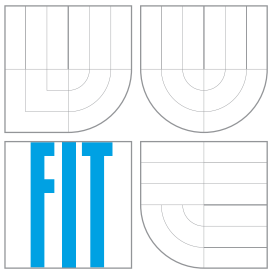
VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÝCH SYSTÉMŮ
FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER SYSTEMS

MOBILNÍ APLIKACE PRO SPRÁVU DNS

MOBILE APPLICATION FOR DNS ADMINISTRATION

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE
AUTHOR

DENIS GALAJDA

VEDOUCÍ PRÁCE
SUPERVISOR

Ing. MICHAL KOVÁČIK

BRNO 2016

Vysoké učení technické v Brně - Fakulta informačních technologií

Ústav počítačových systémů Akademický rok 2015/2016

Zadání bakalářské práce

Řešitel: **Galajda Denis**
Obor: Informační technologie
Téma: **Mobilní aplikace pro správu DNS**
Mobile Application for DNS Administration
Kategorie: Počítačové sítě

Pokyny:

1. Seznamte se se službou a protokolem DNS, nastudujte možnosti a specifika tvorby mobilních aplikací pro Android. Seznamte se s existujícími mobilními aplikacemi podobného charakteru.
2. Analyzujte požadavky na vlastní mobilní aplikaci pro správu DNS.
3. Uvedenou aplikaci navrhněte.
4. Implementujte navrženou aplikaci a otestujte ji na reálných sítích.
5. Zhodnoťte výslednou aplikaci, její ovladatelnost a přehlednost. Navrhněte možnosti dalšího rozšíření.

Literatura:

- Dle pokynů vedoucího

Pro udělení zápočtu za první semestr je požadováno:

- Splnění bodů 1 a 2 ze zadání.

Podrobné závazné pokyny pro vypracování bakalářské práce naleznete na adrese http://www.fit.vutbr.cz/info/szz/

Technická zpráva bakalářské práce musí obsahovat formulaci cíle, charakteristiku současného stavu, teoretická a odborná východiska řešených problémů a specifikaci etap (20 až 30% celkového rozsahu technické zprávy).

Student odevzdá v jednom výtisku technickou zprávu a v elektronické podobě zdrojový text technické zprávy, úplnou programovou dokumentaci a zdrojové texty programů. Informace v elektronické podobě budou uloženy na standardním nepřepisovatelném paměťovém médiu (CD-R, DVD-R, apod.), které bude vloženo do písemné zprávy tak, aby nemohlo dojít k jeho ztrátě při běžné manipulaci.

Vedoucí: **Kováčik Michal, Ing.**, UPSY FIT VUT
Datum zadání: 1. listopadu 2015
Datum odevzdání: 18. května 2016

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
Fakulta informačních technologií
Ústav počítačových systémů a sítí
612 66 Brno, Božetěchova 2

doc. Ing. Zdeněk Kotásek, CSc.
vedoucí ústavu

Abstrakt

Táto bakalárska práca sa zaoberá možnosťami a špecifikami tvorby vlastnej mobilnej aplikácie pre správu DNS na systém Android. Cieľom je vytvorenie prehľadnej aplikácie, jednoduchej na ovládanie, ktorá používateľovi umožní manažovať zóny a záznamy DNS doménového serveru BIND9. Tento problém rieši manipuláciou zónových súborov cez SSH pripojenie, ktorá je ovládaná moderným používateľským rozhraním. Výsledný program umožňuje používateľom pohodlne spravovať DNS server zo svojho chytrého telefónu prakticky hocikde.

Abstract

This bachelor's thesis deals with possibilities and specifics of developing a custom mobile applicaion for DNS administration on Android. The goal is to create an easy-to-use application which will allow the user to manage zones and resource records of BIND9 nameserver. This issue is solved by manipulation of zone files over SSH connection, controlled through the modern user interface. The resulting programm allows the users to comfortably manage a DNS server from their smartphone practically anywhere.

Kľúčové slová

Android, DNS, správa DNS, mobilná apikácia, BIND, SSH

Keywords

Android, DNS, DNS administration, mobile application, BIND, SSH

Citácia

GALAJDA, Denis. *Mobilní aplikace pro správu DNS*. Brno, 2016. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Kováčik Michal.

Mobilní aplikace pro správu DNS

Prehlásenie

Prehlasujem, že som túto bakalársku prácu vypracoval samostatne pod vedením pána Ing. Michala Kováčika. Uviedol som všetky literárne pramene a publikácie, z ktorých som čerpal.

.....
Denis Galajda
16. mája 2016

Poďakovanie

Chcel by som poďakovať vedúcemu mojej bakalárskej práce pánovi Ing. Michalovi Kováčikovi za odborné vedenie, pomoc a rady pri spracovaní tejto práce.

Obsah

Kapitola 1

Úvod

Doba obrovských sálových počítačov je už nenávratne za nami. Dnes je viac než bežné, že človek má vysokovýkonný počítač všade so sebou, a to napr. priamo vo svojom vrecku vo forme chytrého telefónu. Ohromný vzostup mobilných technológií, ktorý nám posledná doba priniesla, ovplyvnil tvarovanie mnohých oblastí dnešného života. Rozsiahle pokrytie internetu, ktorý je v súčastnosti dostupný na nepredstaviteľnom množstve miest, dáva týmto mobilným technológiám veľký potenciál a používateľom otvára možnosti, ktoré nie sú obmedzené miestom. Môžu si napr. počas cesty do zahraničia zistiť na internete informácie o destinácii, či pomocou emailu poslať rodine fotku z dovolenky. To, že sú presmerovaní na správnu internetovú stránku na základe webovej adresy, či to, že elektronická správa odoslaná na konkrétnu emailovú adresu bola doručená práve ich známym, zaisťuje systém serverov DNS. Predmetom tejto práce je vytvorenie aplikácie, ktorá využije výhody mobilných zariadení a poskytne ich používateľom pri správe tohoto systému. Takáto aplikácia umožní jednoducho a rýchlo vykonať elementárne úkony nad DNS serverom napr. z pohodlia svojej sedačky, či na ceste do zamestnania.

Táto práca je členená do niekoľkých logicky oddelených častí. Prvá kapitola čitateľovi predstaví systém DNS, vysvetlí jeho fungovanie a prácu s ním.

Druhá časť sa venuje najrozšírenejšej mobilnej platforme Android, z hľadiska tvorby mobilnej aplikácie. Čitateľa prevedie princípmi, ktoré budú využité pri tvorbe cieľovej aplikácie.

Tej sa venuje kapitola tretia. Táto časť začína analýzou požiadaviek, ktorej výsledkom je ich špecifikácia. Na jej základe je následne vytvorený návrh aplikácie, slúžiaci ako oporný bod pre implementáciu. Kapitolu uzatvára popis testovania vytvoreného programu.

Celá práca je zakončená zhodnotením výslednej aplikácie, jej ovládateľnosti a prehľadnosti, a návrhom možností jej ďalšieho rozšírenia.

Kapitola 2

Systém DNS

Táto kapitola pojednáva o systéme DNS. Prvá sekcia 2.1 definuje tento pojem a vysvetľuje motiváciu za jeho vznikom. Druhá časť 2.2 rozoberá architektúru systému DNS. Kapitolu uzatvárajú sekcie popisujúce formát DNS záznamov a prácu nad nimi. Táto kapitola čerpá informácie hlavne z knihy Síťové aplikace a jejich architektura [3] a technických správ RFC 1034 [5], RFC 1035 [6]. Informácie, ktoré sú predmetom tejto kapitoly, tvoria dôležitú prerekvizitu pre pochopenie súvislostí v ďalších častiach práce.

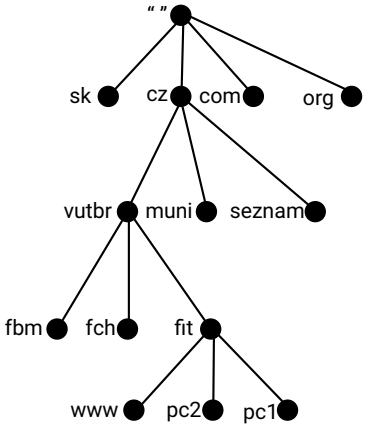
2.1 Služba DNS

Jedným z najdôležitejších kritérií pre komunikáciu na internete, je schopnosť jednoznačne určiť každé sieťové zariadenie. Za týmto účelom sa s príchodom protokolu IP v 70. rokoch začali využívať IP adresy.

Jedná sa o 32-bitové číslo (či 128-bitové u IP adries verzie 6), ktoré tvorí jednoznačný identifikátor pre každé sieťové rozhranie v internete. Tento číselný formát s pevnou dĺžkou je veľmi vhodný pre porovnávanie či prefixové vyhľadávanie. Na druhú stranu je táto identifikácia počítačov pomocou IP adries nevhodná pre používateľa, pretože máloktorý človek si pamätá nejakú IP adresu. Využívanie komplexných textových reťazcov pre adresovanie sieťových zariadení by však vyžadovalo príliš veľa priestoru v hlavičkách paketov a zložité spracovanie.

Za týmto účelom došlo ku vzniku systému DNS, ktorý obsahuje celosvetovú databázu IP adries a ich zrozumiteľných ekvivalentov, doménových mien (označovaných aj ako doménové adresy). Keďže ide ide o veľmi rozsiahlu databázu, tak je distribuovaná na viacero špecializovaných serverov, ktoré sa nazývajú nameservery, doménové servery (menné servery) alebo tiež servery DNS. IP adresa sa zisťuje z doménového mena požiadavkou na server DNS. Proces vyhľadávania v systéme DNS sa nazýva rozolúcia. Medzi základné služby, ktoré poskytuje systém DNS, podľa Matouška[3] patria:

- preklad doménových adries na IP adresy (s využitím záznamov typu A, AAAA),
- preklad IP adries na doménové adresy (s využitím reverzných záznamov typu PTR),
- preklad aliasov počítačov, preklad na tzv. kanonické mená (s využitím záznamov CNAME),
- určenie poštového serveru pre danú doménu (s využitím záznamov typu MX),



Obr. 2.1: Strom DNS (inšpirované z [3])

- podpora rozloženia záťaže medzi viac aplikačných serverov (rotácia záznamov),
- zdieľanie informácie v rámci globálneho priestoru mien
- delegovanie správy domén na jednotlivé subjekty (pomocou záznamov typu NS).

2.2 Architektúra systému DNS

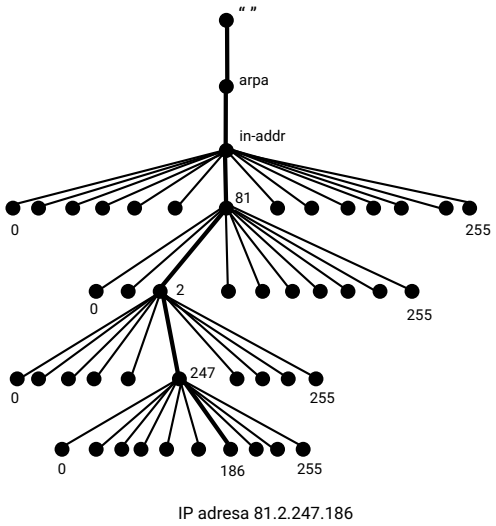
Systém DNS tvoria tri hlavné časti:

- **priestor doménových mien** - zastrešuje usporiadanie, štrukturalizáciu a sprístupnenie dát v systéme DNS,
- **DNS servery** - slúžia ako úložiská dát systému DNS,
- **resolver** - slúži na prístup k týmto dátam,

2.2.1 Priestor doménových mien

Priestor doménových mien je tvorený databázou hierarchicky usporiadanou do podoby koreňového stromu doménových mien (obrázok 2.1). Koreň stromu DNS je označený reťazcom nulovej dĺžky. Uzly stromu sú pomenované textovým reťazcom (bez bodiek a o maximálnej dĺžke 63 znakov). Tieto názvy tvoria časti doménových mien. Uzly s rovnakým bezprostredným predchodcom musia mať medzi sebou unikátne názvy, aby bola zaručená rozlíšiteľnosť uzlov (napr. fit.vutbr.cz., feec.vutbr.cz., fme.vutbr.cz.,). Cesta od listu ku koreňu slúži na vyhľadanie a uloženie doménových adries.

Databáza DNS obsahuje jednotlivé doménové mená usporiadané do domén. Jedna doména je reprezentovaná ako podstrom v priestore doménových mien, pričom doménové meno je vlastne cesta medzi vrcholovým uzlom a koreňom stromu DNS. Doména s vrcholom v uzle vo vzdialenosti jedna od koreňa grafu sa nazýva doména prvej úrovne, doména vo vzdialenosti dva od koreňa stromu DNS je doménou (či subdoménou) druhej úrovne a tak ďalej. Konkrétne sieťové zariadenia vrámci domeny sú reprezentované listami doménového podstromu. Doménové mená sa vyskytujú v dvoch podobách:



Obr. 2.2: Reverzný strom DNS (inšpirované z [3])

- **Absolútne doménové meno** - je tvorené postupnosťou bodkami oddeľovaných mien uzlov na ceste až ku koreňu stromu DNS. Každá absolútna adresa končí bodkou, pretože koreňový uzol má názov vo forme reťazca nulovej dĺžky.
- **Relatívne doménové meno** - je doménové meno bez bodky, ktoré sa interpretuje k relatívnej doméne, v ktorej sa nachádza.

Správa domén (pridávanie koncových uzlov, rušenie uzlov, vytváranie subdomén) je decentralizovaná a deleguje sa na ďalšie organizácie.

Systém DNS musí taktiež zaistiť reverzné (spätné) mapovanie IP adries na doménové mená. Pre účely reverzného mapovania existuje v dátovom priestore DNS špeciálna doména in-addr.arpa. (viď obr. 2.2). Uzly tejto domény reprezentujú čísla IP adresy v štvorbajtovom dekadickom tvare oddelené bodkami (napríklad 81.2.247.186). Doména in-addr.arpa. obsahuje 256 subdomén prvej úrovne, ktoré zodpovedajú prvému bytu IP adresy. Každá z týchto subdomén obsahuje ďalších 256 subdomén druhého rádu, potom tretieho rádu až do hĺbky štyri. IP adresy v strome sú usporiadané, narozdiel od doménového mena, od najvyššieho bytu k najnižšiemu. Doménové mená sa ukladajú od najvyššej domény, ktorá sa uvádza až na koniec.

Strom DNS nie je uložený na jednom mieste v jednej databáze, ale je hierarchický a decentralizovaný. Časti priestoru doménových adries sú fyzicky uložené na lokálnych serveroch DNS, ktoré dohromady tvoria systém DNS. Ten však nebosahuje iba doménové mená. Zahŕňa aj informácie o primárnych a sekundárnych serveroch DNS, správcoch domén, poštových serveroch a podobne. Všetky tieto informácie sú zapisované v textovom formáte do tzv. záznamov DNS (RR, resource records), ktoré popisujú RFC 1034 [5] a RFC 1035 [6]. Fyzické časti priestoru DNS, ktoré sú pod jednoutnou správou, sa nazývajú zóny. Zóna môže obsahovať celú doménu alebo len jej časť časť.

2.2.2 Resolver

Resolver je klientský program, ktorý sa pýta na dáta uložené v systéme DNS. Pomocou neho k týmto dátam pristupujú používateľské programy, ktoré potrebujú informácie z DNS.

Základnou úlohou resolveru podľa Matouška[3] je:

- posielať požiadavky na servery DNS,
- interpretovať odpovede od serveru (prijaté záznamy, chybové hlásenia),
- odovzdať informácie užívateľskému programu, ktorý o dáta žiadal.

Resolver musí byť schopný pristupovať k aspoň jednému serveru DNS. Od serveru získa priamo hľadanú odpoveď alebo mu server vráti odkaz na ďalší server, kde je hľadaná informácia uložená. Resolver tak môže preposielať požiadavky ďalším serverov. Resolver je obvykle implementovaný ako systémová rutina, ktorá je súčasťou operačného systému, ku ktorej priamo pristupujú užívateľské programy, napríklad nslookup či dig.

