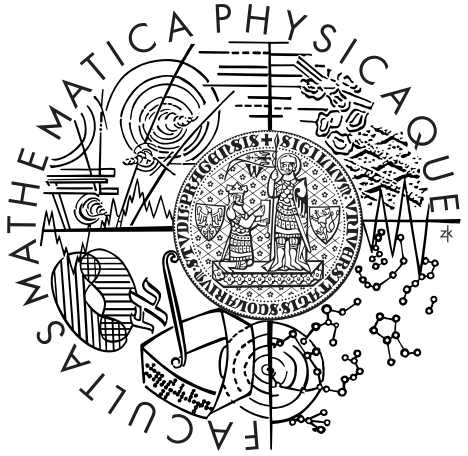
Univerzita Karlova v Praze
Matematicko-fyzikální fakulta

BAKALÁŘSKÁ PRÁCE



Michal Hanzeli

KarlFTP: FTP klient pre mobilné zariadenia so systémom Android

Katedra distribuovaných a spolehlivých systémů

Vedoucí bakalářské práce:	RNDr. Jan Kofroň, Ph.D.
Studijní program:	Informatika
Studijní obor:	správa počítačových systémů

Praha 2014

Ďakujem RNDr. Janu Kofroňovi za odborné vedenie mojej práce, podnetné rady a čas, ktorý mi počas vypracovávania tejto práce venoval. Ďalej by som chcel poďakovať svojim rodičom a Michalovi Bilanskému za podporu v mojom doterajšom štúdiu.

Prohlašuji, že jsem tuto bakalářskou práci vypracoval(a) samostatně a výhradně s použitím citovaných pramenů, literatury a dalších odborných zdrojů.

Beru na vědomí, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorského zákona v platném znění, zejména skutečnost, že Univerzita Karlova v Praze má právo na uzavření licenční smlouvy o užití této práce jako školního díla podle §60 odst. 1 autorského zákona.

V ..... dne ..... Podpis autora

Název práce: KarlFTP: FTP klient pre mobilné zariadenia so systémom Android

Autor: Michal Hanzeli

e-mail autora: hanzeli.michal@gmail.com

Katedra: Katedra distribuovaných a spolehlivých systémů

Vedoucí bakalářské práce: RNDr. Jan Kofroň, Ph.D., Katedra distribuovaných a spolehlivých systémů

Abstrakt: Cílem práce bylo vytvořit program FTP klienta KarlFTP pro mobilní zařízení se systémem Android. KarlFTP doplní existující aplikace o nový přístup k užívaní aplikací tohoto typu, využitím pracovních vláken na pozadí. Poskytne moderní užívatelské prostředí a doplnkové funkce pro snadnější ovládání. Součástí práce je stručný popis funkcí, návrhu a implementace programu. Také uvádíme přehled několika dostupných aplikací a problémy, na které jsme při jejich testování narazili. Program byl vytvořen pro systém Android v jazyku Java.

Klíčová slova: FTP klient, Android, Java

Title: KarlFTP: FTP client for Android devices

Author: Michal Hanzeli

author's e-mail: hanzeli.michal@gmail.com

Department: Department of Distributed and Dependable Systems

Supervisor: RNDr. Jan Kofroň, Ph.D., Department of Distributed and Dependable Systems

Abstract: The goal of this thesis was to create a FTP client application named KarlFTP for mobile devices with Android OS. With KarlFTP we implemented a new approach in which users interact with this type of application by using background threads. Our application offers modern user interface and additional functions which enhance the user experience. The thesis contains brief description of functionality, design and implementation of the program. It also introduces some of the available applications and problems we have encountered while using them. The program was created for Android OS in Java programming language.

Keywords: FTP client, Android, Java

Názov práce: KarlFTP: FTP klient pre mobilné zariadenia so systémom Android

Autor: Michal Hanzeli

e-mail autora: hanzeli.michal@gmail.com

Katedra: Katedra distribuovaných a spoľahlivých systémov

Vedúci bakalárskej práce: RNDr. Jan Kofroň, Ph.D., Katedra distribuovaných a spoľahlivých systémov

Abstrakt: Cieľom práce bolo vytvoriť program FTP klienta KarlFTP pre mobilné zariadenia so sytémom Android. KarlFTP doplní existujúce aplikácie o nový prístup k užívaniu aplikácií tohto typu, využitím pracovných vlákien na pozadí. Poskytne moderné užívateľské prostredie a doplnkové funkce pre ľahšie ovládanie. Súčasťou práce je stručný popis funkcií, návrhu a implementácie programu. Taktiež uvádzame prehľad niekoľkých dostupných aplikácií a problémy, na ktoré sme pri ich testovaní narazili. Program bol vytvorený pre systém Android v jazyku Java.

Kľúčové slová: FTP klient, Android, Java

Obsah

1. Úvod

Prístup k súborom a ich správa je neoddeliteľnou a dôležitou súčasťou práce s nimi. Vznik internetu podstatne zlepšil ich dostupnosť, čím umožnil pracovať s nimi nie len lokálne, ale aj vzdialene. Súbory sa tak môžu nachádzať na jednom mieste a užívateľ má k nim kedykoľvek prístup. V súčasnosti existuje niekoľko služieb, ktoré podporujú zdieľanie súborov. Najpoužívanejšími sú cloudové úložiská a súborové servery využívajúce FTP ako napríklad NAS. V tejto práci sa zameriame práve na použitie FTP.

Cieľom tejto práce je navrhnúť a vytvoriť aplikáciu KarlFTP, ktorá bude fungovať ako ftp klient pre mobilné zariadenia so systémom Android. Dôvodom vzniku tejto práce bolo, že súčasne dostupné aplikácie majú obmedzenú funkcionalitu alebo problémy s fungovaním. Taktiež ich vizuálna stránka nie je príliš atraktívna a nevyužívajú moderné grafické doplnky, ktoré by sa napríklad prispôsobovali rôznym veľkostiam obrazoviek. V našej aplikácii sa pokúsime vyplniť medzery vo funkcionalite medzi dostupnými klientmi aj podporou výberu súborov podľa regulárnych výrazov, synchronizovaného prehliadania a synchronizáciou adresárov. KarlFTP zároveň ponúkne vylepšené používateľské prostredie so špeciálnymi vlastnosťami pre efektívnejšie a intuitívnejšie použitie na rozmerovo odlišných typoch Androidových zariadení ako sú smartphone a tablet.

V prvej kapitole popíšeme softvérovú platformu Android, princíp fungovania a inštalácie aplikácií, FTP protokol a vývojové prostredie Android Studio. Ďalej špecifikujeme našu aplikáciu a popíšeme jej vlastnosti. Následne spravíme analýzu problémov, ktoré pri návrhu vznikli a postup ich riešenia. Ku koncu popíšeme implementáciu aplikácie a v poslednej kapitole sa budeme venovať rozboru existujúcich ftp klientov pre platformu Android. V závere zhodnotíme celkovú prácu a či sa nám podarilo splniť jej ciele.

2. Android a FTP

V tejto kapitole si popíšeme softvérovú platformu Android a jej hlavné vlastnosti, podľa [1]. Zameriame sa tiež na vysvetlenie princípu fungovania a možnosti inštalácie aplikácií v prostredí tohto systému. Posledné dve časti tejto kapitoly budú venované popisu FTP a vývojového prostredia Android Studio.

2.1 Operačný systém Android

Andorid je operačný systém pre mobilné zariadenia, ktorý bol vytvorený spoločnosťou Google a zoskupením Open Handset Alliance. Prvý krát bol tento open source systém predstavený v roku 2007 a od vtedy sú stále vyvýjané jeho nové verzie. Najaktuálnejšia je verzia 4.4 s označením KitKat. Systém Android sa vyskytuje v miliónoch mobilných telefónov a tabletov, vďaka čomu je považovaný za najpoužívanejšiu platformu pre vývojárov aplikácií. Niektoré jeho vlastnosti sa objavili už skôr, no Android je prvé prostredie, ktoré kombinuje:

- Otvorenú a bezplatnú vývojovú platformu založenú na systéme Linux, ktorá je výhodná pre výrobcov zariadení aj vývojárov.
- Architektúru založenú na komponentovom princípe. Tá umožňuje zdieľanie a znovupoužitie niektorých častí aplikácií v rámci iných aplikácií.
- Veľké množstvo zabudovaných služieb ako sú napríklad lokalizácia pomocou GPS, SQL databáza, webové a mapové prehliadače, ktoré sa dajú jednoducho zakomponovať priamo do aplikácie.
- Automatickú správu životného cyklu aplikácií. Android je optimalizovaný pre fungovanie na zariadeniach s menšími nárokmi na batériu a pamäť. Užívateľ teda nemusí dávať pozor na to, či sú spustené aplikácie aktívne alebo nie a je dostatok prostriedkov na ich beh.
- Prenositeľnosť medzi širokou škálou súčasného a budúceho hardvérového vybavenia. Aplikácie pre tento systém sú písané v jazyku Java a vykonávané cez *dalvik* virtuálny stroj. Ten podporuje procesorové architektúry ako ARM, x86 a iné a zároveň vstupné zariadenia ako klávesnice, dotykové obrazovky, pohybové senzory a iné. Taktiež je možné užívateľské prostredie prispôsobovať rozlíšeniam a orientácii obrazoviek.

Uvedieme si vysvetlenie základných pojmov v prostredí Android.
aktivita - základný komponent aplikácie, ktorý poskytuje obrazovku s užívateľským rozhraním, v ktorej užívateľ interaguje s aplikáciou
fragment - komponent aplikácie, ktorý definuje správanie časti užívateľského rozhrania v aplikácii
služba - úloha aplikácie, ktorá je vykonávaná na pozadí bez priamej interakcie s užívateľom. Má dve formy:

- spustená - služba spustená nejakým komponentom aplikácie, pričom táto služba môže na pozadí bežať donekonečna a komponent nad ňou nemá kontrolu

- naviazaná - podobná ako spustená služba ale s tým rozdielom, že je naviazaná na komponent, ktorý ju spustil. Komponent má teda kontrolu nad službou a môže ju prípadne zastaviť.

2.2 Princíp fungovania a inštalácie aplikácie

Pri spustení aplikácie alebo niektorej z jej komponentov Android vytvorí aplikačný objekt. Ten je vytvorený v novom procese s unikátnym ID pod unikátnym užívateľom. Tento objekt sa spustí pred inicializáciou akéhokoľvek komponentu aplikácie a beží, kým sa všetky komponenty neukončia. V prípade, že Android potrebuje ukončiť niektorý z bežiacich procesov, použije nasledujúci prioritný systém. Aplikácia má stav:

- **V popredí** (priorita 1) - aplikácia, s ktorou užívateľ interaguje, alebo má spustenú službu, ktorá je zviazaná s touto aplikáciou
- **Viditeľná** (priorita 2) - aplikácia, s ktorou užívateľ už neinteraguje, ale je stále (čiastočne) viditeľná, alebo má spustenú službu, ktorá je používaná inou neaktívnou, no viditeľnou aktivitou
- **Služba** (priorita 3) - aplikácia so spustenou službou, s ktorou užívateľ neinteraguje, ani nie je viditeľná
- **V pozadí** (priorita 4) - aplikácia so zastavenými aktivitami a bez spustených služieb.
- **Prázdna** (priorita 5) - aplikácia bez aktívnych komponentov

Aktivita, ako hlavný funkčný komponent aplikácie, má svoj beh charakterizovaný ako životný cyklus. Tento cyklus pozostáva z troch hlavných stavov, v ktorých sa môže aktivita nachádzať. Sú to:

- *spustená* - aktivita je v popredí obrazovky a užívateľ s ňou interaguje.
- *pozastavená* - aktivita je čiastočne zakrytá inou aktivitou, ktorá je v popredí. Aktivita je kompletne živá a ako objekt je udržiavaná v pamäti. V prípade, že má systém extrémne nízku voľnú pamäť môže byť zabitá.
- *zastavená* - aktivita je kompletne prekrytá inou aktivitou, ktorá je v popredí. Táto aktivita je taktiež kompletne živá a udržiavaná v pamäti. Ak však systém potrebuje uvoľnenie pamäte, túto aktivitu zabije.

