PŘÍRODOVĚDECKÁ FAKULTA UNIVERZITY PALACKÉHO
KATEDRA INFORMATIKY

BAKALÁŘSKÁ PRÁCE

Klient Stagu pro Android

2012 Radim Pešek

Anotace

V práci popisuji problematiku vývoje aplikací pro operační systém Android, kde představím základní stavební kameny aplikací každého projektu pro tento operační systém a toto vše pak demonstruji na kódu vlastní implementace jednoduchého klienta IS/STAG pro tuto platformu. Aplikace přebírá informace z veřejného rozhraní IS/STAG za využití webových služeb nad IS/STAG. Práce rovněž obsahuje představení operačního systému Android a informačního systému IS/STAG.

Děkuji svému vedoucímu práce Mgr. Janu Outratovi, Ph.D. za podnětné rady, vstřícnost a v neposlední řadě také za vypsání tohoto tématu, které mě zaujalo natolik, že se chci vývoji aplikací pro operační systém Android věnovat i v budoucnu.

Obsah

Seznam obrázků

Seznam tabulek

1. Představení projektu

Projekt seznámí čtenáře s problematiku vývoje aplikací pro operační systém Android a poskytne některá málo dokumentovaná fakta o možnostech vývoje pro tuto platformu. Výsledkem práce je jednoduchý klient STAGu, který zprostředkovává data uložená v IS/STAG pomocí webových služeb nad IS/STAG. Klient byl cílen pro potřeby studentů, takže většina funkcí je určena studentům, nicméně jsem se na poslední chvíli rozhodl (v reakci na poznámku mého vedoucího práce) dodělat i pár funkcí pro učitele, aby měla aplikace širší záběr potencionálních uživatelů.

2. Informační systém IS/STAG

IS/STAG je zkratka představující název Informační Systém STudijní AGendy.

2.1. Co je to IS/STAG

Jedná se informační systém, který byl vyvinut Centrem informatizace a výpočetní techniky na Západočeské univerzitě v Plzni, kde byl také poprvé použit a nasazen do provozu. Postupně se rozšířil na další školy a v roce 2010 ho používalo již 16 škol včetně Univerzity Palackého v Olomouci. Informačním systém pokrývá veškerou studijní agendu od přijímacích řízení až po vydání diplomu a to v prezenční, kombinované formě i formě celoživotního vzdělávání. Lze ho použít pro vysoké školy i pro vyšší odborné školy. Informační systém obsahuje tyto moduly:

- studijní programy, obory plány a předměty
- kompletní evidenci studenta
- přijímací řízení
- rozvrhy
- předzápis
- zkoušky
- semestrální práce
- mobility studentů
- evaluace
- předpisy plateb

- evidenci absolventů, kvalifikačních prací a závěrečných zkoušek

Pro práci s IS/STAG je možné zvolit některou z následujících možností:

1. Nativní klient - aplikace nainstalovaná na pracovní stanici. Je k dispozici pro operační systém Microsoft Windows (XP a vyšší). Nativního klienta používají především rozvrháři, studijní oddělení a správci fakult a kateder.

2. Webové stránky a portál - k přístupu stačí mít internetový prohlížeč. Nabízí funkce určené především pro studenty a učitele.

2.2. Možnosti výměny dat s IS/STAG

IS/STAG nabízí možnosti komunikace s dalšímy počítačovými systémy pomocí veřejného rozhraní. V předchozí kapitole jsem popsal komunikaci na úrovni uživatele, ale v této kapitole to bude komunikace orientovaná na stroje. Díky tomuto rozhraní mohou například vyučující, vedoucí si vlastní databázi bodů či zápočtů, na konci semestru naimportovat soubor ve formátu XLS do IS/STAG a není tedy nutné, aby tyto údaje opisoval do IS/STAG. Další možností využití je umístění seznamu předmětů, učitelů či studijních programů z IS/STAG a jejich umístění na stránky kateder - seznamy tak budou vždy odpovídat aktuálnímu stavu dle IS/STAG.

Výstupy z IS/STAG mohou být v následujících formátech: XML, XLS (Microsoft Excel), CSV (obecný textový soubor) či kalendářový formát iCal.

2.3. Popis rozhraní Webových služeb

Webové služby jsou z uživatelského pohledu místa na internetu s konkrétní adresou (URL), kde běží služba zpracovavající požadavky. Požadavkem může být operace čtení nebo zápisu dat do IS/STAG. Webové služby nad IS/STAG poskytují dva typy aplikačního rozhraní: REST a SOAP. Prvně jmenovaný typ rozhraní je blíže uživatelům a typ SOAP je převážně pro komunikace typu stroj-stroj.

Protože ve své aplikaci Klient Stagu pro Android používám typ rozhraní REST a výstupní formát XML, uvedu příklad přímo z vlatní aplikace tj. rozhraní REST. Mějme tedy následující URL:

```
https://stagservices.upol.cz/ws/services/rest/predmety/
getPredmetyByStudent?osCislo=R090374\&semestr=LS
```

Popis jednotlivých částí URL:

- https://stagservices.upol.cz/ws/services/ - základní adresa webových služeb nad IS/STAG

- rest/predmety/ - adresa konkrétní služby
- getPredmetyByStudent - název konkrétní metody, kterou získáváme nebo zasíláme data (v tomto případě získáváme)
- osCislo=R090374&semestr=LS - parametry metody (osobní číslo studenta R090374 a parametr semest udává letní semestr)

Jako výsledek obdržíme od serveru odpověď ve formátu XML, která bude obsahovat všechny předměty studenta s osobním číslem R090374 v letním semestru. Obsah takového XML souboru vypadá následovně:

```
<ns1:getPredmetyByStudentResponse>
<predmetyStudenta>
  <predmetStudenta>
   <zkratka>YSOFI</zkratka>
   <nazev>Softwarové inženýrství</nazev>
   <katedra>KMI</katedra>
   <rok>2011</rok>
   <kredity>9</kredity>
 </predmetStudenta>
.
.
.
 <predmetStudenta>
   <zkratka>YINFS</zkratka>
   <nazev>Informační systémy</nazev>
   <katedra>KMI</katedra>
   <rok>2011</rok>
   <kredity>9</kredity>
 </predmetStudenta>
</predmetyStudenta>
</ns1:getPredmetyByStudentResponse>
```

Tato data již je možné strojově zpracovávat a použít je ve svých aplikacích například na webových stránkách nebo v mém případě v aplikaci pro platformu Android.

