE_PHARMACIES,
    null, pharmacyCode "=?",
    new String[] { String.valueOf(id) },
    null, null, null, null);
```

Zde je pro získání stejného výsledku použita metoda rawQuery().

```
String strWhere =  " WHERE pharmacyCode='"+id+"'";
String query="SELECT*FROM"+TABLE_PHARMACIES+strWhere;
Cursor cursor = db.rawQuery(query, null);
```

#### 3.2.2.1 Kurzor

Kurzor SQLite databáze je objekt, který nám umožňuje procházet jednotlivé řádky v tabulce podle zadaných parametrů. Objekt Cursor[22] dostaneme zavoláním metody rawQuery() nebo query() a daný kurzor ukazuje na řádky, které odpovídají danému dotazu. Mezi jednotlivými řádky se pohybujeme pomocí příkazů moveToFirst() a moveToNext.

#### 3.2.2.2 Vkládání dat

Pro vkládání a aktualizaci databáze používám objekt ContentValues, který umožňuje definovat klíč a hodnotu jednotlivých záznamů, které chceme do databáze vložit. Klíč představuje identifikátor sloupce v tabulce a hodnota představuje obsah buňky v tomto sloupci. Pro samotné vložení záznamu do databáze používám vestavěnou metodu insert().

```
ContentValues values = new ContentValues();
values.put("route", route);
values.put("text", text);
db.insert(TABLE_ROUTE, null, values);
```

### 3.2.3 SharedPreferences

Kromě databáze aplikace využívá také SharedPreferences[23]. To je další prostor, kam si aplikace mohou ukládat data. SharedPreferences je třída, která umožňuje ukládat a načítat data, která jsou uložena jako dvojice klíč a hodnota. Pomocí této třídy může aplikace ukládat jakékoliv údaje typu: booleans, floats, ints, longs, and strings. Tato data zůstanou v aplikaci uloženy, i když bude aplikace ukončena. V aplikaci to využívám zejména k ukládání uživatelského nastavení.

## 3.3 Práce na pozadí

Pokud chceme provádět dlouhotrvající operace, například přístup k datům (jako jsou MP3, JSON, obrázky) na internetu, musíme implementovat vlákno, které poběží na pozadí. Pokud bychom tyto data chtěli získat pomocí hlavního vykreslovacího vlákna, uživatel by musel počkat než se data stáhnou a poté by mohl teprve pokračovat. Pokud chceme, aby vedlejší vlákno mohlo komunikovat s hlavním vykreslovacím (např. aktualizace pokroku stahování dat), použijeme třídu AsyncTask.

### 3.3.1 AsyncTask

AsyncTask je abstraktní třída, která pomáhá aplikaci zvládnout procesy, které by hlavní vykreslovací vlákno nezvládlo efektivním způsobem. Třída AsyncTask[24] nám umožňuje provádět dlouhotrvající procesy na pozadí a výsledky zobrazovat v hlavním vlákně uživateli bez toho, aby se zpomalil chod aplikace. Tuto třídu můžeme rozdělit do čtyř kroků.

**onPreExecute:** Funkce onPreExecute() se vyvolá předtím, než se začne provádět vlastní práce na pozadí. Tato metoda se obvykle používá k zobrazení dialogového okna informujícího uživatele o pokroku práce, která se vykonává na pozadí.

**doInBackground:** V této funkci se pouští kód, který se vykoná na pozadí.

**onProgressUpdate:** Tato funkce se volá z vlákna, které již běží na pozadí a posílá informace o pokroku hlavnímu vláknu, které je zobrazí uživateli.

**onPostExrcute:** Funkce se vyvolá ve chvíli, kdy skončí práce na pozadí. Většinou zavírá dialogové okno, informující o pokroku a hlásí výsledek operace na pozadí.

Třídu AsyncTask používám pro stahování dokumentů k lékům (příbalové letáky, souhrnné informace a obaly), stahování cen léků, k náročnějším dotazům do databáze (např. hledání substitutů nebo interakcí) a hlavně k aktualizaci databáze.

Problém je v tom, že třída AsyncTask je provázána s aktivitou nebo fragmentem, který ji vytvořil a pokud v průběhu vykonávání procesu na pozadí se aktivita, případně fragment, změní nebo zničí, tak AsyncTask nemůže vrátit žádný výsledek. To může nastat v mnoha případech, ale nejčastější je, že se pouze změní orientace displeje, což vyvolá zničení a znovu vytvoření aktivity. V tu chvíli stará třída AsyncTask nemůže vrátit svůj výsledek a pravděpodobně se vytvoří nová třída, která bude komunikovat s novou aktivitou. To není určitě žádoucí chování.

Tento problém jsem vyřešil obalením třídy AsyncTask fragmentem. Pokud při vytváření fragmentu zavoláme funkci „setRetainInstance(true)“ fragment se nezničí spolu s aktivitou. Pokud aktivitě nastavíme rozhraní daného fragmentu, můžeme navíc posílat informace o průběžném stavu procesu na pozadí do hlavního vykreslovacího vlákna napříč změnami orientace. Pokud aktivita při svém vytváření nenajde daný fragment, vytvoří nový a při dalších vytváření již pracuje s tímto vytvořeným.

Pokud chceme třídu AsyncTask ukončit z hlavního vlákna, nejde zavolat jednoduše metodu na zastavení. Vláknu se musí nastavit pomocí funkce asyncTask.cancel(true), že je zrušené. Vedlejší vlákno se však musí ukončit samo, to znamená, že pravidelně musí kontrolovat, jestli nebylo zrušeno. To dělá metodou „isCanceled()“ a pokud metoda vrátí pravdu, musí se vlákno ukončit.

## 3.4 Propojení se serverem

Aplikace závisí z velké části na serveru. Od něj získává všechny aktualizace databáze. Server má své vlastní API, ke kterému se aplikace připojuje a zjišťuje, jestli na serveru nevznikly nějaké změny, které by měly být přeneseny do aplikace. Server má připravené webové služby, které přijímají a odpovídají na požadavky pomocí rozhraní Representational State Transfer (REST). Server komunikuje pomoci formátu XML nebo JSON.

JSON je velmi lehký, strukturovaný, snadno analyzovatelný a hodně čitelný jazyk. Používá se především, pokud si aplikace vyměňuje data se serverem. Výhodou formátu JSON oproti XML je v jeho jednoduchosti a přehlednosti, hodí se více pro krátké strukturované zprávy. Platforma Android obsahuje knihovnu json.org, která umožňuje efektivně pracovat se soubory JSON. Ale dá se snadno integrovat jiné řešení pro analýzu těchto souborů. Hlavními součástmi knihovny json.org jsou třídy JSONObject a JSONArray. Kontejner JSONArray obsahuje seřazený výpis hodnot, které ohraničují hranaté závorky. Hodnotami v kontejneru mohou být jakékoliv datové typy JSON včetně objektu nebo pole. Každá položka objektu JSONObject má svůj klíč, který je typu JSONString.

XML je zkratka pro Extensible Markup Language. Formát XML je velmi populární a běžně používaný pro sdílení dat na internetu. Android nabízí tři

různé analyzátory XML, jimi jsou DOM, SAX a XMLPullParser. Já jsem zvolil XMLPullParser, protože je účinný a snadno se používá. Jednoduchý XML formulář, který značí cestu podání léku, jsem zpracoval takovýmto kódem. XML

