Česká zemědělská univerzita v Praze

Provozně ekonomická fakulta

Katedra informačních technologií

Diplomová práce

Mobilní aplikace pro operační systém Android

Bc. Hlubuček Lukáš

Zadání

Čestné prohlášení

Prohlašuji, že svou diplomovou práci Mobilní aplikace pro operační systém Android jsem vypracoval samostatně pod vedením vedoucího diplomové práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou citovány v práci a uvedeny v seznamu literatury na konci práce. Jako autor uvedené diplomové práce dále prohlašuji, že jsem v souvislosti s jejím vytvořením neporušil autorská práva třetích osob.

V Praze dne 31.3.2015 odevzdání _____

Poděkování

Rád bych touto cestou poděkoval Ing. Jiřímu Vaňkovi, Ph.D., za vedení této práce.

Mobilní aplikace pro operační systém Android

Mobile applications for the Android operating systém

Souhrn

Tato diplomová práce je zaměřena na problematiku vývoje mobilních aplikací pro vybraný mobilní operační systém Android. Hlavním cílem práce je analýza a porovnání možností vývoje aplikací pro vybraný operační systém.

Dílčími cíli jsou analýza vybraných komponent, které jsou součástí operačního systému, a implementace zvolených komponent.

V závěru jsou vybrané kritické situace shrnuty a rozšířeny o autorovo doporučení, jak postupovat při řešení.

Summary

This thesis is focused on the mobile applications development for the chosen mobile operating system Android. The main aim of this thesis is the analysis and comparison of development possibilities for the chosen operating system.

The partial aims are the analysis of the chosen components, which are a part of the operating system, and implementation of the chosen components.

In the conclusion, there are chosen critical situations that are summarised and extended with the author's recommendations how to proceed during the solving.

Klíčová slova: Android, Java, Google, mobilní zařízení, ADT, Google Play

Keywords: Android, Java, Google, mobile devices, ADT, Google Play

Obsah

1 Úvod....9
2 Cíl práce a metodika....10
2.1 Cíl práce....10
2.2 Metodika práce....10
3 Přehled řešené problematiky....11
3.1 Architektura systému....11
3.1.1 Linux kernel....11
3.1.2 Libraries....12
3.1.3 Android Runtime....12
3.1.4 Application Framework....12
3.1.5 Aplikace....13
3.2 Vývojové prostředky....13
3.2.1 IDE....13
3.2.1.1 Eclipse....13
3.2.1.2 Android Studio....13
3.2.2 SDK....14
3.3 Komponenty aplikace....15
3.3.1 Activity....15
3.3.1.1 Životní cyklus activity....16
3.3.2 Service....18
3.3.2.1 Životní cyklus service....18
3.3.3 Content provider....19
3.3.4 Broadcast receiver....19
3.4 Struktura projektu....19
3.4.1 src....20
3.4.2 res....20
3.4.2.1 Složky drawable....20
3.4.2.2 Složka layout....20
3.4.2.3 Složka values....21
3.4.2.4 Složka values-v11 a values-v14....21
3.4.3 AndroidManifest.xml....22

3.5 Uživatelské rozhraní....22
3.5.1 Vybrané layouty....22
3.5.1.1 Linear Layout....23
3.5.1.2 Relative Layout....23
3.5.1.3 Table Layout....24
3.5.2 Vybrané grafické komponenty....25
3.5.2.1 TextView....25
3.5.2.2 EditText....25
3.5.2.3 Button....26
3.5.2.4 CheckBox....26
3.5.2.5 RadioButton....27
3.5.2.6 Switch, ToggleButton....27
3.5.2.7 ImageView....28
3.5.3 Status bar....28
3.5.4 Notifikace....28
3.5.5 Navigation Bar....28
3.5.6 Action Bar....29
3.5.6.1 Ikona aplikace....29
3.5.6.2 View Control....29
3.5.6.3 Action Buttons....29
3.5.6.4 Action Overflow....29
3.5.7 Navigation Drawer....29
3.5.8 Content Area....30
3.5.9 Devices and Displays....30
3.5.10 Themes....30
3.5.11 App Structure....30
3.5.12 Multi-pane Layouts....31
3.5.13 Navigation....32
3.5.14 Pure Android....33
4 Vlastní práce....34
4.1 Srovnání activity a fragmentu....34
4.1.1 Porovnání AndroidManifest.xml....34
4.1.2 Porovnání MainActivity.java....35
4.1.3 Porovnání OtherUI.java a FragmentOtherUI.java....38

4.1.4 Porovnání activity_main.xml....40
4.1.5 Porovnání ostatních souborů....41
4.2 Ukázka vícepanelového rozložení....44
4.3 Návrhové vzory....45
4.3.1 Singleton....46
4.3.2 Observer....46
4.3.3 Fasáda....47
4.3.4 Dekorátor....48
4.4 Publikování a správa aplikace....49
4.4.1 Registrace vývojářského účtu....49
4.4.2 Vývojářská konzole....50
5 Zhodnocení výsledků a doporučení....51
6 Závěr....53
7 Seznam použité zdrojů....55
8 Seznam obrázků....58
9 Seznam pojmů....59

1 Úvod

V dnešní době mobilní telefon vlastní velká část populace celé planety. A protože vše prochází nějakým vývojovým stádiem, tak i obyčejné telefony, které uměly volat, posílat a přijímat SMS zprávy, vystřídaly telefony, které dostaly nové funkce a vlastnosti, jako jsou barevné displeje, digitální fotoaparáty, možnost přehrávat audio nebo video soubory.

Dnešní uživatelé nepoužívají svůj mobilní telefon jako uživatel z minulého desetiletí. Pro dnešního uživatele může mobilní telefon být jak módní doplněk, tak náhrada za osobní počítač. Mobilní telefon mnohdy nahradil různé kalendáře. Pomocí mobilních operátorů se uživatelé také mohou připojit na internet, a tak je využívat jako e-mailové klienty nebo pro zobrazení internetových stránek a sociálních sítí.

Mobilní telefony se používají při stále více příležitostech a aktivitách v každodenním životě – jako čtečka elektronických knih, navigace, osobní sport tester. Z těchto důvodů mobilní telefony zvyšují svou výkonnost, která přesahuje výkonnost prvních osobních počítačů, ale stejně rychle se zároveň vyvíjí mnoho aplikací, které uživatelé instalují do svých zařízení pro uspokojení těchto aktivit.

Každý úspěšný operační systém potřebuje mít velké množství aplikací, které si jeho uživatelé mohou nainstalovat. Mobilní operační systém Android tuto podmínku jednoznačně splňuje, protože v je obchodu s aplikacemi Google Play je v tuto chvíli k dispozici přes 1 500 000 aplikací.[1]

1 Number of Android applications. *AppBrain* [online]. 2015 [cit. 2015-03-22]. Dostupné z: http://www.appbrain.com/stats/number-of-android-apps

2 Cíl práce a metodika

2.1 Cíl práce

Diplomová práce je zaměřena na problematiku vývoje aplikací pro operační systém Android. Hlavním cílem je analyzovat architekturu a komponenty mobilního operačního systému Android. Dílčími cíli jsou:

- ukázat praktickou implementaci vybraných komponent operačního systému několika způsoby v programovacím jazyce Java,
- analyzovat a zhodnotit jednotlivé implementace a vysvětlit situace, pro které je implementace vhodná,
- popsat průběh publikace a update aplikace pomocí služby Google Play,
- ukázka vybraných návrhových vzorů, které se dají uplatnit při vývoji jakýchkoliv aplikací.

2.2 Metodika práce

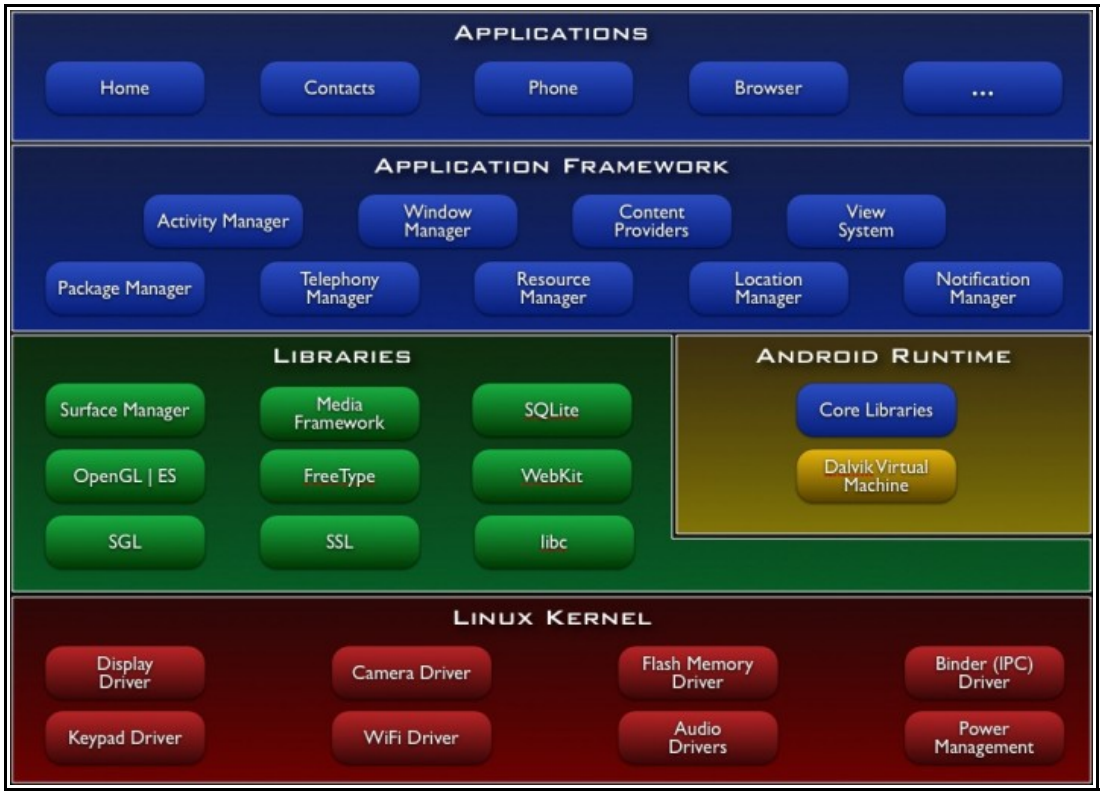
Metodika diplomové práce je založena na studiu a analýze odborných zdrojů, dokumentace, platformy mobilního operačního systému Android a také příkladů implementace vybraných komponent a návrhových vzorů, které se dají uplatnit při vývoji mobilních aplikací.

Na základě těchto zdrojů a vlastních zkušeností bude vytvořena aplikace pro mobilní operační systém Android, na které se budou analyzovat rozdílné formy vývoje aplikací pro operační systém Android.

Pro formulování závěrů diplomové práce byla použita syntéza praktických a teoretických poznatků, nabytých při tvorbě této diplomové práce.

3 Přehled řešené problematiky

3.1 Architektura systému



Obrázek 1: Architektura operačního systému Android zdroj: http://download.e-bookshelf.de/download/0000/5920/58/L-X-0000592058-0001352109.XHTML/index.xhtml

Mobilní operační systém Android je rozdělen do pěti vrstev. Každá vrstva má svůj účel a obstarává určitou část, vrstvy od sebe nemusí být odděleny.

3.1.1 Linux kernel

Tato vrstva obstarává komunikaci softwaru a hardwaru. V jádře jsou zabudovány ovladače, které slouží právě pro tuto komunikaci. Dále má na starosti správu paměti, napájení a řízení procesů.

Linuxové jádro bylo zvoleno z důvodu jednoduchého sestavení, přenositelnosti mezi zařízeními a možnosti modifikovatelnosti.

3.1.2 Libraries

V této vrstvě jsou uloženy základní komponenty celého systému Androidu. Jsou zde knihovny, které obsluhují různé části systému:

- WebKit – pro vykreslování webového obsahu,
- SQLite – knihovna, která spravuje relační databázový stroj,
- FreeType – knihovna se používá pro vykreslování textových fontů,
- Media Libraries – přehrávání audio a video souborů nebo pro zobrazení obrázkových souborů,
- SSL – zajišťuje šifrovanou komunikaci.

3.1.3 Android Runtime

V této vrstvě běží virtuální stroj Dalvik odvozený z Java Virtual Machine. Dalvik Virtual Machine využívá vlastností linuxového jádra, které zajišťuje například správu paměti, vláken atd. Byl vytvořen ze dvou hlavních důvodů.

Prvním důvodem je, že Java Virtual Machine není volně šiřitelný software, druhým optimalizace pro mobilní zařízení, a to hlavně zlepšení energetické náročnosti a výkonu. Jsou zde také uloženy základní Java knihovny, které jsou rozšířeny o knihovny Apache kvůli práci se sítí.

Všechny zdrojové kódy aplikací pro Android jsou nejprve přeloženy do Java byte kódu a následně se ještě jednou přeloží pomocí Dalvik kompilátoru do Dalvik byte, spustitelném v Dalvik Virtual Machine.

3.1.4 Application Framework

V této vrstvě jsou uloženy Java knihovny, které zpřístupní grafické prvky, data jiných aplikací, například kontakty nebo obsah galerie, stavový řádek nebo aplikace běžící na pozadí.

3.1.5 Aplikace

Nejvyšší vrstvu operačního systému tvoří uživatelské aplikace. V Android zařízeních je mnoho aplikací předinstalováno výrobci nebo se dají doinstalovat z Google Play.[2] Další možností, jak přidat novou aplikaci do svého zařízení, je naprogramovat si ji.

3.2 Vývojové prostředky

3.2.1 IDE

Je software, díky kterému programátor může jednodušeji pracovat se zdrojovými kódy aplikací, které tvoří. Ve většině používaných IDE programů je vestavěný našeptávač, který dokončuje rozepsaný příkaz nebo interpretuje příkazy různých verzovacích nástrojů, jako je GIT, SVN, nebo vývojáři umožňuje jednoduše spouštět nebo krokovat výsledný program.

Pro vývoj aplikací pro mobilní operační systém Android se používají vývojové nástroje Eclipse nebo Android Studio, do kterého se musí stáhnout Android Development Tool (ADT), které tvoří SDK pro Android.[3]

3.2.1.1 Eclipse

Tento open source vývojový nástroj je postavený na Java platformě. Základní balík obsahuje nástroje pro vývoj v Javě, ale pomocí mnoha modulů se zde vyvíjet v PHP, C++ a v mnoha dalších jazycích.[4]

Pomocí ADT Bundle, který tvoří SDK pro Android, se dá v Eclipse vyvíjet pro tento mobilní operační systém.