Nejprve jsem čerpal informace ohledně IS/STAG z dokumentace webových služeb Univerzity Palackého [5], nicméně jsem se ještě poohlédl jinde a našel jsem dokumentaci webových služeb Západočeské Univerzity v Plzni [6], která měla bohatší možnosti a byla k dispozici i podrobnější dokumentace. Pravděpodobně je to díky tomu, že na této univerzitě IS/STAG vznikl, a proto je zde nejlépe zdokumentován.

3. Android

3.1. Co je Android

Android je operační systém určený pro mobilní zařízení, který je založený na Linuxovém jádře verze 2.6. Vznikl spojením firmy Android Inc. s firmou Google Inc. a jejich spoluprácí - Google odkoupil původní firmu Android Inc. a udělal z ní v roce 2005 svou dceřinou společnost. V průběhu roku 2007 byl prezentován první produkt tohoto spojení - Android, otevřená platforma pro mobilní zařízení. Ten samý rok byl představen také nástroj pro vývojáře Android SDK, který byl vydán pod licencí open-source. První komerčně dostupný telefon s Androidem byl vyroben firmou HTC a uveden na trh koncem roku 2008 a s ním i SDK verze 1.0. V době vzniku této práce je nejrozšířenější verzí Androidu verze 2.3.x s kódovým označením Gingerbread - více informací o všech dostupných verzích operačního systému Android jsem shrnul do tabulky 1. v sekci Přehled verzí.

Operační systém Android poznáte podle znaku zeleného robota, jak ilustruje obrázek 1. - logo operačního systému Android. Tento symbol je používán i výrobci hardwaru, kteří jím označují zařízení s nainstalovaným operačním systémem Android.

Obrázek 1. Logo operačního systému Android

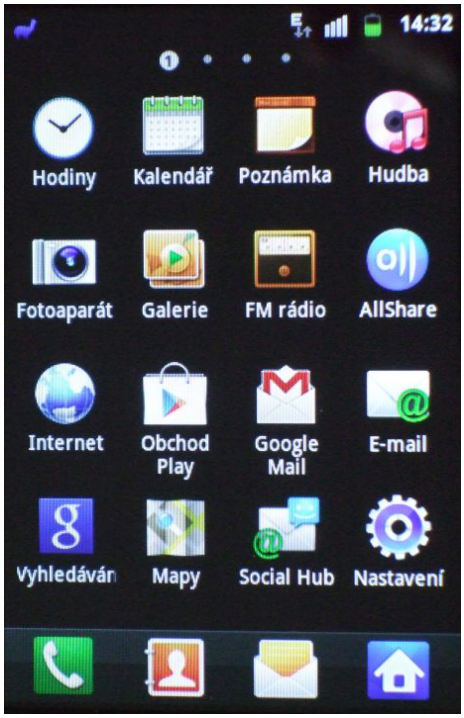
Na obrázku 2. je ukázka prostředí operačního systému Android zobrazující nainstalované aplikace a na obrázku 3. je zobrazeno menu Nastavení, které umožňuje uživateli nastavit chování mobilního zařízení. Konkrétní zobrazení se může lišit podle výrobce, protože každý výrobce si prostředí Androidu vylepšuje svými nástavbami tak, aby uživatelům poskytli v základní výbavě mobilního zařízení více možností či funkcí.

3.1.1. Architektura Androidu

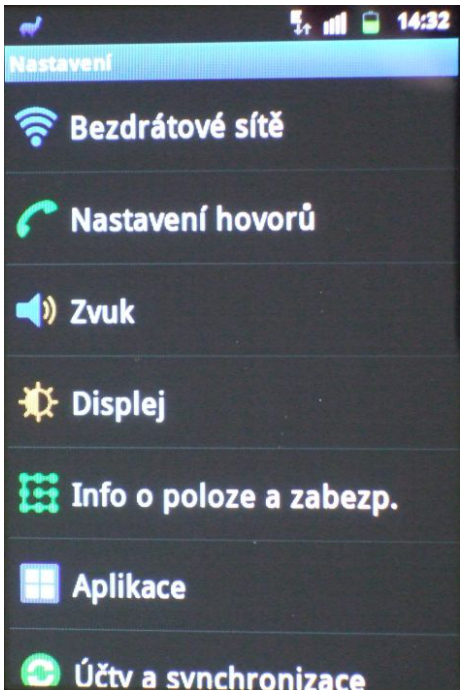
Architektura Androidu je pětivrstvá a tvoří ji tyto vrstvy (vyjmenovány od nejnižší vrstvy): jádro systému, knihovny, Android Runtime (Dalvik), Application framework a nejvyšší vrstvu tvoří tzv. základní aplikace.

Jádro systému je, jak již jsem zmínil, založeno na redukovaném Linuxovém jádře verze 2.6 a obsahuje správu paměti, sítí, procesů, zabudované ovladače. Neobsahuje však grafické uživatelské rozhraní X Window.

Vrstva knihoven je poskytována vývojářům prostřednictvím vrstvy Application framework a obsahuje knihovny pro přehrávání multimediálních souborů, knihovny webového prohlížeče, libc - přizpůsobená pro mobilní zařízení,



Obrázek 2. Ukázka operačního systému Android



Obrázek 3. Menu Nastavení operačního systému Android

odhlehčený databázový engine SQLite, secure socket layer, Knihovnu pro vykreslování bitmapových a vektorových fontů a OpenGL knihovnu pro vykreslování 3D grafiky.

Android Runtime vrstva má v sobě virtualizační stroj známý jako Dalvik (Dalvik Virtual Machine zkráceně DVM), který využívá základní funkce jádra jako je práce s vlákny a správce paměti. Dalvik vznikl kvůli hlavně kvůli licenčním právům na Java Virtual Machine (JVM), ale také kvůli efektivnějšímu řízení spotřeby a výkonu či optimalizaci požadavků na pamět. Všechna tato kritéria hrají klíčovou úlohu u všech mobilních zařizení. Zjednodušeně řečeno každá aplikace spuštěná na Androidu má také svůj vlastní proces a ten běží ve svém DVM.

Vrstva Application framework je nejdůležitější vrstvou pro vývojáře. Zpřístupňuje totiž vývojáři například dostupné služby i prvky uživatelkého rozhraní.

Nejvýše je vrstva základních aplikací, kterou využívají běžní uživatelé operačního systému tj. jedná se o aplikace dodané výrobcem mobilního zařízení nebo později nainstalované balíčky z oficiálního Obchodu Play (dříve známého jako Android Market) či z jiných zdrojů. Přikladem aplikace z jiného zdroje je i moje aplikace Klient Stagu pro Android. Při popisu architektury operačního systému Android jsem vycházel z informací na stránkách Wikipedie [8].