```
<route-list>
    <route>
        <abbr>INF</abbr>
        <czech>infuze</czech>
    </route>
</route-list>
```

```
XmlPullParserFactory factory =
    XmlPullParserFactory.newInstance();
factory.setNamespaceAware(true);
XmlPullParser xpp = factory.newPullParser();
xpp.setInput( new StringReader(  returnRoute ) );
int eventType = xpp.getEventType();
while (eventType != XmlPullParser.END_DOCUMENT) {
    if(eventType != XmlPullParser.START_DOCUMENT)
        if(eventType == XmlPullParser.START_TAG) {
            if(xpp.getName().equals("route")){
                newRouteInfo = new RouteInfo();
            }
            lastTag = xpp.getName();
        } else if(eventType==XmlPullParser.END_TAG){
            if(xpp.getName().equals("route")){
                routeInfoListCustom.add(newRouteInfo);
            }
        } else if(eventType==XmlPullParser.TEXT){
            if(lastTag.equals("abbr"))
                newRouteInfo.setRoute(xpp.getText());
        else
            if(lastTag.equals("czech"))
                newRouteInfo.setText(xpp.getText());
        }
    eventType = xpp.next();
}
```

V kódu můžeme vidět, že XMLPullParser nám pomůže procházet XML dokument pomocí funkce next() a dokáže nám říci, ve které části jednotlivých elementů se nacházíme.

## 3.5 Lokalizace

Aplikace je sice zaměřená pouze na Českou republiku, ale mohou ji používat cizinci, kteří preferují jiný jazyk. Aplikace je zatím pouze v češtině, ale Android nabízí možnost psát všechny pevné texty do jediného souboru. Těchto souborů může být více, a to pro každý jazyk jeden, tedy není problém pouze přeložit názvy a aplikace bude fungovat v jiném jazyku. Dbal jsem tedy, aby všechny texty byly v tomto jednom souboru a nebyly vepsány do samotného kódu aplikace. OS při spuštění zjistí, jaký jazyk je primárně nastaven a podle toho vybere, který soubor bude aplikace používat.

Lokalizovat se nemusejí pouze texty, ale také obrázky, zvuky či jakékoliv jiné statické údaje. Adresářová struktura pro takové soubory může vypadat například takto.