Zmena stavu aktivity je riadená pomocou jednotlivých metód životného cyklu. Tieto metódy sa v rámci implementácie aktivity môžu prepísať, aby aktivita správne reagovala na zmenu stavu. Na nasledujúcom príklade 2.1 si ukážeme kostru aktivity, ktorá obsahuje základné metódy životného cyklu.

```
public class Example extends Activity {

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        // Aktivita sa ako objekt nainicializuje a
        // vytvorí si všekty objekty, ktoré bude
        // potrebovať pre svoju existenciu
        // (napr. definuje si GUI komponenty)
    }
    @Override
    protected void onStart() {
        super.onStart();
        // Aktivita sa pripravuje na stav spustená a vytvorí
        // si GUI, ktoré sa má zobraziť
    }
    @Override
    protected void onResume() {
        super.onResume();
        // Aktivita sa stáva viditeľnou (zobrazí sa GUI) a
        // prechádza do stavu spustená
    }
    @Override
    protected void onPause() {
        super.onPause();
        // Aktivita prechádza do stavu pozastavená a iná
        // aktivita sa dostáva do popredia
    }
    @Override
    protected void onStop() {
        super.onStop();
        // Aktivita prechádza do stavu zastavená a je
        // kompletne prekrytá inou aplikáciou
    }
    @Override
    protected void onDestroy() {
        super.onDestroy();
        // Aktivita sa pripravuje na to, že bude zničená a
        // vykonáva všetky potrebné finalizačné kroky
    }
}
```

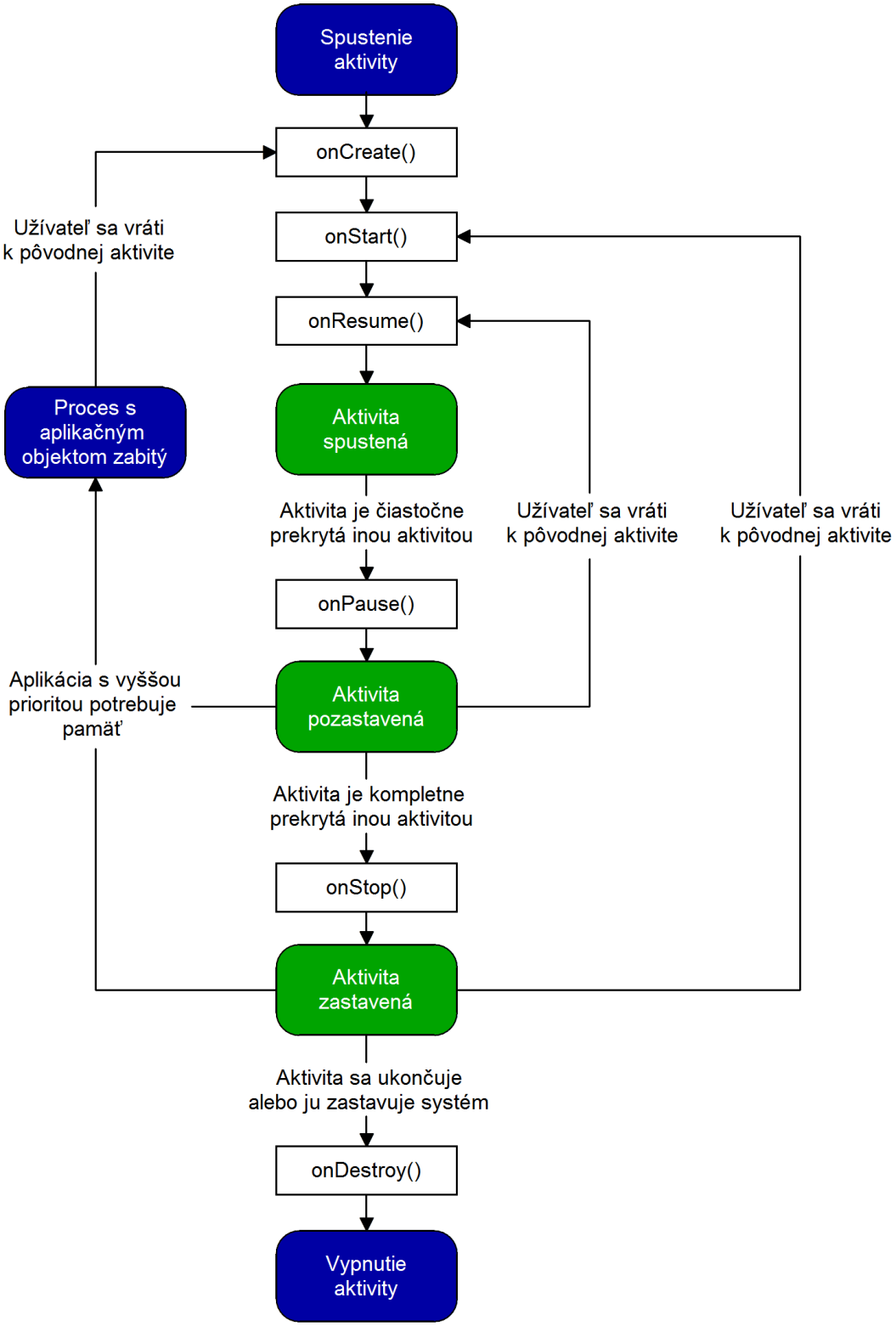
Príklad 2.1: Kostra aktivity

Tieto metódy spoločne definujú a zároveň kontrolujú celý životný cyklus aktivity (obr. 2.1), ktorý pozostáva z troch sľučiek.

- **Celkový čas života** aktivity sa odohráva medzi volaním metód *onCreate()* a *onDestroy()*. V tejto sľučke sa udržiavajú všetky objekty, ktoré definujú

„globálnu“ štruktúru aktivity (napr. definovanie grafických komponent v *onCreate()* a uvoľnenie zdrojov v *onDestroy()*)

- **Viditeľný čas života** aktivity sa odohráva medzi volaním *onStart()* a *onStop()*. Počas tohto času je aktivita (čiastočne) viditeľná a užívateľ s ňou môže interagovať. Táto sľučka je počas životného cyklu volaná niekoľkokrát, keďže aktivita alternuje medzi tým, ako je viditeľná a skrytá pre užívateľa.
- **Čas života v popredí** aktivity sa odohráva medzi volaním *onResume()* a *onPause()*. Aktivita je zobrazená na obrazovke v popredí pred všetkými inými aktivitami a príjma vstup od užívateľa.



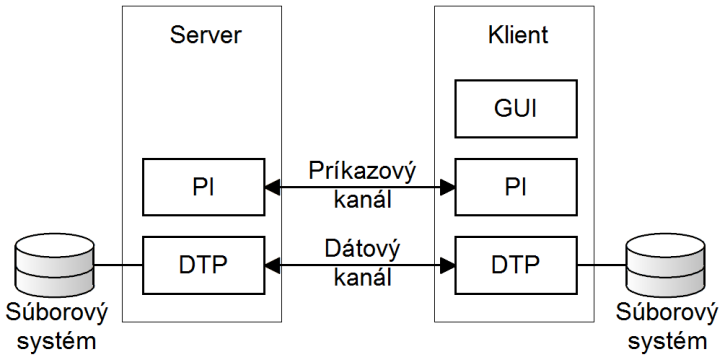
Obr. 2.1: Diagram zobrazujúci životný cyklus aktivity a prechody medzi jej stavmi

Základom pre distribúciu a inštaláciu aplikácie je inštalačný balíček vo formáte APK. Pre vytvorenie tohto balíčka je potrebné, aby sa program najskôr skompiloval a následne sú všetky jeho časti zabalené do jedného súboru s príponou .apk. Súbor okrem kódu obsahuje všetky potrebné dáta pre beh aplikácie ako sú obrázky, zvuky, databázy, certifikáty a manifest súbor. Manifest súbor popisuje meno, verziu, prístupové práva a odkazy na dodatočné knižnice aplikácie. Počas inštalovania sa aplikácii nastavia jednotlivé prístupové práva k dátam v zariadení a jeho periférиám. Inštalácia sa dá previesť tromi spôsobmi:

- Manuálna inštalácia aplikácie v zariadení. Prostredníctvom súborového manažéra sa v zariadení vyhľadá balík apk a pomocou Android inštalatéra sa rozbalí a pripraví na použitie.
- Inštalácia pomocou aplikácie Google Play. Služba umožňuje prehliadať a sťahovať (spoplatnené) aplikácie publikované prostredníctvom Google priamo na android zariadenie. Po stiahnutí aplikácie ju inštalátor automaticky nainštaluje na zariadenie
- Priama inštalácia cez vývojové prostredie. Tento spôsob budeme využívať aj my, keďže nám poskytuje možnosť, ako aplikáciu nainštalovať priamo na testovacie zariadenie a debugovať.

2.3 File transfer protocol

File transfer protokol [3] je štandardný sieťový mechanizmus navrhnutý pre zdieľanie a prenos súborov medzi vzdialenými zariadeniami. Prebieha prostredníctvom siete fungujúcej na architektúre TCP/IP, ako je napríklad Internet. FTP pracuje na princípe modelu klient-server, to znamená, že jedno zariadenie (klient) posiela príkazy a druhé zariadenie (server) čaká na požiadavky, aby ich mohlo vykonať. Počas FTP spojenia sú otvorené dva prenosové kanály(obr. 2.2): príkazový a dátový.



Obr. 2.2: Diagram FTP kanálov

Klient aj server majú pre každý kanál spustený proces, ktorý ich obsluhuje a posiela informácie.

- **DTP** (Data Transfer Process) proces, ktorý má na starosti nadviazanie spojenia a správu dátového kanálu.
- **PI** (Protocol Interpreter) interpretuje protokol a umožňuje kontrolovať DTP cez príkazy príjmané v príkazovom kanály. Serverový PI vytvára spojenie pre príkazový kanál, počúva príkazy, ktoré prichádzajú od klientskeho PI, odpovedá na ne a zahajuje serverový DTP. Klientsky PI je zodpovedný za vytvorenie spojenia s FTP serverom, posielanie FTP príkazov, príjmanie odpovedí od serverového PI a kontrolu klientského DTP.

Pripojenie klienta na server prebieha tak, že klientsky PI zaháji spojenie podľa Telnet protokolu. Klient pošle príkaz so žiadosťou o pripojenie serveru a ten ju interpretuje a spustí svoje DTP. Následne pošle klientovi odpoveď o pripojení s číslom portu, na ktorom bude prebiehať dátový kanál. Klient potom už len na danom porte počúva, kým nezačnú prichádzať dáta.

2.4 Vývojové prostredie Android Studio

Android Studio [4] je integrované vývojové prostredie založené na IntelliJ IDEA, špeciálne navrhnuté pre vývoj aplikácií pre platformu Android. Obsahuje všetky potrebné nástroje pre vývoj a debugovanie ako sú ADT (Android Development Tools), emulátor systému Android, editor grafického rozhrania a iné. Toto prostredie je momentálne dostupné na developerských stránkach Google ako beta verzia. Zároveň sa na týchto stránkach nachádzajú aj podrobné návody, ako toto IDE nainštalovať a vyvíjať aplikácie. Výhodou Android Studia je možnosť navrhovať grafický dizajn aplikácií štýlom WYSIWYG (z anglického What You See Is What You Got). Editor obsahuje možnosti ako priamo vyrenderovať grafické rozhranie na rôzne veľkých obrazovkách bez nutnosti spúšťania aplikácie. Pre testovanie navrhnutej aplikácie je možné vytvoriť rôzne nastavenia emulátora a navrhnúť tak aplikáciu, ktorá sa bude správať špecificky pre rôzne typy skutočných zariadení. Ak však máme k dispozícii reálne zariadenie, môžeme ho pripojiť cez USB k počítaču a pomocou debugovacieho nástroja Android Debug Monitor kontrolovať správanie zariadenia.

3. Špecifikácia programu

V tejto kapitole sa budeme venovať popisu našej aplikácie. Zameriame sa na jej funkčné časti a taktiež ukážeme, ako aplikáciu používať a pracovať s ňou.