3.2. Využití Androidu

Android byl původně určený pro chytré telefony. Nicméně s příchodem nových mobilních zařízení se jeho použití rozšířilo i na tablety. Někteří výrobci hardwaru jej také začali dodávat do cenově dostupných přenosných počítačů, protože je pro uživatele zdarma, což snižuje cenu těchto zařízení a zároveň napomáhá dalšímu rozšíření operačního systému Android mezi bežné uživatele. S popularitou Androidu se také začalo rozšiřovat portfolio produktů, které jsou na trhu dostupné, s tímto operačním systémem. Android se začíná dostávat do televizních přijímačů, herních konzolí a příští rok by se měl začít uplatňovat také v automobilech místo zastaralých a velmi drahých navigačních systémů.

3.3. Přehled verzí

Tabulka zobrazuje přehled klíčových verzí Androidu s jejich kódovými označeními, úrovně API, data vydání a podíl dané verze na trhu. Údaje o podílu jednotlivých verzí na trhu jsou ke dni 5. března 2012 (zdroj: http://developer.android.com[4]). Z tabulky je zřejmé, že nejrozšířenější verzí na trhu je verze 2.3.x s kódovým označením Gingerbread, která má 62% podíl na trhu. S přihlédnutím k tomuto faktu, jsem svou aplikaci vytvořil pro tuto nejpočetnější skupinu zařízení a také proto, že takové zařízení vlastním a proto vše mohu řádně vyzkoušet.

Do verze 2.3.x byl operační systém Android optimalizován pouze pro chytré telefony, ale protože se na trhu velmi rozšířil nový druh mobilního zařízení - tablet, tak přišla firma Google na trh s verzí 3.x.x s kódovým označením Honeycomb, kde byla přidána optimalizace pro tablety a další zařízení s velkou obrazovkou. Protože je verze Honeycomb určená pro tablety, tak se s ní téměř na chytrých telefonech nesetkáte. Nejnovější chytré telefony používají až verzi 4.0.x s kódovým označením Ice Cream Sandwich.

Verze	**Kódové označení**	**API úroveň**	**Podíl na trhu**	**Datum vydání**
Android 4.0.X	Ice Cream Sandwich	14-15	1,6%	19.10.2011
Android 3.X.X	Honeycomb	11-13	3,3%	22.2.2011
Android 2.3.X	Gingerbread	9-10	62,0%	6.12.2010
Android 2.2.X	Froyo	8	25,3%	20.5.2010
Android 2.1	Eclair	7	6,6%	26.10.2009
Android 1.6	Donut	4	0,8%	15.9.2009
Android 1.5	Cupcake	3	0,4%	30.4.2009

Tabulka 1. Přehled klíčových verzí Androidu

4. Vývoj aplikací pro platformu Android

Kapitola obsahuje popis vybraných a použitých stavebních prvků používaných při tvorbě aplikace Klient Stagu pro Android.

4.1. Jak začít programovat

Nejvíce mě v začátcích pomohla kniha Android 2 - Průvodce programováním mobilních aplikací od Marka L. Murphy [1], která je psaná pro začátečníky se základní znalostí Javy. Javu jsem se učil převážně z knihy Sharona Zakhoura a kolektivu[2]. Po zvládnutí základů Javy a Androidu se pro mě stal největším zdrojem informací web pro vývojáře [4], který spravuje přímo firma Google. Stránky pro vývojáře od Googlu jsou jako referenční příručka, a proto je doporučuji používat. Z praktického hlediska je užitečný seriál Vyvíjíme pro Android [3], který je skvělým úvodem do názvosloví a jsou zde i příklady praktických a používaných kódů.

4.1.1. Bez čeho se neobejdete

Neobejte se bez znalostí jazyka Java (nemusí být příliš hluboká), znalosti XML, znalosti prostředí Androidu a problematiky tvorby a stuktury projektů pro operační systém Android.
Ze softwarové výbavy budete dále budete potřebovat:

- nainstalovat *Java SE Development Kit 7* ze stránek firmy Oracle
- *Android SDK Tools*, který je ke stažení na *http://developer.android.com/sdk/*. Tento balík obsahuje vše potřebné, co se týka Androidu. Najdete v něm *Android SDK Manager* sloužící pro správu balíčků, kterým si můžete stáhnout knihovny té verze Androida pro kterou bude aplikaci vyvíjet. Ještě v *Android SDK Tools* naleznete *AVD Manager* (Android Virtual Device Manager). AVD Manager umožňuje správu vašich virtuálních zařízení, na kterých budete v emulátoru zkoušet svoje aplikace.
- mít alespoň textový editor, v kterém budete psát zdrojové kódy. Pokud se tedy nerozhodnete pro některé z možných vývojových prostředí v kapitole 4.1.2.

To, co jsem právě vyjmenoval je úplné minimum, nutné k tomu, abyste se mohli pustit do své první aplikace.

4.1.2. Přehled možných vývojových prostředí

Díky vývojovému prostředí bude vaše programování mnohem efektivnější a bude obsahovat méně chyb, protože v prostředí editoru je obsažena průběžná syntaktická kontrola kódu, různé nápovědy názvů metod, automatické či poloautomatické "dokončovače"a mnoho dalšího. V případě, že se tedy rozhodnete využít komfortu vývojového prostředí, což vřele doporučuji - obzvláště pokud s Androidem začínáte, tak máte na výběr několik možností.

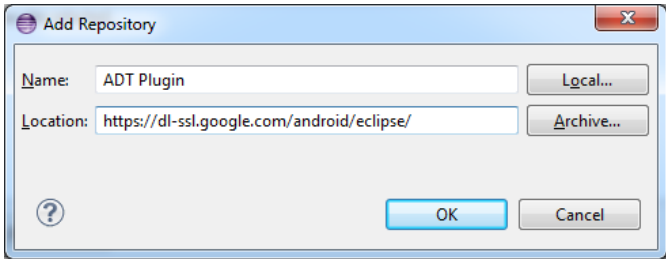
Jak jsem v předchozí kapitole 4.1.1. tvrdil, že je nutná znalost jazyka Java, tak to není až tak úplně pravda a je tady ještě jedna možnost - a to pokud umíte programovat v C#. Pokud se chcete vydat touto cestou, pak vám stačí mít nainstlalován .NET a Visual Studio 2010 a do něj si nainstalovat add-on *MonoDroid*. MonoDroid nelze provozovat s Visual Studiem Express. Jak *MonoDroid* zprovoznit a začít s ním programovat aplikace pro Android pěkně popsal Mahesh Chand ve svém článku *Building Android Applications using C#*.