3.2.1.2 Android Studio

Druhým nástrojem, který je oficiálně podporovaný společností Google pro vývoj

2 Jak vypadá Android uvnitř?. *ANDROIDMARKET* [online]. 2011 [cit. 2014-08-05]. Dostupné z: http://www.androidmarket.cz/android/jak-vypada-android-uvnitr-aneb-co-je-rom-kernel- bootloader-a-dalsi/

3 Ide. *Abc lunuxu* [online]. 2006 [cit. 2014-08-05]. Dostupné z: http://www.abclinuxu.cz/slovnik/ide

4 Eclipse. *Root.cz* [online]. 2002 [cit. 2014-08-05]. Dostupné z: http://www.root.cz/clanky/eclipse-2-ide-na-vsechno/

aplikací pro operační systém Android, je Android Studio. Vývojový nástroj je postavený na Java platformě a existuje ve dvou verzích. První je komerční a druhá je komunitní open source. Na této verzi je postavené prostředí Android Studio.[5]

Tento nástroj je vyvíjen společností JetBrains, opět po stažení Android SDK je možné v komunitní verzi vyvíjet aplikace pro operační systém Android.

3.2.2 SDK

Android SDK je složený z komponent, které slouží pro vytváření, ladění, exportování a podepsání vytvořených aplikací.

- SDK Tools

V této komponentě jsou obsaženy nástroje pro ladění aplikace. Spravují se zde Android Virtual Devices, které vytváří a spravují emulátory Android zařízení. Jsou zde nástroje pro nahrávání vytvořených aplikací jak do emulovaných zařízení, tak do fyzických zařízení.[6]

- ADT Plugin

Tvoří nástroje pro rychlé vytvoření a nastavení projektu. Pro exportování a zabalení aplikace do balíčku .apk. Grafický a textový editor pro tvorbu uživatelského rozhraní.[7]

- Build Tools

Je součástí Android SDK, používá se pro build kódu Android aplikace. Nejnovější verze těchto nástrojů je součástí staženého balíčku SDK a instalován v <SDK> / build-tools / adresáře.

Používá se pro build, debug, spuštění a testování Android aplikace. Tento nástroj připojí zařízení jako superuživatel. Také umožňuje spustit další nástroje jako ProGuard, zipalign.[8]

5 Android Studio. *Android Developers* [online]. 2014 [cit. 2014-08-05]. Dostupné z: https://developer.android.com/sdk/installing/studio.html
6 Tools Help. *Android Developers* [online]. 2014 [cit. 2014-08-05]. Dostupné z: http://developer.android.com/tools/help/index.html
7 ADT Plugin. *Android Developers* [online]. 2014 [cit. 2014-08-05]. Dostupné z: http://developer.android.com/tools/sdk/eclipse-adt.html
8 SDK Build Tools. *Android Developers* [online]. 2014 [cit. 2015-03-02]. Dostupné z: https://developer.android.com/tools/revisions/build-tools.html

ProGuard je nástroj, který zmenšuje velikost výsledného APK souboru. Zmenšení je dosaženo tím, že se odstraní kód, který se nevykonává a odstraní komentáře, také přejmenuje třídy a proměnné. Výhoda je, že výsledný soubor je menší, ale také je mnohem odolnější proti reverznímu inženýrství.[9]

Zipalign je nástroj, který upravuje vnitřní adresářovou strukturu APK souboru. Všechny nekomprimované soubory jsou zařazeny na relativní začátek APK souboru. Výsledkem je, že aplikace zabírá méně operační paměti.[10]

- Platforms

V této komponentě jsou uloženy informace o verzi Android. Pro sestavení aplikace pro konkrétní verzi Androidu je nutné stáhnout a nainstalovat SDK platformu pro tuto verzi. Každá verze zahrnuje systémové obrazy pro specifickou architekturu procesoru. Platformy také obsahují bitovou kopii systému, který obsahuje Google API. V SDK manažeru je k dispozici obraz systému pro většinu verzí Androidu.

Pokud je potřeba otestovat aplikaci v jiném zařízení, než které fyzicky vlastníme, je potřeba stáhnout alespoň jeden systémový obraz mobilního operačního systému Android.[11]

3.3 Komponenty aplikace

Každá aplikace pro operační systém Android je složená ze čtyř komponent, a to activity, service, broadcast receiver, content provider.

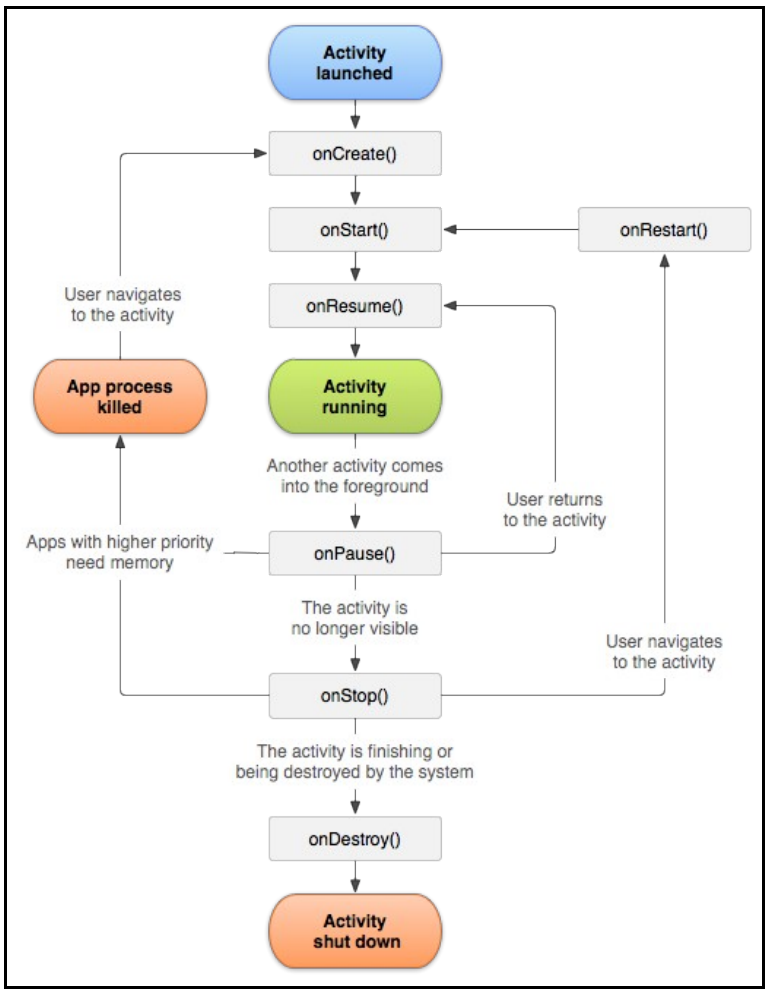
3.3.1 Activity

Odpovídá jedné obrazovce aplikace. V activity je vložené grafické uživatelské rozhraní. Při startu activity se vytvoří nový proces a alokuje se paměť pro prvky UI.

Všechny activities spravuje Activity Manager. Pracuje jako zásobník a ukládá informace o spuštěných activities. Na vrchu je uložena právě aktivní activity.[12]

9 ProGuard. *Android Developers* [online]. 2014 [cit. 2015-03-02]. Dostupné z: https://developer.android.com/tools/help/proguard.html
10 Zipalign. *Android Developers* [online]. 2015 [cit. 2015-03-02]. Dostupné z: https://developer.android.com/tools/help/zipalign.html
11 Platforms. *Android Developers* [online]. 2014 [cit. 2014-08-05]. Dostupné z: http://developer.android.com/tools/revisions/platforms.html
12 Activities. *Android Developers* [online]. 2014 [cit. 2014-08-05]. Dostupné z: http://developer.android.com/guide/components/activities.html

3.3.1.1 Životní cyklus activity



Obrázek 2: Životní cyklus activity zdroj: http://developer.android.com/reference/android/app/Activity.html#ActivityLifecycle

- onCreate()

Je to první metoda, která se používá pro ošetření životního cyklu aplikace. Volá se po vytvoření activity, a proto se zde můžou vytvořit všechny prvky grafického rozhraní, které potřebujeme pro interakci s uživatelem, nebo nastavit všechna statická data, která chceme, aby uživatel viděl.

Tato metoda umí načíst data z objektu, který má v sobě uložené hodnoty z předchozího používání.

- onStart()

Tato metoda se volá, když se aplikace vykresluje na displeji.

Pokud se aplikace zobrazí celá, pokračuje volání metody onResume(), když aplikace zmizí, zavolá se metoda onStop().

- onResume()

Metoda se zavolá, když je aplikace připravená, aby s ní uživatel začal pracovat. Po zavolání této metody je activity na první pozici activity zásobníku.

- onPause()

Metoda je volána pokaždé, když operační systém začne vytvářet novou nebo obnovovat nějakou dříve spuštěnou activity. Metoda slouží pro uložení dat, která chceme, aby se opět načetla při opětovném spuštění naší aplikace. Také je vhodné zastavit animace a vše, co by mohlo zatěžovat CPU přístroje. Metoda by měla proběhnout co nejrychleji, protože další activity nemůže začít, dokud onPause() neskončí.

Další volanou metodou je onResume(), pokud se activity znovu načte, například u otočení displeje, nebo onStop(), když se activity stane neviditelnou pro uživatele.

- OnStop()

Tuto metodu operační systém volá, pokud activity není dlouho zobrazena v důsledku toho, že ji jiná activity překryla.

Také může být zavolána, když je nově vytvořená activity přenášena do popředí nebo když má být tato activity zničena.

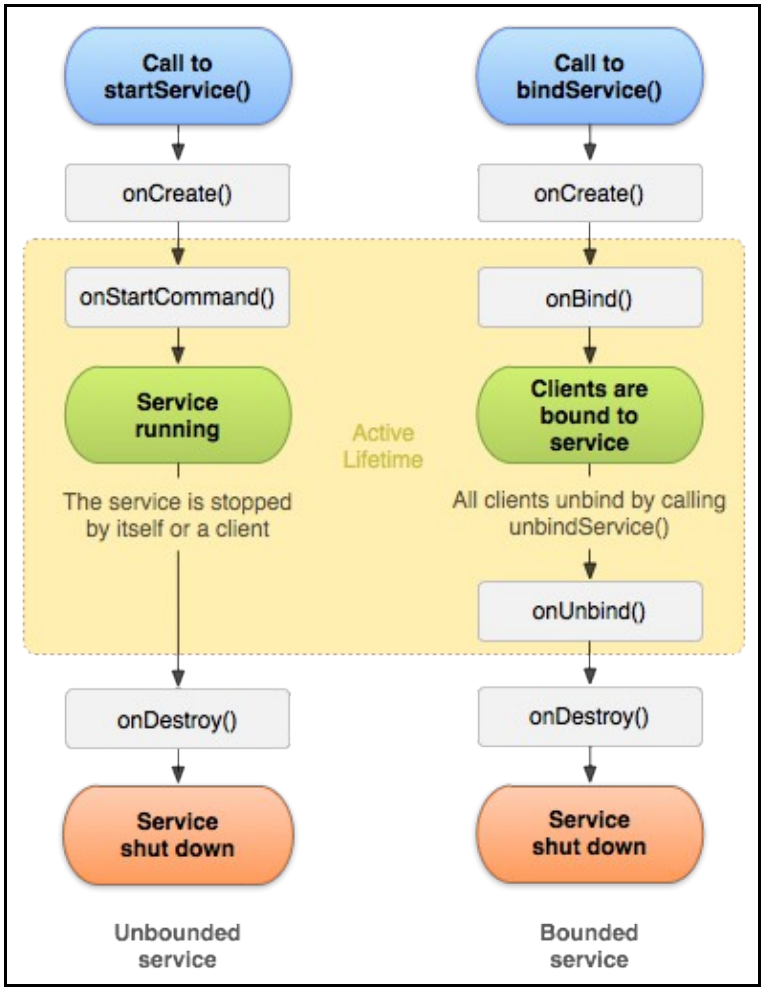
Z této metody může operační systém zavolat onRestart(), v případě další interakce uživatele, nebo je voláno onDestroy() a activity je zničena.

- onDestroy()

Tato metoda je volána tehdy, když programátor zavolá metodu finish() pro danou aktivity nebo když operační systém potřebuje dočasně uvolnit prostředky pro další práci systému.[13]

13 Activity. *Android Developers* [online]. 2014 [cit. 2014-08-05]. Dostupné z: http://developer.android.com/reference/android/app/Activity.html#ActivityLifecycle

3.3.2 Service



Obrázek 3: Životní cyklus service zdroj: http://developer.android.com/guide/components/services.html

Tato komponenta neposkytuje žádné uživatelské rozhraní. Používá se pro dlouho běžící procesy aplikace, které se vykonávají na pozadí v určitých časových smyčkách.

Spuštění service může proběhnout dvěma způsoby: voláním metody startService(), nebo bindService(). Takto spuštěný service může zastavit pouze komponenta, která service vytvořila. Této komponentě se říká klient.

3.3.2.1 Životní cyklus service

I tato komponenta má svůj životní cyklus.

- onCreate() a onDestroy()

Tyto dvě metody jsou volány pokaždé – nezáleží na tom, zda byl service vytvořený startService(), nebo bindService().

- onStartCommand()

Tato metoda je volána, pokud programátor použil startService(). Spustí se nevázaný service.

- onBind()

Metoda je volána pro připojení k vázanému service. Tento service může ukončit pouze komponenta, která jej spustila.

- onUnbind()

Tato metoda se zavolá tehdy, když aplikace už nepotřebuje daný service a odpojí se od něj.[14]

3.3.3 Content provider

Tato komponenta opět nemá uživatelské rozhraní. Používá se pro sdílení dat mezi aplikacemi. K těmto úlohám se používají základní databázové metody (update, insert, delete a query). Protože data nemusí být součástí uživatelského prostředí, výchozí aplikace se dají snadno nahradit nově vytvořenými.[15]

3.3.4 Broadcast receiver

Stejně jako service, tak i broadcast receiver nemá uživatelské rozhraní. Komponenta se využívá pro notifikaci uživatele na stavovém řádku nebo pro spuštění jiné aplikace, například fotoaparátu.[16]

3.4 Struktura projektu

V každém IDE pro vývoj aplikací pro Android jsou cesty k složkám trochu jiné, ale

14 Services. *Android Developers* [online]. 2014 [cit. 2014-08-05]. Dostupné z: http://developer.android.com/guide/components/services.html
15 Vyvíjíme pro Android: Content providery. *Zdroják.cz* [online]. 2012 [cit. 2014-08-05]. Dostupné z: http://www.zdrojak.cz/clanky/vyvijime-pro-android-content-providery/
16 BreadcastReceiver. *Android Developers* [online]. 2014 [cit. 2014-08-05]. Dostupné z: http://developer.android.com/reference/android/content/BroadcastReceiver.html

všechny hlavní složky se jmenují stejně.