3.1 Výber verzie

Prostredie, pre ktoré bola aplikácia vytváraná, je Android vo verzii 4.0.4 a využité je Android API level 15. Pri výbere verzie prostredia bolo hlavným faktorom dostupnosť testovacích zariadení. Použili sme smartphone Sony Ericsson Xperia Mini ST15i s operačným systémom Android 4.0.4 a tablet Prestigio Multipad PMP5570C s verziou 4.1.1. Kedže dopredná kompatibilita je bezproblémová, zvolili sme ako minimálnu verziu 4.0.4. Takto bude aplikácia spustiteľná na všetkých zariadeniach s verziou Androidu 4.0.4 a vyššie. Zároveň je mierená na zariadenia s dotykovou obrazovkou, keďže využíva grafické komponenty špecifické pre tento typ vstupného zariadenia. Aplikácia je stavaná tak, aby podporovala rôzne rozmery obrazoviek vďaka dynamickému užívateľskému rozhraniu, ktoré sa rozmerom prispôsobí.

3.2 KarlFTP

KarlFTP je aplikácia, ktorej hlavnou funkciou je prenos súborov medzi serverom a klientskym zariadením (smartphone, tablet). Zároveň poskytuje prehľadný prehliadač súborov na zariadeni ako, aj na serveri. Mimo toho zabezpečuje:

- pripojenie na server pomocou FTP
- prenos súborov medzi serverom a zariadením prebiehajúci na pozadí aplikácie
- operácie pre premenovanie a mazanie súborov
- operácie pre premenovanie, mazanie a vytváranie priečinkov
- podpora copy/cut/paste operácií pre súbory a priečinky
- synchronizácia priečinkov medzi serverom a zariadením
- výber súborov podľa zadaného vzoru
- podpora synchronizovaného prehliadania

KarlFTP sa snaží užívateľovi poskytnúť prehľadné a intuitívne používateľské prostredie. Použité je tak napríklad swipe view a reagovanie na zmeny v orientácii displeja zariadenia: užívateľ drží zariadenie na výšku, alebo na šírku. Podľa toho sa potom prostredie prispôsobí a zobrazí sa v inom usporiadaní, vhodnejšom pre danú orientáciu. Bližšie si tieto grafické komponenty popíšeme v časti venovanej popisu užívateľského prostredia.

3.3 Popis funkcionality

3.3.1 Pripojenie na server

Aplikácia podporuje pripojenie na server pomocou FTP. Voľbu konkrétneho komunikačného portu užívateľ vykoná pri zadávaní prihlasovacích údajov potrebných pre pripojenie.

3.3.2 Prenos súborov

Najdôležitejšou funkciou, ktorú program vykonáva, je presun súborov do a von zo zariadenia. Ten je implementovaný tak, aby bolo možné posielať v ramci jedného prenosu niekoľko súborov alebo aj priečinkov. Užívateľ tak môže využívať to, že má súbory utriedené v priečinkoch a posielať ich tak celé pomocou jedného spustenia operácie sťahovania alebo nahrávania. Ďalšou podstatnou vlastnosťou operácie prenosu je to, že prebieha na pozadí aplikácie. Aplikácia má na túto operáciu implementovanú prenosovú frontu. Vždy, keď užívateľ spustí nový prenos, pridá sa do fronty a pokiaľ už neprebieha žiadna iná prenosová operácia, nový prenos sa spustí. V opačnom prípade čaká vo fronte, kedy na neho príde rad. Táto vlastnosť prenosu na pozadí je dôležitá hlavne preto, že užívateľ takto nemusí čakať na ukončenie každého prenosu a môže tak s aplikáciou ďalej pracovať. Týmto spôsobom sa aplikácia snaží docieliť lepšiu praktickosť pri používaní programu.

3.3.3 Základná práca so súbormi

Keďže aplikácia slúži aj ako jednoduchý súborový manažér, podporuje základné operácie pre správu súborov. Tie zahrňujú premenovanie a mazanie súborov a priečinkov a tiež tvorbu nových priečinkov. Tieto operácie sú implementované pre lokálne zariadenie a aj pre server. Ďalšou dôležitou funkciou je možnosť voľby spôsobu triedenia zobrazených položiek a to podľa mena, dátumu modifikácie a veľkosti. Pre každú možnosť je ďalej možné určiť smer: zostupne alebo vzostupne. V rámci práce so súbormi je ďalším doplnkom podpora lokálneho presunu súborov. Užívateľ tak môže na zariadení kopírovať, vystrihovať a následne vložiť súbory na požadované umiestnenie na zariadení. Táto operácia taktiež prebieha na pozadí pomocou prenosovej fronty.

3.3.4 Synchronizácia

Pokročilejšou funkciou je synchronizácia pre priečinky. Užívateľ si zvolí priečinok na zariadeni a na serveri a určí, ktorý z nich bude zdroj a ktorý cieľ. Taktiež zvolí jeden z troch typov synchronizácie. Proces následne porovnáva podľa relatívnej cesty a názvu jednotlivé súbory medzi synchronizovanými priečinkami. Porovnávanie prebieha podľa zvoleného typu synchronizácie:

- aktualizácia – ak sú názvy súborov rovnaké, ale súbor v zdroji je novší alebo väčší ako v cieli, tak sa súbor zo zdroja skopíruje do cieľa
- prepis – ak sú názvy súborov rovnaké, súbor v cieli je prepísaný súborom zo zdroja bez ohľadu na veľkosť, alebo čas modifikácie

- synchronizácia – pri tomto type je vytvorená presná zrkadlová kópia zdroja v cieli, to znamená, že ak je súbor novší alebo väčší, kopíruje sa do cieľa. Ak sú subory rovnaké, nevykoná sa nič a ak súbor je v cieli, ale nie je v zdroji, tak je z cieľa zmazaný.

Synchronizácia taktiež prebieha na pozadí aplikácie a nebráni tak užívateľovi v ďalšom používaní.

3.3.5 Výber pomocou vzoru

Táto jednoduchá funkcia slúži k tomu, aby si užívateľ mohol zo zobrazených položiek vyberať rýchlejšie, ak majú súbory podobné názvy (najčastejším prípadom sú fotografie a videá uložené na smartphone). Pomocou zadaného vzorového textu aplikácia prejde práve zobrazený priečinok a vyznačí všetky položky, ktoré vzoru vyhovujú. Následne už môže užívateľ s nimi pracovať rovnako, ako keby si ich vyznačil sám.

3.3.6 Synchronizované prehliadanie

Poslednou z doplnkových funkcií je posun v priečinkoch na zariadení aj na serveri súčasne. Tento synchronizovaný posun funguje na základe porovnávania názvov priečinkov. Ak sa na vzdialenom aj na lokálnom zariadení nachádza priečinok s rovnakým názvom v aktuálne zobrazenom umiestnení stačí, ak sa užívateľ presunie do takéhoto priečinka len na jednom zariadení. Na druhom sa aplikácia presunie sama. Ak sa nevyskytuje, užívateľ sa posunie len tam, kde klikol.

4. Analýza programu

Pri vytváraní našej aplikácie sme museli riešiť niekoľko zásadných problémov, ktoré súviseli s jej správnym fungovaním.

4.1 Pripojenie klienta

Prvým problémom bolo navrhnúť spôsob, akým sa budeme pripájať na server. Potrebovali sme, aby aplikácia dokázala naraz nahrávať/sťahovať súbory zo serveru a komunikovať so serverom pri vypisovaní súborov nachádzajúcich sa na serveri. Vypisovanie obsahu priečinkov je taktiež považované za dátový prenos. Ak totiž klient začal operáciu sťahovania/nahrávania, bol blokovaný na tento jeden prenos a užívateľ nemohol zadať požiadavku na vylistovanie priečinku.

Prvým možným riešením, ako od užívateľa spracovávať súbežné požiadavky, bolo „cashovanie“. Uložili by sme si všetky jeho požiadavky do jednej prioritnej fronty a postupne by sme ich spracovávali. Kratšie trvajúce operácie (výpis súborov, premenovanie a mazanie) by mali najvyššiu prioritu a dlhšie trvajúce (prenos súborov) najnižšiu. Podľa určenej priority by sa operácie postupne z fronty spracúvali. Týmto spôsobom by sme však úplne nedocielili požadovanú funkcionalitu. Aplikácia by síce všetky požiadavky spracovala, no užívateľ by aj tak musel čakať na ich postupné vykonanie.

Druhým riešením bolo vytvorenie dvoch pripojení [1]. Tento spôsob sme využili na rozdelenie operácií nasledovným spôsobom.

- Prvé pripojenie spracúva požiadavky na: výpis obsahu priečinku, premenovanie a zmazanie priečinkov/súborov, vytvorenie nového priečinku.
- Druhé pripojenie spracúva požiadavky na sťahovanie/nahrávanie súborov.

Využívame takto dva prenosové kanály, pričom každý z týchto kanálov je obsluhovaný v separátnych procesoch. Aplikácia takto zvláda vykonávať požiadavky od užívateľa súvisiace so správou súborov a zároveň prenášať súbory.

4.2 Operácie na pozadí

Keďže aplikácia vykonáva operácie súvisiace s prenosom údajov, bolo nutné tieto operácie oddeliť od samotnej aplikácie. Hlavným dôvodom tohto oddelenia bol fakt, že užívateľské prostredie je spravované v jednom UI vlákne. Ak by sme toto vlákno zaťažili aj prenosovými operáciami, tento jedno-vláknový model by dosahoval slabú výkonnosť. Mohlo by taktiež dochádzať k situáciám, kde by toto vlákno bolo blokované prenosom medzi klientom a serverom a UI by nereagovalo. Ak by tento stav trval dlhšie ako 5 sekúnd, android systém by takúto situáciu charakterizoval ako ANR (application not responding) a ponúkol by užívateľovi možnosť túto aplikáciu zabiť, alebo čakať na jej dokončenie. Operácie sme preto od UI vlákna oddelili background vláknami. Dôležitý bol výber správneho typu

[1] pripojením sa myslí pripojenie komunikačné aj dátové spolu ako pár

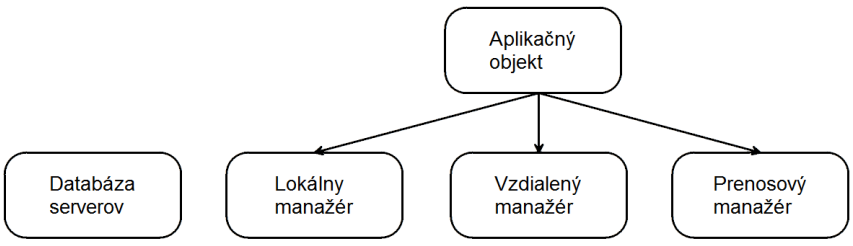
vlákna. Android ponúka rôzne typy vlákien určené pre rôzne časovo náročné úlohy. Podľa ich špecifikácie uvedenej na [2], sme sa rozhodli pre použitie *AsyncTask* vlákna a *Service*.

AsyncTask vlákna zabezpečujú vykonávanie blokujúcich operácií, pričom poskytujú jednoduchý spôsob, ako publikovať ich postup do UI vlákna bez nutnosti špeciálnej obsluhy vlákna. Tento typ vlákna je navrhnutý tak, aby bol použitý na krátko trvajúce operácie (niekoľko sekúnd). Pre tieto vlastnosti bol AsyncTask ideálnou voľbou pre operácie vykonávané prvým pripojením.

Service (služba), popísaná v časti 2.1, je špeciálny komponent aplikačného objektu, ktorý je určený na vykonávanie dlhotrvajúcich operácií. Preto je vhodným spôsobom, ako vykonávať operácie pre druhé pripojenie. Zároveň sme si zvolili typ naviazaného *Service* komponentu, kedže potrebujeme, aby sa pri operácii sťahovania/nahrávania zobrazoval jej percentuálny stav. Ďalšou výhodou je, že služba je oddelená od aplikácie, ktorá ju spustila a prebieha, aj keď sa užívateľ prepne do inej aplikácie.