2.2.3 Server DNS (nameserver)

Ďalšou časťou systému DNS sú servery DNS (nameservers). Server DNS je aplikácia, ktorá uchováva dáta z priestoru doménových mien. Tento priestor je rozdelený do zón a umiestnený na jednotlivé servery DNS. Základnou úlohou DNS servera je odpovedať na požiadavky smerujúce na databázu DNS. Server DNS uchováva dáta vo forme množiny záznamov DNS (resource records). Záznamy sú uložené v lokálnom súbore alebo si ich server načíta z iného servera DNS pomocou prenosu zón. Informácie, ktoré server DNS spravuje a za ktoré je zodpovedný, sa nazývajú sa autoritatívne. Špecifikácia DNS [5] definuje dva základné typy serverov - primárny a sekundárny.

- **Primárny server DNS (master, primary nameserver)**

  Primárny server obsahuje úplné záznamy o doménach, ktoré spravuje. Tieto záznamy sú uložené lokálne v súbore. Server poskytuje autoritatívne (tj. vždy presné) odpovede na tieto domény. Pre každú doménu musí existovať práve jeden primárny nameserver.

- **Sekundárny server DNS (slave, secondary nameserver)**

  Sekundárny server získava dáta od primárneho serveru. Súbor, ktorý obsahuje databázu konkrétnej domény (či subdomény), sa nazýva zónový súbor. Proces prenosu zónových súborov z primárneho servera na sekundárne sa nazýva prenos zón (zone transfer). Sekundárny server musí zabezpečiť pravidelný prenos údajov zóny a aktuálnosť dát. Sekundárny server je tiež autoritatívny server pre danú doménu.

- **Záložný server DNS (caching-only nameserver)**

  Záložný server pracuje ako proxy server. Prijíma otázky od klientov a preposiela ich ďalším serverom DNS. Keď záložný server dostane odpoveď na svoju otázku, uchová si ju a použije ju v budúcnosti. Záložné servery poskytujú neautoritatívne odpovede, tj. odpovede, ktoré môžu byť neúplné a neaktuálne. Zrýchľujú však proces rezolúcie doménového mena.

Platnosť záznamov DNS na sekundárnom a záložnom serveri je časovo obmedzená. Hodnota expirácie je uvedená pri každom zázname. Pokiaľ dôjde k expirácii záznamu, musí server daný záznam zmazať zo svojej databázy, prípadne si ho znova načítať z primárneho servera. Ak by nevykonal aktualizáciu, môže dôjsť k situácii, keď dva rôzne servery DNS odpovedia na rovnakú požiadavku rôznym spôsobom, čo vedie k strate konzistencie dát. Najmä pri zmene v záznamoch DNS (zmena IP adresy, pridanie nového záznamu a pod.)

Je potrebné počítať s tým, že rozšírenie zmien trvá niekoľko hodín, čo je doba, kedy dôjde k expirácii pôvodných záznamov a načítanie nových. Na druhej strane by nastavenie krátkej doby expirácie viedlo k častým otázkam na obnovenie záznamu a preťaženie autoritatívnych serverov. Z dôvodu, že sa zmeny šíria v sieti DNS pomaly, je nutné v prípade, že potrebujeme zistiť aktuálnu hodnotu nejakého záznamu v DNS, kontaktovať priamo primárny alebo sekundárny server DNS.

2.3 Záznamy DNS (Resource Records)

Pre ukladanie informácií v dátovom priestore DNS slúžia záznamy DNS. Záznamy sa ukladajú v textovej podobe v zónových súboroch na serveroch DNS. Každý záznam musí dodržovať formát definovaný štandardom RFC 1035[6]. Záznam obsahuje meno, typ, triedu, ttl[1] a dĺžku záznamu, za ktorými nasledujú informácie, ktoré sa líšia podľa typu záznamu. Najbežnejšie záznamy sú:

Záznam SOA – Start Of Authority

Záznam SOA obsahuje informácie o autoritatívnych dátach pre danú zónu. Každá zóna má práve jeden typ tohto záznamu. Záznam obsahuje meno primárneho serveru DNS pre danú doménu, kontakt na správcu domény (emailovú adresu), sériové číslo pre identifikáciu zmien záznamu a ďalšie informácie pre iné DNS servery.

```
$TTL 3600
@ IN SOA isa.fit.vutbr.cz. root.isa.fit.vutbr.cz.(
              2007110901 ; Serial
              3600 ; Refresh
              900 ; Retry
              3600000 ; Expire
              3600 ) ; Minimum
```

Záznam A – IP Address

Záznam A je základný záznam, ktorý obsahuje preklad doménovej adresy na IP adresu

```
eva.fit.vutbr.cz. IN A~147.229.176.14.
```

Záznam NS – Name Server

NS záznam obsahuje autoritatívne servery pre danú doménu. Autoritatívny server obsahuje platné záznamy pre danú doménu.

```
fit.vutbr.cz. IN NS gate.feec.vutbr.cz.
              IN NS guta.fit.vutbr.cz.
              IN NS kazi.fit.vutbr.cz.
              IN NS rhino.cis.vutbr.cz.
```

[1] Time to live – parameter DNS záznamu, ktorý obsahuje dobu platnosti záznamu

Záznam MX – Mail Exchanger

Záznam MX obsahuje informácie o poštovom serveri danej domény. Doména môže obsahovať viacero poštových serverov a je možné im priradiť rôzne priority.

```
fit.vutbr.cz. IN MX 10 kazi.fit.vutbr.cz. # vyšší priorita
              IN MX 20 eva.fit.vutbr.cz
```

Záznam CNAME – Canonical Name

Záznam typu CNAME označuje, že daná doména je aliasom inej domény a tým pádom majú spoločné DNS záznamy.

```
www     IN CNAME tereza.fit.vutbr.cz.
ldap    IN CNAME tereza.fit.vutbr.cz.
tereza  IN A~147.229.9.22
```

Záznam PTR – Domain Name Pointer

Záznam typu PTR vykonáva spätné (reverzné) mapovanie. To znamená, že prevádza číselnú IP adresu na doménové meno.

```
14.176.229.147.in-addr.arpa. IN PTR eva.fit.vutbr.cz.
```

Záznam TXT – Text

Záznam TXT uchováva v DNS textové dáta týkajúce sa doplňujúcich informácií o doméne, serveri, správcovi a pod.

```
fi.muni.cz. 1773 IN TXT "Masaryk University Brno"
fi.muni.cz. 1773 IN TXT "Botanicka 68a, 602 00, Brno, Czech republic"
fi.muni.cz. 1773 IN TXT "Faculty of Informatics"
```

2.4 Aktualizácia zónových dát

Vytvorenie, pridávanie, editovanie či mazanie dát v zónovom súbore môžeme vykonávať ručne editáciou textového súboru alebo pomocou automatizovaných nástrojov. Dôležité je si uvedomiť, že tieto zmeny sa môžu vykonávať iba na primárnom DNS serveri. Ak vykonáme zmenu napríklad na sekundárnom serveri, prepíše sa táto zmena pri najbližšej synchronizácii. Aktualizáciu zónového súboru tvoria nasledujúce kroky:

1. Aktualizácia sériového čísla v SOA zázname príslušného zónového súboru
2. Pridanie, úprava či odstránenie záznamov typu A, CNAME, MX, apod. v zónovom súbore
3. Pridanie, úprava či odstránenie záznamu typu PTR v reverznom zónovom súbore
4. Reštart primárneho servera DNS - aplikácia si načíta aktualizované dáta do pamäte.

Kapitola 3

Android

Nasledujúci text má za úlohu oboznámiť čitateľa s platformou Android z hľadiska tvorby mobilných aplikácií. Informácie čerpá z oficiálnej dokumentácie[17], vývojárskeho sprievodcu[12], dizajnovej smernice[15] a materiálovej špecifikácie [10].

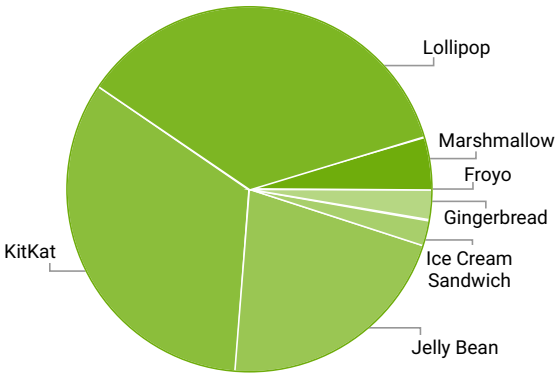
V prvej časti je tento systém všeobecne predstavený a je načrtnutá jeho momentálna situácia. Podkapitola 3.2 čitateľa prevedie architektúrou systému, za účelom vytvorenia predpokladu pre pochopenie jeho fungovania. Časť 3.3 sa zaoberá základnými komponentami samotných aplikácií, na ktoré sa bude neskôr odkazovať implementačná časť práce. Nakoniec je v podkapitole 3.4 načrtnutá problematika prístupu k tvorbe používateľského rozhrania, ktorá bola pre túto platformu vytvorená a je dnes pre ňu štandardom.

3.1 Platforma Android

Android je rozsiahla open source platforma, ktorá vznikla najmä pre mobilné zariadenia (smartphone, PDA, navigácie, tablety). Zahŕňa v sebe operačný systém (založený na jadre Linux), middleware, používateľské rozhranie a aplikácie. Vyvíja ho konzorcium Open Handset Alliance, ktorého cieľom je progresívny rozvoj mobilných technológií, ktoré budú mať výrazne nižšie náklady na vývoj a distribúciu, a zároveň spotrebiteľom prinesú inovatívne používateľsky prívetivé prostredie.

Verzia	Označenie	API	Distribúcia
2.2	Froyo	8	0.1%
2.3.3 - 2.3.7	Gingerbread	10	2,2%
4.0.3 - 4.0.4	Ice Cream Sandwich	15	2,0%
4.1.x	Jelly Bean	16	7,2%
4.2.x		17	10,0%
4.3		18	2,9%
4.4	KitKat	19	32,5%
5.0	Lollipop	21	16,2%
5.1		22	19,4%
6.0	Marshmallow	23	7,5%

Tabuľka 3.1: Tabuľka distribúcie verzií Androidu (zdroj Android Developers [7])



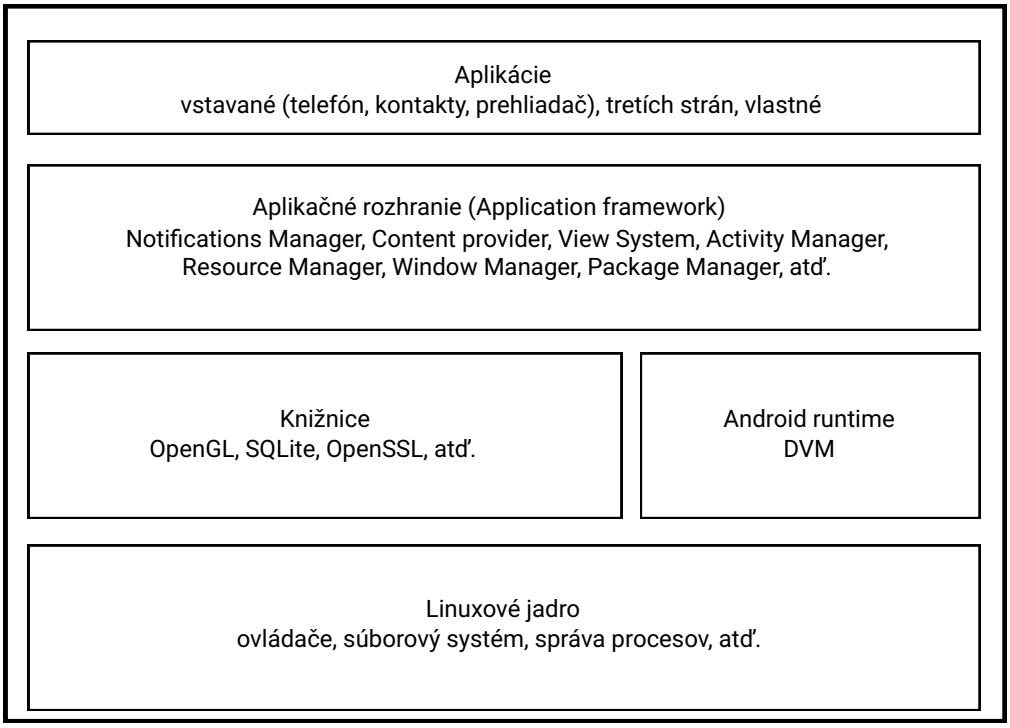
Obr. 3.1: Graf vizualizujúci dáta z tabuľky 3.1

Android je aj v súčastnosti dynamicky rastúcim a vyvíjajúcim sa aktívnym projektom,

ktorý je stále rozširovaný o nové vlastnosti s každou novou verziou. V čase písania tejto práce je najnovšou verziou systému Android 6.0 Marchmallow (API 23). Z tabuľky 3.1 a grafu 3.1 je možné odpozorovať, že najviac rozšírenými (v zmysle distribúcie na zariadeniach) sú verzie Android Lollipop (5.0, 5.1), Android 4.4 KitKat a Android Jelly Bean (4.1.x, 4.2.x a 4.3). Tie spolu aktuálne zastrešujú niečo cez 88% aktívnych zariadení. Využitie nových vlastností systému aj pri vývoji pre staršie verzie je umožnené prostredníctvom sady podporných knižníc, tzv. Support Library. Prilinkovanie tejto sady knižníc umožňuje tieto nové funkcie využívať.

3.2 Architektúra

Operačný systém Android je tvorený zoskupením softvérových komponentov, ktoré je možné zhruba rozčleniť do piatich sekcií a štyroch hlavných vrstiev. Toto ročlenenie je znázornené diagramom na obrázku 3.2.



Obr. 3.2: Diagram architektúry Android OS

Linuxové jadro (Linux Kernel)

Najnižšiu vrstvu tvorí jadro operačného systému postavené na Linuxe. Poskytuje abstraktnú vrstvu medzi používaným hardvérom a softvérom. Zahŕňa všetky základné ovládače, ktoré sa starajú o správu pamäte, procesov alebo periférnych zariadení, ako fotoaparát, klávesnica či obrazovka.

Knižnice

Nad jadrom operačného systému je skupina knižníc napísaných v C/C++ kóde, ktorú využívajú rôzne časti systému. Ich funkcie sú vývojárom poskytnuté prostredníctvom Android Application Framework. Medzi tieto knižnice patria:

- **OpenGL** - knižnica na vykresľovanie 3D grafiky.
- **SQLite** - odľahčená knižnica pre prístup k relačným databázam.
- **Libc** - odvodená BSD štandardná knižnica systému C vyladená pre embedded zariadenia.
- **OpenSSL** - zodpovedajúca za bezepečnosť na internete.
- **Media Libraries** - knižnica pre podporu obrazových súborov a prehrávania videa, či zvuku.
- **FreeType** - knižnica pre vykresľovanie vektorových a bitmapových písem.

Android Runtime

Táto tretia sekcia, ktorá je súčasťou druhej vrstvy, poskytuje kľúčovú súčasť OS nazývanú Dalvik Virtual Machine (DVM). DVM je odnožou virtuálneho stroja Java Virtual Machine, špeciálne navrhnutou a optimalizovanou pre Android. Využíva hlavné funkcie Linuxu, ako správa pamäti či multithreading. Ummožňuje každej Android aplikácii spustiť vlastný proces, ktorý beží na samostatnej instacii DVM, čím je poskytnutá bezpečnosť, izolácia a správa pamäte. Okrem toho táto vrstva taktiež poskytuje skupinu knižníc, ktoré umožňujú vývojárom vytvárať Android aplikácie za použitia programovacieho jazyka Java.