```
MyProject/
    res/
        values/
            strings.xml
        values-es/
            strings.xml
        values-fr/
            strings.xml
```

## 3.6 Přístup k funkcím telefonu

Jelikož se jedná o mobilní aplikaci, zaměřil jsem se na zapojení funkcí mobilního zařízení do provozu aplikace. Pokud například informace o lékárně obsahují její telefonní číslo, aplikace umožní uživateli rovnou do této lékárny zavolat, pouhým stisknutím tlačítka znázorňující telefon. Uživatel bude mít tedy méně práce s opisováním, resp. kopírováním telefonního čísla a následným jeho psaním, resp. vkládáním do aplikace, která umožňuje volat. Pro přístup k těmto funkcím musíme aplikaci udělit oprávnění, které se zobrazí uživateli při instalaci aplikace. Tyto oprávnění se píší do manifestu ve tvaru.

```
<uses-permission android:
    name="android.permission.CALL_PHONE" />
```

K samotnému přístupu k funkcím se používá třída Intent, která je především určena k zahajování nových aktivit, ale také ke spouštění vestavěných funkcí zařízení, jako jsou například zmiňované volání nebo použití kamery. Intent, v překladu záměr, je zpráva, kterou můžete požádat svou nebo jinou aplikaci, aby provedla daný záměr. Intent v sobě nese data potřebná k vykonání záměru a případně cíl volání, tedy aplikaci nebo komponentu, která má tento záměr vykonat.

Pro volání určitého čísla použiji tento kód.

```
Intent callIntent=new Intent(Intent.ACTION_CALL);
callIntent.setData(Uri.parse("tel:"+phoneNumber));
startActivity(callIntent);
```

Na prvním řádku vytvořím záměr, se kterým budu chtít volat. Na druhém přiřadím záměru číslo volaného a na posledním záměr spustím. O zbytek se postará aplikace na volání, která poslouchá, jestli někdo nespustil záměr, který může obsloužit.

Využití kamery se trochu liší od volání a to v tom, že aktivita se nespouští jednoduše pomocí metody startActivity(), ale pomocí startActivityForResult().

```

Intent takePictureIntent
    = new Intent(MediaStore.ACTION_IMAGE_CAPTURE);
startActivityForResult(takePictureIntent,
    REQUEST_IMAGE_CAPTURE);
```

Tato metoda nám umožňuje pracovat s daným obrázkem i po zavření kamery. Jak název napovídá, tak spouštíme aktivitu s tím, že od ní chceme dostat výsledek, kterého docílila. Výsledek tohoto záměru odchytáváme v aktivitě, ze které jsme záměr volali pomocí přetížené metody onActivityResult().

Aplikace také potřebuje přistupovat ke zdrojům zařízení jako je například lokalizační čip. Lokalizaci aplikace potřebuje například pro vyhledávání nejbližších lékáren. Pro určení polohy můžeme použít například GPS nebo Wi-Fi. K samotnému získání polohy potřebujeme opět přidat další oprávnění do manifestu.

```
<uses-permission android:name
    "android.permission.ACCESS_FINE_LOCATION"/>
```

Pro získání informací o poloze s pomocí Wi-fi, musí mít zařízení aktivovanou tuto funkci a mít v dosahu nějakou síť. Google má databázi Wi-fi spotů s jejich souřadnicemi a vypočítává polohu zařízení s pomocí triangulace podle síly signálu jednotlivých dostupných sítí. V dnešních zařízení je rozdíl mezi skutečnou polohou a polohou určenou pomocí Wi-fi přibližně 3-5m. Výhodou tohoto přístupu je jeho rychlost vyhledávání pozice, které se pohybuje v rámci vteřin i méně. Nevýhodou je, že musí být zařízení připojeno na Wi-fi.

Druhá možnost je získat informace o poloze pomocí GPS. Výhodou tohoto přístupu je, že dostaneme nejpřesnější pozici, ale zároveň spotřebováváme větší množství baterie, potřebujeme jasný výhled na satelity a zaměření trvá nějaký čas, ve kterém musí zařízení zjistit přesnou polohu GPS družic.

Pro správné vyřešení problému, jak získávat pozici, jsem vybral obě tyto varianty. Nejprve si aplikace ověří, zda je povolena lokalizace zařízení a poté zjistí, jestli může svou pozici dostat pomocí Wi-fi, pokud se to nezdaří, pokusí se tuto pozici dostat s pomocí GPS.

## 3.7 Využití Google Maps

V aplikaci využívám mapy pro zobrazení okolí a lékáren v něm. Google umožňuje částečně bezplatně využívat jeho mapy v aplikacích. Mapy lze využívat díky Google Maps API Android v2[25] a jejich implementace a přizpůsobení je velmi jednoduché. Pro zobrazení mapy se používá vlastní fragment, který se dá vložit kamkoliv do aktivity, tedy samostatně, nebo přidat nějaké již vytvořené. K mapám aplikace přidává značky, kde se právě uživatel nachází a označuje také vybrané lékárny. K používání Google Maps je zapotřebí API klíč, který jsem získal registrováním na oficiálních stránkách Googlu pro vývojáře. Tím je zajištěno, že každá aplikace bude mít vlastní přístup k používání map. Je to uděláno proto, že ne všechny funkce map jsou zdarma a tímto bude mít Google přehled, jak jsou placené funkce využívány. Tento kód se vloží do manifestu a aplikace může začít používat třídu com.google.android.gms.maps.MapFragment, která nám umožní zobrazit potřebnou mapu.

## 3.8 Uživatelské rozhraní

Aby aplikace vypadala jednotně, všechny styly se nacházejí v samostatném souboru. Tedy pokud bude potřeba změnit styl některých prvků, není potřeba hledat všechny tyto prvky a upravovat jejich styl, ale stačí tento styl pozměnit v tomto souboru a změna se projeví ve všech prvcích.

### 3.8.1 Úvodní obrazovka (Splash Screen)

Splash Screen je aktivita, která se zobrazí při prvním zapnutí aplikace a slouží k pozastavení aplikace, dokud se z internetu nestáhnou potřebná data, nebo se neprovedou jiné procesy potřebné k chodu aplikace. Jelikož tato aplikace má již v sobě zabudovanou databázi, mohl jsem se této Splash Screen aktivitě vyhnout a uživatele nezdržovat stahováním nových dat. Navíc Android Design Guidelines doporučují se těmto typům aktivit vyhnout a to se mi tedy podařilo.

### 3.8.2 Zobrazení obsahu

V návrhu uživatelského rozhraní jsem se rozhodl použít pro zobrazení obsahu záložky, které se nacházejí v horní části obrazovky. Mezi jednotlivými záložkami může uživatel přecházet pomocí tlačítek, ale také jsem pro zjednodušení použil Swipe layout, který umožní přecházet mezi záložkami pouze jednoduchým horizontálním posouváním. Toto gesto je velmi intuitivní a uživatel to jistě ocení.

Všechen obsah se nachází v jedné aktivitě a jednotlivé záložky tvoří fragmenty. V aktivitě jsem použil třídu ViewPager, ve které se zobrazují jednotlivé

fragmenty. Pro přecházení mezi jednotlivými fragmenty jsem použil již vytvořenou třídu FragmentPagerAdapter, od které jsem podědil a doplnil funkce pro procházení fragmentů.

Zde je třída, která ovládá jednotlivé fragmenty.