3.4.1 src

V této složce jsou uloženy všechny zdrojové kódy, které aplikace pro svoji práci potřebuje.

3.4.2 res

Tato složka má v sobě vnořené další složky, do kterých jsou uloženy soubory pro grafiku, uživatelské rozhraní.

3.4.2.1 Složky drawable

Tyto složky jsou určeny pro uložení grafických souborů, které jsou součástí aplikace. Složka je v několika verzích; ve všech by se měly nacházet stejné obrázky, ale v rozdílných rozlišeních. To je způsobené tím, že operační systém Android pracuje na rozdílně velkých zařízeních.

3.4.2.2 Složka layout

V této složce jsou uloženy XML soubory, ve kterých jsou uložená jednotlivá uživatelská rozhraní pro jednotlivé activities. Složka může být opět ve více verzích.

- layout

  Zde se ukládá rozhraní pro uživatelské rozhraní, když telefon držíme na výšku.

- layout-land

  Slouží pro uložení uživatelského rozhraní po překlopení telefonu na šířku. Vývojář může změnit rozložení prvků (například z 2×3 na 3×2), protože jinak by systém klasický layout otočil o 90°.

- layout-large

  V této složce jsou uložená rozložení pro telefony s velkým displejem a tablety.

Vývojář může všechny uživatelské prvky vytvořit větší nebo jinak rozložené.[17]

3.4.2.3 Složka values

Složka slouží pro uložení tří XML souborů.

- dimens

Používá se pro definici velikosti skupin grafických prvků nebo textových odstavců. Pro definování se používají tyto jednotky: dp, sp, pt, px, mm, in. Jednotky dp a sp jsou relativního charakteru; pro vypočítání velikosti je potřeba znát rozlišení nebo velikost displeje. Jednotky pt, px, mm, in jsou absolutního charakteru, pt je 1/72 palce. Jeden pixel je jeden zobrazovací bod displeje.[18]

- strings

Při programování pro Android je zvykem všechny textové řetězce ukládat do externího souboru. Tento přístup je dobrý pro zjednodušení vývoje jazykových mutací aplikace.[19]

- styles

Tento soubor zjednodušuje práci vývojáři, protože nemusí neustále opakovat stejné atributy jednotlivých grafických prvků pro uživatelské rozhraní v activity.[20]

3.4.2.4 Složka values-v11 a values-v14

Operační systém Android existuje v mnoha verzích. U některých verzí api může dojít k nefunkčnosti některých grafických prvků, protože dané api je nepodporuje. Proto jsou v těchto složkách uloženy resources pro dané verze ve v11 (pro verzi api 11 a starší) a ve složce v14 jsou uložena data pro verzi api 14 a víc.[21]

17 Managing Projects. *Android Developers* [online]. 2014 [cit. 2014-08-05]. Dostupné z: http://developer.android.com/tools/projects/index.html
18 More Resource Types. *Android Developers* [online]. 2014 [cit. 2014-08-05]. Dostupné z: http://developer.android.com/guide/topics/resources/more-resources.html#Dimension
19 String Resources. *Android Developers* [online]. 2014 [cit. 2014-08-05]. Dostupné z: http://developer.android.com/guide/topics/resources/string-resource.html
20 Style Resource. *Android Developers* [online]. 2014 [cit. 2014-08-05]. Dostupné z: http://developer.android.com/guide/topics/resources/style-resource.html
21 Styles and themes on values, values-v11 and values-v14 folders. *Stackoverflow* [online]. 2013 [cit. 2014-08-06]. Dostupné z: http://stackoverflow.com/questions/16624317/styles-and-themes- on-values-values-

3.4.3 AndroidManifest.xml

V kořenovém adresáři aplikace se musí nacházet AndroidManifest.xml soubor, kde jsou uloženy základní informace o aplikaci.

Soubor obsahuje jméno Java package (balíku) aplikace a unikátní jméno aplikace. Popisuje activity, service, broadcast receiver a content provider, které jsou součástí aplikace. Také jsou zde vytvořeny intent filtry, které popisují, jaké intenty mohou obsloužit. Díky této deklaraci se operační systém naučí, za jakých podmínek může být daná komponenta spuštěna.

V manifestu jsou uvedena všechna práva pro aplikaci, u kterých je třeba získat souhlas uživatele. Také je zde uložena informace o minimální verzi API, pro kterou je aplikace vytvořená.[22]

3.5 Uživatelské rozhraní

Je reprezentováno všemi grafickými prvky, pomocí kterých aplikace zobrazuje uživateli informace nebo uživatel provádí interakci s aplikací.

Implementace vybraných prvků uživatelského rozhraní jsou provedeny v následující kapitole.

Aplikace pro operační systém Android se dají vytvořit se specifickým vzhledem pro telefony, tablety nebo nově i hodinky.

3.5.1 Vybrané layouty

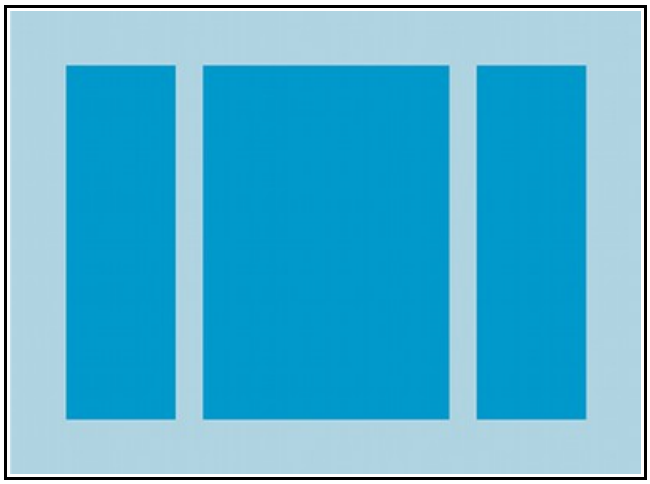
Layout se dá ve stručnosti popsat jako kontejner, do kterého se promítnou nadefinované grafické prvky uživatelského rozhraní. Celé uživatelské rozhraní se dá nadefinovat pomocí dvou způsobů, a to deklarací elementů do XML souboru, nebo programově vytvořením instance třídy grafické komponenty, kterou chceme vytvořit. Deklarace v XML souboru má jednoznačně výhodu v tom, že můžeme vytvořit rozdílná uživatelská rozhraní pro různé velikosti nebo polohy displeje. Další výhodou je oddělení

v11-and-values-v14-folders

22 App Manifest. *Android Developers* [online]. 2014 [cit. 2014-08-05]. Dostupné z: http://developer.android.com/guide/topics/manifest/manifest-intro.html

vzhledu od programového kódu.[23]

3.5.1.1 Linear Layout

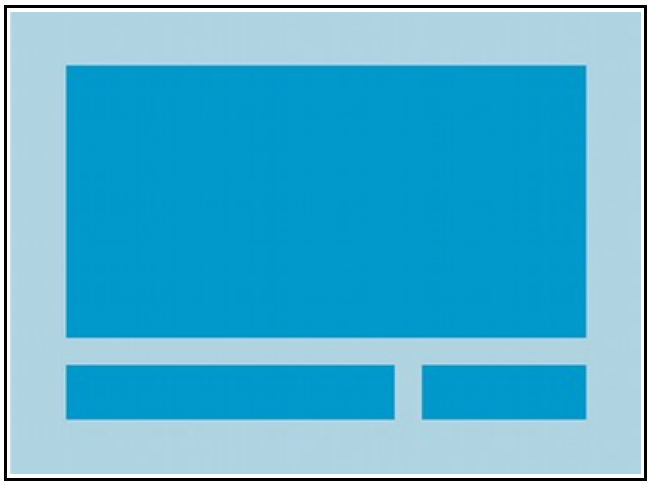


Obrázek 4: Linear layout zdroj: http://developer.android.com/guide/topics/ui/declaring-layout.html

Je druh view, kde jsou potomci řazeni jeden za druhým v jednom směru, který si zvolíme. Na výběr máme vertikální uspořádání shora dolů, nebo horizontální směrem zleva doprava.

Ve vertikální orientaci každý potomek je jeden řádek rodiče. Při vertikální orientaci jsou potomci jako sloupky rodiče.[24]

3.5.1.2 Relative Layout



Obrázek 5: Relative layout zdroj: developer.android.com/guide/topics/ui/declaring-layout.html

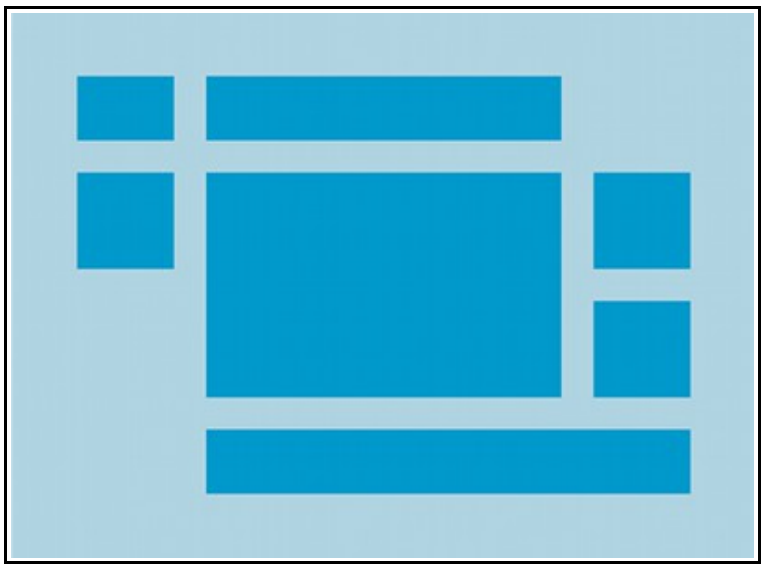
23 Layouts. *Android Developers* [online]. 2014 [cit. 2015-02-18]. Dostupné z: http://developer.android.com/guide/topics/ui/declaring-layout.html

24 Linear Layout. *Android Developers* [online]. 2014 [cit. 2015-02-18]. Dostupné z: http://developer.android.com/guide/topics/ui/layout/linear.html

V tomto view se potomci pozicují relativně oproti ostatním potomkům. To znamená, že mají referenci na nějakého jiného potomka rodiče nebo přímo vůči rodiči.

Toto umisťování prvků má oproti jinému prvku své výhody – udržuje hierarchický zápis v XML plošší, neboť se pro změnu směru orientace nemusí vytvořit nový uzel XML souboru.[25]

3.5.1.3 Table Layout



Obrázek 6: Table layout zdroj: https://developer.android.com/guide/topics/ui/layout/grid.html

Tento druh view zobrazuje své potomky do řádků a sloupků. Potomci první úrovně jsou řazeny shora dolů. Převážně to jsou objekty typu TableRow. Potomci druhé úrovně jsou řazeny zleva doprava.

Výhoda zde je, že se zpřehlední zápis, protože řádky jsou od sebe jednoznačně odděleny a dají se lépe upravovat. Buňky řádku můžeme jednoduše spojovat nebo jednotlivé prvky můžeme zneviditelnit.[26]

Tento layout má nevýhodu tehdy, když je potřeba vytvořit v řádky s rozdílným počtem sloupků. V těchto případech musíme tabulku například rozdělit na dvanáct sloupků a prvkům v z řádku přiřadit jejich poměrovou váhu v řádku.

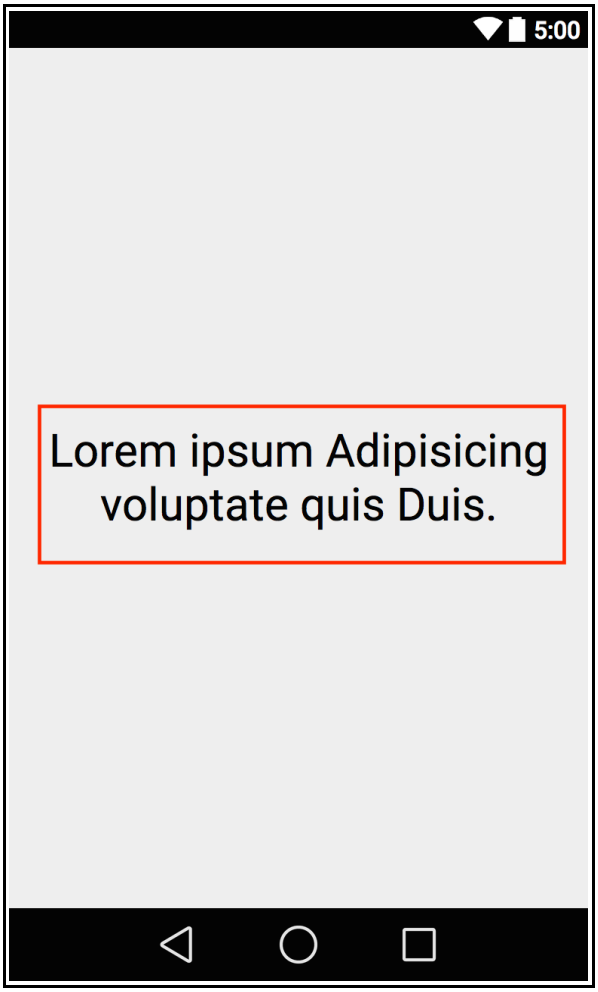
25 Relative Layout. *Android Developers* [online]. 2014 [cit. 2015-02-18]. Dostupné z: http://developer.android.com/guide/topics/ui/layout/relative.html

26 Table. *Android Developers* [online]. 2014 [cit. 2015-02-19]. Dostupné z: https://developer.android.com/guide/topics/ui/layout/grid.html

3.5.2 Vybrané grafické komponenty

Tyto vybrané komponenty jsou klíčové pro interakci uživatele s aplikací a aplikace s uživatelem.

3.5.2.1 TextView



Obrázek 7: TextView zdroj: vlastní

Slouží pro zobrazení textových informací uživateli. Na výše uvedeném obrázku je červeně označený.

3.5.2.2 EditText

EditText zpřístupňuje uživateli vkládání textu nebo jeho editaci. Po doteku této komponenty se zobrazí kurzor a automaticky se zobrazí klávesnice.