Protože drtivá většina vývojářů si zvolila programovat aplikace pro Android v jazyce Java, zvolil jsem tuto cestu i já. Dalším zásadním důvodem proč programovat v jazyce Java je fakt, že Java kód se v Androidu spouští "přímo"a váš kód nemusí procházet "konvertorem"jakým je MonoDroid.
Nejpoužívanějším, a z mého pohledu i nejpropracovanějším, vývojovým prostředím je produkt *Eclipse* a proto jsem si ho zvolil pro vývoj své aplikace Klient Stagu pro Android. Podrobněji se mu budu věnovat v následující kapitole 4.1.3.

Další možností je vývojové prostředí *Netbeans* od firmy Oracle, které je open source. Podporuje více programovacích jazyků, ale primárně je určen pro vývoj aplikací v jazyce Java.

4.1.3. Vývojové prostředí Eclipse

Eclipse je open source vývojová platforma, která se zaměřuje na programování v jazyce Java. Eclipse má možnost rozšiřitelnosti pomocí pluginů a to je právě jeho silnou stránkou, protože díky pluginům lze Eclipse rozšířit tak, aby se v něm dalo vyvíjet pro platformu Android. Tento plugin se jmenuje ADT Plugin (ADT - Android Development Tools) a lze ho nainstalovat pohodlně přímo z prostředí Eclipse. Nejprve je nutné nainstalovat Eclipse, takže si jej nejdříve stáhneme z *http://www.eclipse.org/downloads/*. Je zde na výběr několik verzí produktu Eclipse - doporučuji verzi Eclipse Classic. Po stažení a naistalování produktu Eclipse jej spusťte a nastavte si pracovní adresář k čemuž vás Eclipse vyzve. Jakmile se dostanete do vývojového prostředí nainstalujte si ADT Plugin. To uděláte následovně: zvolte v menu *Help* > *Install New Software*, v okně stiskněte tlačítko *Add* a vyplňte údaje dle obrázku 4. a potvrďte. V okně Available Software zatrhněte *Developer Tools*, potvrďte a dokončete instalaci pluginu. Po instalaci ADT Pluginu je nutné restartovat Eclipse. Tímto máte prostředí



Obrázek 4. Přidání ADT Pluginu

Eclipse připraven a můžete se pustit do programování pro platformu Android.

4.1.4. Založení nového projektu v Eclipse

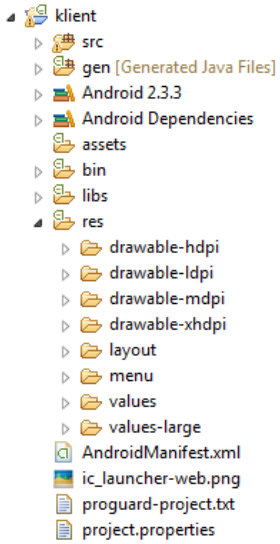
Založení projektu je díky průvodci jednoduché a rychlé narozdíl od ručního vytváření všech potřebných souborů a adresářů. Nový projekt vytvoříme takto: zvolíme v menu položku *File* > *New* > *Project*, v okně si nalistujem sekci Android a tam zvolíme *Android Application Project*. V dalším kroku musíme vyplnit název aplikace, jméno projektu a jméno balíčku se nám předvyplní automaticky, ale lze je, v případě potřeby, změnit. Potvrdíme a v následujícím kroku si můžeme zvolit ikonu naší aplikace, pod kterou ji najdeme v seznamu programů. Další kroky můžeme potvrdit a zanechat implicitní hodnoty. Tímto jsme vytvořili základní adresářovou strukturu naší nové aplikace včetně všech potřebných souborů, výsledek je vidět na obrázku 5.

4.1.5. Popis adresářové struktury Android projektu

Každý projekt obsahuje pevně danou strukturu adresářů a souborů. Tu je vidět na obrázku 5. Postupně si ji představíme po jednotlivých adresářích od kořenového adresáře:

- `AndroidManifest.xml`: XML dokument popisující naši aplikaci a její komponenty (všechny Activity, práva aplikace - permissions, služby atp.)
- `/src/`: v tomto adresáři jsou uloženy všechny zdrojové soubory (*.java) našich Activity, vlastních tříd atd. Příklad takovéhoto souboru je v kapitole 4.3.1.
- `/gen/`: sem si překladače ukládají zdrojový kód, který vygenerují
- `/assets/`: adresář pro statické soubory, které chceme přibalit k aplikaci určené k použití na zařízení
- `/bin/`: adresář, kde se aplikace uchovává po jejím zkompilování

- `/libs/`: zde se ukládají knihovny (např. třetích stran), které naše aplikace potřebuje
- `/res/`: toto je nejpoužívanější adresář společně z adresářem src. Jsou zde všechny potřebné prostředky jako jsou ikony, soubory s návrhem grafického uživatelského rozhraní - layouty, návrhy menu, seznamy řetězců atd.



Obrázek 5. Adresářová struktura projektu

Pojďme se nyní podrobněji podívat do obsahu adresáře `/res/`:

- `/res/layout`: zde jsou XML soubory s návrhy grafického uživatelského rozhraní (GUI)
- `/res/menu`: opět místo pro XML soubory, ale tentokrát se specifikací menu
- `/res/values`: zde jsou XML soubory s řetězci, poli apod.
- `/res/drawable`: adresář pro obrázky PNG, JPG atp.
- `/res/xml`: adresář s dalšími XML soubory, které chceme k aplikaci přibalit
- `/res/raw`: určeno pro různé typy souborů

Při odkazování se na prostředky (resources), z adresáře `/res/`, v programovém kódu používáme tento systém (názvy souborů se uvádí bez přípony):

- `/res/layout`: R.layout.nazevsouboru
- `/res/menu`: R.menu.nazevsouboru

- `/res/values`: R.nazevsouboru.nazevhodnotyvsouboru
- `/res/drawable`: R.drawable.nazevsouboru
- `/res/xml`: R.xml.nazevsouboru
- `/res/raw`: R.raw.nazevsouboru

Při odkazování se na prostředky (resources), z adresáře `/res/`, v XML layoutech používáme tento systém (názvy souborů se uvádí bez přípony):

- `/res/values`: @nazevsouboru/nazevhodnotyvsouboru - například @string/btnWizardText nás odkazuje do adresáře /res/values na soubor string.xml (jedná se o soubor s řetězci - vhodné pro multilinguální programy) a hodnotu řetězce btnWizardText
- `/res/drawable`: @drawable/nazevsouboru

4.2. Možné přístupy k programování GUI

Vývojář se může postavit, k programování GUI pro platformu Android, dvěma různými způsoby. Prvním z nich je programové řešení a druhé deklarativní. Porovnání obou způsobů demonstruji na tvorbě stejného prvku GUI a to na menu. Vizuální výsledek programového i deklarativního způsobu řešení je stejný a je jím menu na obrázku 13.