```
public class TabPagerAdapter
    extends FragmentPagerAdapter {

    PharmacyFragment pharmacy = null;

    final int PAGE_COUNT = 3;
    // Tab Titles
    private String tabtitles[]
        =new String[]{"Lékárny","Léky","Oblíbené"};
    Context context;
    public TabPagerAdapter(FragmentManager fm) {
        super(fm);
    }
    @Override
    public int getCount() {
        return PAGE_COUNT;
    }
    @Override
    public CharSequence getPageTitle(int position){
        return tabtitles[position];
    }
    @Override
    public Fragment getItem(int index) {
        switch (index) {
            case 0:
                return new PharmacyFragment();
            case 1:
                return new DrugFragment();
            case 2:
                return new FavoriteFragment();
        }
        return null;
    }
    public PharmacyFragment getPharmacy() {
        return pharmacy;
    }
}
```

Pokud chceme zobrazit jednotlivé záložky pomocí ActionBaru, musí se povolit NAVIGATION_MODE_TABS a vytvořit počet instancí ActionBar.Tab

odpovídající celkovému počtu záložek. Kód aktivity, ve které se fragmenty zobrazují, vypadá takto.

```
tabPagerAdapter =
    new TabPagerAdapter(getSupportFragmentManager());
actionBar.setNavigationMode
    (ActionBar.NAVIGATION_MODE_TABS);

mViewPager = (ViewPager) findViewById(R.id.pager);
mViewPager.setAdapter(tabPagerAdapter);
mViewPager.setOnPageChangeListener
    (new ViewPager.SimpleOnPageChangeListener() {
    @Override
        public void onPageSelected(int position) {
            tabPosition = position;
                actionBar.setSelectedNavigationItem
                    (position);
        }
});