Aplikáčné rozhranie (Application Framework)

Vrsta aplikačného rozhrania pokytuje aplikáciám množstvo služieb vo forme tried jazyka Java a umožňuje vývojárom využiť tieto služby v ich programoch. Aplikačné rozhranie zahŕňa tieto nasledujúce kľučové služby:

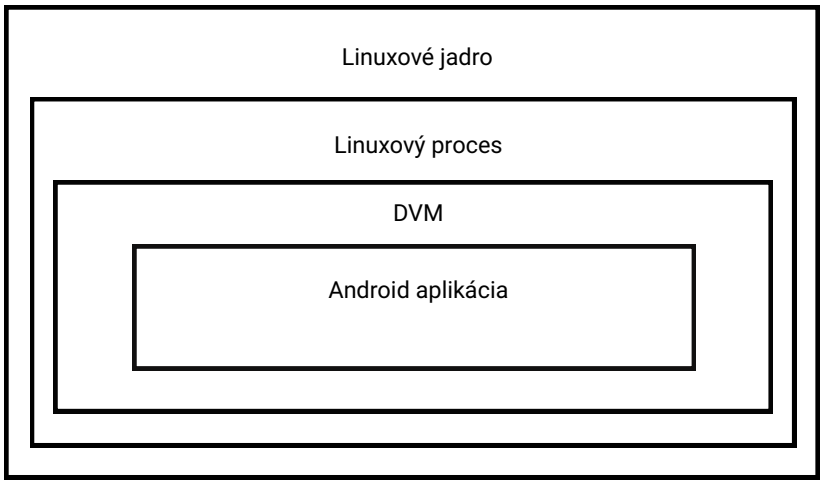
- **View System** - rozšíriteľný súbor komponentov pre vytvorenie používateľského rozhrania.
- **Activity Manager** - má na starosti všetky aspekty životného cyklu aplikácie.
- **Resource Manager** - poskytuje prístup k vstavaným zdrojom nekódového charakteru ako reťazce, nastavenia farieb a rozmiestnenie užívateľského rozhrania.
- **Content Provider** - umožňuje aplikáciám zverejňovať a zdieľať dáta s inými aplikáciami.
- **Notifications Manager** - umožňuje zobrazovať upozornenia a notifikácie používateľovi.
- **Window Manager** - spravuje všetky okná aplikácií.
- **Package Manager** - obsahuje zoznam všetkých aplikácií nainštalovaných na zariadení.

Aplikácie

Aplikácie tvoria najvyššiu vrstvu operačného systému Android. Sú to všetky nainštalované aplikácie na zariadení, ktoré využívajú bežní používatelia. Môže ísť o predinštalované programy ako aplikácia pre príjmanie a odosielanie SMS, kontakty či internetový prehliadač, alebo ďalšie aplikácie tretích strán.

3.3 Základné komponenty aplikácie

Ako už bolo spomenuté, Android beží na linuxovom jadre a Android aplikácie sú napísané v programovacom jazyku Java. Tieto aplikácie bežia vrámci instancií virtuálneho stroja DVM, ktorý zas spadá pod proces spravovaný linuxovým jadrom, ako je názorne ukázané na obrázku 3.3.



Obr. 3.3: DVM

Samotné aplikácie pozostávajú zo stavebných komponentov, ktoré sú pre konkrétnu aplikáciu voľne zoskupené v aplikačnom manifeste AndroidManifest.xml. Ten obsahuje popis všetkých komponentov aplikácie a spôsobu ich interakcie. Podľa oficiálnych vývojárskych stránok projektu Android, popisujúcih aplikačné základy[13], tvoria nasledujúce štyri komponenty základné stavebné prvky v Android aplikáciách:

- `Activity` - prvok, ktorý reprezentuje užívateľské rozhranie a zastrešuje interakciu medzi používateľom a obrazovkou zariadenia.
- `Service` - komponent, ktorý zabezpečuje spracovanie na pozadí súvisiace z aplikáciou.
- `Broadcast Receiver` - má na starosti komunikáciu medzi operačným systémom a aplikáciou.
- `Content Provider` - zastrešuje správu dát.

3.3.1 Activity

Komponent `Activity` reprezentuje jednu obrazovku s užívateľským rozhraním, ktorá je implementovaná ako podtrieda rovnomennej triedy `Activity`. V skratke `Activity` vykonáva

akcie na obrazovke. Napr. aplikácia Kontakty môže obsahovať jednu `Activity`, ktorá zobrazuje zoznam všetkých kontaktov, ďalšiu pre vytvorenie kontaktu a ďalšiu pre zobrazenie detailu kontaktu. Ak aplikácia pozostáva z viac ako jednej `Activity`, tak by jedna z nich mala byť označená ako tá, ktorá je zobrazená po spustení aplikácie. Tá môže následne spustiť ďalšiu `Activity` pomocou metód `startActivity()` a `startActivityForResult()`. V tom prípade dôjde k vloženiu volajúcej `Activity` na vrchol zásobníku, pre jej rýchle obnovenie pri spätnej akcii a jej následné ukončenie. Životný cyklus `Activity` znázorňuje obrázok 3.4. V oficiálnej dokumentácii[11] je uvedené, že sa `Activity` môže nachádzať v štyroch štádiách:

- **Aktívna (Running)** - v tomto štádiu je `Activity` aktívna, nachádza sa na popredí obrazovky a na vrchole zásobníku.
- **Pozastavená (Paused)** - v prípade, že `Activity` nieje na popredí, avšak je zároveň stále viditeľná, nachádza sa v stave pozastavenia. V tomto štádiu je stále plne funkčná, naďalej obsahuje prislušné informácie, avšak pri nedostatku pamäte môže byť systémom ukončená, či zničená.
- **Zastavená (Stopped)** - toto štádium nastáva vtedy, keď je `Activity` úplne prekrytá a tým pádom nieje vôbec viditeľná. Stále si uchováva svoje informácie, no v prípade nedostatku pamäte môže dôsť rovnako ako v pozastavenom stave k jej nútenému zániku.
- **Neaktívna** - `Activity` nachádzajúca sa v stave pred spustením. Do tohoto stavu prechádza aj v prípade núteného ukončenia pozastavenej, či ukončenej `Activity` pri nedostatku pamäte.

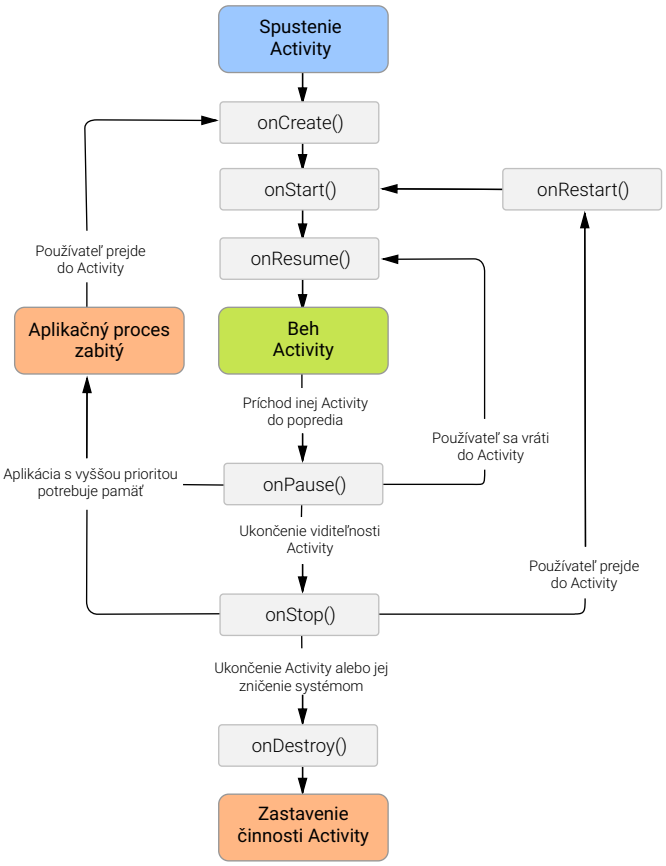
3.3.2 Service

`Service` je komponent, ktorý beží na pozadí za účelom vykonania dlho trvajúcich operácií. Typickým príkladom je hranie hudby na pozadí, kým je používateľ v inej aplikácii. `Service` je implementovaná ako podtrieda triedy s rovnakým názvom `Service`. `Service` nereprezentuje žiadny prvok používateľského rozhrania. Je úplne nezávislá od spúšťajúcej aplikácie a môže bežať aj po jej ukončení. Podľa dokumentácie[18] poznáme dva typy `Service`:

- **zahájená (started)** – je "zahájená"z iného komponentu prostredníctvom `startService()` a ukončí sa sama až po ukončení operácie a to aj v prípade, že zahajujúci komponent už zanikol,
- **naviazaná (bound)** – pomocou `bindService()` sa naviaže na existujúce komponenty aplikácie, ktoré jej následne poskytujú dáta. `Service` ukončí svoju činnosť, keď zaniknú všetky komponenty, na ktoré je naviazaná.

3.3.3 Content Provider

`Content Provider` na požiadanie poskytuje dáta jednej aplikácie ostatným. Takéto požiadavky sú obsluhované metódami triedy `ContentResolver`, ktorej instanciu je možné získať prostredníctvom `getContentResolver()`. Tieto dáta môžu byť uložené v súborovom systéme, databáze alebo niekde úplne inde. `Content Provider` je vybraný na základe `Content URI` v nasledujúcom tvare[14]:



Obr. 3.4: Životný cyklus Actvity (inšpirované z [11])

`content://authority/optionalPath/optionalId`

Časť `authority` identifikuje požadovaný `Content Provider`, `optionalPath` určuje požadovaný typ dát a prostredníctvom `optionalId` je vyselektovaný jedinečný záznam podľa unikátneho čísla. `ContentResolver` poskytuje metódy pre nasledujúce operácie:

- `insert()`
- `update()`
- `delete()`
- `query()`

3.3.4 Broadcast Receiver

`Broadcast Receiver` odpovedá na broadcastové správy od iných aplikácií alebo systému. Aplikácie iniciujú broadcast za účelom oboznámenia ostatných aplikácií o dokončení sťahovania dát a o ich dostupnosti. `Broadcast Receiver`, ktorý túto komunikáciu zachytí, zaháji príslušnú akciu. Komponenty `Broadcast Receiver` sú podtriedami triedy `BroadcastReceiver` a každá správa broadcastu je objektom triedy `Intent`. `Broadcast Receiver` je možné registrovať dvoma spôsobmi:

- **dynamicky** - prostredníctvom `registerReceiver()`,
- **staticky** - uvedením elementu `<receiver>` v Android manifeste.

3.3.5 Ďalšie komponenty

Okrem spomenutých hlavných komponentov existujú ešte ďalšie, ktoré sa podieľajú na stavbe spomenutých entít, ich logiky a prepojenia:

- **Fragment** - reprezentuje časť užívateľského rozhrania v `Activity`.
- **View** - UI prvok zobrazený na obrazovke, zahŕňajúci zoznamy či tlačidlá.
- **Layout** - hierarchické usporiadanie prvkov `View`, ktoré určuje ich formát a vzhľad.
- **Intent** - správa, ktorá slúži na prepojenie komponentov.
- **Resources** - externé zdroje ako reťazce, konštanty a obrázky.
- **Manifest** - konfiguračný súbor aplikácie.

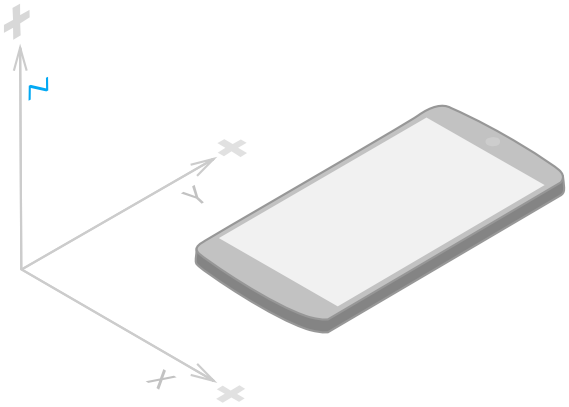
3.4 Material design

S príchodom verzie Android 5.0 Lollipop bol svetu predstavený nový prístup k používateľskému rozhraniu - Material design. Podľa oficiálnej špecifikácie[10] sa jedná o vizuálny jazyk pre používateľov, ktorý spája klasické princípy dobrého designu s inováciou a možnosťami technológie a vedy. Má za cieľ zjednotiť dizajn skrz všetky platformy, či sa už jedná o webové stránky, mobilné alebo desktopové aplikácie. Tento prístup je postavený na troch princípoch:

- **Metafora hmoty (tzv. materiál)** - je zjednotená teória hospodárneho rozmiestnenia objektov a systému pohybu. Plochy a hrany materiálu dodávajú prvkom vizuálne vlastnosti založené na realite. Jeho flexibilita im však dodáva vlastnosti presahujúce možnosti hmotného sveta a to bez porušenia fyzikálnych zákonov. Presun objektov v priestore, ich interakcie a priestorové vzťahy sú vyobrazené pomocou svetla, plôch a základných pohybov.
- **Základné prvky printového dizajnu** - typografia, mriežky (grids), voľný priestor, mierka, farby a využitie obraznosti. Kombináciou týchto elementov dochádza ku vzniku hierarchie, dodaniu významu a vytvoreniu ústredných bodov. Pri ich použití s dôrazom na používateľské úkony je hlavná funkcionalita okamžite zjavná a používateľom poskytujú vrámci používateľského prostredia orientačné body a medzníky.
- **Pohyb dodávajúci význam** - primárne používateľské ukony sú reprezentované inflexnými bodmi, ktoré iniciujú pohyb a tým transformujú celý dizajn. Všetky úkony sa odohrávajú v jednotnom prostredí a objekty sú pri zmenách súvisle transformované a reorganizované. Všetok pohyb má svoj význam a slúži k presunu pozornosti pri zachovanej spojitosti.

Materiálové prostredie

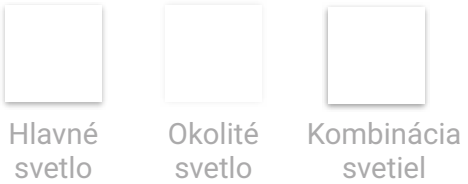
Prostredie, v ktorom materiál existuje, je trojrozmerný priestor, čo znamená, že všetky objekty majú rozmery `x`, `y` a `z`. Os `z` je kolmá vzhľadom k rovine obrazovky zariadenia a jej hodnota rastie kladne smerom k pozorovateľovi (viď. obr. 3.5). Každý kus materiálu zaberá jednu pozíciu na osi `z` a má štandardnú hrúbku 1dp[1].



Obr. 3.5: 3D priestor s osami x, y a z (vytvorené na základe [9])

Toto prostredie osvetľujú dva zdroje virtuálneho svetla (názornú ukážku je možné vidieť na obrázku):

- **Hlavné svetlo** - priame svetlo s konkrétnym smerovaním a tomu zodpovedajúcim vrhaním tieňov.
- **Okolité svetlo** - vytvára jemné tiene zo všetkých uhlov.



Obr. 3.6: Vrhanie tieňov rôznych svetelných zdrojov (vytvorené na základe [9])

Vlastnosti materiálu

Všetok materiál má isté nemeniteľné vlastnosti a prirodzené správanie:

- rozmery materiálu `x` a `y` sú variabilné ale rozmer `z` má uniformnú hrúbku o rozmere 1dp,

[1] Density-independent pixel (dp) - jednotka zobrazenia prvkov na obrazovkách s rôznou hustotou pixelov. Je ekvivalentná jednému pixelu na obrazovke s hustotou pixelov 160 a je k tejto hustote priamo úmerná.