4.3 Návrh aplikácie

Pri vývoji aplikácie bolo dôležité vytvoriť vhodnú kostru aplikácie (obr. 4.1). Prvým krokom bolo pripojenie aplikácie k serveru. To znamenalo, že sme potrebovali objekt, ktorý by mal uložené informácie o jednotlivých serveroch, ku ktorým sa užívateľ bude chcieť pripájať. Na to sme vytvorili jednoduchú databázu, ktorú samotný android poskytuje. Ďalej bolo potrebné pripojenie udržiavať. Pre tento účel sme vytvorili objekt manažéra. Ten by spravoval nie len pripojenie, ale aj základné operácie nad súbormi. Kedže bolo nutné pristupovať k lokálnym a vzdialeným súborom, boli zo základného manažéra odvodené dva špeciálne typy, lokálny a vzdialený manažér. Lokálny manažér pre spravovanie súborov na zariadení a vzdialený manažér s prvým pripojením na server pre prácu so súbormi na serveri. Pre nahrávanie/sťahovanie súborov zo serveru sme vytvorili prenosového manažéra, ktorý dostal druhé pripojenie na server a prenosovú frontu. K jeho úlohám ďalej pribudla obsluha servisného vlákna, ktoré má vykonávať úlohy pre tohto manažéra. Aplikácia sa pri prechode konfiguračnými zmenami deštruuje a znova inicializuje. Toto správanie však bolo nežiadúce. Pre udržiavanie celkového stavu sme si preto vytvorili vlastný objekt aplikácie, ktorý nezaniká. Preto bol tento objekt vhodný, aby obsahoval našich troch manažérov a zachovával tak ich stav. Takto sme dostali kostru nášho programu, do ktorej sme mohli vložiť všetky doplnkové operácie. Na ňu sme ďalej napojili grafické rozhranie a vytvorli takto našu aplikáciu.



Obr. 4.1: Diagram zobrazujúci kostru našej aplikácie

4.4 Grafické rozhranie

Návrh grafického rozhrania spočíval v zvolení vhodných grafických komponentov pre jednotlivé časti aplikácie. Jednotlivé komponenty sme volili podľa toho, ktoré časti aplikácie bude užívateľ najviac využívať a aby bolo jednoduché sa medzi nimi prepínať. Andorid ponúka dva typy komponentov pre stavbu grafického rozhrania. Sú to aktivita a fragment. Pri konštrukcii sme sa museli držať aj nasledujúcich pravidiel:

1. Aktivita musí byť základným zobrazovacím komponentom, čiže fragment sám bez nadradeného komponentu nemôže existovať
2. Aktivita nemôže obsahovať vnorené aktivity.
3. Fragment nemôže obsahovať vnorené fragmenty.

Použitím týchto znalostí sme navrhli nasledujúcu grafickú štruktúru. Pri používaní aplikácie predpokladáme, že najmenej bude frekventovaná práca so serverovou databázou. Na jej obsluhu použijeme dve obrazovky: jedna úvodná, v ktorej užívateľ vyberie uložený server z databázy a druhá editačná, kde o serveri upraví informácie. Úvodná a editačná obrazovka majú jednoduchú kompozíciu, preto sa každá z nich bude zobrazovať prostredníctvom vlastnej aktivity.

V našej aplikácii budú najviac využívaní manažéri. Prepínanie medzi nimi bude teda najfrekventovanejšie. Štruktúra obrazoviek manažérov je zložitejšia, takže sa vytvoril priestor na návrh rozhrania odlišnými postupmi. Prvou možnosťou bolo priradiť každému manažérovi jednu aktivitu. Táto metóda však bola značne neefektívna kvôli frekventovanému prechodu medzi jednotlivými obrazovkami manažérov. Prepínanie medzi aktivitami prebieha nasledovne: vždy po prepnutí do ďalšej aktivity sa predošlá uloží do zásobníka, aby sa k nej dalo vrátiť po stlačení systémového tlačidla *Späť*. Užívateľ by pri častom prepínaní obrazoviek veľmi rýchlo zásobník zaplnil rovnakými objektami, čo by veľmi zvýšilo pamäťovú náročnosť aplikácie. Zároveň by sa prepínala celá obrazovka, čo by mohlo na užívateľa pôsobiť mätúco. Druhou možnosťou bolo využitie fragmentov, ktoré sa na zásobník neukladajú. Základom musela byť aktivita podľa 1. pravidla. Do nej sme zakomponovali tri fragmenty, jeden pre každého manažéra, vo forme záložiek. Takýto koncept má výhodu v tom, že užívateľ sa môže jednoducho prepínať medzi manažérmi, pričom sa stále nachádza na jednej obrazovke, na ktorej fragmenty zaberajú len určitú časť a tak je stále zobrazené akési pozadie od aktivity. Fragmenty sa pri prepínaní neukladajú a tak nezaťažujú pamäť. Zároveň sme takto získali možnosť použitia swipe views a dodali tak aplikácii lepšiu ovládateľnosť.

4.5 Serverová databáza

Pre pripojenie k serveru musí užívateľ o serveri zadať prihlasovacie údaje. Aby tieto údaje nemusel zadávať vždy, keď aplikáciu spustí, bolo by vhodné si tieto údaje pamätať, a to pre každý server, na ktorý sa bude pripájať. Pre udržiavanie perzistentných dát na zariadení ponúka android štyri možnosti: zdieľané preferencie, interné a externé úložiská a SQLite databázu. Pre naše potreby vyhovovala SQLite databáza. Zdieľané preferencie sú primitívne dáta ukladané štýlom kľúč-hodnota. Ak by sme chceli ukladať údaje o serveroch, museli by sme pre každý

server a každý jeho údaj generovať unikátny kľúč, čo by bolo dosť neefektívne. Pri externom úložisku by bola aplikácia závislá na pamäťovej karte a po jej vysunutí by uložené dáta neboli dostupné. Ak by sme využili interné úložisko, museli by sme navrhnúť vlastný formát pre uloženie dát v textovom súbore a tiež parser. SQLite databáza má jednoduché použitie. Nevyžaduje dodatočnú inštaláciu, ani administráciu. Museli sme vytvoriť len SQL výrazy pre vytvorenie a aktualizáciu databázy a ďalej už bola databáza spravovaná automaticky systémom.

4.6 Knižnica FTP

Knižnicu pre pripojenie k FTP serveru sme vyberali takú, aby bola s bezplatnou licenciou. Vybrali sme knižnicu commons-net od spoločnosti Apache, keďže je stabilná, poskytuje FTP pripojenie a je rozšírená v open-source programoch.

5. Popis implementácie

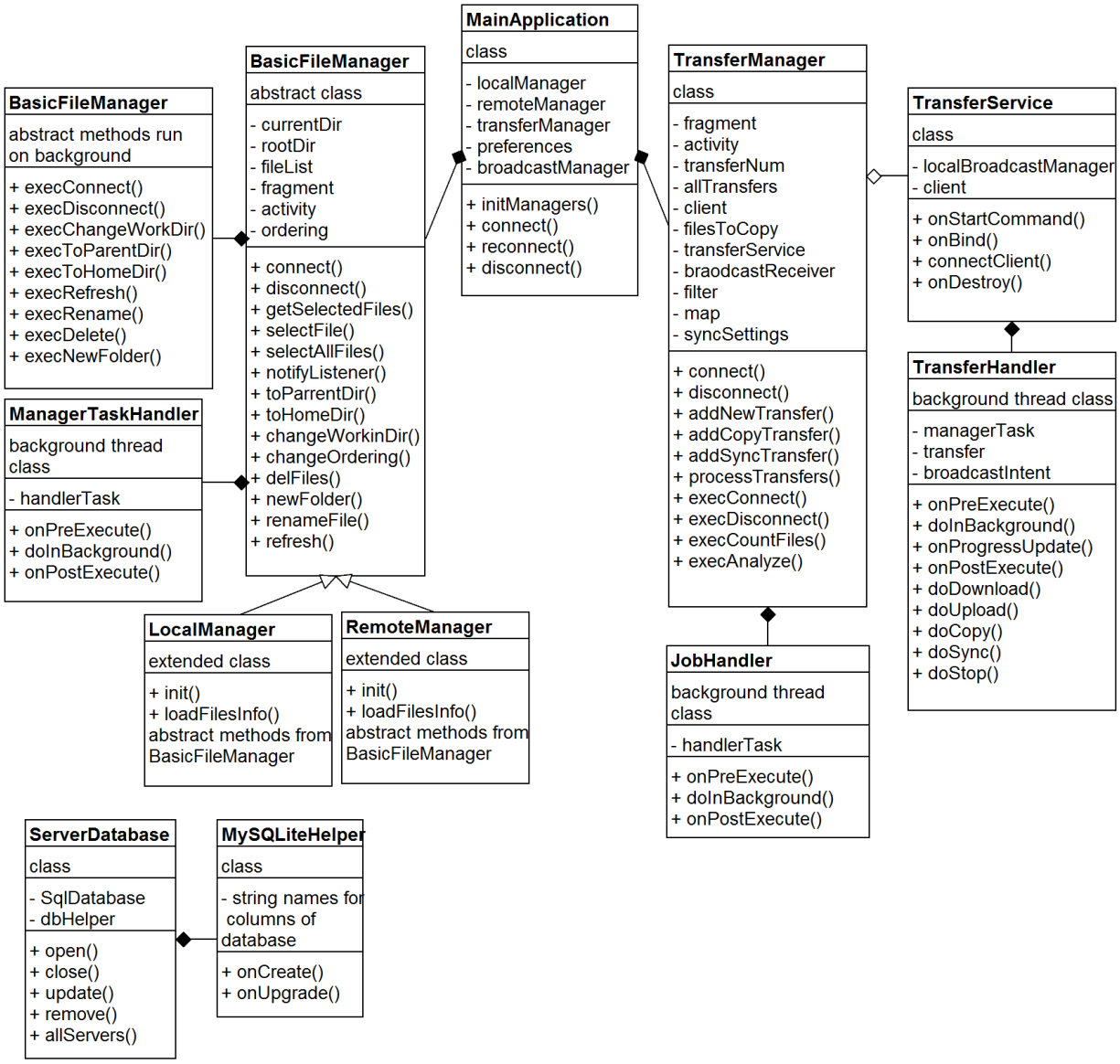
V poslednej kapitole si popíšeme implementáciu jednotlivých častí aplikácie.