4.2.1. Programový způsob tvorby GUI

Tento způsob spočívá v tom, že celá tvorba menu je pouze v programovém kódu naší Activity. Menu tedy generujeme přímo kódem, který uvedeme v metodě *onCreateOptionsMenu*. Výhodou tohoto řešení je na první pohled menší množství kódu, které je navíc pěkně pohromadě v naší Activity. Nevýhodou tototo způsobu tvorby GUI je fakt, že v kódu Activity máme dohromady jak funkční kód, tak definici grafiky. Z toho plyne, že případné změny (například pořadí položek v menu) musíme řešit zásahem v kódu Activity.
Kód nutný pro vytvoření menu pomocí programového řešení (umístěný v naší Activity):

```
@Override
public boolean onCreateOptionsMenu(Menu menu)
{
 super.onCreateOptionsMenu(menu);
 menu.add(0,OPTIONS_ID,0,R.string.menu_options)
 .setIcon(android.R.drawable.ic_menu_preferences);
 menu.add(0,HELP_ID,0,R.string.menu_help)
```

```
 .setIcon(android.R.drawable.ic_menu_help);
 menu.add(0,EXIT_ID,0,R.string.menu_exit)
 .setIcon(android.R.drawable.ic_menu_close_clear_cancel);
 menu.add(0,ABOUT_ID,0,R.string.menu_about)
 .setIcon(android.R.drawable.ic_menu_info_details);
 return true;
}
```

Další, nikoliv zásadní, nevýhodou programového řešení je jiný způsob identifikace jednotlivých prvků. Prvky jsou identifikovány čísly, které si musíme definovat. Například OPTIONS_ID=0, HELP_ID=1 atd. Na toto je nutné mystet při obsluze události *onOptionsItemSelected*. U deklarativního způsobu se při obsluze této události odvoláváme na přístušné id, které jsme si v XML souboru definovali (parametr android:id).

Výhodou programového řešení je, že s ním můžeme řešit i situace, které deklarativním způsobem vůbec vytvořit nelze. Více se o této výhodě zmiňuji v kapitole 4.2.2.

4.2.2. Deklarativní řešení tvorby GUI

Toto řešení je založeno na součinnosti programového kódu Activity a XML souborů. Odstraňuje některé z nedostatků programového způsobu tvorby GUI. Tím nejzásadnějším je fyzické oddělení definice grafiky a funkčního kódu do dvou různých souborů. Z hlediska praktického je deklarativní řešení výhodnější, i když je pro vývojáře o trochu pracnější. Tento způsob programování jsem si zvolil i já, a proto jej uvidíte i v mých vzorových kódech v této práci. Kód umístěný v naší Activity pro vytvoření menu pomocí deklarativního řešení:

```
@Override
public boolean onCreateOptionsMenu(Menu menu)
{
 MenuInflater inflater = getMenuInflater();
 inflater.inflate(R.menu.main_menu, menu);
 return true;
}
```

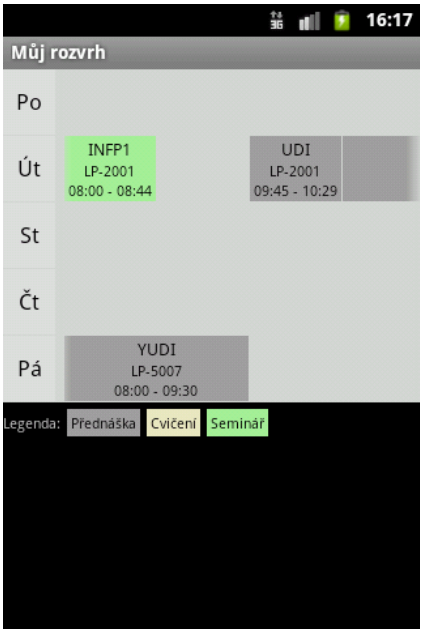
Jak je vidět, v kódu se pouze "odkazujeme"na soubor main_menu.xml s grafickým návrhem menu (R.menu.main_menu = /res/menu/main_menu.xml).

Kód příslušného XML souboru main_menu.xml umístěného v adresáři */res/menu*:

```
<?xml version="1.0" encoding="utf-8"?>
<menu xmlns:android="http://schemas.android.com/apk/res/android" >
```

```
 <item android:id="@+id/options"
  android:icon="@android:drawable/ic_menu_preferences"
  android:title="@string/menu_options" />
 <item android:id="@+id/help"
   android:icon="@android:drawable/ic_menu_help"
   android:title="@string/menu_help" />
 <item android:id="@+id/exit"
  android:icon="@android:drawable/ic_menu_close_clear_cancel"
  android:title="@string/menu_exit" />
 <item android:id="@+id/info"
  android:icon="@android:drawable/ic_menu_info_details"
  android:title="@string/menu_about" />
</menu>
```

Nevýhodou tohoto způsobu tvorby GUI je, že ho nelze použít na taková zobrazení, kdy přesně neznáme počet a umístění prvků. Příkladem z mé aplikace Klient Stagu pro Android je Rozvrh hodin (obrázek 6.) - zde dopředu nevíme kolik předmětů má daný student v rozvrhu, jak jsou dlouhé ani den v týdnu každého předmětu. Nicméně to lze řešit kompromisem, kdy jako základ použijeme deklarační způsob a předměty generujeme programově.