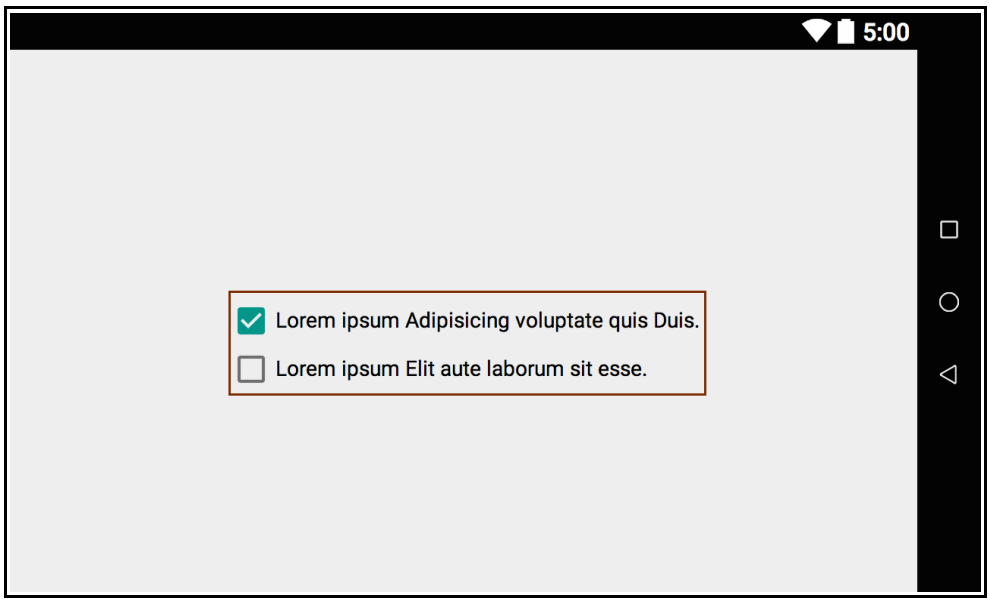
Této komponentě můžeme přiřadit vlastnost, jakou klávesnici chceme nechat

zobrazit. Klávesnice se umí přepnout do režimu zadávání obyčejného textu, e-mailové adresy, webové stránky, číslic nebo telefonního čísla. Klávesnici můžeme nastavit tak, aby psala první písmeno věty jako velké, u každého slova velké písmeno nebo že vstup je heslo. V tomto případě se budou zobrazovat zástupné znaky místo znaků vložených.[27]

3.5.2.3 Button

Tlačítka jsou vložena do rodičovského view, aby uživatel aplikace potvrdil nějaký vstup nebo spustil nějakou naprogramovanou interakci aplikace. Vytvořená tlačítka mohou obsahovat ikonu, text nebo obojí.[28]

3.5.2.4 CheckBox



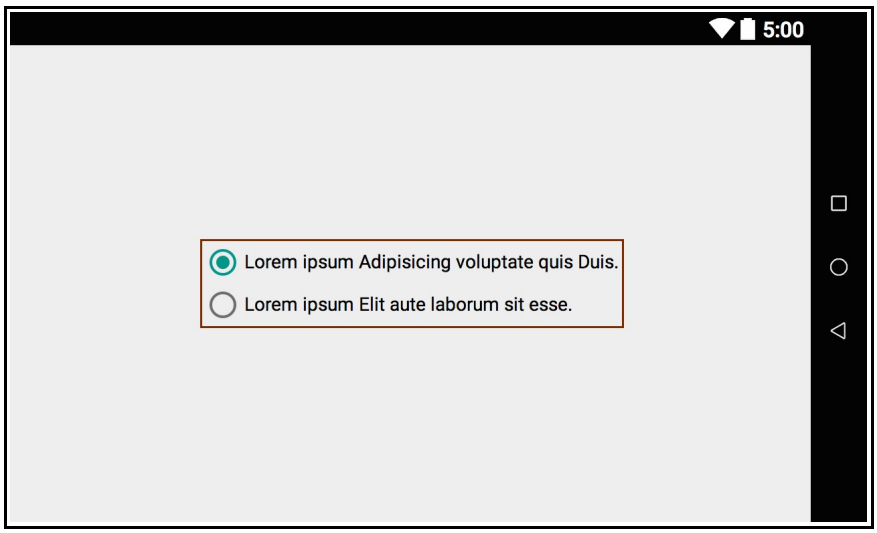
Obrázek 8: CheckBox zdroj: vlastní

Na výše obrázku je CheckBox červeně orámovaný.

Toto view dovoluje uživatele označit žádnou nebo více možností. Všechny z těchto možností nabývají hodnot true, nebo false. Tato komponenta je vhodná například pro odškrtávání položek ze seznamu a podobné případy.[29]

27 Text Fields. *Android Developers* [online]. 2014 [cit. 2015-02-19]. Dostupné z: http://developer.android.com/guide/topics/ui/controls/text.html
28 Buttons. *Android Developers* [online]. 2014 [cit. 2015-02-19]. Dostupné z: http://developer.android.com/guide/topics/ui/controls/button.html
29 Checkbox. *Android Developers* [online]. 2014 [cit. 2015-02-19]. Dostupné z: http://developer.android.com/guide/topics/ui/controls/checkbox.html

3.5.2.5 RadioButton

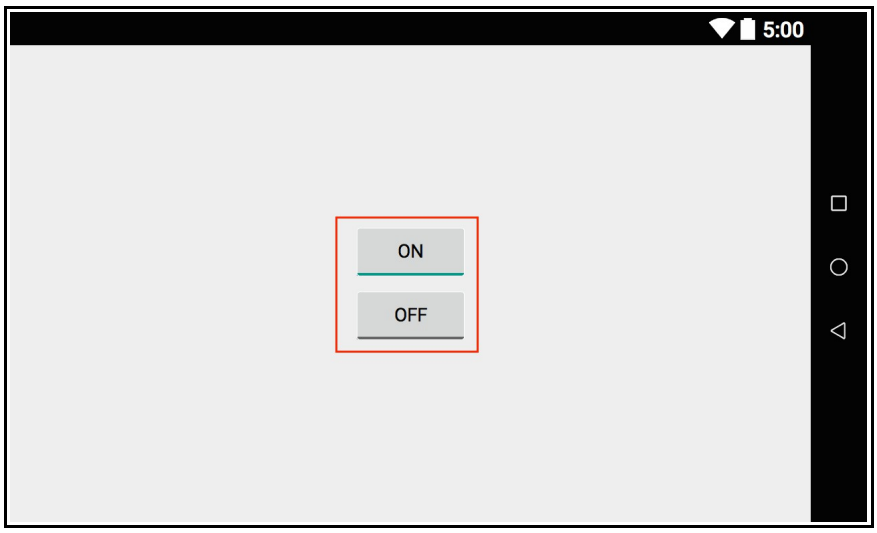


Obrázek 9: RadioButton zdroj: vlastní

Na výše obrázku je RadioButton červeně orámovaný.

Tuto grafickou komponentu můžeme programátor využít, pokud potřebuje od uživatele získat jednu odpověď z několika možností. Protože RadioButton může být označena pouze jedna odpověď je potřeba skupiny těchto odpovědí zabalit do xml tagu RadioGroup, který jednotlivé otázky od sebe odliší.[30]

3.5.2.6 Switch, ToggleButton



Obrázek 10: ToggleButton zdroj: vlastní

30 Radio Buttons. *Android Developers* [online]. 2014 [cit. 2015-02-19]. Dostupné z: http://developer.android.com/guide/topics/ui/controls/radiobutton.html

Na výše obrázku je ToggleButton červeně orámovaný.

Tato dvoustavová komponenta se používá v systému Android verze 4.0 (api level 15). V dřívějších verzích se používala komponenta ToggleButton. Komponenta je velmi podobná svým chováním CheckBoxu, protože opět nabývá stavů označeno (true), nebo neoznačeno (false).[31]

3.5.2.7 ImageView

Slouží pro zobrazení obrázků načtených z resources aplikace nebo odchycených pomocí content provideru. ImageView se postará o přepočítání měřítka obrázku, aby se dal použít v rámci vytvořených rozložení. Další možností je použít různé filtry pro vložené obrázky.[32]

3.5.3 Status bar

Po stažení rolety zde uživatelé mohou vidět detaily notifikací. V nestaženém stavu se vlevo zobrazují ikony aplikací, které uživatele upozorňují, že nastala nějaká notifikace. Z pravé strany jsou zobrazeny systémové hodiny, indikátor stavu baterie a indikátor síly a druhu připojení na internet a do GSM sítě.

3.5.4 Notifikace

Notifikace jsou vhodné pro stručné zprávy – ty při kliknutí spustí aplikaci, která notifikaci vytvořila. Obvykle slouží pro zobrazení aktualizací nebo připomínky z aplikace.

Pro odstranění notifikací ze seznamu se používá jednoduché horizontální swipe gesto nebo zde bývá tlačítko, které smaže všechny notifikace najednou.

3.5.5 Navigation Bar

Slouží pro zobrazení softwarových tlačítek, která slouží pro krok zpět, zobrazení domovské obrazovky a zobrazení naposledy spuštěných aplikací. Tato lišta je dostupná

31 Toggle Buttons. *Android Developers* [online]. 2014 [cit. 2015-02-19]. Dostupné z: http://developer.android.com/guide/topics/ui/controls/togglebutton.html

32 ImageView. *Android Developers* [online]. 2014 [cit. 2015-02-19]. Dostupné z: http://developer.android.com/reference/android/widget/ImageView.html

pouze pro zařízení, která nemají tradiční hardwarová tlačítka, která by je zastoupila.

3.5.6 Action Bar

Uživatel používá jako ovládací centrum aplikace. Action bar se může měnit, a proto můžeme všechny části aplikace vytvořit jiné.

Action bar se dá rozdělit do nekonečně mnoha podob, ale v dokumentaci je doporučené rozdělení na „hlavní action bar“, „vrchní pruh“ a „spodní pruh“. V hlavní části se mohou vyskytovat tyto prvky: ikona aplikace, view control, action buttons, action overflow. Jako přepínač view můžeme využít vrchní pruh action baru. Do dolního pruhu můžeme přesunout action buttons a action overflow.

3.5.6.1 Ikona aplikace

Vytváří identitu aplikace, může zde být vložený jakýkoliv obrázek. Pomocí ikony se uživatel také může v aplikaci pohybovat zpět.

3.5.6.2 View Control

Tato část action baru se dá nazvat jako interaktivní – můžeme sem vložit widget, který bude mít funkci dropdown menu pro navigaci na další funkce, nebo zde může být vstupní textové pole, které může prohledávat zobrazený text nebo seznam.

3.5.6.3 Action Buttons

Do této části může programátor vložit tlačítka, která budou spouštět vybrané funkce aplikace pro vybrané view.

3.5.6.4 Action Overflow

Slouží jako kontejner pro schování méně používaných action buttons.

3.5.7 Navigation Drawer

Tento prvek je vhodné použít, pokud je navigace složitější a při implementaci do

action baru by byla nepřehledná. Navigation drawer je většinou schovaný za levou hranou displeje, a pokud to uživatel vyžádá, vysune se.

3.5.8 Content Area

Reprezentuje prostor, kde se zobrazí obsah view, se kterým chce uživatel pracovat.[33]

3.5.9 Devices and Displays

Operační systém Android je nainstalovaný na různých druzích zařízení, jako jsou mobilní telefony a tablety, které mají rozdílné velikosti a rozlišení displeje. Pro tyto případy máme v projektu složky MDPI (~160 DPI), HDPI (~240 DPI), XHDPI (~320 DPI), XXHDPI (~480 DPI), kam nahrajeme ikony v příslušných rozlišeních, a operační systém si na základě jemnosti displeje vybere ikonu z příslušné složky a vloží se do view, kam patří.[34]

3.5.10 Themes

Themes je mechanismus, který operační systém Android používá pro udržení stejného vzhledu ve všech použitých views v aplikaci. Ve složce style.xml můžeme nastavit základní vzhledové vlastnosti prvků, například velikost a barvu textu, barvu pozadí nebo odsazení od okraje prvku. Operační systém Android má dvě základní barevná schémata, ze kterých si programátor může vybrat: „Holo Light“, „Holo Dark“. Tato dvě schémata si programátor může upravit pomocí souboru style.xml.[35]

3.5.11 App Structure

Při vývoji aplikace platí pravidlo maximálně tří kliknutí. Toto pravidlo platí také pro aplikace pro operační systém Android. Aplikace by měla být rozdělena na tři stupně: „Top level views“, „Category views“ a „Detail/edit view“.

---

33 Phones & Tablets. *Android Developers* [online]. 2015 [cit. 2015-03-15]. Dostupné z: https://developer.android.com/design/handhelds/index.html

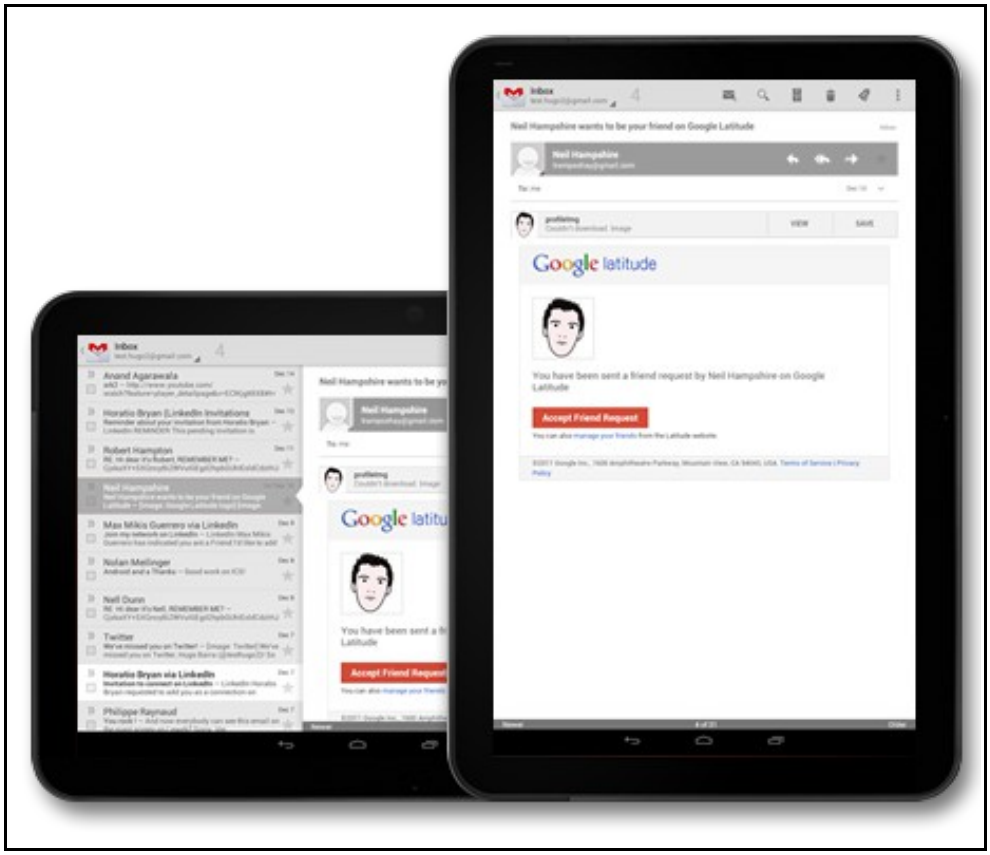
34 Devices and Displays. *Android Developers* [online]. 2015 [cit. 2015-03-15]. Dostupné z: https://developer.android.com/design/style/devices-displays.html

35 Themes. *Android Developers* [online]. 2015 [cit. 2015-03-15]. Dostupné z: https://developer.android.com/design/style/themes.html

Top level views se zobrazí uživateli po spuštění aplikace. Pro navigaci na „Detail/edit view“ se používá „Fixed tabs widget“, který výsledné views seskupuje do záložek, a uživatel si tak může vybrat, kterou si spustí. Zde je doporučeno použít maximálně pět až sedm záložek; při vyšším počtu by záložkové rozdělení bylo nepřehledné. Při větším počtu spouštěných funkcí je možné použít „Spinners widget“, který funguje jako drop-down menu, nebo se dá použít Category view, které bude fungovat jako výčet funkcí, jež jsou spustitelné.[36]

„Category view“ je nepovinné a může být sloučené s top view v jedno.