- každý materiál vrhá tieň, ktorý je prirodzeným výsledkom vyvýšenia materiálu v priestore (na osi `z`),
- obsah je na materiále zobrazený v ľubovoľnom tvare a farbe, pričom žiadnym spôsobom materiálu nepridáva na hrúbke,
- obsah sa môže správať nezávislo na materiále, ale je obmedzený vrámci jeho hraníc,
- materiál je pevný:
  - vstupné udalosti ovplyvňujú iba jeho popredie a neprechádzajú cezeň,
  - jednotlivé materiály cez seba nemôžu prechádzať,
- viacero materiálov nemôže súčasne zaberať to isté miesto v priestore,
- materiál môže dynamicky meniť svoj tvar, veľkosť a farbu - k týmto zmenám však dochádza len v rovine,
- materiál sa neskladá a neohýba,
- jednotlivé kusy materiálu sa môžu zlúčiť do jedného,
- môže byť ľubovoľne vytvorený či zničený vrámci celého prostredia,
- môže sa pohybovať pozdĺž hociktorej z osí,
- pohyb po osi `z` je typicky výsledkom používateľskej interakcie s materiálom.

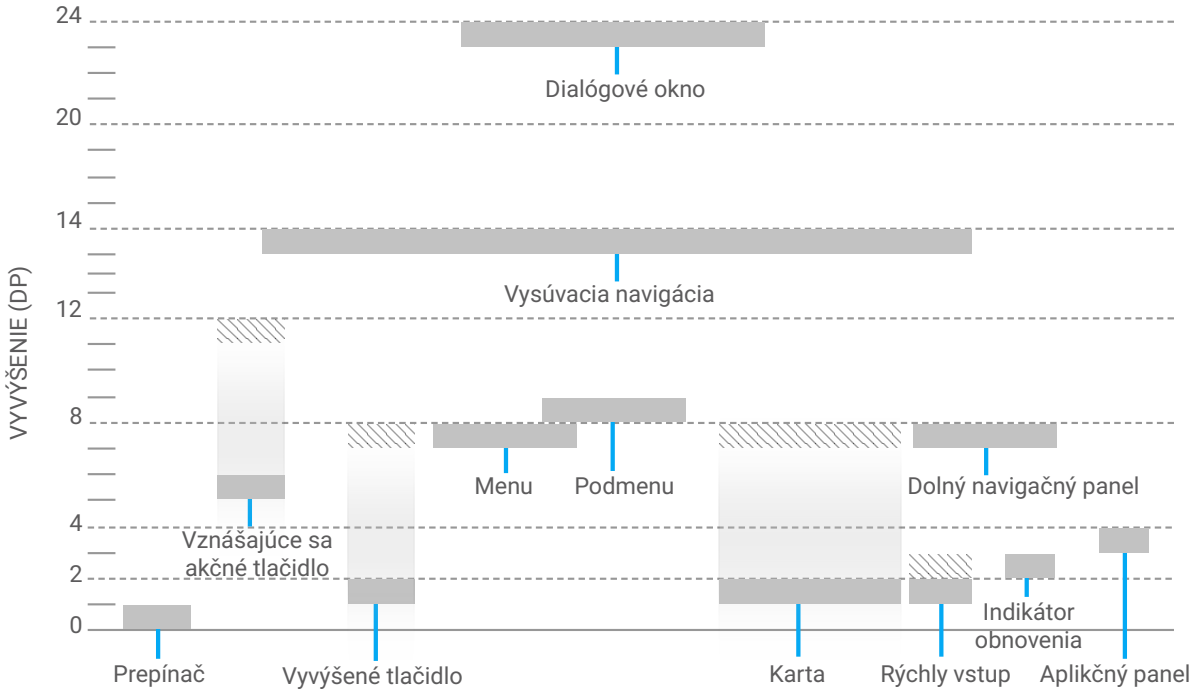
Vyvýšenie materiálu

Umiestenie materiálového objektu v priestore na osi `z` je určené vlastnosťou, ktorá sa nazýva **vyvýšenie**. Vyvýšenie reprezentuje relatívnu hĺbku, resp. vzdialenosť, medzi dvomi plochami pozdĺž osi `z`, ktorá sa typicky meria v dp. Keďže každý materiálový prvok má hrúbku 1dp, berie sa v úvahu vzdialenosť medzi vrchnými stranami plôch. Dôležitým faktom je, že vyvýšenie každého potomka je relatívne k vyvýšeniu jeho rodiča.

Každý materiálový objekt, nezávisle na veľkosti, má predvolenú pozíciu **vyvýšenia v pokojnom stave**, ktorá sa nemení. Toto vyvýšenie je konzistentné pre všetky komponenty rovnakého druhu naprieč všetkými aplikáciami (tzn. vyvýšenie napr. aplikačnej lišty je pre všetky aplikácie rovnaké).

Niektoré druhy komponetov môžu svoje vyvýšenie meniť vrámci reakcie na používateľský vstup alebo systémové udalosti. Ide o tzv. **responzívne vyvýšenie**, ktoré je implementované pomocou dynamického odstupu. Responzívne vyvýšenie je, tak ako vyvýšenie v pokojnom stave, konzistetné naprieč všetkými aplikáciami pre všetky komponenty toho istého typu. Po ukončení vstupnej udalosti sa komponent nachádzajúci na pozícii responzívneho vyvýšenia navráti na miesto zodpovedajúce pokojnému stavu. Diagram porovnania vyvýšení jednotlivých typov komponentov je znázornený na obrázku 3.7.

Hodnota vyvýšenia určuje výzor tieňa, ktorý objekt vytvára. Tento tieň je jedinou vizuálnou stopou, ktorá indikuje vzdialenosť medzi jednotlivými plochami. Táto jeho vlastnosť je dôležitým faktorom obzvlášť pri pohybe objektov, kde naznačuje smer posunu a zároveň prezrádza, či sa vzdialenosť objektov zväčšuje alebo zmenšuje.



Obr. 3.7: Diagram porovnania vyvýšení jednotlivých komponentov (vytvorené na základe [8])

Vzťahy objektov

Spôsob organizovania objektov v aplikácii určuje ich správanie medzi sebou. Objekty sa môžu pohybovať nezávisle od ostatných alebo môžu byť obmedzené hierarchicky nadradeným objektom.

Všetky objekty sú súčasťou hierarchie postavenej na vzťahoch rodič-potomok. Potomok v tomto vzťahu odkazuje na objekt podriadený rodičovskému prvku. Objekty môžu byť potomkami systému alebo iného objektu, pričom objekty na rovnakej úrovni hierarchie nazývame súrodenci.

V tejto hierarchii platia nasledovné pravidlá:

- každý objekt má práve jedného rodiča,
- každý objekt môže mať ľubovoľný počet potomkov,
- potomok dedí transformačné vlastnosti rodiča, ako napr. pozícia, rotácia, mierka a vyvýšenie.

Kapitola 4

Mobilná aplikácie pre správu DNS

Nasledujúca kapitola pojednáva o aplikácii, ktorej vytvorenie je predmetom tejto práce.

V prvej časti 4.1 je rozobraná analýza požiadaviek na takúto aplikáciu. Na začiatku sa nachádza prehľad existujúcich mobilných aplikácií podobného charakteru. Následne sú čitateľovi predstavené najpoužívanejšie softvérové riešenia DNS a tie, ktoré sú vybrané za cieľové, sú za účelom analýzy požiadaviek bližšie rozobraté. Na základe týchto informácií je na konci súhrn požiadaviek predmetnej aplikácie.

Podkapitola 4.2 popisuje návrh a implementáciu tejto aplikácie na základe vytvorených požiadaviek. Postupne po častiach čitateľovi opisuje správanie a fungovanie jednotlivých časti programu doplnené o názorné ukážky.

Kapitolu uzatvára časť 4.3 venovaná testovaniu a zhodnoteniu jeho výsledkov, na základe ktorých, je doplnená o návrhy možných rozšírení vytvorenej aplikácie.

Celá kapitola sa opiera o teoretické znalosti zhrnuté v doterajších kapitolách 2 a 3, ktoré sú na relevantných miestach doplnené o nové poznatky.

4.1 Analýza požiadaviek

Táto sekcia sa zaoberá analýzov požiadaviek na vytvorenie mobilnej aplikácie pre správu DNS serveru.

V súčasnej dobe už existuje niekoľko mobilných aplikácií zaoberajúcich sa touto tematikou. Analýza týchto konkuretných riešení v podsekcii 4.1.1, je prvým prepokladom pre rozbor požiadaviek pre novú aplikáciu s podobným zameraním.

Druhá kapitola, zaoberajúca sa platformou Android, rozviedla požiadavky a predpoklady pre vytvorenie mobilnej aplikácie. V prvej kapitole bol predstavený systém DNS ako taký, avšak v softvérovej podobe existuje niekoľko jeho implementácií, ktoré sa od seba rôzne líšia. Čo táto existencia množstva rôzličných distribúcií znamená pre správu DNS, je rozobrané v podsekcii 4.1.2, ktorá čerpá nové informácie z [4], [1] a [2]. Na základe výsledkov tejto časti je následne možné analyzovať konkrétne možnosti a požiadavky pre správu DNS.

Vďaka všetkým získaným poznatkom a vyvodeným záverom je možné nakoniec zakončiť analýzu požiadaviek vytvorením špecifikácie pre vytvorenie novej aplikácie v časti 4.1.4.

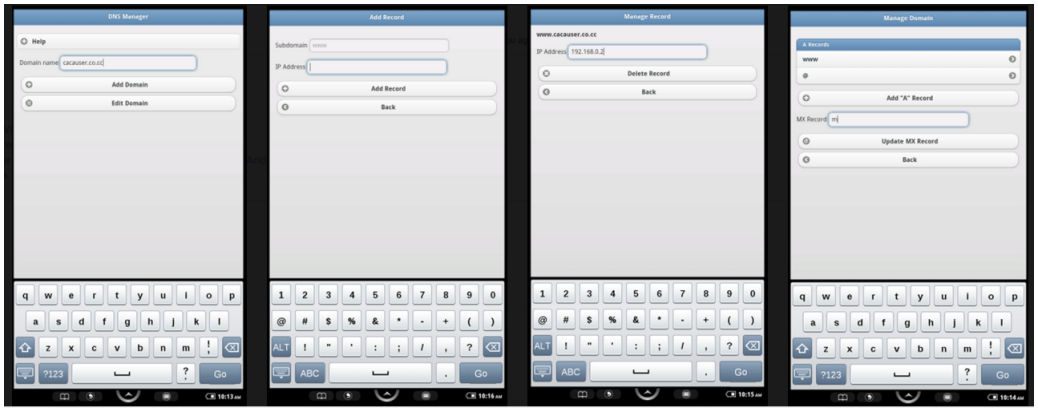
4.1.1 Prehľad existujúcich mobilných aplikácií

V súčasnej dobe je k dispozícii niekoľko druhov softvéru pre správu DNS. Do tejto množiny patria rozličné programy, od jednoduchých aplikácií s pár možnosťami až po komplexné riešenia systému DNS. Zväčša sa však jedná a webové alebo desktopové klientské aplikácie.

Na poli natívnych mobilných aplikácií to už tak ružové nieje. V nasledujúcej podkapitole by som chcel stručne priblížiť existujúce mobilné aplikácie podobného rázu, ktoré som objavil počas môjho trhového prieskumu.

DNS Manager

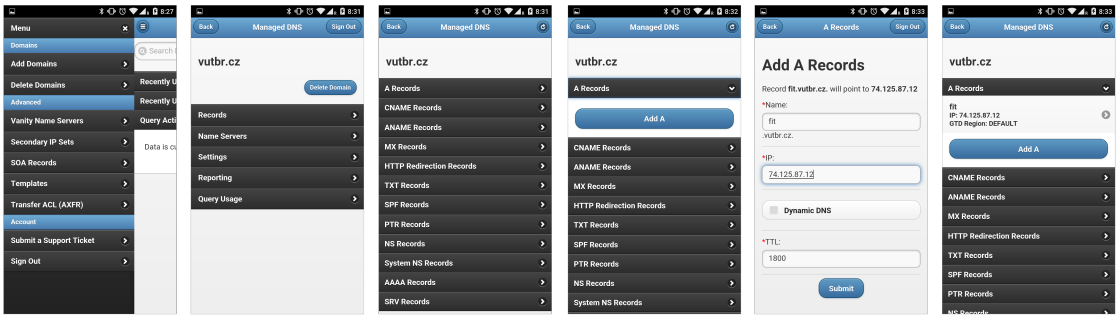
V drvivej väčšine aplikácií dostupných na Google Play Store, ktoré sami seba označujú za správcov DNS ide len o nastavenie nového DNS serveru pre resolver na zariadení. V prípade aplikácie DNS Manager od štúdia Open Merchant Account Ltd to tak však nieje. Tvorcovia tejto aplikácie o nej uvádzajú, že je s ňou možné editovanie DNS Ä"záznamov, "MX"záznamov a dokonca aj vytvorenie nových domén priamo za behu. To potvrdzujú aj prezentačné screenshoty aplikácie na obrázku 4.1. Podľa nich je možné usúdiť, že sa aplikácia vyznačuje minimalistickým dizajnom, avšak samotný popis napovedá, že aj obmedzenou funkcionalitou. Ku veľkej škode vyzerá byť aplikácia už nefunkčná. K tomuto záveru som dospel empirickým testovaním na 4 rôznych Android zariadeniach s rôznymi verziami OS, kde v každom prípade došlo k zlyhaniu aplikácie ihneď po štarte. Aplikácia sa už na Google Play Store nenachádza, stále je však k dispozícii na Amazon Store.



Obr. 4.1: Ukážka aplikácie DNS Manager

DNS Made Easy

DNS Made Easy je intuitívna, na prvý pohľad jednoducho ovládateľná aplikácia pre vzdialenú správu DNS serveru, s veľkým množstvom možností. Je tu k dispozícii široká škála možných záznamov (ako je možné vidieť na priloženom obrázku 4.2) a aplikácia poskytuje napr. aj možnosť grafickej reprezentácie štatistických dát o vyťaženosti servera. Háčikom tejto aplikácie však je, že je pomocou nej možné manažovať len domény spravované rovnomennou službou DNS Made Easy - nieje možné ju použiť na administráciu vlastného DNS servera. Aplikácia slúži ako klietske ovládacie prostredie pre klientov tejto paušálnej služby.

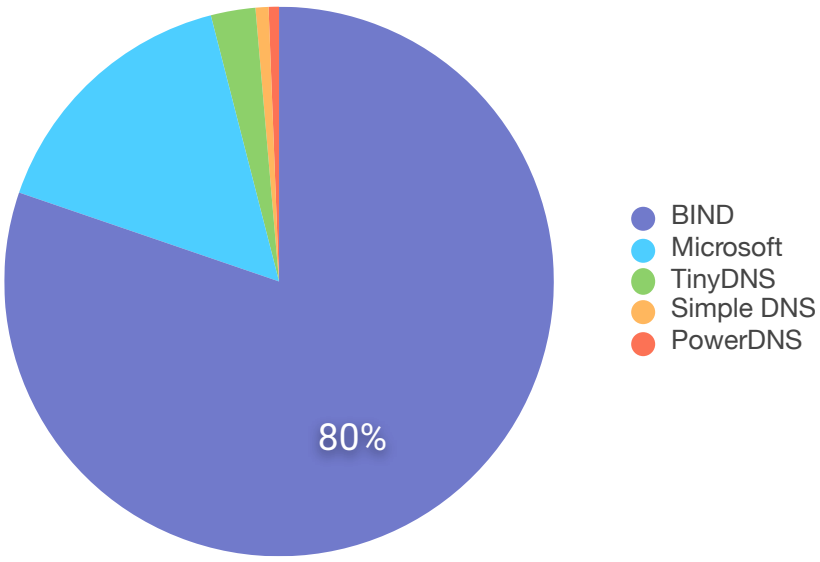


Obr. 4.2: Ukážka aplikácie DNS Made Easy

4.1.2 Prehľad distribúcií DNS softvéru

V úvode tejto sekcie 4.1 pojednávajúcej o analýze požiadaviek bolo zmienené, že dnešnej dobe existuje viacero rozličných softvérových riešení systému DNS. Jedná sa o nezávyslé implementácie DNS protokolu, schopné rezolúcie doménových mien. Podľa posledných štatistík z januára 2016[19], ktoré uverejnila organizácia ISC (Internet Systems Consorcium), je narozšírenejšou distribúciou softvér BIND (Berkeley Internet Name Domain). Za ním nasledujú programy Microsoft DNS, TinyDNS, Simple DNS a Power DNS. BIND však svojím zastúpením medzi distribúciami s celkovými 80% jednoznačne dominuje (viď. graf 4.3) a je často označovaný ako DNS štandard pre unixové systémy.