Obrázek 6. Rozvrh hodin

4.3. Základní stavební kameny

4.3.1. Activity

Activity je hlavní část každé aplikace, bez ní nelze žádnou aplikaci spustit. Activity si můžeme představit jako malířské plátno nebo jednu obrazovku naší aplikace. Activity může obsahovat třeba jen funkční kód a nebo obsahuje i ovládací prvky jako jsou tlačítka, přepínače, textová pole a další. Activity jsou ukládány v rámci projektu do podadresáře *src*, kde je každá Activity uložená ve svém souboru například UkazkaActivity.java. Nejjednodušší kód Activity by mohl být například tento:

```
package com.example.android.mojeaplikace;

import android.app.Activity;
import android.os.Bundle;

public class UkazkaActivity extends Activity {

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
    }

}
```

První sekce *package* je jméno balíku což je vlastně jméno naší aplikace - v našem případě mojeaplikace. V druhé časti *import* se vkládají potřebné knihovny, které v rámci naší Aktivity budeme používat. A v poslední třetí části je definice třídy *UkazkaActivity*, která je potomkem třídy *android.app.Activity* jenž importujeme v předchozím bloku. Z toho vyplívá, že každá naše Activity je potomkem zmiňované třídy *android.app.Activity*. Uvnitř naší třídy definujeme metody, kde jedna z těch základních je metoda *onCreate*, která se volá ihned po vytvoření UkazkaActivity. V *onCreate* je důležité zavolat metodu *super.onCreate(savedInstanceState)*, kterou je vhodné volat co nejdříve. Následující metoda *setContentView(R.layout.main)* nám spojuje naši *UkazkaActivity* s XML souborem layoutu. Layout soubor našeho konkrétního příkladu se jmenuje *main.xml* a najdete ho ve vašem projektu v adresáři */res/layout*.

Aby UkazkaActivity něco dělala, je nutné do zmiňovaného souboru *main.xml* vložit například takovýto kód:

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent" >
```

```
    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Ahoj světe!" />

</LinearLayout>
```

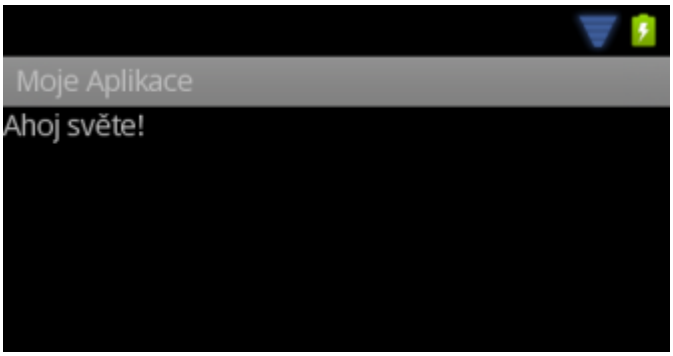
V XML souboru definujeme jen prvky grafického uživatelského prostředí (GUI), takže jsou odděleny programová část a GUI. V našem příkladu je definován jen jeden prvek TextView, který zobrazuje text uvedený v parametru *android:text*. Poslední, co musíme definovat, je soubor *AndroidManifest.xml*. V něm musí být zapsané všechny Activity a u jedné z nich musí být uveden element *intent-filter*. Příklad jednoduchého souboru *AndroidManifest.xml* s jednou Activity - UkazkaActivity:

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.android.mojeaplikace"
    android:versionCode="1"
    android:versionName="1.0" >

    <uses-sdk android:minSdkVersion="10" />

    <application
        android:icon="@drawable/ic_launcher"
        android:label="@string/app_name" >
        <activity
            android:name=".UkazkaActivity"
            android:label="@string/app_name" >
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>
</manifest>
```

Všechny tyto tři soubory tvoří, samozřejmě po zkompilování, aplikaci pro operační systém Android, která po spuštění zobrazí v levém horním rohu text *Ahoj světe!*. Spuštěnou aplikaci ilustruje obrázek 7.

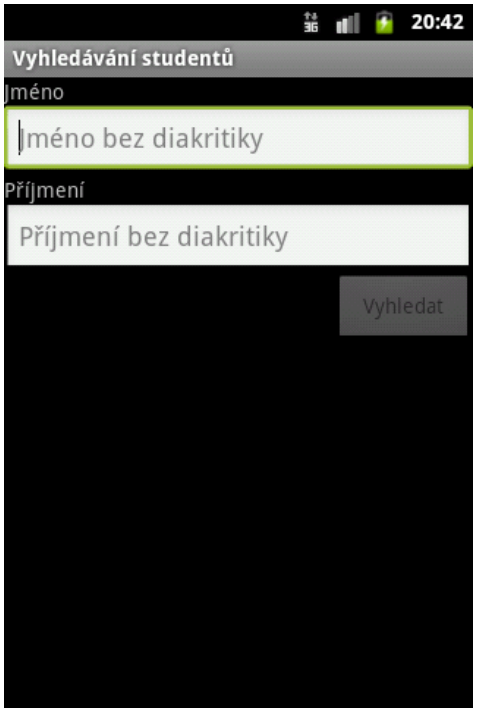


Obrázek 7. Obrazovka ukazkové Activity

4.3.2. Widget

Android nabízí sadu prvků grafického uživatelského rozhraní (GUI), obdobně jako je tomu například na platformě Windows, která obsahuje základní Widgety. Jsou to například tyto prvky (nejedná se o kompletní seznam, ale výběr nejpoužívanějších prvků):

- *TextView* (popisek) - na obrazovce aplikace zobrazí popisný text. Tento prvek je vidět na obrázku 7. a je reprezentován textem *Ahoj světe!*
- *Button* (tlačítko) - tento prvek vytvoří běžné tlačítko, které bez potřebného kódu ve vaší Activity nebude nic spouštět.
- *ImageView* (obrázek) - jedná se o analogii prvku TextView, ale výsledkem je obrázek
- *ImageButton* (obrázkové tlačítko) - opět se jedná o analogii prvku Button, jen s tím rozdílem, že je na tlačítku umístěný obrázek
- *EditText* (textová pole) - pomocí textových polí můžeme aplikaci zadávat jakýkoliv text. Na obrázku 8. je například použit v Activity pro vyhledávání studentů.
- *CheckBox* (zatržítka) - běžně známé zaškrtávací políčko používané pro volby ano/ne, zapnuto/vypnuto atp. Příkladem je položka *Průvodce nastavením po spuštění* na obrázku 16.
- *RadioButton* (kulaté přepínače) - jedná se o prvky, kterými se vybírá z několika voleb právě jedna volba (viz obrázek 9.). V XML souboru se sdružují pomocí elementu *RadioGroup*.