5.1 Použité technológie

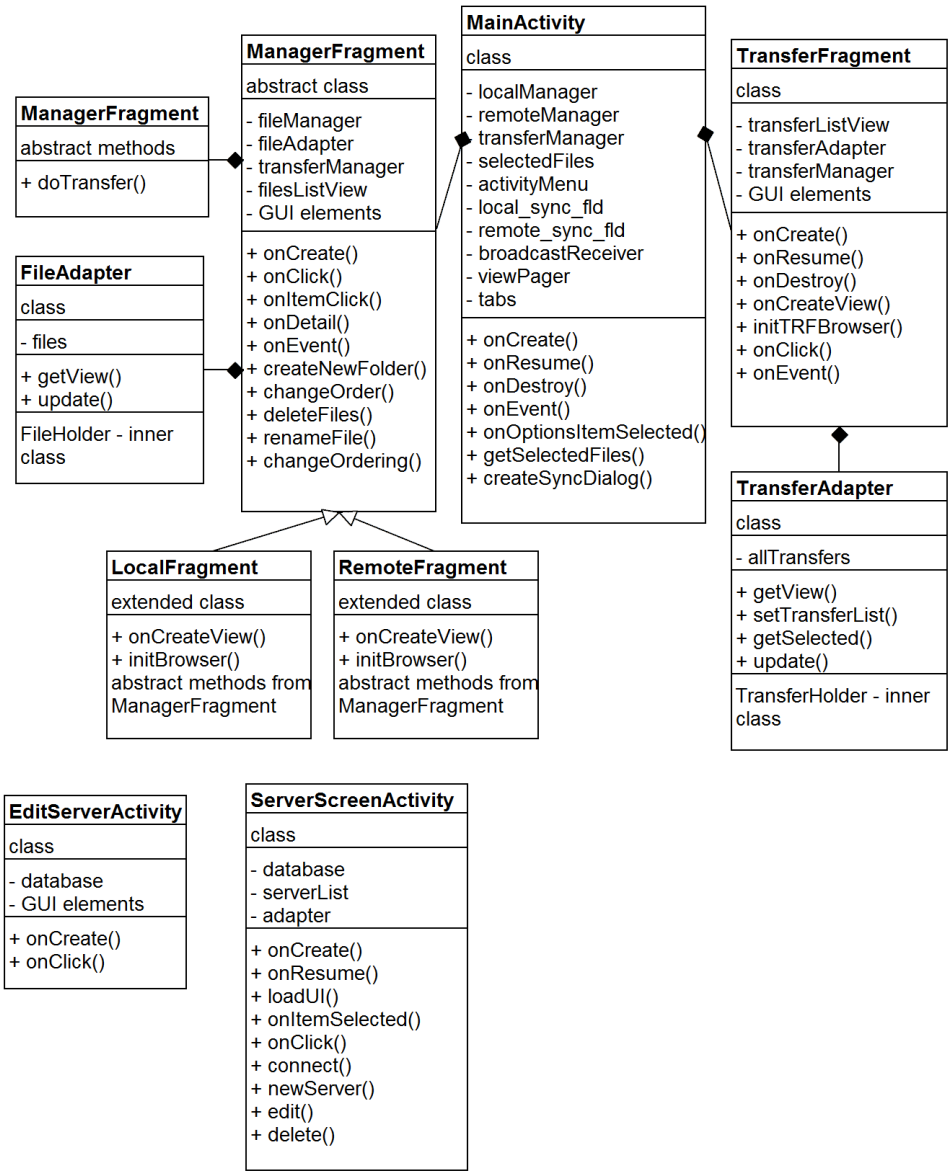
Aplikácia využíva technológie určené pre vývoj android aplikácií. Logická časť aplikácie využíva Javu 7 a SQLite Database. Vzhľad aplikácie je zapísaný v XML súboroch, pomocou ktorých sa v sytéme Android definujú rozloženia komponentov v grafickom rozhraní. Pre komunikáciu so serverom je použitá knižnica od spoločnosti Apache, konkrétne org.apache.commons vo verzii 3.3.

5.2 Kostra aplikácie

Aplikácia je rozdelená do dvoch úrovní. Prvá je logická úroveň (obr. 5.1), ktorá zabezpečuje vykonávanie všetkých operácií so súbormi alebo priečinkami. Tá sa ďalej dá rozdeliť do troch častí: **MainApplication** a jej inštancie manažérov - **TransferManager, LocalManager, RemoteManager**, databáza s uloženými servermi **ServerDatabase** a obsluha service vlákna **TransferService**. Druhá je grafická úroveň (obr. 5.2). Tá zabezpečuje zobrazovanie všetkých komponentov užívateľského rozhrania a komunikáciu medzi užívateľom a logickou úrovňou. Je rozdelená na dve časti: správa databázy serverov **ServerScreenActivity** a **EditServerActivity**; **MainActivity** a jej fragmenty - **TransferFragment, LocalFragment, RemoteFragment**. Tieto dve časti sú priamo naviazané na ich výkonné časti v logickej úrovni. Mimo tohto rozdelenia sú implementované doplnkové objekty umožňujúce fungovanie oboch úrovní.



Obr. 5.1: Znázornenie logickej úrovne



Obr. 5.2: Znázornenie grafickej úrovne

5.3 MainApplication

Hlavným prvkom celej logickej časti aplikácie je objekt **MainApplication**. Jeho úlohou je udržiavanie globálneho stavu aplikácie. Tento stav je nutné udržiavať kvôli konfiguračným zmenám medzi ktorými aplikácia prechádza, ak sa mení orientácia displeja. Všetky grafické komponenty sú pri tejto zmene deštruované a incializované znova s novou konfiguráciou displeja. Stav **MainApplication** objektu sa pri tejto zmene zachováva, rovnako ako aj po celú dobu, kým je aplikácia spustená. Z tohoto dôvodu obsahuje inštancie manažérov**localManager, remoteManager, transferManager**, ktorých stav sa nemôže stratiť. Pri prepnutí do hlavnej obrazovky sa spustí, pomocou metódy *initManagers()*, vytvorenie nových objektov pre jednotlivé manažére. Zároveň sa **MainApplication** postará o spustenie pripojenia oboch klientov pomocou *connect()*. Pri odchode z hlavnej obrazovky metódou *disconnect()* klientov odpojí. Ďalšou výhodou je, že tento ob-

jekt je prístupný z ktoréhokoľvek miesta v kóde a je tak použitý na volanie metód medzi manažérmi.

5.4 Komunikácia medzi aplikáciou a serverom

Pre komunikáciu medzi aplikáciou a serverom je z knižnice org.apache.commons použitý klientsky objekt **FTPClient**. Pomocou prihlasovacích údajov zadaných pre daný server sú pripájané klienty pre **TransferManager** aj **RemoteManager**.

5.5 Manažéri

5.5.1 Lokálny a vzdialený manažér

Lokálny a vzdialený manažér majú rovnakú funkcionalitu, preto sú implementovaní ako potomkovia abstraktnej triedy **BasicFileManager**. Trieda združuje spoločné komponenty, ktoré určujú stav a funkcionalitu oboch manažérov. Tie musia mať informáciu o tom, v ktorom priečinku sa nachádzajú – **currentDir**, cestu k domovskému adresáru – **rootDir**, zoznam súborov v aktuálnom priečinku – **fileList** a typ usporiadania – **ordering**. Obaja manažéri majú podobné vykonávanie ich základných funkcií. Rozdielom je to, že **RemoteManager** využíva na vykonanie operácie inštanciu klienta a **LocalManager** nie. **BasicFileManager** obsahuje triedu **ManagerTaskHandler**, ktorá spúšťa jednotlivé operácie na pozadí. Tieto opérácie sú definované ako abstraktné a každý z manažérov ich implementuje podľa svojich potrieb. Ak má teda niektorý z manažérov vykonať jednu zo spoločných operácií, zavolajú sa najskôr spoločné metódy. Operácie sa pripravia a následne prenechajú na vykonanie na pozadí manažérom, pre ktorého sú určené. Oba manažére sú prepojené s hlavnou aktivitou – *activity*, ich určenými fragmentami – *fragment*. Pomocou metódy *notifyListener()* im posielajú informácie o zmenách v zozname súborov – **fileList** alebo o eventoch, ktoré nastali a na ktoré majú adekvátne reagovať.

5.5.2 Prenosový manažér

Prenosový manžér **TransferManager** je trieda, ktorá obsahuje prenosovú frontu **allTransfers** a jej hlavnou úlohou je spracovávať položky v nej. Sú tri typy položiek. Prvým typom je sťahovanie/nahrávanie. Ak od lokálneho alebo vzdialeného manažéra dostane prenosový manažér súbory pre túto operáciu, vytvorí pomocou metódy *addNewTransfer()* novú položku (prenos) so stavom *čaká* v **allTransfers**. Pre súbory v tejto položke sa najskôr pomocou background vlákna **JobHandler** a jeho metódy *execCountFiles()* spočíta ich celková veľkosť. Táto veľkosť je potom používaná pri percentuálnom výpočte prenesených bajtov. Po spočítaní je položka zaradená na koniec fronty kde čaká, kým na ňu príde rad. Druhým typom je kopírovanie súborov v rámci lokálneho zariadenia. Po tom, ako **TransferManager** dostane od **LocalManager** zoznam súborov, ktoré sa majú prekopírovať, uloží si ich do premennej **fileToCopy**. Následne čaká na to, kedy bude známa cesta, kam sa súbory majú prekopírovať. Ak cestu dostane,

pomocou metódy *addCopyTransfer()* vytvorí novú položku so stavom *čaká* v **allTransfers** a rovnako, ako pri prvom type veľkosť súborov spočíta. Posledným typom je synchronizácia. Po získaní potrebných informácií (zdrojový a cieľový priečinok, typ synchronizácie) manažér spustí na pozadí metódu *execAnalyze()*, ktorá porovná synchronizované priečinky. Porovnávanie sa vykonáva podľa typu nasledovne. Nech máme porovnávané súbory f1 zo zdroja a f2 z cieľa, potom:

- aktualizácia – ak f1.názov == f2.názov && f1.časModifikácie > f2.časModifikácie, tak f2 nahradiť súborom f1, inak preskočiť
- prepis – ak f1.názov == f2.názov, tak f2 nahradiť súborom f1, inak preskočiť
- synchronizácia – ak f1.jeVZdroji && !f2.jeVCieli, tak f1 presunť do cieľa; ak !f1.jeVZdroji && f2.jeVCieli, tak f2 v cieli zmazať; ak f1.názov == f2.názov && f1.časModifikácie > f2.časModifikácie, tak f2 nahradiť súborom f1, inak preskočiť

Porovnávanie názvov prebieha pomocou inštancie **map** objektu **HashMap**, kde kľúčom je relatívna cesta k súboru a hodnotou obejkt **SyncItem** obsahujúci informácie o súbore v zdroji a cieli potrebné pre synchronizáciu. Po analýze priečinkov je vytvorená nová položka so stavom *čaká* v **allTransfers** metódou *addSyncTransfer()*. Po pridaní aspoň jednej položky do prenosovej fronty manažér v metóde *processTransfers()* frontu spracuje. Prvú položku z nej vyberie, predá ju **TransferService** a čaká, kým nedostane potvrdenie o dokončení prenosu, kedy položku prepne do stavu *hotovo*. Následne vyberie novú prvú položku a znova ju predá na spracovanie. Takto pokračuje kým fronta, nie je prázdna.

5.6 TransferService

Trieda **TransferService** umožňuje vykonávať všetky dlhodobé operácie na pozadí aplikácie. Po vytvorení a naviazaní na prenosového manažéra, dostane od **TransferManager** nový prenos, ktorý má vykonávať. Na vykonávanie využíva vlákno na pozadí spravované vnútornou triedou **TransferHandler**. Tá podľa typu operácie spustí jej vykonávanie. Najskôr v metóde *onPreExecute()* zmení stav prenosu na *vykonáva sa*. Následne v *doInBackground()* spustí konkrétnu metódu podľa typu operácie:

- *doDownload()* - sťahovanie zo serveru
- *doUpload()* - nahrávanie na server
- *doCopy()* - kopírovanie súborov v rámci zariadenia
- *doSync()* - synchronizácia priečinkov

Počas vykonávania operácie metóda *onProgressUpdate()* pomocou **broadcastIntent** zverejňuje prostredníctvom správy percentuálny pokrok operácie. Túto správu zachytáva **TransferManager** a podľa ID položky vo svojom zozname danú položku aktualizuje. Po skončení operácie sa zavolá metóda *onPostExecute()*, ktorá službu ukončí. Zároveň prenosovému manažérovi oznámi, že môže začať spracovávať ďalšiu položku vo svojom zozname.

5.7 ServerDatabase

Táto trieda spravuje databázu s informáciami o jednotlivých serveroch, ktoré užívateľ do aplikácie uloží. Na komunikáciu s touto databázou slúži pomocný odbjekt *MySQLiteHelper*. Pre prácu s ňou je implementovaných 5 metód, ktoré zabezpečujú: otvortenie (*open()*) a zatvorenie (*close()*) databázy, jej aktualizovanie (*update()*), odstránenie záznamu (*remove()*) a výpis všetkých záznamov (*allServers()*). Samotná databáza je tvorená jednou tabuľkou obsahujúcou alias servera, jeho adresu, meno a heslo užívateľa, číslo portu, označenie anonymného pripojenia, domovský priečinok pre zariadenie a server.

5.8 MainActivity

Trieda **MainActivity** je grafický komponent zobrazujúc hlavnú obrazovku. Obsahuje objekt **viewPager**, ktorý zabezpečuje napojenie fragmentov na ňu. Trieda je zodpovedná za fungovanie menu aplikácie, kde sa nachádzajú položky s doplnkovými funkciami. Tieto funkcie sa aplikujú na všetkých troch manažérov, preto na nich má referencie. Ak nastane konfiguračná zmena a užívateľ otočí zariadenie, **MainActivity** zareaguje a postará sa o prekreslenie celej obrazovky do novej konfigurácie tým, že spustí vytvorenie samej seba [1] a svojich fragmentov v *onCreate()* metóde. Ak nastane nejaká udalosť, o ktorej musí aplikácia užívateľa upozorniť, metóda *onEvent()* príslušnú udalosť dostane a vypíše na obrazovku upozornenie.