for (int i=0; i~< tabPagerAdapter.getCount();i++){
    ActionBar.Tab tab = actionBar.newTab()
        .setText(tabPagerAdapter.getPageTitle(i))
                        .setTabListener(tabListener);
                        actionBar.addTab(tab);
    }
```

Pomocí třídy ViewPager a jeho listeneru vytvořeného metodou setOnPageChangeListener() aplikace zjišťuje, na kterou záložku uživatel přešel a zobrazí správný obsah.

## 3.9 Vyhledávání

Pro zjednodušení vyhledávání léků jsem místo jednoduchého textového pole implementoval AutoCompleteTextView, které funguje jako našeptávač. Pro uživatele to znamená, že nemusí vypisovat celý název léku, ale stačí jeho část a aplikace mu nabídne možnosti, které odpovídají jeho zadání a on si z nich může vybrat.

Z databáze vyberu základní jméno všech léků, vyřadím duplicity a tento seznam léků dám jako parametr adaptéru, který následně zobrazí uživateli léky odpovídající jeho vyhledávání.

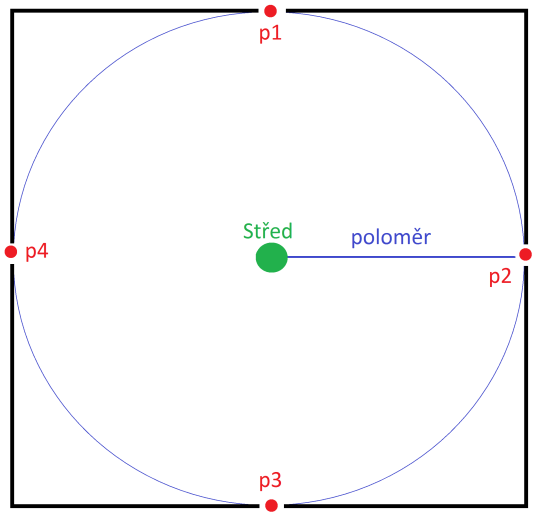
### 3.9.1 Lékárny v okolí

Aplikace vyhledává lékárny v okolí podle GPS souřadnic a je tedy potřeba po zapnutí aplikace rychle tyto lékárny vyhledat. Uživatel si může nastavit vzdálenost, do které mají být lékárny nalezeny. Záznamů lékáren bude v databázi v řádů tisíců a je nutno získat jejich vzdálenost a nenechat uživatele čekat, než se všechny lékárny seřadí. SQLite nepodporuje trigonometrické funkce k výpočtu vzdálenosti mezi dvěma body a nabízejí se tedy různé možnosti, jak tento problém vyřešit.

Jednou možností je importovat nativní knihovnu pro práci s SQLite databází a nadefinovat vlastní funkci, která bude vypočítávat vzdálenost mezi aktuální pozicí a jednotlivými lékárnami.

Další možností je zoptimalizovat algoritmus na vyhledávání lékáren v okolí. Můžeme využít informace, že všechny lékárny se budou vyskytovat do vzdálenosti nastavené uživatelem. Pozici jednotlivých lékáren máme uloženou v tabulce jako dvojici hodnot značící zeměpisné souřadnice (zeměpisnou šířku a délku). Jsou to úhlové vzdálenosti od rovníku a nultého poledníku. Optimalizace algoritmu spočívá v tom, že odstraním všechny lékárny, které se nevejdou do čtverce s hranou 2krát větší než je nastavená uživatelem a středem se souřadnicemi aktuální pozice. K tomu jsou potřeba zjistit body p1 ,p2, p3 a p4, které se nacházejí ve vzdálenosti nastavené uživatelem od středu a s úhly 0 °, 90 °, 180 ° a 270 °. Tyto body zjistíme ze souřadnic středu a maximální vzdálenosti lékáren pomocí funkce, která s pomocí poloměru Země tyto body vypočítá. Tato funkce proběhne jen jednou a proces zobrazování okolních lékáren nijak nezpomalí. Po získání těchto bodů můžeme jednoduše vyfiltrovat lékárny, které se v této čtvercové oblasti nenacházejí. Pomocí jednoduchého porovnání zjistíme, jestli jsou zeměpisné šířky a délky větší nebo menší než jednotlivé body. Dále pracujeme jen se zlomkem lékáren, u kterých zjistíme jejich vzdálenost od aktuální pozice. Odstraníme ty, které mají větší vzdálenost, než je maximální povolená, a tedy se nevejdou do kruhu se středem se souřadnicemi aktuální pozice.

Zvolil jsem tuto druhou možnost, protože je zajímavější a rychlejší než získávat souřadnice všech lékáren a počítat jejich vzdálenost od aktuální pozice.




Obrázek 3.1: Vyhledávání lékáren

Kapitola 4

# Testování

## 4.1 Testování autorem

Po celý čas implementace byly jednotlivé části aplikace testovány průběžně převážně na mobilním telefonu JiaYu G4S s verzí OS Android 4.2 a úhlopříčkou 4,7 palce. Pro testování zpětné kompatibility a funkčnosti na starších a méně výkonných zařízeních jsem používal Samsung Galaxy Tab s verzí OS Android 2.2 a úhlopříčkou 7 palců. Pro zobrazení výstupů testů jsem používal třídu Log, která pracuje s několika typy výstupů. Pro zobrazení chybových nebo jiných hlášek přímo na zařízení jsem použil třídu Toast. Tato třída umožní zobrazit text přímo uživateli na displej.

Po doimplementování každé aktivity, které se zobrazují uživateli, jsem je testoval na všechny možné interakce s uživatelem.