Obrázek 8. Ukázka widgetu EditText a Button

Obrázek 9. Ukázka widgetu RadioButton

4.3.3. Intent

Intent (záměr) je instance třídy *android.content.Intent* a jedná se vlastně o asynchonní zprávy, které umožňují požadovat jedné komponentě systému Android funkci od jiné komponenty systému Android. Typickým příkladem záměru je zaslaní zprávy, aby systém Android spustil nějakou další Activity. Příklad záměru - spuštění Activity s názvem Help.class, která může být umístěna například v metodě *onClick* nějakého tlačítka v aplikaci:

```
startActivity(new Intent(this, Help.class));
```

Dalším příkladem využití záměru je například přidání položky do kalendáře. Takový kód by vypadal takto:

```
Intent intent = new Intent(Intent.ACTION_EDIT);
intent.setType("vnd.android.cursor.item/event");
intent.putExtra("beginTime", cal.getTimeInMillis());
intent.putExtra("allDay", true);
intent.putExtra("endTime", cal.getTimeInMillis()+60*60*1000);
intent.putExtra("title", fullObject.getTxtBig().toString());
intent.putExtra("eventLocation", location);
startActivity(intent);
```

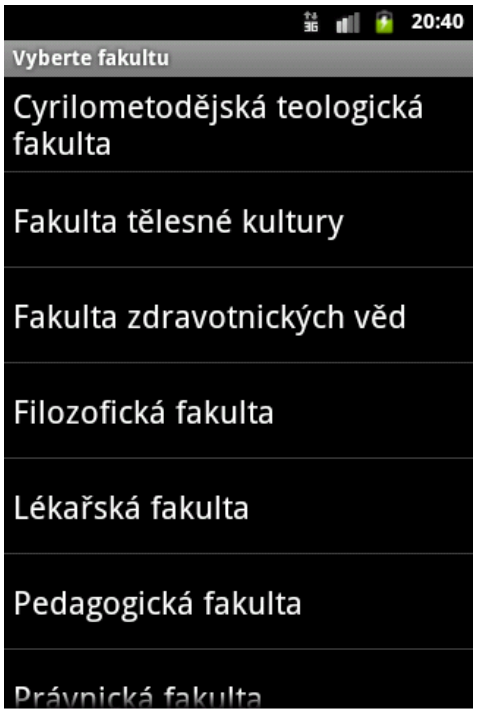
Záměr může také obsahovat data například při volání komponenty internetového prohlížeče je součástí záměru také URL, které má prohlížeč zobrazit. Nebo můžeme předávat pomocí záměru parametry z jedné Activity do druhé - toto demonstruji na následujícím kódu:

```
i = new Intent(HlavniMenuActivity.this, DepartmentInfo.class);
i.putExtra("requestCode", 0);
startActivityForResult(i,0);
```

Tento kód spustí z Activity HlavniMenuActivity další Activity DepartmentInfo a předává jí parametr requestCode.

4.3.4. ListView

ListView je skupinové zobrazení položek, které lze rolovat (scrollovat). Používá se především pro různé výpisy seznamů položek. Jde o jeden z nejpoužívanějších prvků v aplikacích na platformě Android. ListView může zobrazovat jednoduchý seznam (viz obrázek 10.), nicméně může zobrazovat i složitější datové struktury (několik položek v jednom bloku) jako je na obrázku 11. - zobrazuje skupinu pěti údajů a jedné poznámky, která je přítomna jen u některé ze skupin.



Obrázek 10. Ukázka jednoduchého seznamu ListView

Obrázek 11. Ukázka komplexního seznamu ListView

Jednotlivé položky jsou automaticky vkládány do seznamu díky objektu *Adapter*, který bere data z určeného zdroje (pole, databáze) a konvertuje je do odpovídajícího zobrazení. V případě jednoduchého seznamu je vhodné využít toho, že ListView je potomkem třídy ListActivity a pak není nutné si programovat vlastní *Adapter*. U složitějších seznamů položek, jako je již zmíněný ListView na obrázku 11., je nutné si napsat vlastní *Adapter*. Toto budu blíže popisovat v následující kapitole *Adapter*. Příklad jednoduchého ListActivity:

```
import android.app.Activity;
import android.os.Bundle;
import android.widget.ArrayAdapter;
import android.widget.ListView;

public class VyberFakultuActivity extends ListActivity {
  ArrayList<String> facultiesList = new ArrayList<String>();
  public void onCreate(Bundle icicle) {
   super.onCreate(icicle);
   ... //naplnění seznamu facultiesList
   this.setListAdapter(new ArrayAdapter<String>
     (this,android.R.layout.simple_list_item_1,
           facultiesList.toArray(new String[0])));
  }
.
.
}
```

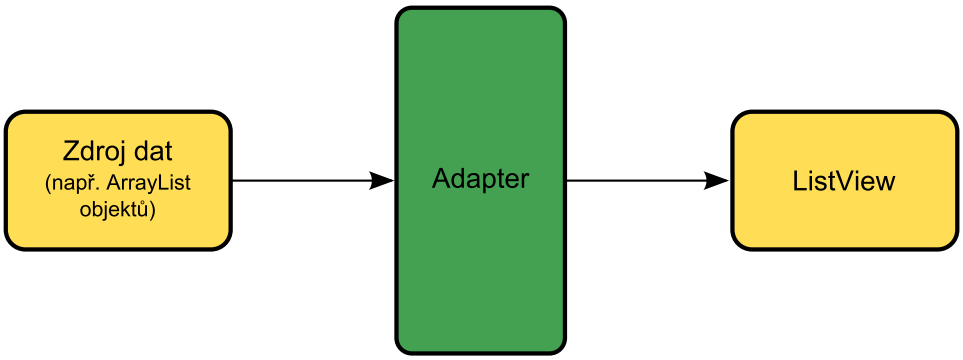
Tento jednoduchý seznam položek dokonce ani nevyžaduje příslušný XML soubor s layoutem narozdíl od složitějších seznamů, kde je nutné kromě naprogramování *Adapter*u ještě vytvořit 2 soubory s XML layoutem: jedné skupiny položek v seznamu (jeden "řádek"seznamu) a pak celkový layout seznamu.

4.3.5. Adapter

Adapter se používá při tvorbě komplexních ListView seznamů a jeho rolí je, že "sedí"mezi našim zdrojem dat a naší obrazovkou. Příkladem takového seznamu je například seznam zkouškových termínů na obrázku 11., kterou použiji jako demonstrační. Schematicky je role adaptéru znázorněna na obrázku 12.