5.9 Manager Fragments

Každý manažér má ako grafický komponent pridelený fragment. Rovnako, ako v logickej úrovni, je pre lokálneho a vzdialeného manažéra vytvorený abstraktný **ManagerFragment**, ktorý pomocou metód *onClick()* a *onItemClick()* reaguje na stláčanie tlačidiel a položiek v zozname súborov. **LocalFragment** a **RemoteFragment** sú potomkami abstraktného fragmentu, pričom si inštanciujú vlastné tlačidlá (**GUI elements**) a zoznamy súborov (**filesListView** udržiavaný pomocou **FileAdapter**), no využívajú spoločné metódy nadradenej triedy. Každý fragment je napojený na svojho manažéra, ktorému spúšťa jednotlivé operácie. Pre komunikáciu v smere manažér -> fragment slúži metóda *onEvent()*, ktorú manažér zavolá a odovzdá fragmentu typ udalosti, ktorá sa odohrala. Abstraktnú metódu *doTransfer()* má každý fragment zvlášť naimplementovanú, keďže **LocalFragment** spúšťa nahrávanie súborov a **RemoteFragment** ich sťahovanie. Pre zobrazenie zoznamu súborov **filesListView** slúži **FileAdapter**, ktorého úlohou je tento zoznam pre fragment udržiavať. **TransferFragment** má odlišnú štruktúru, preto je vytvorený samostatne. Zobrazuje zoznam prenosov **transferListView**, ktorý je udržiavaný v objekte **TransferAdapter**. Fragment je napojený na svoj prenosový manažér, ktorému odovzdáva volania *onClick()* metódy. Opačný smer komunikácie obsluhuje *onEvent()* metóda, ktorá zachytáva udalosti od manažéra.

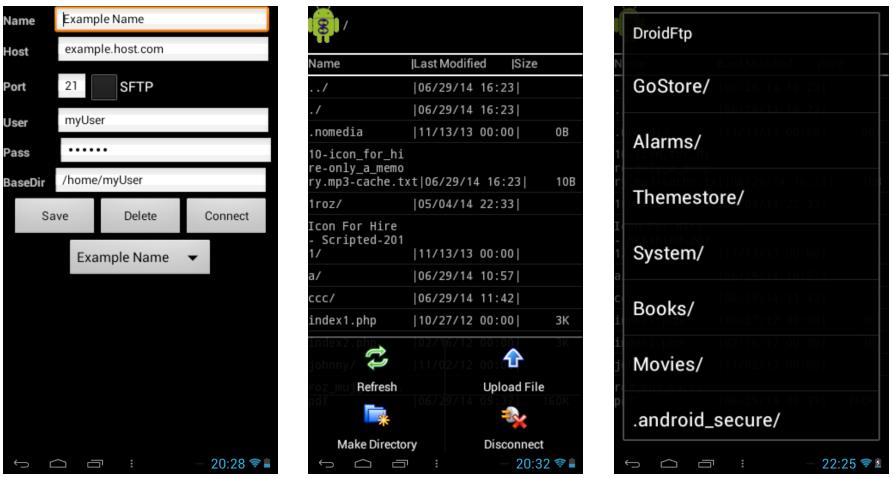
[1] vytvorenie prebieha podľa životného cyklu aktivity popísaného v kapitole 2.2

6. Existujúce aplikácie

V súčasnosti existuje niekoľko aplikácií, ktoré fungujú ako FTP klient. Tieto aplikácie sú dostupné prostredníctvom služby Google Play a môžeme ich teda vyskúšať na mobilnom zariadení. Než podrobne popíšeme našu aplikáciu, spravíme analýzu týchto existujúcich riešení.

6.1 DroidFtp

Prvou skúšanou je aplikácia DroidFTP (obr. 6.1). Tento program, spomedzi testovaných, má najhoršiu použiteľnosť. Užívateľské prostredie je veľmi nepraktické a neprehľadné a tak je zložité určiť, kde sa vybraný súbor bude sťahovať, alebo posielať. Pre prenos je možné vybrať vždy len jeden súbor alebo priečinok, čo je veľmi obmedzujúce. Umiestnenie, kde sa majú súbory sťahovať, je možné určiť len pred pripojením na server a nie je možné počas pripojenia toto nastavenie zmeniť. Nahrávanie už tento problém nemá, keďže sa posielaný súbor uloží do práve prehliadaného priečinka. Prenos prebieha na pozadí, no nezobrazí sa žiadny progress bar a užívateľ tak nie je schopný určiť, či prebieha. Program okrem prenosu súborov neposkytuje žiadne ďalšie funkcie ako napríklad copy/paste podpora alebo synchonizácia. Celkovo je práca s aplikáciou veľmi problémová a neintuitívna.

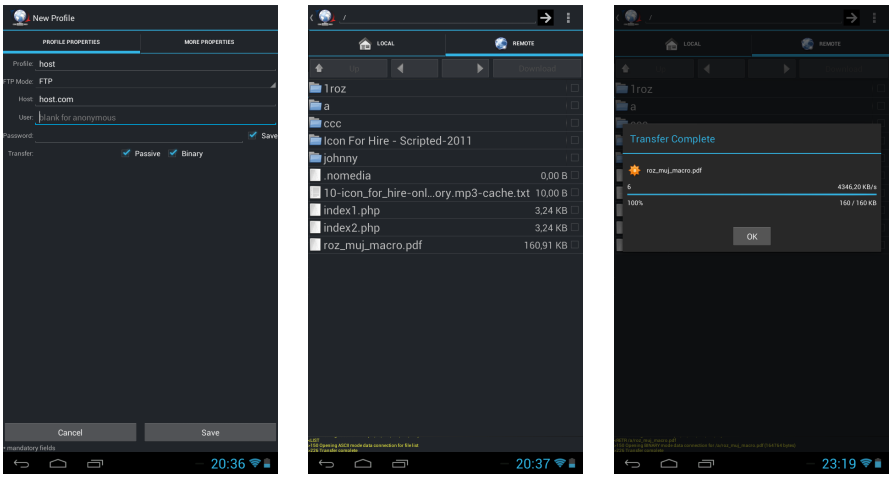


Obr. 6.1: DroidFtp

6.2 FtpCafe

Aplikácia FtpCafe (obr. 6.2) už je oveľa praktickejšia ako predošlá. Užívateľské prostredie je jednoduché a prehľadné a užívateľ tak vždy vie, kde sa na zariadení alebo serveri nachádza. Nechýbajú základné funkcie ako premenovávanie a mazanie súborov, vytváranie priečinkov a úprava triedenia. Veľmi praktickou funkciou je označovanie súborov na základe vzoru v názvoch súborov. Taktiež dobrým doplnkom sú tlačidlá na posun späť a dopredu podľa toho, ako sa užívateľ pohybuje v priečinkoch. Prenos súborov je bezproblémový, a keďže aplikácia podporuje

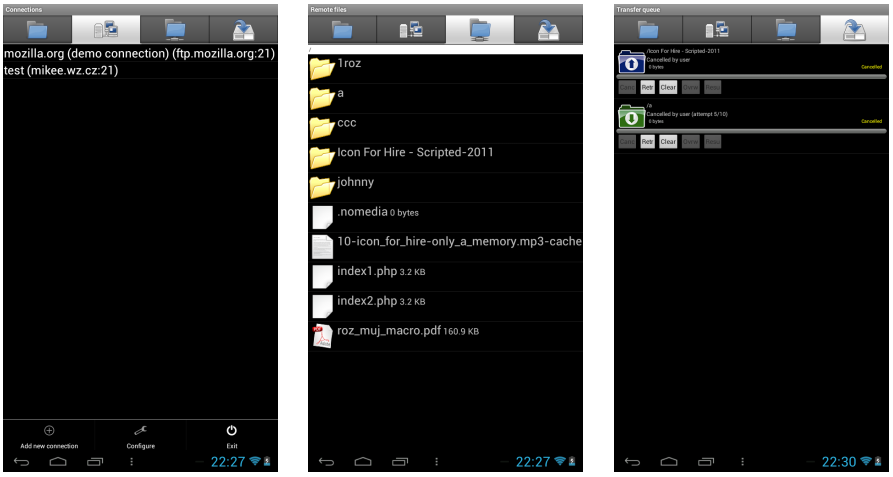
selektovanie súborov, je možné do jedného prenosu vložiť niekoľko položiek. Aplikácia zobrazuje v spodnej časti log správy o komunikácii zo serverom. Táto časť obrazovky je však veľmi úzka a užívateľ má tak problém scrollovať tento výpis. Ďalším problémom je, že prenos prebieha jedine v popredí aplikácie. Ak užívateľ spustí sťahovanie alebo posielanie súborov, nemôže už s programom ďalej pracovať a musí čakať, kým sa operácia nedokončí. Taktiež chýba copy/paste funkcia a možnosť synchronizácie. Na druhú stranu sú podporované rôzne typy pripojenia ako FTP, FTPS (implicit a explicit) a SFTP. Zároveň je možné exportovať a importovať uložené pripojenia.



Obr. 6.2: Greyhound FTP

6.3 Greyhound FTP

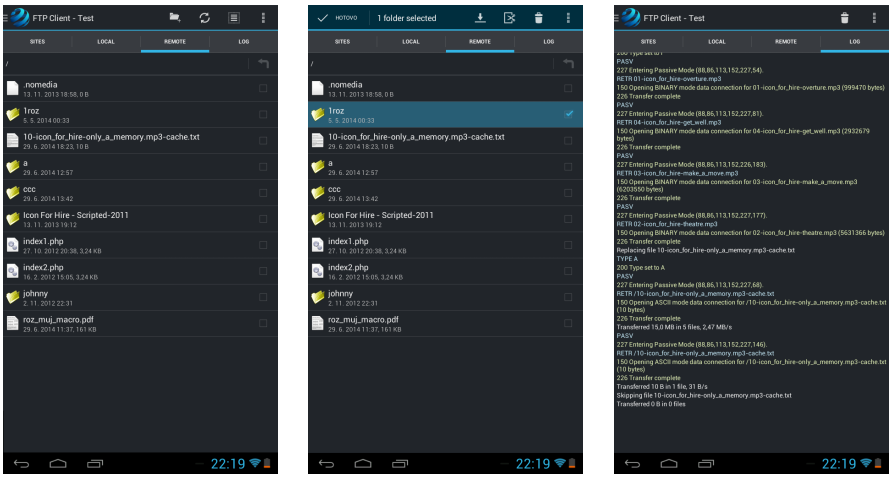
Ďalšia testovaná aplikácia je Greyhound FTP (obr. 6.3). Prvou vlastnosťou, ktorou sa odlišuje od ostatných skúšaných programov, je funkčná podpora prenosov na pozadí. Užívateľ takto môže pracovať s aplikáciou aj počas sťahovania/posielania. Zároveň má kontrolu nad prenosom a môže tak sledovať jeho priebeh, prípadne ho zastaviť a znovu spustiť. Táto funkcia výrazne prispieva k praktickosti celej aplikácie. Čo ju však obmedzuje, podobne ako pri DroidFTP, je možnosť sputiť prenos len na jednom priečinku alebo súbore. Ak chce teda užívateľ poslať viacero súborov, musí ich všetky pracne popridávať do prenosovej fronty. Ďalšia zaujímava funkcia je cashovanie obsahu adresárov. Aplikácia tak reaguje rýchlejšie, keďže si zoznam súborov ukladá do pamäte a nesťahuje ho vždy zo serveru. Užívateľské prostredie je až príliš jednoduché a neposkytuje podrobnejšie informácie o jednotlivých súboroch. Pozitívom však je možnosť upraviť si veľkosť písma a riadkov v zozname súborov. Typy pripojenia sú dostupné len dve a to FTP a SFTP. Aplikácia má podporovať remote view, čo je možnosť otvoriť súbor bez nutnosti stiahnutia zo serveru na zariadenie. Túto funkciu sa nám kvoli nefunkčosti nepodarilo otestovať. Chýbajúcimi funkciami sú opäť copy/paste a synchronizácia.