V súčasnosti existuje viac než 20 rôznych softvérových distribúcií DNS systému[4]. Mnohé tieto distribúcie sa ďalej delia na vnútorné verzie, ktorých obsluha sa môže medzi sebou líšiť. Vzhľadom k tomuto faktu, je možné vyvodiť, že vytvorenie mobilnej aplikácie umožňujúcej efektívnu správu DNS serverov naprieč všetkými distribúciami by bolo viac než komplikované a výkonovo zbytočne náročné. Zacielením na jednu distribúciu s jednotnou formou manipulácie, je možné tento problém eliminovať. Ideálnym kandidátom je najrozšírenejší DNS softvér v súčastnosti, BIND 9, ktorým je možné pokryť valnú väčšinu všetkých DNS serverov.



Obr. 4.3: 5 najrozšírenejších distribúcií systému DNS (vytvorené na základe [19]))

4.1.3 Správa BIND 9

Softvérová distribúcia BIND 9 v sebe zahŕňa dve ústredné časti:

- `liblwres` – knižnica zastrešujúca funkcie resolveru,
- `named` – samotný nameserver, ktorý je predmetom správy vytváranej aplikácie.

Prostredie BINDu je ovládané prostredníctvom konfiguračného súboru `named.conf`. Ten umožňuje nastevenie serveru pomocou ohromného množstva konfiguračných entít v podobe rôznych výrazov a klauzúl[1], typicky v nasledujúcom formáte:

```
<klauzula-1> ["<meno-klauzuly-1>"] [<trieda-klauzuly-1>] {
   <výraz-1>;
   <výraz-2>;
   <výraz-N>;
};
<klauzula-N> ["<meno-klauzuly-N>"] [<trieda-klauzuly-N>] {
   <výraz-1>;
   <výraz-2>;
   <výraz-N>;
};
```

Mimo iné, obsahuje tento súbor zoznam všetkých zón, ktoré DNS server spravuje. Jednu zónu v konfiguračnom súbore reprezentuje klauzula `zone`. Všetky serverom spravované zóny (typu master) sa typicky nachádzajú v textových súboroch umiestnených na serveri, keďže BIND sám o sebe nemá databázovú podporu.

Cieľová aplikácia by oprávnenému používateľovi mala umožniť k týmto zónovým dátam pristupovať a podľa potreby nimi manipulovať. K týmto úkonom sa dá pristupovať viacerými spôsobmi.

Priama editácia relevantných textových súborov

Prvým a klasickým prístupom je priama editácia zónových súborov (spomínaná v časti 2.4). Používateľ priamo na serveri pristupuje k textovým súborom s informáciami o zónach (zónové a konfiguračné súbory) a v prípade potreby ich upravuje v textovom editore. Nevýhodou tohoto prístupu je hlavne chybovosť ľudského faktoru, kedy používateľ môže ľahko urobiť chybu pri ručnom zapisovaní výrazov. Tento problém by však mohol byť ľahko eliminovaný vytvorením používateľského rozhrania, ktoré by malo vytváranie týchto výrazov na starosti, na základe kontrolovaných používateľských vstupov.

Nástroje `nsupdate` a `rndc`

Druhým možným prístupom je použitie konfiguračných nástrojov, ktoré BIND 9 podporuje. Ide o nástroje:

- `nsupdate` (Dynamic DNS update utility) – nástroj umožňujúci dynamické úpravy zónových súborov (práca s DNS záznammi),
- `rndc` (Remote Name Daemon Control) – nástroj na vzdialenú konfiguráciu serveru `named` (okrem iného podporujúci aj vytváranie a mazanie zón).

[1] Úplný zoznam klauzúl a výrazov je uvedený v BIND 9 Administrator Reference Manual[2]

Ich použitie by poskytlo jednoduchší prístup k veľkému množstvu konfiguračných možností, ktoré tieto nástroje zastrešujú. Veľkou nevýhodou tohto spôsobu je však to, že je plne závyslý od veľmi špecifických a vopred pripravených konfiguračných nastavení cieľového servera.

Prídavný modul DLZ (Dynamically Loadable Zones)

Jedná sa o doplnok programu BIND 9 podporovaný od verzie 9.4, ktorý umožňuje získavať zónové dáta priamo z externej databázy. DLZ ovládače existujú pre viacero rôznych databázových backendov vrátane PostgreSQL, MySQL, a LDAP. V tomto prípade by aplikácia jednoducho manipulovala priamo s dátami v databáze. Problémom je, podobne ako v prípade práce s konfiguračnými nástrojmi, obmedzenie na veľmi špeicificky nakonfigurované servery.

4.1.4 Špecifikácia požiadaviek

Na základe získaných vedomostí a vyvodených záverov je možné požiadavky na cieľovú aplikáciu zhrnúť nasledovne. Cieľom je vytvorenie novej Android aplikácie, ktorá bude spĺňať tieto kritéria:

- spustiteľná pre optimálny počet zariadení, ideálne bez potreby rozsiahlych zásahov do nastavení zariadenia (napr. root telefónu),
- dokáže pracovať s BIND 9 servermi, ideálne bez nutnosti inštalácie nových doplnkov na strane servera,
  - umožní používateľovi, ak má používateľ príslušné práva, upravovať a mazať zóny, ktoré server priamo spravuje (zóny typu master) a takéto zóny vytvárať - ideálne bez potreby špeciálnych konfiguračných nastavení,
  - umožní používateľovi, ak má používateľ príslušné práva, pridávať, upravovať a mazať DNS záznamy vrámci zón - ideálne bez potreby špeciálnych konfiguračných nastavení.

Požiadavky boli doplnené o kritérium v zmysle vyhnutia sa ďalším potrebným úkonom, ktoré by tvorili prerekvizity pre chod aplikácie. Tento bod tvorí základ pre jednoduchosť a intuitívnosť použitia aplikácie. V ideálnom prípade používateľovi stačí aplikáciu nainštalovať a poznať príslušné prístupové údaje.

4.2 Návrh a implementácia

Podkapitola 4.2 popisuje návrh a implementáciu tejto aplikácie na základe vytvorených požiadaviek. Postupne po častiach čitateľovi opisuje správanie a fungovanie jednotlivých časti programu doplnené o názorné ukážky.

4.2.1 Návrh na základe špecifkovaných požiadaviek

Nasledujúca sekcia pojednáva o návrhu aplikácie, ktorej požiadavky boli špecifikované v pasáži 4.1.4.

Prvou uvedenou požiadavkou je spustiteľnosť na optimálnom počte zariadení. To je dosiahnuteľné optimálnym zacielením na dostupné verzie API platformy Android, ktoré boli spolu s ich distribúciami uvedené v časti 3.1. Z tabuľky 3.1 je možné vydedukovať, že zacielením na minimálnu podporovanú verziu číslo 16, dôjde pokrytiu vyše 95% všetkých zariadení. Vylúčením starších verzií s malým zastúpením je možné eliminovať zbytočnú stratu podpory významných funkcionálnych prvkov, ktoré sa objavili s príchodom novších verzií a umožňuje využitie knižnice Support Library. Tým sa zaistí konzistentný používateľský zážitok aplikácie naprieč všetkými podporovanými verziami systému.

Druhou požiadavkou je správa BIND 9 DNS serverov. Konkrétne sa má jednať o prácu so zónami a DNS záznammi. Možné spôsoby dosiahnutia týchto úkonov sú popísané v časti 4.1.3. Pre účely cieľovej aplikácie je zvolený prístup priamej manipulácie so súbormi. Toto rozhodnutie vedie k obmedzeniu rozsahu funkcionality v prospech použiteľnosti aplikácie, kde je počet úkonov potrebných pre spojazdnenie aplikácie v spolupráci s DNS serverom najmenší. Túto prácu so súbormi na vzdialenom serveri je možné realizovať prostredníctvom protkolov SSH a SFTP. Jediným predpokladom sú adekvátne prístupové práva k súborom, ktorými chce používateľ manipulovať. Využitie tohoto prístupu v navrhovanej aplikácii by mohlo vyzerať nasledovne:

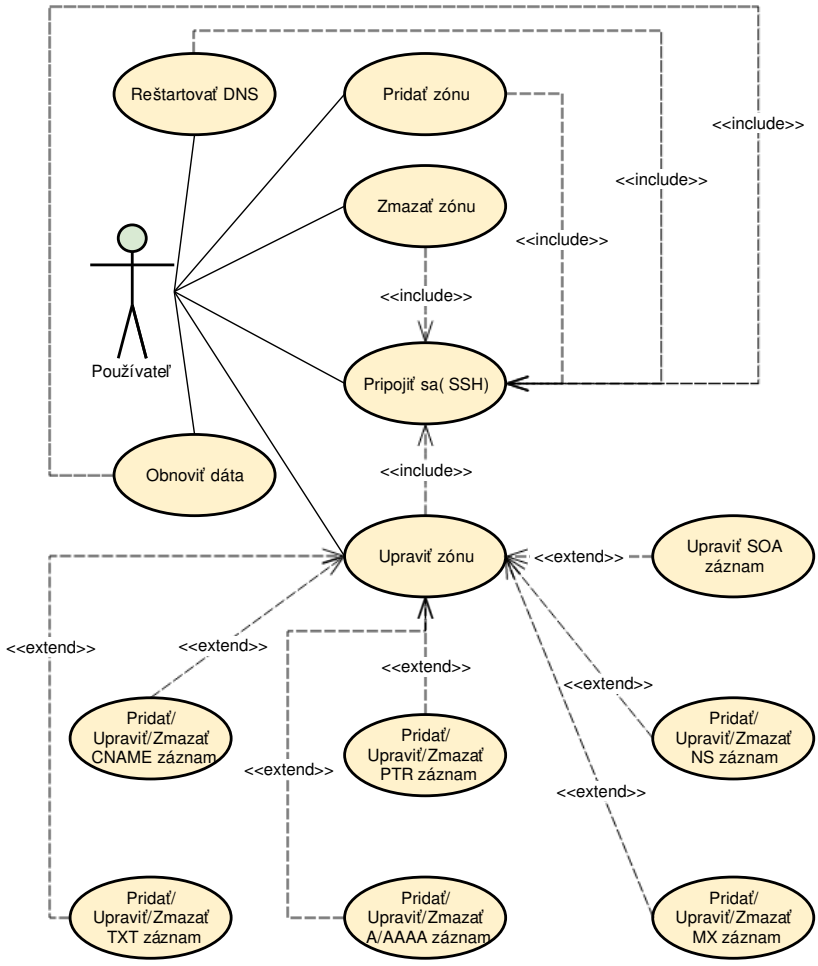
1. Používateľ sa prostredníctvom SSH pripojí na cieľový server.
2. Po pripojení je spracovaný zoznam všetkých serverom spravovaných zón, ktorý je používateľovi poskytnutý.
3. Ak si používateľ želá prezrieť detailné informácie o zóne, dôjde k stiahnutiu obsahu vybraného zónového súboru pomocou SFTP a jeho následnému spracovaniu.
4. Ak má prihlásený používateľ zapisovacie práva na konkrétne zónové súbory, môže prostredníctvom na to uspôsobených častí používateľského rozhrania s týmito súbormi manipulovať (skrz kombináciu SSH a SFTP). Po ukončení práce môže dať službe DNS bežiacej na serveri pokyn na reštart, za účelom načítania zmien.

Pri práci s DNS záznammi vrámci zón, je potrebné zobrať do úvahy, že v súčastnosti existuje veľké množstvo typov DNS záznamov. Obsah DNS záznamov je na tomto atribúte závyslý. Vzhľadom na špecifickosť týchto údajov je potrebné vytvoriť pre každý podporovaný typ samostatné grafické rozhranie s prispôsobenou obsluhou. Podpora celej množiny všetkých typov DNS záznamov je tým pádom problematická a viedla by k veľkému navýšeniu robustnosti celej aplikácie. Z toho dôvodu pokladám za vhodnejšie obmedziť podporu na základné, najčastešie sa vyskytujúce DNS záznamy, ktoré boli popísané v časti 2.3.

4.2.2 Diagram prípadu užitia

Používateľská interakcia s navrhovaným sytémom je vyjdrená diagramom prípadu užitia na obrázku 4.4, ktorý zhŕňa používateľove možnosti. Vystupuje tu jediný aktér v roli používateľa. Ten má po pripojení na server nasledujúce možnosti:

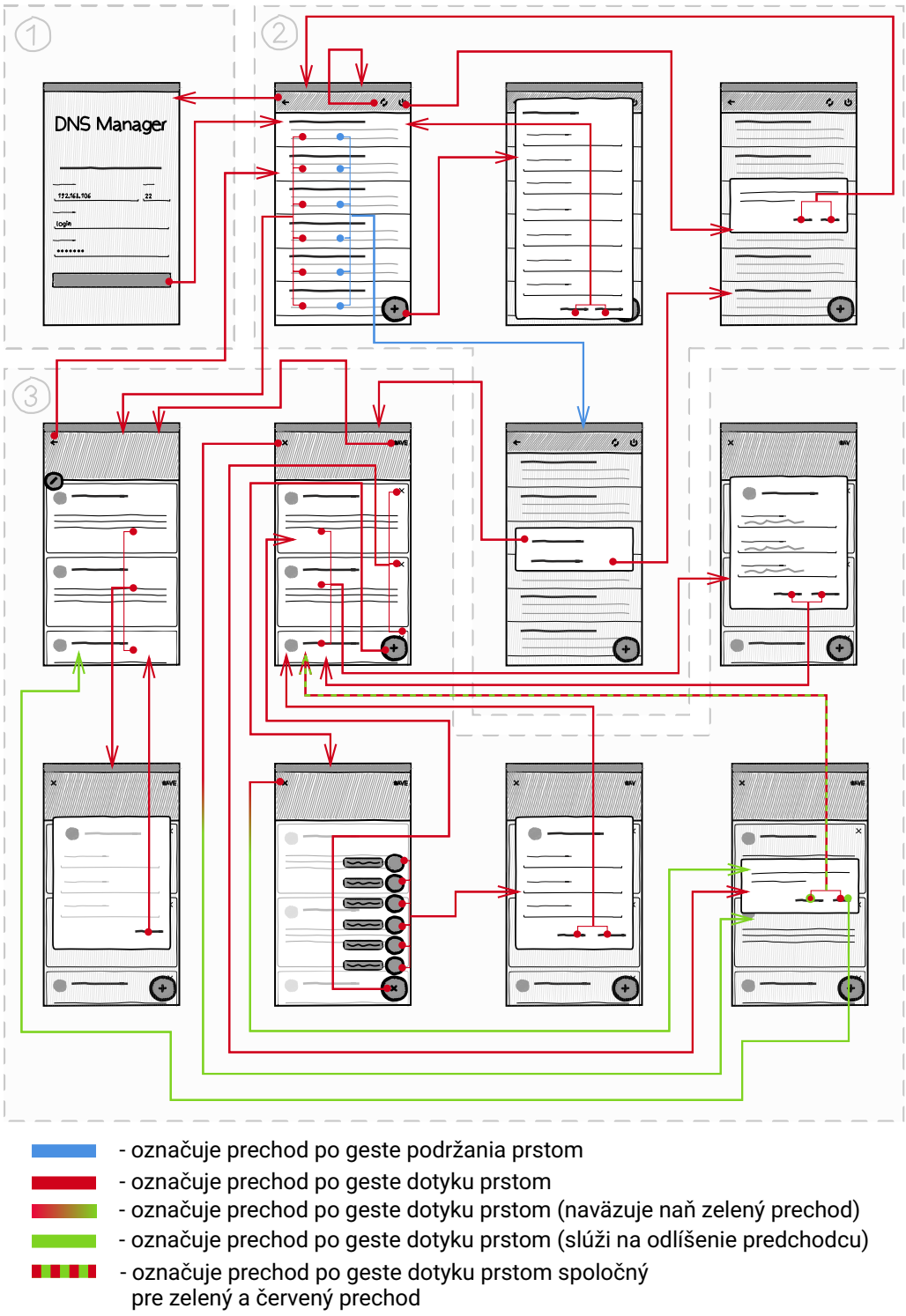
- vytvoriť novú zónu,
- vymazať existujúcu zónu,
- poslať serveru pokyn na reštartovanie bežiacej služby DNS, za účelom načítania vykonaných zmien,
- obnoviť dáta obdržané zo servera (tzv. refresh),
- upraviť zónu - vrámci tejto úpravy, môže naviac používateľ vytvoriť, upraviť alebo nadobro zmazať hociktorý z podporovaných typov DNS záznamov, s výnimkou záznamu typu SOA. Ten je pre svoj špecifický charakter možné iba upravovať.