## 4.2 Usability test

Poté, co jsem sám důkladně otestoval aplikaci, byl proveden usability test. Účel usability testu je prověřit funkčnost a hlavně použitelnost aplikace. Pro tento test jsem zvolil testery, kteří by měli být cíloví uživatelé pro tuto aplikaci, tedy budou ji s největší pravděpodobností využívat.

### 4.2.1 Popis testu

Účelem testu bylo zjistit, zda je ovládání aplikace intuitivní a zda vybraný tester dokáže jednoduše získat požadované informace. Každý testovaný dostal scénář s různými úkoly a cíly, kterých měl dosáhnout. Scénáře byly navrhnuty tak, aby otestovaly funkcionalitu celé aplikace.

### 4.2.2 Testeři

Pro testování jsem zvolil uživatele, kteří mají zkušenosti s OS Android a jsou ve věkové skupině nad 15 let. Tento věk jsem zvolil, protože mladší uživatelé lékárny nenavštěvují příliš často. Aplikace najdi-lékárnu.cz byla nainstalována na jejich vlastní zařízení a tím odpadla bariéra neznalosti nového zařízení. Parametry jednotlivých zařízení naleznete v tabulce 4.1

| Tester | Název zařízení | Verze OS | Velikost displeje |
|---|---|---|---|
| **1** | HTC Desire 500 | 4.1.2 | 4.3 ” |
| **2** | Lenovo 5000 | 4.4 | 5.0 ” |
| **3** | LG G3 | 4.4.2 | 5.5 ” |
| **4** | Samsung Galaxy S5 | 4.4 | 5.1 ” |
| **5** | Zopo ZP700 | 4.2 | 4.7 ” |

Tabulka 4.1: Testovací zařízení

### 4.2.3 Scénáře

Každý scénář obsahuje zadání a ideální řešení.

1. Najděte jméno a adresu nejbližší lékárny.
   **Ideální řešení:** Na hlavní obrazovce v záložce lékárny se tato informace nachází u první lékárny v seznamu nejbližších lékáren.

2. Vyhledejte otevírací dobu aktuálního dne v lékárně Horní Počernice.
   **Ideální řešení:** Na hlavní obrazovce v záložce lékárny v ActionBaru zmáčkne tlačítko pro vyhledávání a napíše Horní Počernice. V seznamu klikne na vyhledanou lékárnu a mezi informacemi o otevírací době nalezne potřebnou informaci.

3. Vyhledejte cenu léku Ibalgin 200 v Praze 9.
   **Ideální řešení:** Na hlavní obrazovce v záložce léky pomocí vyhledávání nalezne Ibalgin 200. V seznamu vybere tento lék a v detailu léku klikne na tlačítko Zobrazit ceny. V seznamu měst nalezne řádek s popisem Praha 9 a po kliknutí na tento řádek se mu požadovaná informace zobrazí.

4. Zjistěte, zda Ibalgin 200 interaguje s Aspirinem.
   **Ideální řešení:** Na hlavní obrazovce v záložce léky pomocí vyhledávání nalezne Ibalgin 200. V seznamu vybere tento lék a v datailu léku klikne na tlačítko Interakce. V seznamu se nachazí Aspirin a tedy interaguje s Ibalginem 200.

5. Přidejte Ibumax 200 mg mezi oblíbené léky.
   **Ideální řešení:** Na hlavní obrazovce v záložce léky pomocí vyhledávání

nalezne Ibumax 200 mg. V seznamu vybere tento lék a v detailu léku klikne na hvězdičku, která se nachází vedle názvu.

6. Zjistěte informace o léku z přiloženého čárového kódu.
   **Ideální řešení:** Na hlavní obrazovce v záložce léky stiskne tlačítko znázorňující fotoaparát, které se nachází v ActionBaru. Poté pomocí kamery naskenuje čárový kód.

7. Nastavte vzdálenost vyhledávání lékáren na 15 km.
   **Ideální řešení:** Na hlavní obrazovce klikne na tlačítko znázorňující ozubené kolečko, které se nachází v ActionBaru. Poté pomocí posuvné lišty nastaví vzdálenost zobrazování lékáren na 15 km.

### 4.2.4 Výsledky

#### 4.2.4.1 Tester 1

Tester je muž, 22 let a studuje IT. Byl již s aplikací seznámem během jejího vývoje a podílel se částečně na jejím testování v průběhu implementace. Předpokládá se již určitá znalost používání.

1. Výsledek byl dosažen ideální cestou.
2. Výsledek byl dosažen ideální cestou.
3. Výsledek byl dosažen ideální cestou.
4. Tester nevěděl, co znamená interakce léků, ale i tak výsledku dosáhl.
5. Výsledek byl dosažen ideální cestou.
6. Testerovi nebylo jasné, kde má menu tlačítko pro skenování hledat. Prošel záložku lékárny, detail léku i oblíbené. Poté zkoušel hardwarové tlačítko pro vyvolání menu. Asi po minutě požadované tlačítko našel a konstatoval, že předpokládal, že menu tlačítka, budou na všech záložkách stejné.
7. Výsledek byl dosažen ideální cestou.

#### 4.2.4.2 Tester 2

Tester je žena, 19 let a běžný uživatel chytrého telefonu s OS Android. Tester o vývoji aplikace věděl, ale aktivně s ní nikdy nepracoval.