Obr. 6.3: Greyhound FTP

6.4 FTP Client

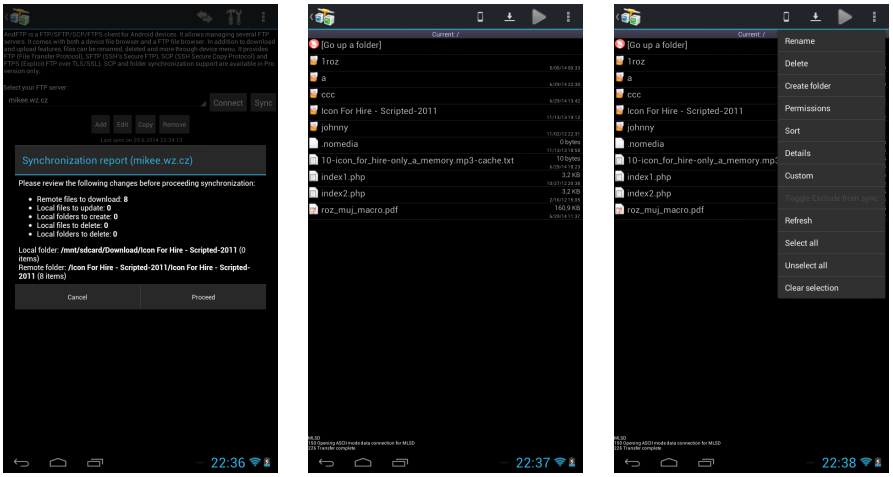
Aplikácia FTP Client (obr. 6.4) sa vyznačje hlavne veľmi praktickým a prehľadným prostredím. Využíva dynamické zobrazenie a ovládacie ikony sa zobrazujú podľa konkrétnych potrieb. Kým nie je vybraný nejaký súbor, zobrazené sú len ikony pre vytvorenie nového priečinku, obnovenie obsahu a výber všetkých súborov v zozname. Ak však užívateľ zvolí súbor, zobrazia sa možnosti ako kopírovanie, vystrihnutie, premenovanie a zmazanie súboru. Ovládanie aplikácie tak je veľmi jednoduché a intuitívne, pričom využíva moderné grafické doplnky ako swipe views a dynamický action bar. Ďalšími vhodnými doplnkami sú, ako na jedinej aplikácii, plnohodnotné zobrazenie logu pripojenia k serveru a podpora viacerých jazykov. Program zároveň dobre komunikuje s užívateľom. Ak vyprší čas spojenia, aplikácia to oznámi spolu s možnosťou obnovy spojenia. Čo však chýba, je prenos súborov na pozadí. FTP Client má tak rovnaký problém, ako napríklad FTPCafe a užívateľ teda po spustení prenosu s aplikáciou už viac interagovať nemôže, dokiaľ sa operácia neskončí. Medzi pozitívne vlastnosti však patrí import/export nastavení serverov, široká ponuka typov pripojenia (FTP,FTPS, SFTP,SMB/CIFS) a podpora copy/paste na lokálnom zaridení.



Obr. 6.4: FTP Client

6.5 AndFTP

Najsťahovanejšou ftp klientskou aplikáciou je AndFTP (obr. 6.5). Hlavnými dôvodmi sú jeho stabilita a to, že ako jediný poskytuje možnosť synchronizácie. Užívateľské prostredie je prehľadné a poskytuje dostatočný prehľad o jednotlivých položkách v zozname súborov. Pomohlo by však, aby sa aplikácia prispôsobovala rozmerom obrazovky. Podobne, ako aj v FtpCafe, je log zobrazený v úzkom okienku, čo sťažuje jeho použiteľnosť. Vo funkcionalite nechýbajú základné funkcie ako premenovanie a mazanie súborov a vytváranie nových priečinkov. V rámci dodatočných funkcií je podporovaná aj funkcia copy/paste na lokálnom zariadení. Tak, ako všetky spomínané aplikácie, okrem FtpCafe, prebiehajú prenosy iba na popredí aplikácie. Medzi dostupné typy pripojení patrí FTP, SFTP, FTPS a SCP.



Obr. 6.5: AndFTP

7. Záver

V tejto práci sme navrhli a vytvorili aplikáciu KarlFTP, ktorá funguje ako ftp klient pre mobilné zariadenia so systémom Android. Pri používaní a porovnávaní existujúcich aplikácií sme ich zhodnotili po funkčnej a užívateľskej stránke. Zistili sme tak nedostatky, ktoré určitým spôsobom obmedzovali ich fungovanie a použiteľnosť. Vymysleli sme preto vlastný program. V našom riešení sme implemtovali nový spôsob fungovania aplikácie, ktorý umožňuje vykonávať viacero operácií súčasne. Zároveň sme program vylepšili o doplnkové funkcie uľahčujúce užívateľovi ovládanie. Pri testovaní aplikácie sme zistili problémy, ako sú pomalý prenos dát a občasná nestabilita pripojenia, ktoré súvisia s kvalitou pripojenia zariadenia k internetu. Program by sa mohol v budúcnosti rozšíriť o podporu iných typov pripojenia ako su FTPS alebo SFTP. Aj napriek spomínaným problémom bola práca úspešná.

Zoznam použitej literatúry

[1] Burnette, Ed. *Hello, Android: Introducing Google's Mobile Development Platform*. 3. vydanie. Pragmatic Programmers, 2010. ISBN 978-1-934356-56-2.

[2] Android Development Guide 2014
`http://developer.android.com/`

[3] Postel, J. a Reynolds, J. *File Transfer Protocol*. RFC 959. Október 1985.

[4] Android Studio
`https://developer.android.com/sdk/installing/studio.html`

8. Prílohy

8.1 Užívateľská dokumentácia

Užívateľské prostredie aplikácie pozostáva zo štyroch základných častí. Každá z týchto častí poskytuje užívateľovi možnosť ako obsluhovať jednotlivé funkcie programu. Zároveň poskytujú prehľad o tom, čo aplikácia práve robí a v akom stave sa nachádza. Tieto časti sú:

- úvodná obrazovka
- obrazovka pre zadávanie parametrov pripojenia na server
- hlavná obrazovka aplikácie s prehliadačmi
- menu lišta

Pre správnu funkciu aplikácie je potrebné pripojenie na internet, najlepšie pomocou WiFi pripojenia.

8.1.1 Úvodná obrazovka

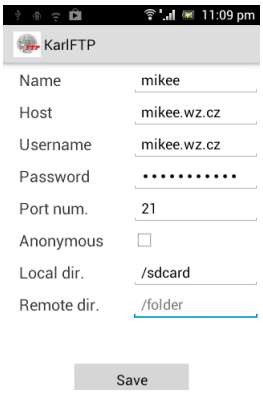
Po spustení aplikácie sa zobrazí úvodná obrazovka s logom (obr. 8.1). Tá slúži ako rozcestník pre prechod do hlavnej obrazovky, alebo do časti s nastaveniami pripojenia. Obsahuje štyri tlačidlá a jednu drop down lištu. Po kliknutí na lištu sa zobrazí zoznam s názvami uložených pripojení. Užívateľ si jeden zvolí (implicitne je vybraný prvý v zozname) a následne klikne na tlačidlo *Connect* a spustí sa hlavná obrazovka, alebo tlačidlo *Edit* a zobrazí sa obrazovka s uloženými parametrami pre vybrané pripojenie. Ak však klikne na tlačidlo *New*, dostane sa taktiež do parametrovej obrazovky, avšak s prázdnymi textovými poliami pre zadanie parametrov pre nové pripojenie. Posledným je tlačidlo *Delete*, ktoré má za úlohu zmazať vybrané pripojenie.

Obr. 8.1: Úvodná obrazovka

8.1.2 Obrazovka s parametrami pripojenia

Táto obrazovka (obr. 8.2) obsahuje textové polia, do ktorých užívateľ zadáva všetky údaje, ktoré sú potrebné pre pripojenie aplikácie k serveru. Tie pozostávajú

z: názvu pripojenia, adresy serveru, typu pripojenia, užívateľského mena a hesla. Ak chce užívateľ použiť anonymné prihlásenie, zaškrtne políčko *Anonymous* a meno s heslom zadávať nemusí. Ako voliteľné sú parametre s cestou k lokálnemu a vzdialenému priečinku, v ktorých sa aplikácia automaticky načíta po spustení. V spodnej časti sa nachádza už len tlačidlo *Save* pre uloženie zadaných parametrov a návrat späť na úvodnú obrazovku. Ak sa chce užívateľ vrátiť na úvodnú obrazovku bez uloženia pripojenia, stlačí systémovú šípku pre návrat späť.



Obr. 8.2: Obrazovka s parametrami pripojenia

8.1.3 Hlavná obrazovka

Po prejdení do hlavnej obrazovky musí užívateľ počkať, kým aplikácia oznámi, že boli pripojený obaja klienti(obrazky 8.3d, 8.3e) alebo nastal problém s pripojením a klientov sa pripojiť nepodarilo. Ak sa pripojenie nepodarilo, užívateľ môže skúsiť klientov znovu pripojiť pomocou tlačidla *Reconnect* v menu aplikácie. Hlavná obrazovka sa skladá z dvoch častí. Prvá časť zobrazuje manažéra pre súbory na lokálnom zariadení 8.3a a serveri 8.3b a tiež manažéra pre prenosovú frontu 8.3c. Toto zobrazenie je vo forme záložiek, kde každý manažér má vlastnú záložku. Užívateľ má dve možnosti ako medzi týmito záložkami prechádzať. Môže kliknúť priamo na záložku, alebo využit grafický komponent nazývaný swipe views. Slúži na to, aby užívateľ pomocou horizontálneho posunu prstom po obrazovke plynule prechádzal medzi jednotlivými záložkami. Tento princíp ovládania je bežný práve v prípadoch, ak sa obrazovka skladá z niekoľkých podobných častí, pričom zobrazená je len jedna.

Záložky pre lokálne zariadenie a server sa zobrazujú rovnakým spôsobom. Každá obsahuje skrolovací zoznam so súbormi, ktoré sa nachádzajú v tom priečinku, kde sa užívateľ práve nachádza. Cesta k tomuto priečinku je zobrazená v textovom poli nad zoznamom. Vľavo od tohto poľa sa nachádzajú tlačidlá pre návrat do domovského adresára (tlačidlo 1) a pre návrat o úroveň vyššie v súborovom strome (tlačidlo 2). Každá položka v zozname charakterizuje jeden súbor. Obsahuje zaškrtávacie okienko pre zvolenie súboru, ikonu zobrazujúcu typ súboru, názov súboru a tlačidlo 3 pre zobrazenie okna s dodatočnými informáciami o danom súbore, ako sú jeho veľkosť a čas poslednej modifikácie.

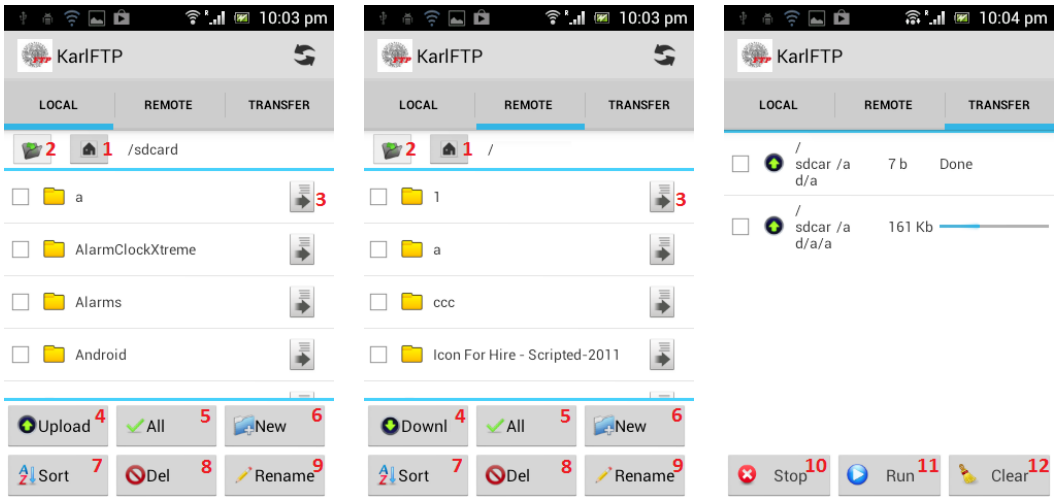
Záložka pre prenosovú frontu je taktiež vo forme skrolovacieho zoznamu. Tento zoznam obsahuje informácie o každom procese, ktorý sa nachádza v prenosovej fronte. Každá položka obsahuje po poradí zaškrtávacie okienko, obrázok znázorňujúci typ operácie (sťahovanie, posielanie, kopírovanie alebo synchronizácia),

textové polia zobrazujúce cestu k zdrojovému a cieľovému priečinku a stav operácie (čaká, počíta, vykonáva, zastavená). Taktiež sa tam nachádza progress bar, ktorý zobrazuje, aká časť operácie sa vykonala.