3.5.12 Multi-pane Layouts



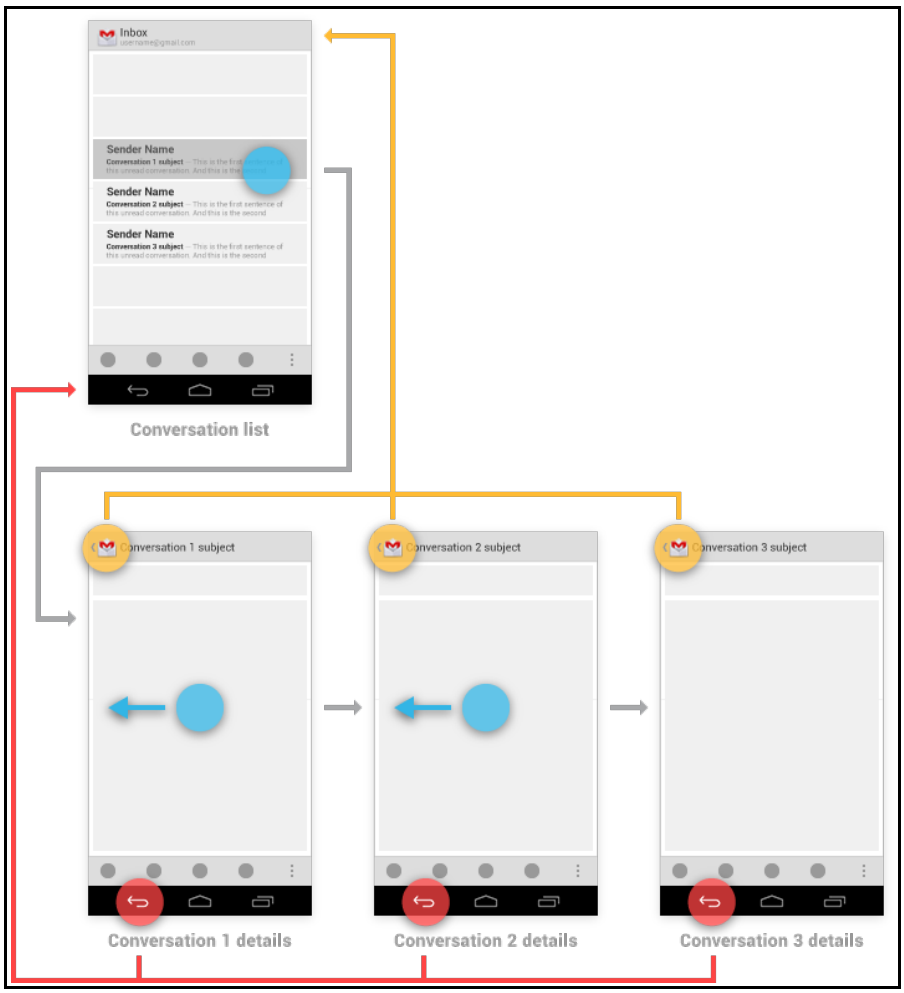
Obrázek 11: Ukázka multi-pane layouts zdroj: https://developer.android.com/design/patterns/multi-pane-layouts.html

Zařízení s operačním systémem Android mají různé velikosti, proto, aby aplikace na velkých displejích neplýtvaly místem, vznikly panely. Při používání panelů můžeme view složit z několika dílčích částí. Celá tato koncepce má tu výhodu, že na velkých

36 App Structure. *Android Developers* [online]. 2015 [cit. 2015-03-16]. Dostupné z: https://developer.android.com/design/patterns/app-structure.html

zařízeních složené views lépe vyplní zobrazenou plochu zařízení.[37]

3.5.13 Navigation



Obrázek 12: Krok zpět v aplikaci zdroj:
https://developer.android.com/design/patterns/navigation.html

Ve verzi operačního systému 3 byla přidán takzvaný up botton, který slouží pro navigaci zpět v rámci aplikace. Předchozí verze pracovaly pouze s fyzickým, nebo softwarovým tlačítkem zpět. Navigace pomocí up botton a tlačítka zpět je trochu rozdílná, protože up botton se vrací o úroveň výše, čili z detail view se uživatel vrátí na category view nebo na top view. Tlačítko zpět funguje trochu jiným způsobem, a to tak, že prochází zobrazená views v chronologickém pořadí, ve kterém byla zobrazena uživateli.[38]

37 Multi-pane Layouts. *Android Developers* [online]. 2015 [cit. 2015-03-16]. Dostupné z: https://developer.android.com/design/patterns/multi-pane-layouts.html
38 Navigation with Back and Up. *Android Developers* [online]. 2015 [cit. 2015-03-16]. Dostupné z: https://developer.android.com/design/patterns/navigation.html

3.5.14 Pure Android

Každá platforma má navrženou specifickou sadu uživatelských komponent a stylů vzhledu. Například na iOS mají tlačítka kulaté rohy, některé používají barevný přechod pro jejich „Title bar“. Grafické návrhy uživatelského rozhraní by se neměly napodobovat mezi platformami. V případě, že je potřeba upravit vybrané komponenty, mělo by se postupovat velmi opatrně.

Ačkoliv mobilní operační systémy vypadají každý jinak, mají společné funkce jako sdílet, vyhledat, zpět, smazat atd. Pro tyto funkce má rovněž každý operační systém svou sadu ikon.

Při návrhu uživatelského rozhraní by vývojář neměl používat „tabhost“, který je ukotvený v dolní části displeje, protože tento widget používá iOS.

Ostatní operační systémy pro návrat ze zobrazené uživatelského rozhraní softwarové tlačítko v Action baru. Android by pro tuto funkci měl používat ikonu aplikace.

V nastavení aplikace mobilní operační systém Android nepoužívá žádné indikátory, které by naznačily, že se na položku dá kliknout.

Jak bylo výše několikrát zmíněno, operační systém Android se používá na rozdílných zařízeních, jako jsou mobilní telefony, tablety nebo další příslušenství. Vývojář by proto neměl zapomenout vytvořit „multi-pane“ rozložení a vložit všechny ikony do aplikace v různých rozlišeních.[39]

39 Pure Android. *Android Developers* [online]. 2015 [cit. 2015-03-18]. Dostupné z: https://developer.android.com/design/patterns/pure-android.html

4 Vlastní práce

Z načerpaných znalostí a zkušeností byla vytvořena demonstrativní aplikace. Demonstrativní aplikaci jsem pomocí nástroje GIT rozdělil do několika větví, které budu v této části dále analyzovat a popisovat.

Všechny zdrojové kódy aplikace budou dostupné v přepisu této práce nebo klonováním repozitáře pomocí nástroje GIT.

Repositář je dostupný na https://github.com/HlubyLuk/dpDemo.git

4.1 Srovnání activity a fragmentu

Používání fragmentů má jednoznačnou výhodu, protože jedna activity může obsahovat nekonečně mnoho fragmentů, které je možné používat kdykoliv a kdekoliv. Další výhoda fragmentů je, že programátor nemusí psát a opakovat bloky kódu, které se budou vykonávat v nových activities. Další výhodou používání fragmentů je modularita aplikace, protože každý fragment může reprezentovat jednu funkcionalitu.

Pro demonstraci výhod a nevýhod používání fragmentů byly vytvořeny v GIT repozitáři větve act a fra, které budou porovnány pomocí příkazu `git diff act fra`. Po spuštění příkazu bude rozdílný obsah větve act označen znakem mínus a rozdílný obsah větve fra bude označen znakem plus. Shodný text v obou větvích je černý text.

4.1.1 Porovnání AndroidManifest.xml

```
diff --git a/app/src/main/AndroidManifest.xml
b/app/src/main/AndroidManifest.xml
index f4e0238..8ab891f 100644
--- a/app/src/main/AndroidManifest.xml
+++ b/app/src/main/AndroidManifest.xml
@@ -15,12 +15,5 @@
                 <category
android:name="android.intent.category.LAUNCHER" />
             </intent-filter>
         </activity>
-        <activity
-            android:name=".OtherUI"
-            android:parentActivityName=".MainActivity">
```

V manifestu větve act je potřeba přidat tag <activity>, kde pomocí atributů android:name zadefinujeme jméno nové activity a android:parentActivityName funguje jako odkaz na activity, která se má spustit při zmáčknutí tlačítka zpět nebo UP button.

Pokud programátor v AndroidManifest.xml nevytvoří novou activity a uživatel by se ji pokusil spustit, aplikace se zastaví a v logovacím souboru najdeme chybu AndroidRuntime: FATAL EXCEPTION.

```
-            <meta-data
-                android:name="android.support.PARENT_ACTIVITY"
-                android:value="cz.hlubyluk.dp.MainActivity" />
-        </activity>
     </application>
 </manifest>
```

Tag <meta-data> je potřeba použít kvůli zpětné kompatibilitě pro starší zařízení, která mají operační systém Android ve verzi 4 a nižší. Při nepoužití tohoto tagu je UP button nefunkční a vůbec se nezobrazí.

4.1.2 Porovnání MainActivity.java

```
diff --git a/app/src/main/java/cz/hlubyluk/dp/MainActivity.java
b/app/src/main/java/cz/hlubyluk/dp/MainActivity.java
index 480d3f2..6f4ebb7 100644
--- a/app/src/main/java/cz/hlubyluk/dp/MainActivity.java
+++ b/app/src/main/java/cz/hlubyluk/dp/MainActivity.java
@@ -1,17 +1,16 @@
 package cz.hlubyluk.dp;

-import android.content.Intent;
 import android.os.Bundle;
+import android.support.v4.app.FragmentTransaction;
 import android.support.v4.widget.DrawerLayout;
 import android.support.v7.app.ActionBarActivity;
 import android.support.v7.app.ActionBarDrawerToggle;
 import android.view.MenuItem;
 import android.view.View;
 import android.widget.Button;
-import android.widget.LinearLayout;


-public class MainActivity extends ActionBarActivity implements
View.OnClickListener {
+public class MainActivity extends ActionBarActivity implements
```

```
IClickedButton {
```

Pro zpracování nastavených událostí je potřeba implementovat veřejný interface; v případě vývoje pomocí nových activities stačí interface View.OnClickListener u vývoje ve fragmentech se pro implementaci vytvoří vlastní interface – zde to je IclickedButton.

```
     DrawerLayout drawerLayout;
     ActionBarDrawerToggle actionBarDrawerToggle;
@@ -20,18 +19,18 @@ public class MainActivity extends
ActionBarActivity implements View.OnClickListe
     protected void onCreate(Bundle savedInstanceState) {
         super.onCreate(savedInstanceState);
         setContentView(R.layout.activity_main);
+        if (savedInstanceState == null) {
+            FragmentTransaction fragmentTransaction =
getSupportFragmentManager().beginTransaction();
+            fragmentTransaction.replace(R.id.dlCont,
FragmentHome.NewInstance());
+            fragmentTransaction.replace(R.id.dlNavi,
FragmentNavigation.newInstance());
+            fragmentTransaction.commit();
+        }
```

Zpracování fragmentů probíhá transakčním způsobem. Při použití této metody se musí vytvořit instance třídy FragmentTransaction, která zpracuje všechny změny po zavolání metody commit().

```
         getSupportActionBar().setDisplayHomeAsUpEnabled(true);
         getSupportActionBar().setHomeButtonEnabled(true);
         drawerLayout = (DrawerLayout) findViewById(R.id.dlRoot);
         actionBarDrawerToggle = new ActionBarDrawerToggle(this,
drawerLayout, R.string.mdOpen, R.string.mdClose);
         drawerLayout.setDrawerListener(actionBarDrawerToggle);
         actionBarDrawerToggle.syncState();
-        LinearLayout navigation = (LinearLayout)
findViewById(R.id.dlNavi);
-        for (int i = 0; i < navigation.getChildCount(); i++) {
-            if (navigation.getChildAt(i) instanceof Button) {
-                navigation.getChildAt(i).setOnClickListener(this);
-            }
-        }
```

Pro tlačítka, která slouží pro spuštění nové aktivity, jsme nastavili listener metodou setOnClickListener(), pracující jako veřejný interface objektů. Také nám zpřístupní možnost asynchronně zpracovávat vzniklé události u objektů, které mají nastavený tento

listener, a to pomocí implementované metody onClick().

```
     }

     @Override
@@ -43,19 +42,17 @@ public class MainActivity extends
ActionBarActivity implements View.OnClickListe
     }

     @Override
-    public void onClick(View view) {
-        Intent intent = null;
+    public void clicked(View view) {
+        FragmentTransaction fragmentTransaction =
getSupportFragmentManager().beginTransaction();
         switch (view.getId()) {
             case R.id.mdBT1:
-                intent = new Intent(this, OtherUI.class);
-                intent.putExtra("text", ((Button) view).getText());
+                fragmentTransaction.replace(R.id.dlCont,
FragmentOtherUI.newInstanc(((Button) view).getText()));
                 break;
             case R.id.mdBT2:
-                intent = new Intent(this, OtherUI.class);
-                intent.putExtra("text", ((Button) view).getText());
+                fragmentTransaction.replace(R.id.dlCont,
FragmentOtherUI.newInstanc(((Button) view).getText()));
                 break;
         }
-        startActivity(intent);
+        fragmentTransaction.addToBackStack(null).commit();
         drawerLayout.closeDrawers();
     }
 }
```

V případě, že se pro vývoj zvolí metoda spouštění nových activities, je potřeba objekt Intent, který při konstrukci dostane první vstupní parametr, sloužící pro získání kontextu aplikace, ve chvíli, kdy se Intent tvoří. Druhý parametr je ukazatelem na třídu, kterou má být spuštěna.