1. Výsledek byl dosažen ideální cestou.
2. Výsledek byl dosažen ideální cestou.
3. Výsledek byl dosažen ideální cestou.

4. Terster nevěděl, co znamená interakce léků, ale i tak výsledku dosáhl.
5. Výsledek byl dosažen ideální cestou.
6. Výsledek byl dosažen ideální cestou.
7. Výsledek byl dosažen ideální cestou.

#### 4.2.4.3 Tester 3

Tester je žena, 15 let a začátečník v používání chytrého telefonu s OS Android. Tester o vývoji aplikace věděl, ale aktivně s ní nikdy nepracoval.

1. Výsledek byl dosažen po kliknutí na první záznam v seznamu.
2. Výsledek byl dosažen ideální cestou.
3. Tester správně našel daný lék, ale poté klikl na tlačítko „Najít nejlepší cestu v okolí“. Poté se vrátil a výsledku dosáhl správnou cestou.
4. Výsledek byl dosažen ideální cestou.
5. Výsledek byl dosažen ideální cestou, ale testerovi trvalo nějaký čas najití daného tlačítka.
6. Testerovi trvalo nějaký čas na najití daného tlačítka, ale výsledek byl nakonec dosažen.
7. Výsledek byl dosažen ideální cestou.

#### 4.2.4.4 Tester 4

Tester je muž, 35 let a běžný uživatel chytrého telefonu s OS Android.

1. Výsledek byl dosažen ideální cestou.
2. Tester měl problém s vyhledáním lékárny. Nebylo mu jasné hledání pomocí lupy dostupné v ActionBaru. Po zaváhání nakonec výsledku dosáhl.
3. Výsledek byl dosažen ideální cestou.
4. Výsledek byl dosažen ideální cestou.
5. Výsledek byl dosažen ideální cestou.
6. Tester zaváhal při hledání tlačítka na skenování, stejně jako u bodu 2.
7. Výsledek byl dosažen ideální cestou.

#### 4.2.4.5 Tester 5

Tester je muž, 28 let a běžný uživatel chytrého telefonu s OS Android.

1. Výsledek byl dosažen ideální cestou.
2. Výsledek byl dosažen ideální cestou.
3. Výsledek byl dosažen ideální cestou.
4. Výsledek byl dosažen ideální cestou.
5. Výsledek byl dosažen ideální cestou.
6. K dosáhnutí výsledku muselo být testerovi napovězeno.
7. Výsledek byl dosažen ideální cestou.

### 4.2.5 Vyhodnocení

Z usability testování je patrné, že umístění tlačítka pro skenování kódů ActionBaru není vhodné, protože uživatelé nepředpokládají, že se jednotlivé položky ActionBaru budou na jednotlivých záložkách měnit. Jako problém nemohu považovat časté váhání při vyhledávání interakce z důvodu neznalosti tohoto slova. Uživatel, který bude tuto informaci potřebovat ji lehce najde.

# Závěr

Cílem práce bylo vytvořit aplikaci pro OS Android, která bude komunikovat s portálem najdi-lékárnu.cz, bude přebírat jeho funkcionalitu a bude rozšiřovat použití systému na mobilní zařízení s OS Android.

Před samotnou tvorbou systému jsem zanalyzoval již existující řešení na trhu. Poté jsem provedl návrh celé aplikace, ve kterém jsem uplatnil poznatky z analýzy již existujících řešení a vyvaroval se chyb, které v nich byly. Následně proběhl samotný vývoj aplikace. Testování začalo již v průběhu implementace, testováním jednotlivých aktivit. Na závěr proběhlo testování aplikace jako celku.

Tento projekt celkově hodnotím kladně. Podařilo se mi splnit zadání práce a navíc jsem navrhl a realizoval funkce, které jsou možné pouze na mobilních zařízení jako bylo například skenování čárových kódů.

Díky této práci jsem se naučil vyvíjet aplikace pro OS Android, prohloubil jsem své vědomosti v oblasti analýzy a návrhu softwaru. Byla to pro mě dobrá a užitečná zkušenost. Rád bych se vývojem aplikací pro OS Android věnoval i do budoucna.

## Pokračování v práci

Na aplikaci bych rád i nadále pracoval a pomáhal rozšiřovat funkcionalitu, která bude přidávána na webové rozhraní a zároveň bude vhodná k přidání na mobilní verzi. Jedna z těchto funkcí by mohla být funkce na hodnocení lékáren, kde by sami uživatelé mohli hodnotit jednotlivé lékárny, přístup zaměstnanců a celkovou kvalitu služeb. S touto funkcí souvisí i přihlašování, které by se muselo také dodělat.

# Literatura


[1] Státní ústav pro kontrolu léčiv: Databáze lékáren - SUKL [online]. 2014, [Citováno 2014-10-18]. Dostupné z: http://www.sukl.cz/modules/apotheke/

[2] Státní ústav pro kontrolu léčiv: Databáze léků - SUKL [online]. 2014, [Citováno 2014-10-18]. Dostupné z: http://www.sukl.cz/modules/medication/