Druhou časťou sú tlačidlá pre základné operácie so súbormi. Tie sa líšia podľa toho, v ktorej záložke sa užívateľ nachádza. Pre záložky s manažermi obsluhujúcimi zariadenie a server sú to:

- tlačidlo 4 - spustenie operácie sťahovania/odosielania
- tlačidlo 5 - výber všetých zobrazených súborov a priečinkov
- tlačidlo 6 - vytvorenie novéhu priečinku
- tlačidlo 7 - zmena poradia triedenia (vzostupne/zostupne)
- tlačidlo 8 - vymazanie vybraných súborov a priečinkov
- tlačidlo 9 - premenovanie jedného zvoleného súboru alebo priečinku

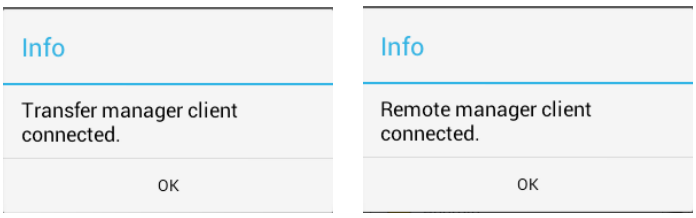
Pre záložku s prenosovou frontou sú to dva tlačidlá a to tlačidlo 10 pre zastavenie, 11 pre nové spustenie vybranej operácie a tlačidlo 12 pre vymazanie vybraných položiek zo zoznamu. Hlavné okno má špeciálnu vlastnosť, že po otočení zaria-



(a) Obrazovka lokálneho manažéra

(b) Obrazovka vzdialeného manažéra

(c) Obrazovka prenosového manažéra

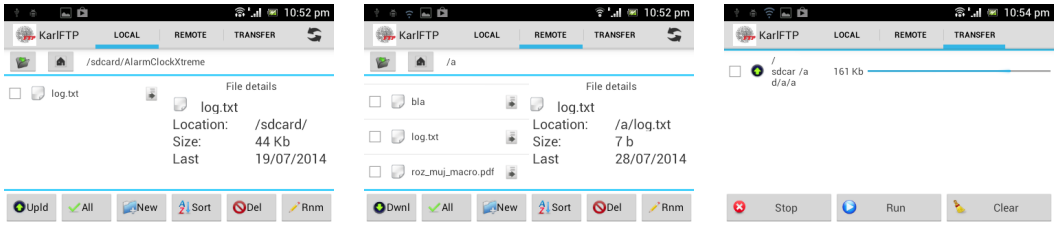


(d) Transfer klient pripojený

(e) Remote klient pripojeny

Obr. 8.3: Časti hlavnej obrazovky

denia na šírku 8.4 sa mierne zmení rozloženie grafických komponentov. Okno s dodatočnými informáciami o súbore je pevne ukotvené vpravo do zoznamu súborov, avšak je prázdne. Ak však užívateľ klikne na tlačidlo 3, údaje o konkrétnom súbore sa tam doplnia.



(a) Lokálny manažér na šírku

(b) Vzdialený manažér na šírku

(c) Prenosový manažér na šírku

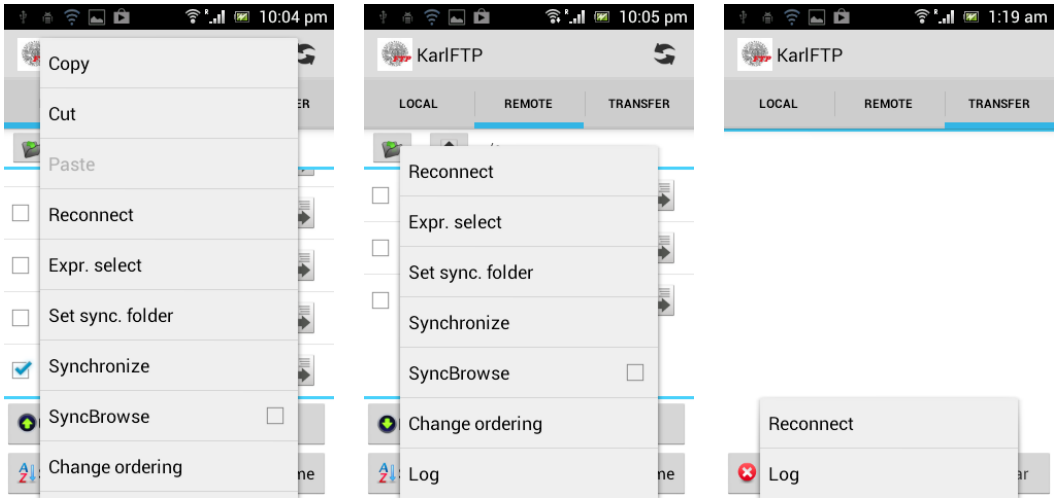
Obr. 8.4: Časti hlavnej obrazovky na šírku

8.1.4 Menu lišta hlavnej obrazovky

Menu je špecifické podľa toho, v ktorej časti hlavnej obrazovky sa užívateľ nachádza. Pre časť s lokálnym 8.5a a vzdialeným 8.5b manažérom vyzerá nasledovne. Prvou položkou je tlačidlo pre obnovenie zoznamu súborov v hlavnej obrazovke. Otvorenie menu je možné dvoma spôsobmi a záleží na type zariadenia. Na zariadeniach, ktoré majú systémové menu tlačidlo, sa menu ponuka zobrazí po jeho stlačení. Ak však toto tlačidlo nemá, je v aplikácii v pravom hornom rohu zobrazené menu tlačidlo vo forme obrázka s tromi bodkami, ktoré má rovnakú funkciu ako chýbajúce systémové. V zobrazenom menu sa nachádza desať položiek reprezentujúcich jednotlivé funkcie aplikácie.

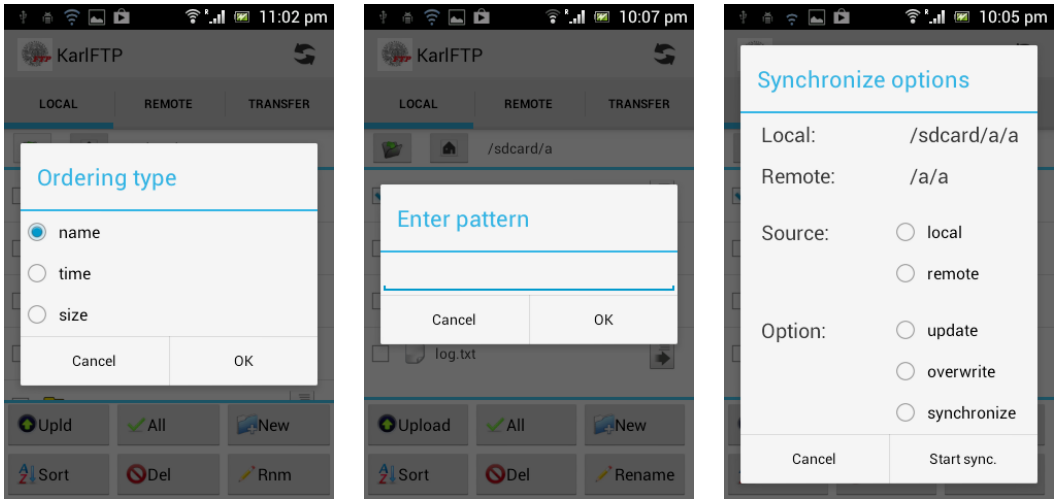
Zhora začínajú tlačidlá *Copy* a *Cut* a tlačidlo *Paste*, ktoré je štandardne zablokované. Pomocou prvých dvoch užívateľ vybrané súbory alebo priečinky zvolí pre kopírovanie alebo vystrihnutie, pričom sa odblokuje tlačidlo *Paste*. Následne pomocou neho užívateľ vloží označené súbory/priečinky na požadované umiestnenie. Tieto tri tlačidlá sa nachádzajú len v časti s lokálnym manažérom. Ďalšou položkou v menu je tlačidlo *Reconnect* pre opätovné pripojenie pri výpadku spojenia. Nasleduje tlačidlo pre výber súborov podľa vzoru *Expr. select*. Po jeho stlačení sa užívateľovi zobrazí dialógove okno 8.6b, v ktorom zadá požadovaný vzor a voľbu potvrdí (tlačidlo *Ok*) alebo zamietne (tlačidlo *Cancel*). Následne sa nájdené položky v zozname súborov označia. Ako vzor sa očakáva regulárny výraz.

Tlačidlo *Set sync. folder* slúži na zvolenie priečinku, ktorý sa bude synchronizovať na zariadení a na serveri. Pre túto operáciu je možné vybrať práve jeden priečinok na serveri a práve jeden na zariadení. Ak užívateľ vyberie viac, žiaden alebo súbor, aplikácia ho upozorní na zlú voľbu. Po zvolení oboch prienčinkov na synchronizovanie sa v menu odblokuje tlačidlo *Synchronize* pre spustenie synchronizácie. Po stlačení sa zobrazí dialógove okno 8.6c pre zadanie dodatočných nastavení. Tie zahŕňajú voľbu zdroja (*local* alebo *remote*) a typ synchronizácie (*update*/*overwrite*/*synchronize*). Po nastavení týchto parametrov užívateľ potvrdí spustenie synchronizácie (tlačidlo *Start*), alebo sa vráti do hlavnej obrazovky (tlačidlo *Cancel*). Nasleduje tlačidlo *SyncBrowse* pre spustenie funkcie synchronizovaného prehliadania, tlačidlo *Change ordering* pre zobrazenie typu triedenia 8.6a a *Log* tlačidlo slúžiace na zobrazenie logu pripojenia. Ak sa užívateľ na hlavnej obrazovke nachádza v záložke s prenosovým manažérom 8.5c, menu obsahuje tlačidlá *Reconnect* a *Log*.



(a) Menu lokálneho manažéra (b) Menu vzdialeného manažéra (c) Menu prenosového manažéra

Obr. 8.5: Menu pre jednotlivé manažére



(a) Nastavenie typu triedenia (b) Zadávanie regulárneho vzoru (c) Nastavenie synchronizácie

Obr. 8.6: Dialógové okná spúšťané v menu

8.2 Inštalácia aplikácie

Prílohou práce je prenosové médium s našou aplikáciou **KarlFTP**. Pre inštaláciu na zariadenie musia byť splnené nasledovné podmienky:

- nainštalovaný súborový manažér, doporučuje sa **Total Commander** inštalovaný pomocou **Google Play** aplikácie
- povolená položka Unknown sources, ktorá sa nachádza v Settings -> Security

Postup inštalácie:

1. stiahnuť inštalačný balíček **KarlFTP.apk** z prenosového média do zariadenia do vybraného priečinku

2. v zariadení pomocou aplikácie **Total Commander** otvoriť **KarlFTP.apk**
3. po vyzvaní potvrdiť inštaláciu
4. po nainštalovaní otvoriť **KarlFTP** v zozname všetkých nainštalovaných aplikácií

8.3 Obsah priloženého CD

V priloženom CD sa nachádza:

- inštalačný balíček s aplikáciou
- zdrojový kód aplikácie
- PDF verzia bakalárskej práce