Obr. 4.4: Diagram prípadu užitia navrhovanej aplikácie

4.2.3 Prototyp používateľského rozhrania

Návrh používateľského rozhrania z pohľadu navigácie a rozmiestenia prvkov je reprezentovaný prototypom tzv. "drôtených modelov"resp. wireframov na obrázku 4.5.



Obr. 4.5: Wireframy navrhovanej aplikácie

Aplikácia je rozdelená na 3 obrazovky.

1. **Úvodná obrazovka** – umožňuje používateľovi pripojenie na server. Obsahuje:
   - textové polia pre vstupné údaje,
   - potvrdzujúce tlačidlo.

   Po úspešnom pripojení presmeruje používateľa na obrazovku "Zoznam zón".

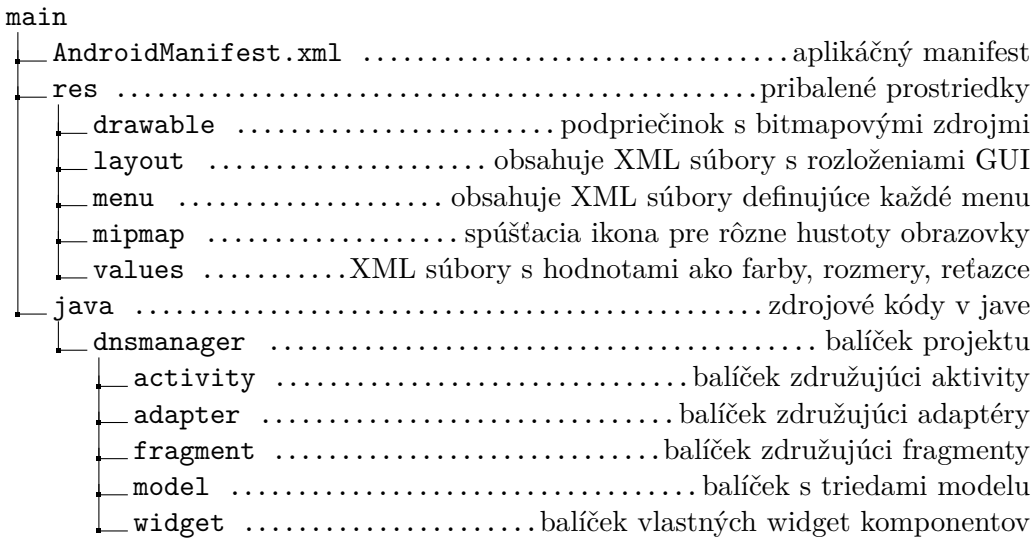
2. **Zoznam zón** – zobrazuje výpis zón vo forme interaktívneho zoznamu. Obsahuje:
   - aplikačný panel - sprostredkúva 3 tlačidlá:
     - navigačné tlačidlo pre návrat späť,
     - tlačidlo pre obnovu dát zo servera,
     - tlačidlo pre reštartovanie služby DNS na serveri, ktorého zvolenie vyvolá konfirmačné dialógové okno,
   - vznášajúce sa akčné tlačidlo - vyvolá dialógové okno pre vytvorenie novej zóny,
   - zoznam so zónami - každá položka zoznamu zobrazuje základné informácie o zóne. Zvolenie položky gestom dotyku prstom presmeruje používateľa na obrazovku "Detail zóny"v štandardnom režime. Podržaním prsta na položke dôjde vyvolaniu kontextového menu s možnosťami vymazania a editácie záznamu. Pri zvolení vymazania je vyvolané konfirmačné dialógové okno pre potvrdenie voľby. V prípade výberu možnosti editácie, dôjde k presmerovaniu na obrazovku "Detail zóny"v editačnom režime.

3. **Detail zóny** - slúži na správu záznamov DNS vrámci zóny. Môže sa nachádzať v dvoch režimoch: štandardnom alebo editačnom. V štadardnom režime obsahuje:
   - aplikačný panel - obsahuje meno zóny a navigačné tlačidlo pre návrat,
   - zoznam kariet - každá karta zobrazuje informácie o DNS zázname. Pri zvolení karty dotykom prsta sa zobrazí needitovateľné príslušné dialógové okno DNS záznamu,
   - horné vznášajúce sa akčné tlačidlo - prepne obrazovku do editačného režime.

   V editačnom režime obrazovka obsahuje:
   - aplikačný panel s dvoma tlačidlami:
     - uložiť zmeny - po uložení zmien prepne obrazovku do štandardného režimu,
     - zrušiť - vyvolá konfirmačné dialógové okno a v prípade potvrdenia prepne obrazovku do štandardného režimu,
   - zoznam kariet - každá karta zobrazuje informácie o DNS zázname. V editačnom režime má každá karta (mimo karty SOA záznamu) aj tlačidlo pre vymazanie. Jeho zvolením dôjde k vyvolaniu konfirmčného dialógového okna. Pri zvolení karty dotykom prsta sa zobrazí príslušné editovateľné dialógové okno DNS záznamu,
   - vznášajúce sa akčné menu - pri rozbalení zobrazí zoznam tlačidiel reprezentujúcich jednotlivé DNS záznamy, ktoré vyvolávajú príslušné dialógové oná pre vytvorenie záznamov.

4.2.4 Štruktúra programu

Štruktúra aplikácie zodpovedá štandardnému rozloženiu Android projektu. Rozdelenie programu je znázornené na obrázku 4.6.



Obr. 4.6: Adresárová štruktúra aplikácie

Z tohoto obrázku je možné odpozorovať, že štruktúru aplikácie tvoria 3 hlavné časti:

- **Súbor** `AndroidManifest.xml` – jedná sa o aplikačný manifest, ktorého úloha je popísaná v sekcii 3.3.
- **Priečinok** `res` – adresár obsahujúci prostriedky, napríklad ikony, rozloženia používateľských rozhraní apod., ktoré sú pribalené ku skompilovanému kódu aplikácie.
- **Priečinok** `java` – adresár uchovávajúci zdrojové kódy aplikácie v jazyku Java.

V adresári `res` sa nadhádzajú podpriečinky `drawable`, `layout`, `menu`, , `mipmap` a `values`. V `drawable` sú uložené obrázkove prostriedky použité v aplikácii. V skutočnosti neide o jediný podpriečinok, ale skupinu podadresárov so súbormi pre zariadenia s rôznymi hustotami obrazoviek. To isté platí aj o `mipmap`, kde je uložená spúšťacia produktová ikona v rôznych rozlíšeniach. V časti `values` sa nachádzajú XML súbory, ktoré obsahujú jednoduché hodnoty ako textové reťazce, čísla, farby apod. Sú to:

- `colors.xml` – definuje farebnú paletu aplikácie,
- `dimens.xml` – obsahuje zoznam rôznych rozmerov použitých v aplikácii ,
- `strings.xml` – rôzne textové reťazce použité v aplikácii,
- `styles.xml` – definuje vlastné štýly grafického rozhrania.

Definície rozložení používateľských rozhraní sa nachádzajú v XML súboroch umiestnených v podpriečinku `layout`. Organizujú komponenty pre všetky obrazovky, časti obrazoviek,

alebo dialógové okná. Podadresár `menu` združuje XML definície každého menu v aplikácii. Ide napr. o rôzne kontextové menu, či menu možností v aplikačnom panely.

Priečinok `java` obsahuje balíček projektu `dnsmanager`. Ide o jeho relatívne meno, pričom absolútny názov balička je `cz.vutbr.fit.stud.xgalaj02.dnsmanager`. Pri pomenovávaní sa využíva kombinácia reverzného zápisu domény autorskej spoločnosti (v tomto prípade autorov e-mail) a názvu aplikácie/projektu. Táto konvencia zaručuje unikátnosť mien balíčkov a rozlíšiteľnosť aplikácií na zariadení, či obchode Google Play. Zdrojové kódy aplikácie sú rozdelené do 5 častí: `activity` , `adapter`, `fragment`, `model` a `widget`.

Modul `activity` obsahuje všetky triedy použité v aplikácii, ktoré sú potomkami triedy `Activity`. Ako už bolo spomenuté v kapitole 3 venujúcej sa systému Android, tieto triedy predstavujú jednotlivé obrazovky aplikácie. Balíček `fragment` združuje triedy rozširujúce komponent `Fragment`, ktoré reprezentujú časti obrazoviek alebo dialógové okná. Modul `adapter` zahŕňa všetky adaptéry využité v aplikácii. Tie vytvárajú prepojenie medzi komponentami ako `ListView`, či `RecyclerView` a ich dátami. Balíček `widget` vznikol za účelom oddelenia vlastných programových widgetov ako napr. separátory. Modul `model` obsahuje triedy reprezentujúce modelovú časť aplikácie. Ide o funkcionalitu logiky, ktorá beží na pozadí. Zastrešuje napr. sieťovú komunikáciu, či prácu s jednotlivými záznammi.

Bližšie sa týmto modulom, ich obsahu a triedam, venuje nasledujúca časť 4.2.5.

4.2.5 Popis balíčkov a tried

Ako už bolo uvedené v podsekcii 4.2.4, program je rozdelený do 5 modulov.

- `activity` – obsahuje všetky aktivity aplikácie,
- `adapter` – združuje použité adaptéry,
- `fragment` – zoskupuje fragmenty v aplikácii,
- `model` – reprezentuje model aplikácie,
- `widget` – vlastné grafické komponenty používateľského rozhrania.

Zoznam týchto baličkov spolu s ich obsahom je zobrazený na obrázku 4.7. Modul `activity` obsahuje triedy, ktoré reprezentujú obrazovky navrhnuté v časti 4.2.3. Úvodnú obrazovku s pripojením na server implementuje trieda `ConnectActivity`. Zoznam zón je zastúpený aktivitou `ZoneListActivity` a obrazovka Detail zóny zas triedou `ZoneDetailActivity`.

Balíček `adapter` tvoria dve triedy:

- `ResourceRecordRecyclerViewAdapter`
- `ZoneRecyclerViewAdapter`

`ZoneRecyclerViewAdapter` je adaptérom k zoznamu zón na obrazovke Zoznam zón. Zabezpečuje načítavanie a vykresľovanie položiek zoznamu spolu s obsluhov používateľských vstupných udalostí nad nimi. Podobnú úlohu zastáva `ResourceRecordRecyclerViewAdapter`, ktorý má na starosti zoznam záznamov DNS na obrazovke Detail zóny.

Triedy modulu `fragment` je možné typologicky rozčleniť na niekoľko skupín. Prvou skupinou sú fragmenty reprezentujúce konfirmačné dialógové okná, ktoré slúžia na potvrdenie, resp. zrušenie, používateľom vyvolanej akcie. Ide o akcie, ktoré majú na systém veľký alebo nenávratný dopad (mazanie alebo reštart). Patria sem triedy:

- `DeleteRecordDialogFragment` – potvrdenie akcie vymazania DNS záznamu zo zóny .
- `DeleteZoneDialogFragment` – potvrdenie akcie vymazania zóny.
- `DiscardChangesDialogFragment` – potvrdenie ukončenia úprav bez uloženia zmie.
- `RestartBINDDialogFragment` – pokyn pre reštart DNS servera.

Ďalšou skupinou tried v balíčku fragmentov sú dialógové okná DNS záznamov. Tieto komponenty zobrazujú detailné informácie o jednotlivých záznamoch DNS špecifické pre každý typ. Taktiež slúžia ako rozhrania pre vytváranie nových záznamov a editáciu existujúcich. Základom tejto skupiny je trieda `RecordDialogFragment`, ktorú všetky ostatné triedy, špecifické pre jednotlivé typy záznamov, rozširujú. Tieto dialógové okná sa môžu nachádzať v troch stavoch[2] (vytváranie, zobrazovanie alebo editácia záznamu), ktoré mierne upravujú ich správanie a podobu. Triedy, ktoré dedia od `RecordDialogFragment` a sú súčasťou tejto skupiny sú:

- `RecordAialogFragment` – dialógové okno A/AAAA záznamu.
- `RecordCNAMEDialogFragment` – dialógové okno CNAME záznamu.
- `RecordMXDialogFragment` – dialógové okno MX záznamu.
- `RecordNSDialogFragment` – dialógové okno NS záznamu.
- `RecordPTRDialogFragment` – dialógové okno PTR záznamu.
- `RecordSOADialogFragment` – dialógové okno SOA záznamu.
- `RecordTXTDialogFragment` – dialógové okno TXT záznamu.

Samostatnou kategóriou je dialógové okno, ktoré tvorí používateľské rozhranie pre vytvorenie novej zóny. Implementuje ho trieda `NewZoneDialogFragment`, ktorá používateľovi umožňuje pridať na server novú zónu. Takáto zóna bezprostredne po vytvorení obsahuje po jednom zázname typu SOA, NS a A.

Fragmentom nepatriacim do žiadnej zo zmienených skupín je `ZoneDetailFragment`. Ten tvorí časť obrazovky Detail zóny, konkrétne obaľuje obsahovú časť so zoznamom DNS záznamov.

Modul `widget` pozostáva z jedinej triedy `DividerItemDecoration`. Táto trieda implementuje dekorátor slúžiaci na vizuálne oddelenie položiek, ktorý využíva obrazovka Zoznam zón.

Poslednou časťou je model aplikácie obsiahnutý v rovnomennom balíčku `model`. Je tvorený nasledujúcimi triedami:

- `AsyncTaskWithCallback`,
- `ConfigFile`,
- `ConnectionManager`,
- `Named`,

[2] S výnimkou dialógového okna SOA záznamu, ktoré implementuje iba zobrazovací a editačný režim

- `NamedZone`.

`AsyncTaskWithCallback` je abstraktná trieda rozširujúca triedu `AsyncTask`. Ide o formu asynchronného vlákna, ktorá sa využíva v Androide. Táto rozširujúca abstraktná trieda dopĺňa k štandardnej triede verejné rozhranie (public interface), ktorého zakomponovaním je možné vláknu predať callback, ktorý sa má uskutočniť po dokončení činnosti vlákna. Táto trieda je použitá hlavne pri sieťovej komunikácii so serverom, ktorá musí prebiehať v samostatnom procese mimo obsluhy používateľského rozhrania.

```
cz.vutbr.fit.stud.xgalaj02.dnsmanager
├── activity
│   ├── ConnectActivity.java
│   ├── ZoneListActivity.java
│   └── ZoneDetailActivity.java
├── adapter
│   ├── ResourceRecordRecyclerViewAdapter.java
│   └── ZoneRecyclerViewAdapter.java
├── fragment
│   ├── DeleteRecordDialogFragment.java
│   ├── DeleteZoneDialogFragment.java
│   ├── DiscardChangesDialogFragment.java
│   ├── NewZoneDialogFragment.java
│   ├── RecordADialogFragment.java
│   ├── RecordCNAMEDialogFragment.java
│   ├── RecordDialogFragment.java
│   ├── RecordMXDialogFragment.java
│   ├── RecordNSDialogFragment.java
│   ├── RecordPTRDialogFragment.java
│   ├── RecordSOADialogFragment.java
│   ├── RecordTXTDialogFragment.java
│   ├── RestartBINDDialogFragment.java
│   └── ZoneDetailFragment.java
├── model
│   ├── AsyncTaskWithCallback.java
│   ├── ConfigFile.java
│   ├── ConnectionManager.java
│   ├── Named.java
│   └── NamedZone.java
└── widget
    └── DividerItemDecoration.java
```

Obr. 4.7: Rozdelenie baličkov a tried aplikácie

Sieťovú komunikáciu má na starosti trieda `ConnectionManager` implementujúca návrhový vzor Jedináčik (Singleton). Počas celého behu aplikácie existuje jediná instacia tejto triedy, ktorá na seba odkazuje. Tým sa zabráni strate referencií a zániku objektu. Na komunikáciu využíva funkcie knižnice `com.jcraft.jsch`, ktorá implementuje protokol SSH2 v čistej Jave.