Do objektu Intent můžeme vložit jednoduchá data, která chceme předat do nové activity.

Pokud používáme aktivity, spustí se nová activity pomocí metody startActivity(intent).

Při používání fragmentů je potřeba pomocí FragmenteTransaction.replace(int containerViewId, Fragment fragment) vyměnit obsah FrameLayout, který jde rozlišit pomocí atributu android:id v XML definici hostující activity. Po zavolání metody commit(), se vytvoří nový fragment, který jsme předali metodě replace() jako druhý parametr.

4.1.3 Porovnání OtherUI.java a FragmentOtherUI.java

```
diff --git a/app/src/main/java/cz/hlubyluk/dp/OtherUI.java
b/app/src/main/java/cz/hlubyluk/dp/OtherUI.java
deleted file mode 100644
index f83c357..0000000
--- a/app/src/main/java/cz/hlubyluk/dp/OtherUI.java
+++ /dev/null
@@ -1,19 +0,0 @@
-package cz.hlubyluk.dp;
-
-import android.os.Bundle;
-import android.support.v7.app.ActionBarActivity;
-import android.widget.TextView;
-
-/**
- * Created by HlubyLuk on 28.02.15.
- */
-public class OtherUI extends ActionBarActivity {
-    @Override
-    protected void onCreate(Bundle savedInstanceState) {
-        super.onCreate(savedInstanceState);
-        setContentView(R.layout.other_ui);
-        String text = getIntent().getStringExtra("text");
```

V nově spuštěné activity v proběhne metoda onCreate(). Zde si můžeme z objektu Intent, který spustil novou activity, převzít všechna vložená data, která byla předána při vytvoření v předchozí activity.

```
-        TextView textView = (TextView) findViewById(R.id.ouiTV);
-        textView.setText(text);
-    }
-}
```

Získaný textový řetězec vložíme do předem připraveného TextView.

```
diff --git a/app/src/main/java/cz/hlubyluk/dp/FragmentOtherUI.java
b/app/src/main/java/cz/hlubyluk/dp/FragmentOtherUI.java
new file mode 100644
index 0000000..143572e
--- /dev/null
```

```
+++ b/app/src/main/java/cz/hlubyluk/dp/FragmentOtherUI.java
@@ -0,0 +1,32 @@
+package cz.hlubyluk.dp;
+
+import android.os.Bundle;
+import android.support.annotation.Nullable;
+import android.support.v4.app.Fragment;
+import android.view.LayoutInflater;
+import android.view.View;
+import android.view.ViewGroup;
+import android.widget.TextView;
+
+/**
+ * Created by HlubyLuk on 28.02.15.
+ */
+public class FragmentOtherUI extends Fragment {
+
+    public static FragmentOtherUI newInstanc(CharSequence text) {
+        FragmentOtherUI fragment = new FragmentOtherUI();
+        Bundle bundle = new Bundle();
+        bundle.putCharSequence("text", text);
+        fragment.setArguments(bundle);
+        return fragment;
+    }
+
```

Výše uvedená metoda newInstance() se používá, protože operační systém Android vyžaduje prázdné konstruktory. Pro předání jednoduchých datových typů se dá využít objekt Bundle, do kterého namapujeme námi předávané hodnoty.

```
+    @Override
+    public View onCreateView(LayoutInflater inflater, @Nullable
ViewGroup container, @Nullable Bundle savedInstanceState) {
+        View view = inflater.inflate(R.layout.fragment_other_ui,
container, false);
+        String text = getArguments().getString("text");
+        TextView textView = (TextView)
view.findViewById(R.id.ouiTV);
+        textView.setText(text);
+        return view;
+    }
+}
```

U fragmentového přístupu se obsah TextView předával pomocí objektu Bundle. Obsah tohoto objektu je dostupný fragmentu jako jeho argumenty.

4.1.4 Porovnání activity_main.xml

```
diff --git a/app/src/main/res/layout/activity_main.xml
b/app/src/main/res/layout/activity_main.xml
index b9a1b48..61c3911 100644
--- a/app/src/main/res/layout/activity_main.xml
+++ b/app/src/main/res/layout/activity_main.xml
@@ -4,27 +4,11 @@
     style="@style/mpmp"
     tools:context=".MainActivity">

-    <RelativeLayout
+    <FrameLayout
         android:id="@+id/dlCont"
-        style="@style/mpmp">
+        style="@style/mpmp" />

-        <TextView
-            style="@style/tvCIP"
-            android:text="@string/maTV" />
-    </RelativeLayout>
-
-    <LinearLayout
+    <FrameLayout
         android:id="@+id/dlNavi"
-        style="@style/drawer">
-
-        <Button
-            android:id="@+id/mdBT1"
-            style="@style/mpwc"
-            android:text="@string/naviBT1" />
-
-        <Button
-            android:id="@+id/mdBT2"
-            style="@style/mpwc"
-            android:text="@string/naviBT2" />
-    </LinearLayout>
+        style="@style/drawer" />
 </android.support.v4.widget.DrawerLayout>
```

Zde je vidět, jak vypadá xml soubor, který bude reprezentovat views zobrazené uživateli. Struktura je rozdělena na tři úrovně. První úroveň tvoří kořenový element, který je zde DrawerLayout. Tento layout nám zpřístupňuje vysouvací boční panel, který může posloužit jako navigace pro aplikaci. V další vrstvě jsou vloženy dva elementy, které slouží jako zobrazovací prostory pro nadefinované komponenty, s nimiž bude uživatel pracovat.

Při použití fragmentů se do main_activity.xml zadefinují tagy <FrameLayout>; budou fungovat jako prostor pro vytvořené fragmenty. Pokud se tento fragment při práci uživatele s aplikací nebude měnit, můžeme vložit tag <fragment>, v němž pomocí atributu class: spustíme fragment, který chceme, aby se zobrazil ve zvoleném prostoru.

4.1.5 Porovnání ostatních souborů

Soubory HomeFragment.java, NavigationFragment.java, home_fragment.xml a navigation_fragment.xml jsou součástí větve fra.

Jak bylo výše řečeno, vývoj aplikací pomocí fragmentů má další výhodu v tom, že se nemusí opakovat tolik kódů, neboť jednotlivé fragmenty tvoří dílčí moduly, které můžeme vkládat do <FrameLayout> a které jsou nadefinované v activity_main.xml.

```
diff --git a/app/src/main/java/cz/hlubyluk/dp/FragmentNavigation.java
b/app/src/main/java/cz/hlubyluk/dp/FragmentNavigation.java
new file mode 100644
index 0000000..31cb9c0
--- /dev/null
+++ b/app/src/main/java/cz/hlubyluk/dp/FragmentNavigation.java
@@ -0,0 +1,49 @@
+package cz.hlubyluk.dp;
+
+import android.app.Activity;
+import android.os.Bundle;
+import android.support.annotation.Nullable;
+import android.support.v4.app.Fragment;
+import android.view.LayoutInflater;
+import android.view.View;
+import android.view.ViewGroup;
+import android.widget.Button;
+import android.widget.LinearLayout;
+
+/**
+ * Created by HlubyLuk on 28.02.15.
+ */
+public class FragmentNavigation extends Fragment implements
View.OnClickListener {
+
+    IClickedButton clickedButton;
+
+    public static FragmentNavigation newInstance() {
+        FragmentNavigation fragment = new FragmentNavigation();
```

```
+        return fragment;
+    }
+
+    @Override
+    public void onAttach(Activity activity) {
+        super.onAttach(activity);
+        if (activity instanceof IClickedButton) {
+            clickedButton = (IClickedButton) activity;
+        }
+    }
+
```

Metoda onAttach() slouží pro asociaci fragmentů s activity a naplnění instance interface, který tuto komunikaci fragmentu a MainActivity.java zprostředkovává, tedy fragment je schopný odesílat zprávy do své activity.

Také tento případ je ukázka návrhového vzoru observer, který bude podrobněji analyzován později.

```
+    @Override
+    public View onCreateView(LayoutInflater inflater, @Nullable
ViewGroup container, @Nullable Bundle savedInstanceState) {
+        View view = inflater.inflate(R.layout.fragment_navigation,
container, false);
+        LinearLayout linearLayout = (LinearLayout)
view.getRootView();
+        for (int i = 0; i < linearLayout.getChildCount(); i++) {
+            if (linearLayout.getChildAt(i) instanceof Button) {
+                linearLayout.getChildAt(i).setOnClickListener(this);
+            }
+        }
+        return view;
+    }
+
+    @Override
+    public void onClick(View view) {
+        clickedButton.clicked(view);
+    }
+}
```

Po stisknutí tlačítka, se zpracuje vzniklá událost v metodě onClick(View view), následuje takzvané callback volání, čili spuštění metody interface, který doručí zprávu do hostující activity.

```
diff --git a/app/src/main/java/cz/hlubyluk/dp/IClickedButton.java
b/app/src/main/java/cz/hlubyluk/dp/IClickedButton.java
```

```
new file mode 100644
index 0000000..c9091cb
--- /dev/null
+++ b/app/src/main/java/cz/hlubyluk/dp/IClickedButton.java
@@ -0,0 +1,10 @@
+package cz.hlubyluk.dp;
+
+import android.view.View;
+
+/**
+ * Created by HlubyLuk on 28.02.15.
+ */
+public interface IClickedButton {
+    public void clicked(View view);
+}
```

Interface je speciální druh třídy. Tato třída slouží jako soubory metod, které objekty používají pro komunikaci s ostatními objekty.

Zápis interface je název interface a poté se píší metody, které chceme, aby zvolený objekt implementoval. Metody mají prázdné tělo.[40]

Výše uvedená ukázka je jednoduchý příklad callback interface, který bude doručovat data do hostující activity.

Tento interface se dá také využít jako součást návrhového vzoru observer.

40 What is an Interface?. *The JavaTM Tutorials* [online]. 2015 [cit. 2015-03-09]. Dostupné z: http://docs.oracle.com/javase/tutorial/java/concepts/interface.html

4.2 Ukázka vícepanelového rozložení

Obrázek 13: ukázka vícepanelového rozložení zdroj: vlastní

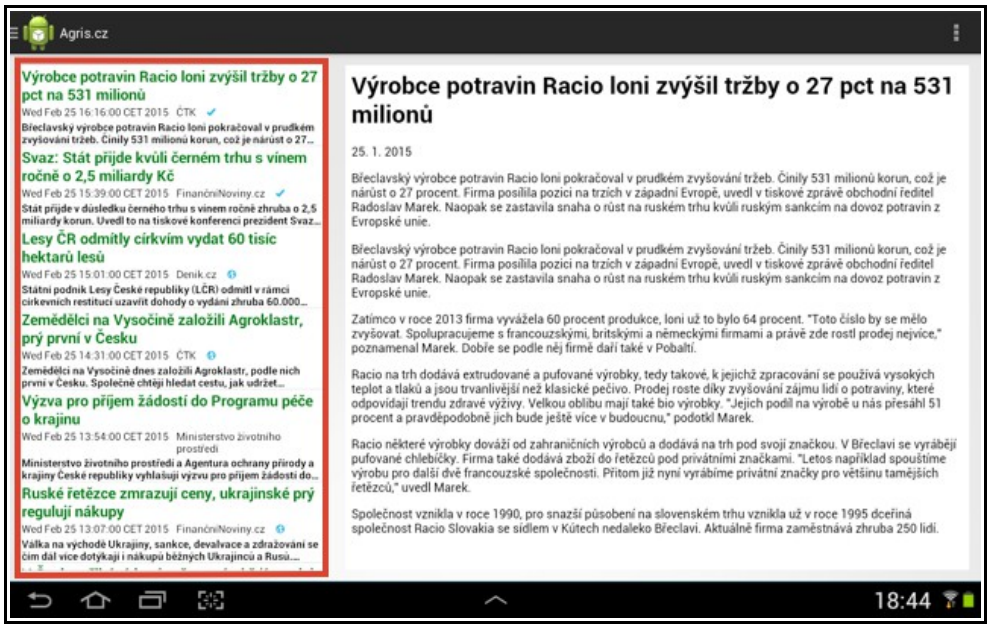
Protože aplikace může být nainstalována i na zařízeních s velkým displejem , je možno upravit výsledné view tak, aby zobrazovalo více informací najednou, viz výše vložený obrázek. Ve výše uvedeném příkladu je zobrazeno category view. V levé části je zobrazen navigační fragment, který je na zařízeních s malým displejem schovaný za levou hranou displeje.

```
<LinearLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="horizontal"
    android:weightSum="4">

    <FrameLayout
        android:id="@+id/flNavigation"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:layout_weight="3" />

    <FrameLayout
        android:id="@+id/flContent"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:layout_weight="1" />
```

Pro vytvoření vícepanelového vzhledu je nejjednodušší cestou použít fragmenty, protože díky tomu se tato šablona dá využít pro všechna views, pro různé úrovně používání zde to je například zde to je category view a detail view. V tomto případě pro zobrazení informací ve více panelech je potřeba v šabloně pro zobrazenou activity vytvořit dvakrát <FrameLayout>; jeden pro zobrazení navigačního fragmentu označený jako android:id="@+id/flNavigation" a druhý pro zobrazení obsahu označený android:id="@+id/flContent".



Obrázek 14: ukázka vícepanelového rozložení zdroj: vlastní

Na obrázku je ukázka, jak může vypadat detail view při využití více panelů. V této ukázce je sloučeno category view a detail view. Červeně orámovaný je seznam článků, který v předchozím stavu tvořil hlavní část okna.

4.3 Návrhové vzory

Návrhové vzory jsou pravidla a doporučení, které se používají v objektově orientovaných programovacích jazycích pro řešení problémů spojených s návrhem a vývojem softwaru.