[3] najdi-lékárnu.cz [online]. [Citováno 2014-10-08]. Dostupné z: http://www.najdi-lekarnu.cz/

[4] Google play [online]. [Citováno 2014-11-05]. Dostupné z: https://play.google.com/store

[5] Windows Phone [online]. [Citováno 2014-11-06]. Dostupné z: https://www.windowsphone.com/cs-cz

[6] App Store [online]. [Citováno 2014-11-10]. Dostupné z: http://store.apple.com/cz

[7] tutorialspoint: Android Architecture [online]. [Citováno 2014-12-01]. Dostupné z: http://www.tutorialspoint.com/android/android_architecture.htm

[8] Jamal Eason: Android Studio 1.0 | Android Developers Blog [online]. 2014, [Citováno 2015-05-02]. Dostupné z: http://android-developers.blogspot.cz/2014/12/android-studio-10.html

[9] TutorialsPoint: Java - Overview [online]. [Citováno 2014-11-20]. Dostupné z: http://www.tutorialspoint.com/java/java_overview.htm

[10] Allen, G.: *Android 4: Průvodce programováním mobilních aplikací*. Brno: COMPUTER PRESS, 2013, ISBN 978-80-251-3782-6.

[11] Poměr OS v prodaných zařízeních [online]. [Citováno 2015-04-12]. Dostupné z: http://www.idc.com/prodserv/smartphone-os-market-share.jsp

[12] Android Developer: Verze [online]. [Citováno 2015-01-12]. Dostupné z: http://developer.android.com/guide/topics/manifest/uses-sdk-element.html

[13] Poměr verzí OS Android [online]. [Citováno 2015-04-12]. Dostupné z: http://developer.android.com/about/dashboards/index.html

[14] Matěj Konečný: Vyvíjíme pro Android: Začínáme - Zdroják [online]. [Citováno 2015-03-16]. Dostupné z: http://www.zdrojak.cz/clanky/vyvijime-pro-android-zaciname/

[15] Android Developers: Managing the Activity Lifecycle | Android Developers [online]. [Citováno 2015-01-23]. Dostupné z: http://developer.android.com/training/basics/activity-lifecycle/index.html

[16] Android Developers: Design | Android Developers [online]. [Citováno 2015-01-24]. Dostupné z: https://developer.android.com/design/index.html

[17] Fluid UI [online]. [Citováno 2015-02-15]. Dostupné z: https://www.fluidui.com/

[18] SQLite [online]. [Citováno 2015-01-23]. Dostupné z: https://www.sqlite.org

[19] SQLite: Datatypes In SQLite Version 3 [online]. [Citováno 2015-03-03]. Dostupné z: https://www.sqlite.org/datatype3.html

[20] Knihovna zXing, GitHub [online]. [Citováno 2015-02-01]. Dostupné z: https://github.com/zxing/zxing/

[21] Rychlost SQLite databáze [online]. [Citováno 2015-04-30]. Dostupné z: https://www.sqlite.org/speed.html

[22] Android Developers: Cursor | Android Developers [online]. [Citováno 2015-01-17]. Dostupné z: http://developer.android.com/reference/android/database/Cursor.html

[23] Android Developers: SharedPreferences | Android Developers [online]. [Citováno 2014-12-19]. Dostupné z: http://developer.android.com/reference/android/content/SharedPreferences.html

[24] Android Developers: AsyncTask | Android Developers [online]. [Citováno 2014-11-15]. Dostupné z: http://developer.android.com/reference/android/os/AsyncTask.html

[25] Google Developers: SGoogle Maps Android API v2 — Google Developers [online]. [Citováno 2015-01-20]. Dostupné z: https://developers.google.com/maps/documentation/android/

Příloha **A**

# Seznam použitých zkratek

**UI** User interface

**XML** Extensible markup language

**OS** Operační systém

**JSON** JavaScript Object Notation

**iOS** iPhone Operation System

**GPS** Global Positioning System

**DVM** Dalvik Virtual Machine

**SDK** Software Development Kit

**JDK** Java Development Kit

**JRE** Java Runtime Enviroment

**SMS** Short Message Service

**SQL** Structured Query Language

**ATC** Anatomicko-terapeuticko-chemická skupina

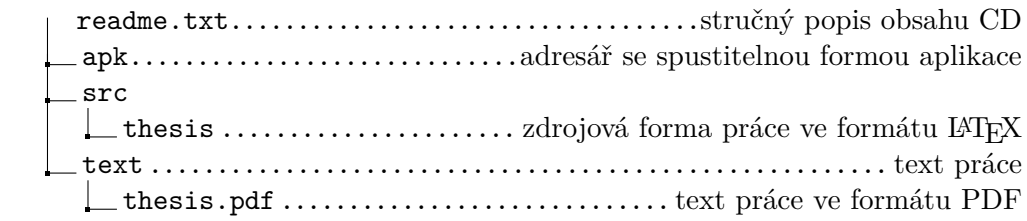
**QR** Quick Response

**MP3** MPEG Layer 3

**MPEG** Moving Picture Experts Group

**API** Application Programming Interface

**REST** Representational State Transfer


readme.txt ..... stručný popis obsahu CD
apk ..... adresář se spustitelnou formou aplikace
src
thesis ..... zdrojová forma práce ve formátu LaTeX
text ..... text práce
thesis.pdf ..... text práce ve formátu PDF