Prácu s DNS v aplikácii zastrešujú triedy `Named`, `ConfigFile` a `NamedZone`. `Named` reprezentuje nameserver, ktorý beží na vzdialenom počítači. Ten obsahuje informácie ako:

- absolútnu cestu služby na serveri,
- zoznam primárnych zón spadajúcich pod správu nameservera,
- zoznam využívaných konfiguračných súborov.

Okrem uskladnenia týchto informácii zastrešuje aj prácu nad týmito údajmi (ako pridanie-/odobranie zón apod.).

Objekty triedy `ConfigFile` reprezentujú konfiguračné súbory, ktoré BIND 9 nameserver využíva. V praxi sa jedná o súbor `named.conf` a ďalšie súbory, ktoré v ňom sú zahrnuté prostredníctvom direktívy `include`. Instancie tejto triedy v sebe ukladajú nasledujúce informácie:

- absolútnu cesta k súboru,
- prefix destinačnej absolútnej cesty, ktorý je použitý v prípade výskytu relatívnych ciest vrámci súboru,
- obsah konfiguračného súboru,
- zapisovacie prístupové práva používateľa k danému konfiguračnému súboru.

Poslednou triedou je `NamedZone`, ktorá reprezentuje zónu DNS servera. Táto trieda zastrešuje kompletnú manipuláciu vrámci zón a prácu s DNS záznammi. Na to program používa objekty a funkcie knižnice `dnsjava`[3], ktorá predstavuje implementáciu DNS v Jave. Okrem práce s dátami zón a DNS záznamov, poskytuje táto trieda informácie o umiestnení zónových súborov a ich prístupových právach.

Okrem spomenutých knižníc `JSch` a `dnsjava` používa aplikácia aj iné moduly prevzaté od tretích strán. Každá z knižníc je open source a vydaná buď pod licenciou Apache License v2 alebo The MIT License (MIT). Nasleduje zoznam týchto knižníc:

- JSch – `jsch-0.1.53`
- dnsjava – `org.xbill.dns_2.1.7`
- Butterknife – `com.jakewharton:butterknife:7.0.1`
- Material Ripple – `com.balysv:material-ripple:1.0.2`
- Material-ish Progress – `com.pnikosis:materialish-progress:1.7`
- Material Dialogs – `com.github.afollestad.material-dialogs:core:0.8.5.8`
- Material Contextual Action Bar – `com.afollestad:material-cab:0.1.11`

4.2.6 Implementáčné detaily hlavných častí aplikácie

Táto podsekcia sa venuje bližšiemu popisu fungovania aplikácie prezentovanej v predchádzajúcich častiach. Uvádza balíčky a programy predstavené v podsekciách 4.2.4 a 4.2.5 do celkového kontextu fungovania programu vo vzájomnej spolupráci podprogramov.

[3] názov balíčka knižnice `dnsjava` v aplikácii je `org.xbill.dns_2.1.7`

Pripojenie na server

Pri spustení aplikácie je používateľovi prezentovaná úvodná obrazovka, ktorá mu umožňuje pripojiť sa k vzdialenému serveru, na ktorom beží nameserver `bind9`. Tú implementuje program `ConnectActivity`, ktorej používateľské rozhranie bolo navrhnuté v časti 4.2.3. Pre pripojenie musí najprv používateľ zadať nasledujúce údaje:

- **adresu servera** - vo forme IP adresy alebo doménového mena,
- **port** - číslo portu, na ktorom sa má vytvoriť pripojenie,
- **login** - prihlasovacie meno používateľa,
- **heslo** - heslo používateľského účtu.

Po potvrdení údajov stlačením tlačidla pre pripojenie vykoná program kontrolu formátu vstupov. Pri neplatnom formáte dochádza k nastaveniu chybového príznaku pre nevalidné polia a pokus o pripojenie je vyhodnotený ako neúspešný. V prípade platných vstupov program pokračuje v úsilí o pripojenie na server. Postupuje nasledovne:

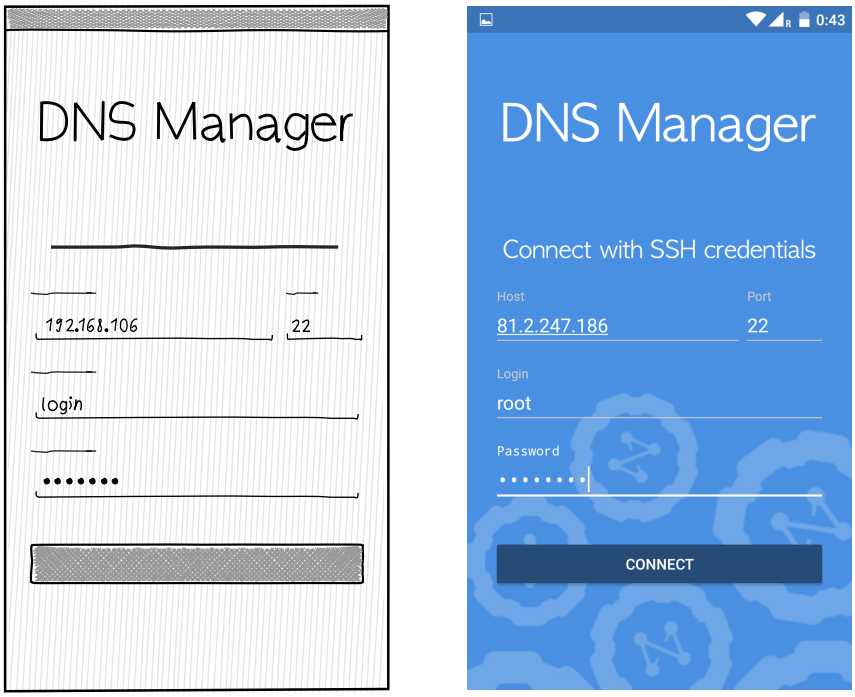
1. `ConnectionManager` zodpovedný za sieťovú komunikáciu inicializuje svoje hodnoty a pokúša sa o vytvorenie spojenia vrámci ktorého otvára kanál SSH.
2. V prípade úspešného spojenia program posiela požiadavok zisťujúci, či na danom serveri beží služba `bind9`.
3. Následne je vyhľadaný a spracovaný hlavný konfiguračný súbor `named.conf` s celým jeho obsahom. Uložia sa absolútne cesty k súborom a zapisovacie prístupové práva používateľa.
4. Je vytvorený zoznam serverom spravovaných primárnych zón s prístupovými právami k ich zónovým súborom.
5. Pokus o pripojenie je vyhodnotený ako úspešný. Zahajuje sa spustenie `ZoneListActivity`, ktorej sa predávajú spracované dáta.

V prípade nezdaru hociktorého z týchto bodov, je pokus o pripojenie ukončený a vyhodnotený ako neúspešný. O tom je následne používateľ patrične oboznámený prostredníctvom správy v dolnej časti obrazovky (tzv. toast). Tento algoritmus je predmetom diagramu v prílohe B. Používateľske rozhranie bolo implementované podľa navrhnutého prototypu 4.5. Jeho realizáciu je možné vidieť na obrázku 4.8.

Akcie obrazovky Zoznam zón

Po úspešnom pripojení na server je používateľovi prezentovaný zoznam spravovaných zón. Táto obrazovka implementovaná programom `ZoneListActivity`, je zobrazená na obrázku 4.9. Zoznam zón má vo svojej réžii 4 z pohľadu implementácie zaujímavé akcie.

- reštart nameservera,
- vytvorenie novej zóny,
- vymazanie existujúcej zóny,
- zobrzenie detailu zóny.



Obr. 4.8: Úvodná obrazovka

Reštart je implementovaný ako odoslanie príkazu na reštartovanie služby `bind9` prostredníctvom SSH kanálu v `ConnectionManager`.

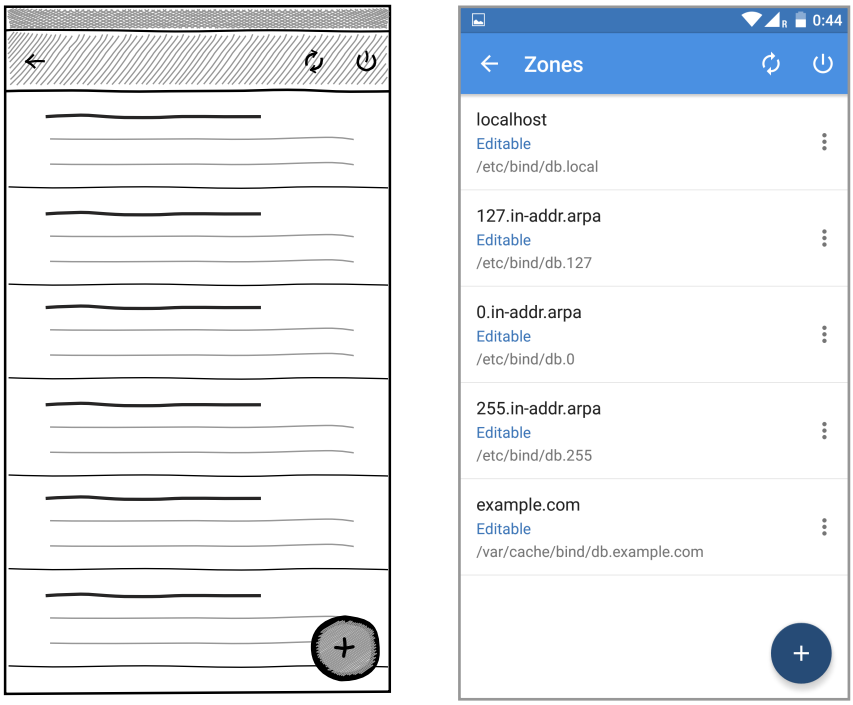
Vytváranie zón prebieha skrz dialógové okno dostupné z tejto obrazovky. Používateľ v ňom musí vyplniť všetky požadované údaje. Následne pri ich potvrdení prebehne validácia vstupov. Ak je formát všetkých vstupov v poriadku, program sa z nich pokúsi vytvoriť novú zónu. Využíva k tomu funkcie a objekty knižnice `dnsjava`, ktoré sú súčasťou triedy `NamedZone`. Postup je nasledovný.

1. z overených vstupov vytvor záznam typu SOA, NS a A,
2. vytvor z týchto záznamov novú zónu,
3. preveď zónový objekt do formy zónového súboru,
4. pokús sa uložiť zónový súbor na serveri,
5. nájdi v zozname konfiguračných súborov prvý, ku ktorému má používateľ zapisovacie práva,
6. zapíš do nájdeného konfiguračného súboru informácie o novej zóne,
7. obnov dáta zo servera.

V prípade zlyhania hociktorého z krokov, je používateľ informovaný o chybe.

Opačným prípadom akcie je **mazanie zón**, ktoré je tiež dostupné z tejto obrazovky. Postup pri vymazaní zóny je nasledovný:

1. odober zónu zo zoznamu zón v `Named`,



Obr. 4.9: Používateľské rozhranie obrazovky Zoznam zón

2. nájdi konfigurčný súbor, ktorý obsahuje informácie o zóne,
3. vymaž zónový súbor,
4. uprav nájdený konfiguračný súbor (vymaž informácie o zóne).

Samotný zoznam so zónami je implementovaný ako `RecyclerView`, za ktorého správanie zodpovedá objekt triedy `ZoneRecyclerViewAdapter`. Pri jednotlivých položkách môže používateľ vidieť nasledujúce informácie:

- názov zóny,
- cestu fyzického umiestnenia zónového súboru na serveri,
- informáciu o tom, či prihlásený používateľ môže upravovať konkrétny zónový súbor.

Pri **zobrazení detailu zónového záznamu** dôjde k stiahnutiu zónového súboru zo serveru. Tieto dáta sú následne spracované za pomoci knižnice `dnsjava` a uložené do príslušného objektu `NamedZone`. Ten je predaný štartujúcej obrazovke implementovanej triedou `ZoneDetailActivity`, na ktorú je používateľ presmerovaný.

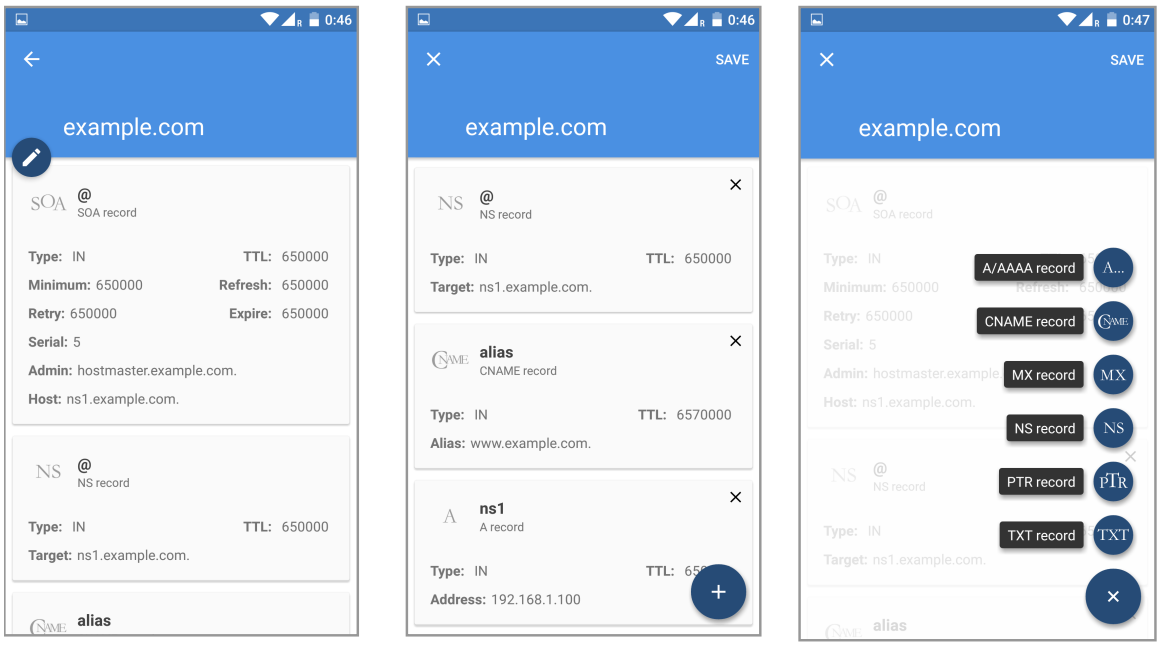
Práca so záznammi DNS

Grafické používateľské rozhranie pre prezeranie a manipuláciu záznamov DNS vrámci zóny reprezentuje obrazovka Detail zóny (viď. obr.). Implementuje ju `ZoneDetailActvity`, ktorá sa môže nachádzať v dvoch stavoch: štandardnom a editačnom. Prostredníctvom tejto obrazovky môže používateľ vytvárať, upravovať a mazať záznamy DNS.