Návrhové vzory jde rozřadit do tří hlavních skupin, a to tvořivé, strukturální a behaviorální.[41]

41 Design Pattern Overview. *Tutorialspoin*t [online]. 2015 [cit. 2015-03-22]. Dostupné z: http://www.tutorialspoint.com/design_pattern/design_pattern_overview.htm

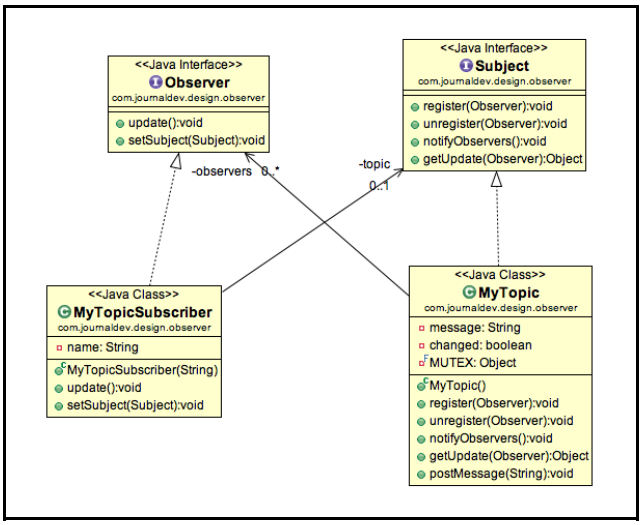
4.3.1 Singleton

Je příklad jednoduchého návrhového vzoru, který tvoří objekty. Tento návrhový vzor je vhodné implementovat tehdy, když potřebujeme mít pouze jednu instanci dané třídy. Tento vzor se dá použít pro vytvoření instance správce oken, dále na vytvoření tiskové řady či pro vytvoření spojení s databází a podobně.

```
public class ClassicSingleton {
   private static ClassicSingleton instance = null;
   protected ClassicSingleton() {
      // Exists only to defeat instantiation.
   }
   public static ClassicSingleton getInstance() {
      if(instance == null) {
         instance = new ClassicSingleton();
      }
      return instance;
   }
}
```

Typickým rysem tohoto návrhového vzoru je přístup k této instanci pomocí globálního přístupového bodu, který je zde statická metoda getInstance(). Při jeho implementaci se používá neveřejného konstruktoru, čili třída "Singleton" instancuje sama sebe, protože nikdo jiný nemá k jejímu konstruktoru přístup.[42]

4.3.2 Observer



Obrázek 15: UML diagram návrhového vzoru observer zdroj: http://www.journaldev.com/1739/observer-design-pattern-in-java-example- tutorial

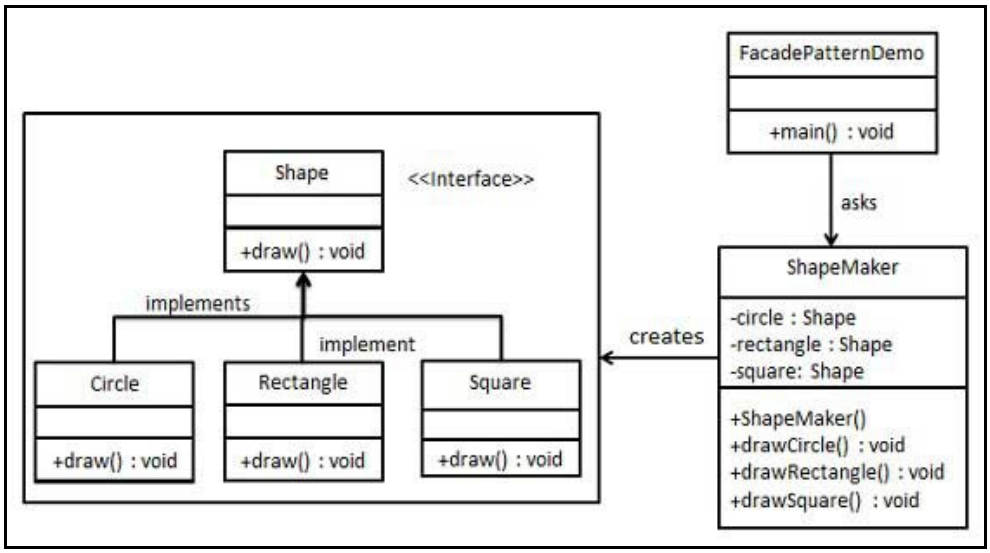
42 Simply Singleton. *JavaWorld* [online]. 2015 [cit. 2015-03-22]. Dostupné z: http://www.javaworld.com/article/2073352/core-java/simply-singleton.html

Návrhový vzor observer se používá v situacích, kdy nějaký objekt obsahuje kardinalitu jedna ku několika dalším objektům, které na něm závisí. Tento návrhový vzor řeší to, že upozorní všechny závislé objekty na změnu, která vznikla. Observer řeší chování objektů.

Tento návrhový vzor má využití pro View-Model-Controller framework, který musí nějak reagovat změnou view při změně model.

Implementace se dá rozdělit na dvě části: model zde představuje takzvaný předmět a view je zde pozorovatel, který si může každý model (předmět) registrovat pro zasílání změn modelu.[43]

4.3.3 Fasáda



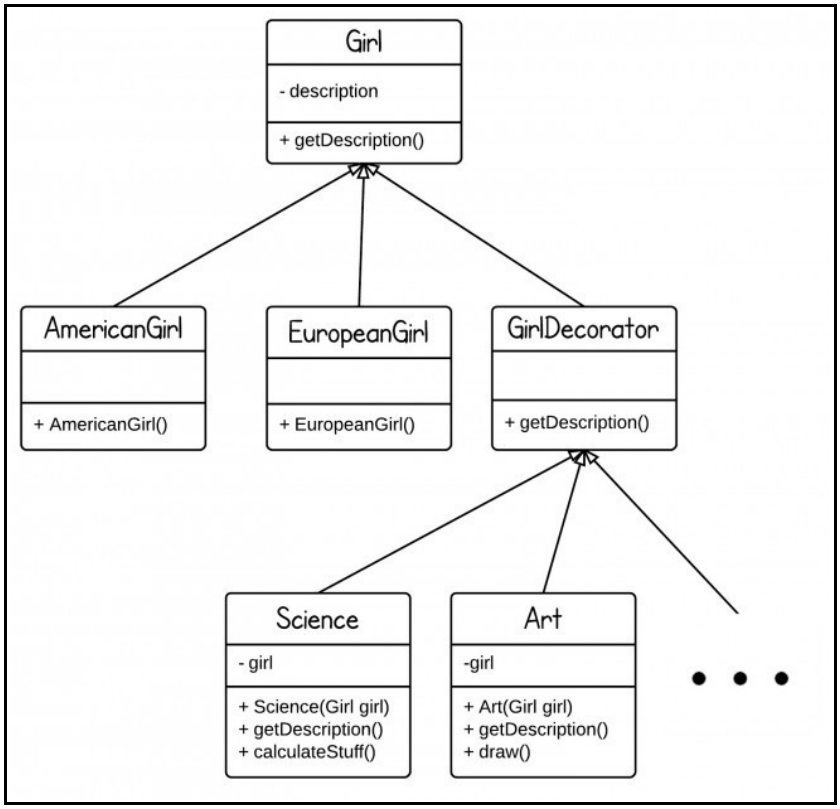
Obrázek 16: UML diagram návrhového vzoru fasáda zdroj: http://www.tutorialspoint.com/design_pattern/facade_pattern.htm

Jedná se o strukturální návrhový vzor. Největší využití se dá najít u aplikací, které mají hodně služeb; fasádní třída jim bude poskytovat přístup na tyto služby. Také se dá využít, když potřebujeme pomocí fasádní třídy vytvořit rozdílné objekty, jako ve výše přiloženém obrázku, kde se pomocí rozhraní Shape můžeme dotázat na vlastnosti, které budou implementovány v daném objektu.[44]

43 Observer Design Pattern in Java – Example Tutorial. *JournalDev* [online]. 2013 [cit. 2015-03- 22]. Dostupné z: http://www.journaldev.com/1739/observer-design-pattern-in-java-example- tutorial

44 Design Patterns Facade Pattern. *Tutorialspoint* [online]. 2015 [cit. 2015-03-24]. Dostupné z: http://www.tutorialspoint.com/design_pattern/facade_pattern.htm

4.3.4 Dekorátor



Obrázek 17: UML diagram návrhového vzoru dekorátor zdroj: http://www.programcreek.com/2012/05/java-design-pattern-decorator-decorate-your-girlfriend/

Dekorátor je návrhový vzor, který objektům dynamicky přidává nové vlastnosti a metody. Zlepšuje přizpůsobitelnost objektů, tak aby lépe popisovaly reálné předměty. Celý návrhový vzor funguje na principu dědění a vytváření malých, jednoduchých rozšíření, která jsou lépe udržitelná než jeden velký objekt, který by měl všechny vlastnosti a metody od začátku implementované.

```
package designpatterns.decorator;

public class Main {

	public static void main(String[] args) {
		Girl g1 = new AmericanGirl();
		System.out.println(g1.getDescription());

		Science g2 = new Science(g1);
		System.out.println(g2.getDescription());

		Art g3 = new Art(g2);
		System.out.println(g3.getDescription());
	}
```

```
}
```

Ve výše uvedené ukázce je praktické použití tohoto návrhového vzoru. Po vytvoření objektu Girl je tento objekt rozšířen o vlastnost, která říká, odkud pochází, a v dalších krocích se objektu přidávají vlastnosti jeho dovedností.

Výsledný výpis do konzole by mohl vypadat takto:

```
+American
+American+Like Science
+American+Like Science+Like Art
```

Znak + ve výpisu odděluje postupně přidávané vlastnosti objektu Girl.[45]

4.4 Publikování a správa aplikace

Pro možnost publikovat aplikaci na Google Play je potřeba zaregistrovat si vývojářský účet. Tento proces se může rozdělit na tři jednoduché kroky, a to „Registraci“, „Uhrazení poplatku“ a „Developer Console“.

4.4.1 Registrace vývojářského účtu

Před registrací je potřeba mít zřízený účet u společnosti Google. Pokud chce svou aplikaci publikovat společnost, je potřeba vytvořit nový účet, který bude vystupovat jako samostatná osoba.

Po vyplnění základního formuláře o identitě vývojáře a po přečtení licenčních podmínek pro vybraný region, kde se bude aplikace publikovat, je potřeba zaplatit registrační poplatek – ten činí 25 amerických dolarů. Registrační poplatek musí být zaplacený pomocí služby Google Wallet.

Pokud publikovaná aplikace se bude prodávat za peníze, je potřeba ve službě Google Wallet aktivovat obchodní verzi této služby.[46]

45 Java Design Pattern: Decorator – Decorate your girlfriend. *Program Creek* [online]. 2015 [cit. 2015-03-24]. Dostupné z: http://www.programcreek.com/2012/05/java-design-pattern-decorator-decorate-your-girlfriend/

46 Get Started with Publishing. *Android Developers* [online]. 2015 [cit. 2015-03-19]. Dostupné z: http://developer.android.com/distribute/googleplay/start.html

4.4.2 Vývojářská konzole

Po přihlášení do vývojářské konzole se jako první zobrazí seznam publikovaných aplikací. Pokud se publikuje úplně nová aplikace, stiskneme tlačítko „Přidat novou aplikaci“, kde vyplníme jméno aplikace a nahrajeme příslušný APK soubor. Po potvrzení dialogu nebo vybrání publikované aplikace se dostaneme na stránku, kde se vyplní povinná pole, která slouží jako popis vytvořené aplikace a informace o vývojáři aplikace. Teď stačí vše jen potvrdit a aplikace je publikována na službě Google Play. Po zveřejnění aplikace může vývojář kdykoliv upravit všechny nastavené hodnoty jako je cena aplikace nebo popis aplikace.

V záložce APK se zobrazí tři podzáložky, a to „PRODUKCE“, „TESTOVACÍ BETA VERZE“ a „TESTOVÁNÍ ALFA VERZE“. Tyto tři podzáložky slouží pro rozlišení testovacích verzí od produkční verze.

Testování všech aplikací pro operační systém Android probíhá pomocí komunity a komunitních stránek na službě Google+. Pro každou testovací verzi můžeme vytvořit jiný okruh testerů.

Po zveřejnění aplikace můžeme nastavit, v kterých zemích se vytvořená aplikace bude publikovat. Také se dá nastavit, pro jaká zařízení bude dostupná.

Ve vývojářské konzoli jsou dostupné statistiky a výpisy o pádech aplikace. Statistiky zobrazují počet instalací a odinstalací aplikace, na jakých verzích operačního systému je aplikace nainstalována, počet unikátních uživatelů a podobné metriky.

Záložka „Selhání a chyby AND“ slouží pro zobrazení chybových logů a zpráv, které uživatelé k těmto chybám napsali. Také se zde dá zobrazit, na jakém zařízení s jakou verzí operačního systému k chybám došlo.[47]

47 Developer Console. *Android Developers* [online]. 2015 [cit. 2015-03-19]. Dostupné z: http://developer.android.com/distribute/googleplay/developer-console.html

5 Zhodnocení výsledků a doporučení

Pro praktickou ukázku byla vytvořena demonstrativní aplikace, která je veřejně přístupná pomocí verzovacího nástroje GIT z veřejného repozitáře, dostupného z https://github.com/HlubyLuk/dpDemo.git.

Na této aplikaci byly demonstrovány dva rozdílné přístupy implementace, které jsou pro tento mobilní operační systém možné, a to pro každé nové view spustit novou activity a vývoj, který používá pro zobrazení nového view fragment.

Shrnutí výsledků pokusů a analýz z celého průběhu vývoje aplikací pro operační systém Android a vybraných komponent tohoto mobilního operačního systému.

Výsledky tohoto pokusu jsem rozdělil na několik bodů, které mohou fungovat jako autorovo doporučení:

- Protože celý vývoj aplikace má nějaký životní cyklus, je vhodné se těchto bodů držet. Před začátkem vývoje je potřeba udělat podrobnou analýzu toho, co výsledná aplikace bude umět.
- V druhé části této analýzy je potřeba všechny funkce rozřadit do jednotlivých úrovní používání, a to do top view, category view, detail view. Pro všechny úrovně je třeba vytvořit šablonu, jak bude vypadat na různě velkých zařízeních.
- Pro řešení dílčích funkcí a obsahu jednotlivých zobrazených úrovní je vhodné použít fragmentový přístup k vývoji aplikací pro operační systém Android. Pro přechod na vyšší úroveň pak použít novou activity.
- Další doporučení se bude týkat už vývoje aplikace, a to používání verzovacích nástrojů, které dovolí více vývojářům spolupracovat na jednom projektu.
- V průběhu vývoje se doporučuje dodržovat vytvořené a jednotné workflow.
- Před vývojem dílčích funkcionalit je potřeba vycházet z předběžné analýzy. Tuto analýzu je možné ještě přezkoumat a rozšířit o detaily, které tato funkcionalita bude vykonávat.