Vytváranie záznamov prebieha prostredníctvom špecializovaných dialógov podobne ako vytváranie novej zóny. Po kontrole validity vstupov je vytvorený nový záznam, ktorý je

pridaný do zoznamu zdrojových záznamov zóny. V prípade **mazania záznamov** dochádza k vyhľadaniu záznamu v tomto zozname a jeho následnému vymazaniu. **Editácia záznamov** je implementovaná ako kombinácia predchádzajúcich dvoch akcií. Najprv sa vytvorí upravená kópia záznamu a následne dôjde k odstráneniu originálu.

Všetky tieto operácie prebiehajú len v rámci dát uložených na zariadení. Aby došlo premietnutiu zmien na serveri, je nutné úpravy dokončiť uložením. Vtedy dôjde k inkrementácii sériového čísla v zázname SOA a zónový súbor je nahradený novým, reflektujúcim vykonané zmeny. Nové dáta sa následne stiahnu zo servera do aplikácie.



Obr. 4.10: Používateľské rozhranie obrazovky Detail zóny v štandardnom a editačnom režime

4.2.7 Produktová ikona

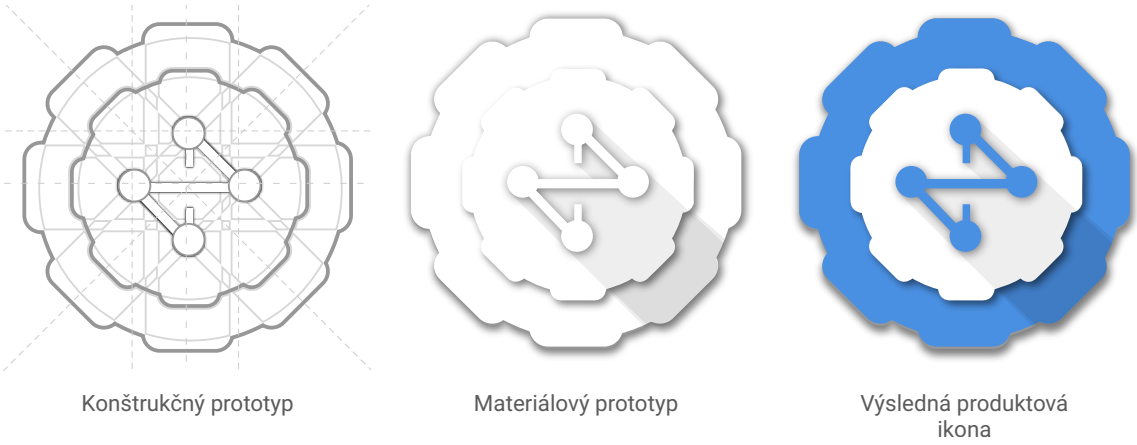
Produktová ikona (resp. logo) aplikácie je prostriedkom vizuálneho vyjadrenia identity produktu. Jej účelom je odkomunikovať používateľovi hlavnú myšlienku a účel aplikácie. V tomto konkrétnom prípade teda, že ide o nástroj na správu DNS serveru.

Myšlienku správy, administrácie, resp. nastavenia som sa rozhodol komunikovať prostredníctvom tvaru matice, ktorý sa za týmto účelom používa bežne. Používatelia sa s ním stretávajú často pri aplikáciách, ktoré niečo spravujú, či nastavujú. Z tohoto dôvodu som usúdil, že by tento symbol bol vhodný aj ako základný tvar pri "nastavení"DNS serveru. To, že ide práve o nastavenie sieťovej služby má podtrhnúť práve centrálny symbol, ktorý predstavuje prepojené uzly siete. Inšpiráciu pre použitie tohoto tvaru som čerpal od spoločnosti Google Inc., ktorá jeho obdobu pôvodne použila v časti svojho produktu Google Cloud Platform s názvom Cloud DNS[4].

Pri tvorbe loga k vytvorenej aplikácii som sa snažil nasledovať smernicu k produktovým ikonám materiálového dizajnu[16]. Prvým krokom bolo vytvorenie niekoľkých variánt zostáv konštrukčného prototypu za použitia malej palety jednoduchých univerzálnych prvkov

[4]https://support.google.com/cloud/answer/6256226?hl=en

(kružnice, obdĺžniky, priamky). Podľa navrhnutých konštrukcií bolo následne možné vyhotoviť materiálové modely, spĺňajúce kritériá materiálového prostredia popísané v časti 3.4. Nasledovala aplikácia farieb, ktoré boli vybrané podľa farebnej palety samotnej aplikácie. Výsledné logo bolo vybrané na základe používateľského prieskumu, kde mali respondenti určiť, ktorá varianta sa im páči najviac. Vybranú variantu loga spolu s medziproduktami procesu jeho tvorby je možné vidieť na obrázku 4.11.



Obr. 4.11: Medziprodukty a výsledok tvorby produktovej ikony vytvorenej aplikácie

4.3 Testovanie

Testovanie vytvorenej aplikácie prebiehalo na dvoch úrovniach:

- testovanie funkcionality,
- testovanie použiteľnosti.

Všetko testovanie bolo uskutočnené na 3 rôznych mobilných zariadeniach:

- telefón OnePlus Two – s verziou Android systému 5.1.1,
- tablet Google Nexus 7 – s verziou Android systému 6.0.1,
- telefón Xiaomi Mi3 – s verziou Android systému 4.4.4.

Testované boli reálne siete s 3 rôznymi servermi s bežiacou službou `bind9`:

- server v lokálnej sieti s operačným systémom Linux Ubuntu 14.04 LTS (Trusty Tahr),
- server v lokálnej sieti s operačným systémom Raspbian Jessie ,
- VPS s operačným systémom Ubuntu Server 14.04 LTS 64bit.

4.3.1 Testovanie funkcionality

Toto testovanie prebiehalo formou manuálnch testov s rôznymi zariadeniami a na rôznych serveroch. Išlo o vykonávanie základných úkonov aplikácie (ako manipulácia so zónami a záznammi) v rôznych podmienkach.

Jeho cieľom bolo odhalenie možných chýb a nečakaného správania. Zamerané bolo taktiež na pozorovanie odchýlok používateľského rozhrania v rámci rôznych verzií operačných systémov mobilných zariadení a rôznych hustôt obrazoviek.

4.3.2 Testovanie použiteľnosti

Základ tohoto testovania tvoril dopredu stanovený testovací scénar, ktorým sa každý testujúci používateľ riadil. Jeho interakcia s aplikáciou bola pozorovaná a zaznamenaná pre ďalšiu analýzu. Výber respondentov bol zameraný na osoby aspon zbežne familiárnymi s fungovaním systému DNS, ktoré mali predchádzajúce skúsenosti s používaním mobilnej platformy Android. Pre tento test bol vytvorený nasledujúci scenár:

1. Pripojte sa pomocou aplikácie na vzdialený server pomocou poskytnutých prihlasovacích údajov
2. Zoznámte sa s prostredím aplikácie ľubovoľnými úkonmi.
3. Vytvorte na serveri novú zónu.
4. Do vytvorenej zóny pridajte 2 nové záznamy ľubovoľného typu a zmeny uložte.
5. Jeden zo záznamov, vytvorených v prechádzajúcom kroku, vymažte.
6. Upravte TTL SOA záznamu v upravovanej zóne.
7. Ukončite úpravy bez uloženia zmien.
8. Zónu vytvorenú v kroku 3 vymažte.

4.3.3 Výsledky testovania

Používateľské testovanie ukázalo, že aplikácia je z hľadiska navigácie rozvrhnutá dobre. Každý subjekt sa dokázal v aplikácii bez problémov orientovať. Ako istou prekážkou pri plnení úloh, sa však ukázala nižšia miera intuitívnosti formátov vstupných údajov pre nové záznamy (hlavne pri vytváraní novej zóny). Jedná sa síce o údaje, ktoré sú podmienené istou formou znalosti problematiky, ale fakt, že sa na prvý pokus nepodarilo vytvoriť zónu ani jednému účastníkovi testu, nie je lichotivý. Tento problém by mohli vyriešiť implicitné nápovedy. Tieto výsledky sa odzrkadlia pri návrhu budúcich možných rozšírení.

Kapitola 5

Záver

Cieľom tejto práce bolo vytvorenie mobilnej aplikácie pre správu DNS, ktorá by jednoducho a prehľadne umožňovala prácu so zónami a DNS záznammi. Po analýze možných riešení bola na realizáciu tohoto programu zvolená metóda priamej manipulácie zónových súborov cez SSH pripojenie.

Za účelom čo najlepšieho používateľského zážitku a dosiahnutia čo najlepšej použiteľnosti, bola pri tvorbe používateľského rozhrania dôsledne nasledovaná dizajnová smernica vizuálneho jazyka Material Design.

Aplikácia sa ukázala byť prehľadná a jednoduchá na použitie, čím bol hlavný cieľ splnený, aj keď používateľské testovanie odhalilo priestor na zlepšenie.

Do budúcna by sa aplikácia mohla rozšíriť o ďalšie funkcie. V prvom rade by sa mali doladiť nedostatky odhalené testovaním. Rozšírenie podpory viacerých typov DNS záznamov. Ďalším možným rozšírením je podpora viacerých DNS serverov naraz. Používateľ by si vrámci svojho profilu uložil viacero pripojení na rôzne nameservery, ktoré by sa uložili do databázi a mohol by medzi nimi jednoducho prepínať. V budúcnosti by sa aplikácia mohla taktiež rozšíriť o možnosť pokročilých konfiguračných nastavení DNS ako povolenie dynamických úprav, dnssec, chroot apod. V tomto prípade bude treba prísť s efektívnym spôsobom práce nad konfiguračnými súbormi.

Literatúra

[1] Aitchison, R.: *Pro DNS and BIND 10*. Books for professionals by professionals, Apress, 2011, ISBN 9781430230489.

[2] Consortium, I. S.: *BIND 9 Administrator Reference Manual*. Internet Systems Consortium, mar 2016.

[3] Matoušek, P.: *Síťové aplikace a jejich architektura*. VUTIUM, 2014, ISBN 978-80-214-3766-1.

[4] Mens, J.: *Alternative DNS Servers: Choice and Deployment, and Optional SQL/LDAP Back-Ends*. Uit Cambridge Limited, 2008, ISBN 9780954452995. URL https://books.google.cz/books?id=UAlTNgAACAAJ

[5] Mockapetris, P.: Domain names - concepts and facilities. STD 13, RFC Editor, November 1987, http://www.rfc-editor.org/rfc/rfc1034.txt. URL http://www.rfc-editor.org/rfc/rfc1034.txt

[6] Mockapetris, P.: Domain names - implementation and specification. STD 13, RFC Editor, November 1987, http://www.rfc-editor.org/rfc/rfc1035.txt. URL http://www.rfc-editor.org/rfc/rfc1035.txt

[7] Dashboards | Android Developers [online]. `http://developer.android.com/about/dashboards/index.html`, 2016-05-02 [cit. 2016-05-06].

[8] Elevation and shadows - What is material? - Google design guidelines [online]. `http://www.google.com/design/spec/what-is-material/elevation-shadows.html`, 2016-05-02 [cit. 2016-05-06].

[9] Environment - What is material? - Google design guidelines [online]. `http://www.google.com/design/spec/what-is-material/environment.htm`, 2016-05-02 [cit. 2016-05-06].

[10] Introduction - Material design - Google design guidelines [online]. `http://www.google.com/design/spec/material-design/introduction.html`, 2016-05-02 [cit. 2016-05-06].

[11] Activity | Android Developers [online]. `http://developer.android.com/reference/android/app/Activity.html`, 2016 [cit. 2016-05-06].

[12] API Guides | Android Developers [online]. `http://developer.android.com/guide/index.html`, 2016 [cit. 2016-05-06].

[13] Application Fundamentals | Android Developers [online]. http://developer.android.com/guide/components/fundamentals.html, 2016 [cit. 2016-05-06].

[14] ContentUris | Android Developers [online]. http://developer.android.com/reference/android/content/ContentUris.html, 2016 [cit. 2016-05-06].

[15] Google Design [online]. https://design.google.com/, 2016 [cit. 2016-05-06].

[16] Icons - Style - Google design guidelines [online]. https://www.google.com/design/spec/style/icons.html, 2016 [cit. 2016-05-06].

[17] Reference | Android Developers [online]. http://developer.android.com/reference/packages.html, 2016 [cit. 2016-05-06].

[18] Services | Android Developers [online]. http://developer.android.com/guide/components/services.html, 2016 [cit. 2016-05-06].

[19] Domain Server Software Distribution [online]. https://ftp.isc.org/www/survey/reports/2016/01/fpdns.txt, 2016 [cit. 2016-05-06].

Prílohy

Zoznam príloh

Príloha A

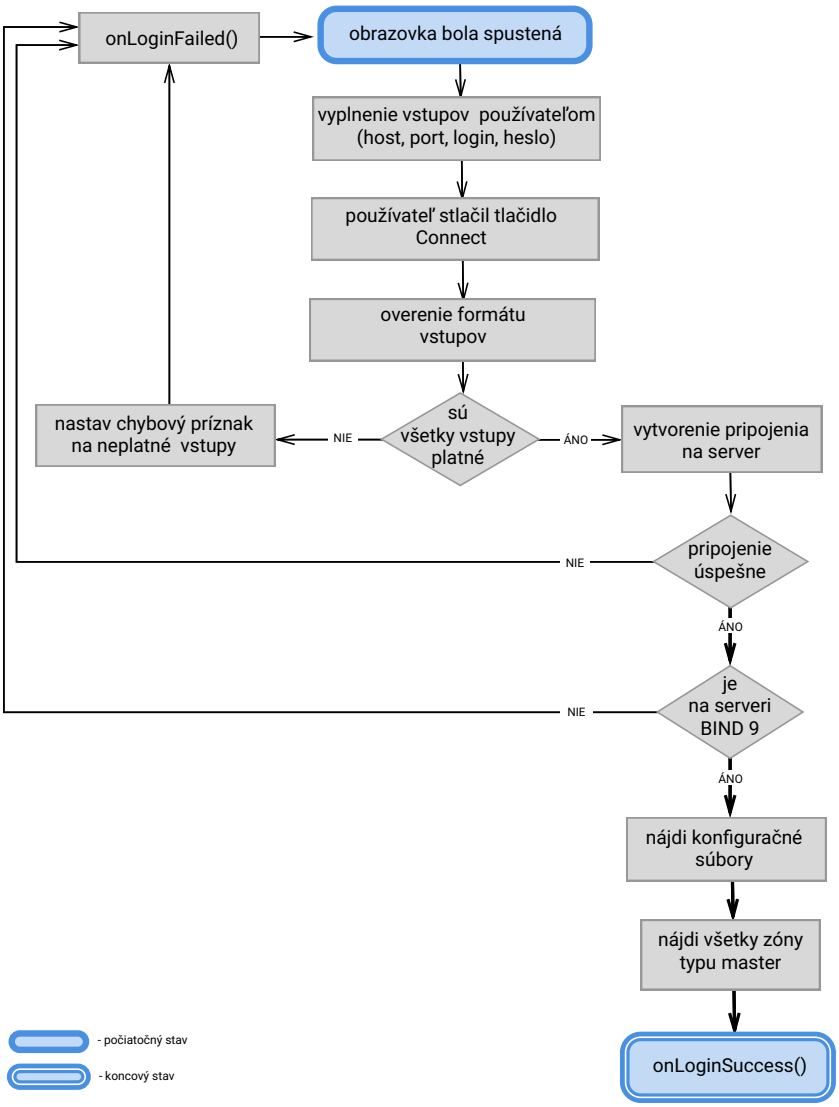
Obsah CD

Priložené CD obsahuje nasledujúcu štruktúru súborov.

- `dnsmanager` – zdrojové kódy aplikácie
- `manual.pdf` – návod na použitie aplikácie
- `tex` – zdrojové kódy textovej časti práce
- `pdf` – elektronická verzia práce
- `apk` – priečinok s inštalačným súborom aplikácie

Príloha B

Diagram algoritmu pripojenia aplikácie na server



Obr. B.1: Diagram pripojenia na server