- Při vytváření finálního vzhledu se doporučuje vycházet ze základních vzhledových principů operačního systému Android.
- Výsledný vhled vytvořit pomocí stylovacího souboru style.xml, protože tímto se rozdělí instance grafických komponent od jejich vzhledových vlastností.

6 Závěr

Mobilní operační systém Android je otevřenou platformou, která se postupem času z mobilního operačního systému stala spíše celou platformou, protože dnes už operační systém Android není pouze v mobilních telefonech a tabletech, ale je součástí hodinek, televizí nebo slouží jako multimediální systém v automobilech.

Tato veliká fragmentace zařízení je jednoznačně jednou z největších nevýhod této platformy, protože vývojář musí obsloužit velké množství rozlišných zařízení a znát také specifické metody ovládání těchto zařízení.

- Vývoj aplikací pomocí fragmentů má jasné výhody, a to, že kód je lépe škálovatelný, udržitelný a přehlednější, protože každý fragment je objekt, který reprezentuje nějaké informace zobrazené uživateli. Tyto objekty mají své stavy a chování a po jeho vytvoření se dá vložit do activity.

- Jako verzovací nástroj byl v tomto případě zvolen GIT, který zvládne vytvořit několik paralelních verzí jednoho projektu nebo několik různých verzích repozitářů a přidělení různých přístupových práv pro programátory.

- V tomto případě bylo workflow rozděleno na vytvoření funkčního programu se základním vzhledem. Po dokončení funkcionality programu vytvořit vhled. Dle autorova názoru je vhodné, když se tento přístup promítne do repozitáře ve službě například GIT, a to tak, že máme větev, do níž jsou přidávány commit, které řeší pouze funkcionalitu. V další větvi si programátor může vyřešit vzhled a třetí větev představuje produkční verzi aplikace.

- Při stylování vzhledu grafických komponent vznikne nejsnadněji udržitelný vzhled, když se pro vlastnosti vzhledu použije externí stylopis, který je pro platformu Android uložen v souboru style.xml. Pro vložení textového obsahu do grafických komponent je nejjednodušší vytvořit textové položky v souboru string.xml, na který se budou grafické komponenty odkazovat. Tento soubor také slouží pro vytvoření jazykových mutací vytvořené aplikace. Výsledný vzhled by se neměl inspirovat a používat vzhled z jiných mobilních platforem, protože uživatelé by mohli být při používání aplikace zmatení.

- Při programování výsledných funkcionalit je vhodné používat návrhové vzory, které pomohou vytvořit správný a dobře udržitelný kód, který bude funkcionalita vykonávat.

7 Seznam použité zdrojů

1. Number of Android applications. *AppBrain* [online]. 2015 [cit. 2015-03-22]. Dostupné z: http://www.appbrain.com/stats/number-of-android-apps
2. Jak vypadá Android uvnitř?. *ANDROIDMARKET* [online]. 2011 [cit. 2014-08-05]. Dostupné z: http://www.androidmarket.cz/android/jak-vypada-android-uvnitr-aneb-co-je-rom-kernel-bootloader-a-dalsi/
3. Ide. *Abc lunuxu* [online]. 2006 [cit. 2014-08-05]. Dostupné z: http://www.abclinuxu.cz/slovnik/ide
4. Eclipse. *Root.cz* [online]. 2002 [cit. 2014-08-05]. Dostupné z: http://www.root.cz/clanky/eclipse-2-ide-na-vsechno/
5. Android Studio. *Android Developers* [online]. 2014 [cit. 2014-08-05]. Dostupné z: https://developer.android.com/sdk/installing/studio.html
6. Tools Help. *Android Developers* [online]. 2014 [cit. 2014-08-05]. Dostupné z: http://developer.android.com/tools/help/index.html
7. ADT Plugin. *Android Developers* [online]. 2014 [cit. 2014-08-05]. Dostupné z: http://developer.android.com/tools/sdk/eclipse-adt.html
8. SDK Build Tools. *Android Developers* [online]. 2014 [cit. 2015-03-02]. Dostupné z: https://developer.android.com/tools/revisions/build-tools.html
9. ProGuard. *Android Developers* [online]. 2014 [cit. 2015-03-02]. Dostupné z: https://developer.android.com/tools/help/proguard.html
10. Zipalign. *Android Developers* [online]. 2015 [cit. 2015-03-02]. Dostupné z: https://developer.android.com/tools/help/zipalign.html
11. Platforms. *Android Developers* [online]. 2014 [cit. 2014-08-05]. Dostupné z: http://developer.android.com/tools/revisions/platforms.html
12. Activities. *Android Developers* [online]. 2014 [cit. 2014-08-05]. Dostupné z: http://developer.android.com/guide/components/activities.html
13. Activity. *Android Developers* [online]. 2014 [cit. 2014-08-05]. Dostupné z: http://developer.android.com/reference/android/app/Activity.html#ActivityLifecycle
14. Services. *Android Developers* [online]. 2014 [cit. 2014-08-05]. Dostupné z: http://developer.android.com/guide/components/services.html
15. Vyvíjíme pro Android: Content providery. *Zdroják.cz* [online]. 2012 [cit. 2014-08-05]. Dostupné z: http://www.zdrojak.cz/clanky/vyvijime-pro-android-content-providery/
16. BreadcastReceiver. *Android Developers* [online]. 2014 [cit. 2014-08-05]. Dostupné z: http://developer.android.com/reference/android/content/BroadcastReceiver.html
17. Managing Projects. *Android Developers* [online]. 2014 [cit. 2014-08-05]. Dostupné z: http://developer.android.com/tools/projects/index.html
18. More Resource Types. *Android Developers* [online]. 2014 [cit. 2014-08-05]. Dostupné z: http://developer.android.com/guide/topics/resources/more-resources.html#Dimension
19. String Resources. *Android Developers* [online]. 2014 [cit. 2014-08-05]. Dostupné z: http://developer.android.com/guide/topics/resources/string-resource.html
20. Style Resource. *Android Developers* [online]. 2014 [cit. 2014-08-05]. Dostupné z: http://developer.android.com/guide/topics/resources/style-resource.html
21. Styles and themes on values, values-v11 and values-v14 folders. *Stackoverflow* [online]. 2013 [cit. 2014-08-06]. Dostupné z: http://stackoverflow.com/questions/16624317/styles-and-themes- on-values-values-v11-and-values-v14-folders
22. App Manifest. *Android Developers* [online]. 2014 [cit. 2014-08-05]. Dostupné z: http://developer.android.com/guide/topics/manifest/manifest-intro.html

23. Layouts. *Android Developers* [online]. 2014 [cit. 2015-02-18]. Dostupné z: http://developer.android.com/guide/topics/ui/declaring-layout.html
24. Linear Layout. *Android Developers* [online]. 2014 [cit. 2015-02-18]. Dostupné z: http://developer.android.com/guide/topics/ui/layout/linear.html
25. Relative Layout. *Android Developers* [online]. 2014 [cit. 2015-02-18]. Dostupné z: http://developer.android.com/guide/topics/ui/layout/relative.html
26. Table. *Android Developers* [online]. 2014 [cit. 2015-02-19]. Dostupné z: https://developer.android.com/guide/topics/ui/layout/grid.html
27. Text Fields. *Android Developers* [online]. 2014 [cit. 2015-02-19]. Dostupné z: http://developer.android.com/guide/topics/ui/controls/text.html
28. Buttons. *Android Developers* [online]. 2014 [cit. 2015-02-19]. Dostupné z: http://developer.android.com/guide/topics/ui/controls/button.html
29. Checkbox. *Android Developers* [online]. 2014 [cit. 2015-02-19]. Dostupné z: http://developer.android.com/guide/topics/ui/controls/checkbox.html
30. Radio Buttons. *Android Developers* [online]. 2014 [cit. 2015-02-19]. Dostupné z: http://developer.android.com/guide/topics/ui/controls/radiobutton.html
31. Toggle Buttons. *Android Developers* [online]. 2014 [cit. 2015-02-19]. Dostupné z: http://developer.android.com/guide/topics/ui/controls/togglebutton.html
32. ImageView. *Android Developers* [online]. 2014 [cit. 2015-02-19]. Dostupné z: http://developer.android.com/reference/android/widget/ImageView.html
33. Phones & Tablets. *Android Developers* [online]. 2015 [cit. 2015-03-15]. Dostupné z: https://developer.android.com/design/handhelds/index.html
34. Devices and Displays. *Android Developers* [online]. 2015 [cit. 2015-03-15]. Dostupné z: https://developer.android.com/design/style/devices-displays.html
35. Themes. *Android Developers* [online]. 2015 [cit. 2015-03-15]. Dostupné z: https://developer.android.com/design/style/themes.html
36. App Structure. *Android Developers* [online]. 2015 [cit. 2015-03-16]. Dostupné z: https://developer.android.com/design/patterns/app-structure.html
37. Multi-pane Layouts. *Android Developers* [online]. 2015 [cit. 2015-03-16]. Dostupné z: https://developer.android.com/design/patterns/multi-pane-layouts.html
38. Navigation with Back and Up. *Android Developers* [online]. 2015 [cit. 2015-03-16]. Dostupné z: https://developer.android.com/design/patterns/navigation.html
39. Pure Android. *Android Developers* [online]. 2015 [cit. 2015-03-18]. Dostupné z: https://developer.android.com/design/patterns/pure-android.html
40. What is an Interface?. *The JavaTM Tutorials* [online]. 2015 [cit. 2015-03-09]. Dostupné z: http://docs.oracle.com/javase/tutorial/java/concepts/interface.html
41. Design Pattern Overview. *Tutorialspoint* [online]. 2015 [cit. 2015-03-22]. Dostupné z: http://www.tutorialspoint.com/design_pattern/design_pattern_overview.htm
42. Simply Singleton. *JavaWorld* [online]. 2015 [cit. 2015-03-22]. Dostupné z: http://www.javaworld.com/article/2073352/core-java/simply-singleton.html
43. Observer Design Pattern in Java – Example Tutorial. *JournalDev* [online]. 2013 [cit. 2015-03- 22]. Dostupné z: http://www.journaldev.com/1739/observer-design-pattern-in-java-example- tutorial
44. Design Patterns Facade Pattern. *Tutorialspoint* [online]. 2015 [cit. 2015-03-24]. Dostupné z: http://www.tutorialspoint.com/design_pattern/facade_pattern.htm
45. Java Design Pattern: Decorator – Decorate your girlfriend. *Program Creek* [online]. 2015 [cit. 2015-03-24]. Dostupné z: http://www.programcreek.com/2012/05/java-design-pattern-decorator-decorate-your-girlfriend/
46. Get Started with Publishing. *Android Developers* [online]. 2015 [cit. 2015-03-19]. Dostupné z:

http://developer.android.com/distribute/googleplay/start.html
47. Developer Console. *Android Developers* [online]. 2015 [cit. 2015-03-19]. Dostupné z: http://developer.android.com/distribute/googleplay/developer-console.html

8 Seznam obrázků

Obrázek 1: Architektura operačního systému Android zdroj: http://download.e-bookshelf.de/download/0000/5920/58/L-X-0000592058-0001352109.XHTML/index.xhtml
Obrázek 2: Životní cyklus activity zdroj: http://developer.android.com/reference/android/app/Activity.html#ActivityLifecycle
Obrázek 3: Životní cyklus service zdroj: http://developer.android.com/guide/components/services.html
Obrázek 4: Linear layout zdroj: http://developer.android.com/guide/topics/ui/declaring-layout.html
Obrázek 5: Relative layout zdroj: developer.android.com/guide/topics/ui/declaring-layout.html
Obrázek 6: Table layout zdroj: https://developer.android.com/guide/topics/ui/layout/grid.html
Obrázek 7: TextView zdroj: vlastní
Obrázek 8: CheckBox zdroj: vlastní
Obrázek 9: RadioButton zdroj: vlastní
Obrázek 10: ToggleButton zdroj: vlastní
Obrázek 11: Ukázka multi-pane layouts zdroj: https://developer.android.com/design/patterns/multi-pane-layouts.html
Obrázek 12: Navigace v aplikaci zdroj: https://developer.android.com/design/patterns/navigation.html
Obrázek 13: ukázka vícepanelového rozložení zdroj: vlastní
Obrázek 14: ukázka vícepanelového rozložení zdroj: vlastní
Obrázek 15: UML diagram návrhového vzoru observer zdroj: http://www.journaldev.com/1739/observer-design-pattern-in-java-example- tutorial
Obrázek 16: UML diagram návrhového vzoru fasáda zdroj: http://www.tutorialspoint.com/design_pattern/facade_pattern.htm
Obrázek 17: UML diagram návrhového vzoru dekorátor zdroj: http://www.programcreek.com/2012/05/java-design-pattern-decorator-decorate-your-girlfriend/

9 Seznam pojmů

Pojem	Význam
Action bar	Prvek, který slouží pro pohyb v aplikaci.
Action buttons	Tlačítka v action baru.
Action overflow	Rozbalovací tlačítka v action baru.
Activity	Grafická komponenta se kterou pracuje uživatel.
Android Runtime	Běhové prostředí Android aplikací.
API - Application Programming Interface	Soubor funkcí, které může programátor volat.
Aplication framework	Vrstva, která zpřístupňuje data ostatních aplikací.
Applications	Nainstalované aplikace.
Broadcast receiver	Komponenta, která přijímá sdílená data.
Content provider	Komponenta, která sdílí data mezi aplikacemi.
Dropdown menu	Vysouvací menu.
Fragment	Objekt, který se vkládá do activity.
GIT	Nástroj pro paralelní vývoj kódu.
IDE - Integrated Development Environment	Vývojové prostředí.
Interface	Rozhraní objektu.
Layout	Podklad na který se vynáší grafické prvky uživatelského rozhraní.
Libraries	Systémové knihovny.
Navigation bar	Systémová tlačítka zpět, domu, spuštěné aplikace.
Navigation drawer	Vysouvací prvek z boku displeje.
Observer	Návrhový vzor pozorovatel.
SDK - Software Development Kit	Nástrojů, pro tvorbu aplikací vybrané platformy.
Service	Dlouho běžící komponenta na pozadí.
Singleton	Návrhový vzor jedináček.
Spinners widget	Widget, který se po kliknutí zobrazí vložená data
Status bar	Vrchní lišta, pro ikony notifikací a zobrazení.
Swipe	Tahové gesto.
UP button	Vrchní lišta, která slouží pro ikony notifikací.
View controler	Prvek, který přepíná zobrazené view.