Ve své Activity ExamList v metodě *onCreate* definujeme pomocí *setContentView* příslušný XML layout (exam_list - ležící v adresáři */res/layout*). Jeho kód je jednoduchý a obsahuje pouze jeden objekt a tím je právě ListView.
Kód souboru exam_list.xml:

```
<?xml version="1.0" encoding="utf-8"?>
```



Obrázek 12. Blokové schéma Adapteru

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:orientation="vertical" >
    <ListView
        android:id="@+id/examListView"
        android:layout_width="fill_parent"
        android:layout_height="fill_parent" />
 </LinearLayout>
```

Zmiňovaný objekt ListView je v XML souboru pojmenován parametrem *android:id* názvem *examListView*, které musíme v metodě *onCreate* taktéž definovat, aby bylo možné nastavit adaptér. Tato definice je vidět na následujícím kódu v druhém řádku (tento kód je umístěn v *onCreate* metodě naší Activity).

```
setContentView(R.layout.exam_list);
final ListView lv = (ListView) findViewById(R.id.examListView);
lv.setAdapter(new ExamListBaseAdapter(ExamList.this, itemsList));
```

Třetím řádkem nastavujeme našemu zobrazení *examListView*, definovaném v souboru exam_list.xml, svůj adaptér s názvem *ExamListBaseAdapter*. Argumenty našeho adaptéru jsou *ExamList.this* (název naší Activity) a ArrayList zkouškových termínů - *itemsList*.

Ještě než se podíváme na kód našeho adaptéru (*ExamListBaseAdapter*), ukáži XML kód layout souboru exam_list_row.xml, který definuje, jak má vypadat řádek našeho seznamu.
Kód souboru exam_list_row.xml, ležícího adresáři */res/layout*:

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
```

```
android:layout_height="match_parent"
android:padding="5dp" >

<TextView
    android:id="@+id/name"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_alignParentLeft="true"
    android:layout_toLeftOf="@+id/date"
    android:textSize="22dp" />

<TextView
    android:id="@+id/examiner"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_alignTop="@+id/location"
    android:layout_below="@id/name" />

<TextView
    android:id="@+id/date"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_alignParentRight="true"
    android:layout_alignTop="@id/name" />

<TextView
    android:id="@+id/time"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_alignParentRight="true"
    android:layout_below="@+id/date" />

<TextView
    android:id="@+id/location"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_alignParentRight="true"
    android:layout_below="@+id/time" />

<TextView
    android:id="@+id/note"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
```

```
        android:layout_below="@+id/location"
        android:layout_marginTop="5dp"
        android:background="@color/white"
        android:textColor="@android:color/black" />
</RelativeLayout>
```

Kompletní zdrojový kód vlastního adaptéru najdete v příloze A.. V definici třídy adaptéru, která je potomkem třídy *BaseAdapter*, je definováno spojení s vlastním zdrojem dat - ArrayList objektů *Exam*, který jsem pojmenoval *exam-ArrayList*. Následují metody potřebné pro správnou funkci adaptéru, pak metoda getView starající se o provázání datové položky s vizuálním prvkem v souboru exam_list_row.xml - toto má za úkol tento příkaz:

```
convertView = mInflater.inflate(R.layout.exam_list_row, null);
```

Nedílnou součástí adaptéru je statická třída *ViewHolder*, přes kterou k tomuto propojení dochází.

4.4. Další prvky použité v projektu

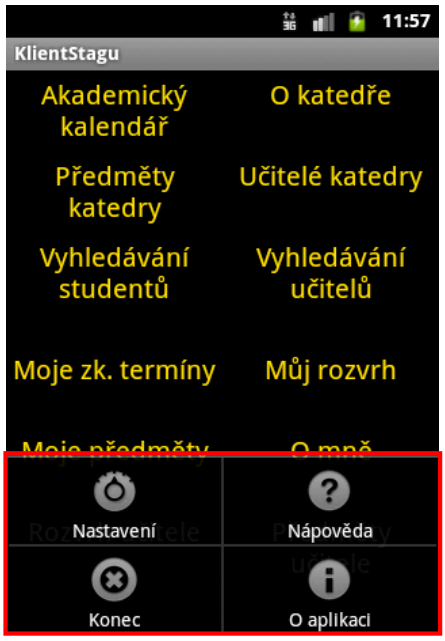
4.4.1. Menu

Menu se otvírá stiskem hardwarového tlačítka Menu na mobilním zařízení. V systému Android je nazývano jako *Menu voleb* nebo také *Option menu* a jedná se o nabídku vyjíždějící ze spodu obrazovky tak, jak je zobrazeno na obrázku 13.. Menu může mít 1 až 6 položek, která jsou přímo viditelná. V případě, že menu obsahuje více než 6 položek, pak se v menu objeví nová volba *Další*, která zpřístupní zbývající položky.

Nejjednodušší způsob jak vytvořit takovéto menu je pomocí XML layoutu, který se vytváří v adresáři */res/menu*, a příslušného kódu ve vaší Activity. Do své Activity vložíte novou metodu *onCreateOptionsMenu*, kde definujete název XML layoutu (main_menu), který chcete použít jako menu. Příslušný kód vypadá následovně:

```
@Override
public boolean onCreateOptionsMenu(Menu menu)
{
 MenuInflater inflater = getMenuInflater();
 inflater.inflate(R.menu.main_menu, menu);
 return true;
}
```

Pro úplnost ještě potřebný kód XML layoutu v souboru main_menu.xml:



Obrázek 13. Menu voleb (červeně vyznačeno)

```
<?xml version="1.0" encoding="utf-8"?>
<menu xmlns:android="http://schemas.android.com/apk/res/android" >
   <item android:id="@+id/options"
      android:icon="@android:drawable/ic_menu_preferences"
          android:title="Nastavení" />
    <item android:id="@+id/help"
       android:icon="@android:drawable/ic_menu_help"
          android:title="Nápověda" />
    <item android:id="@+id/exit"
  android:icon="@android:drawable/ic_menu_close_clear_cancel"
          android:title="Konec" />
    <item android:id="@+id/info"
       android:icon="@android:drawable/ic_menu_info_details"
          android:title="O aplikaci" />
</menu>
```

K výše uvedenému XML layoutu menu bych ještě doplnil, že parametrem *android:icon* definujeme ikonu, která je v tomto případě systémová, ale můžeme použít i svou vlastní nebo menu udělat pouze textové vynecháním parametru. Menu můžeme udělat i jen čistě obrázkové, vynecháním parametru *android:title*

Aby bylo menu funkční (”klikatelné”) je nutné ve vaší Activity vytvořit obslužnou metodu *onOptionsItemSelected*. Například takto:

```
@Override
```

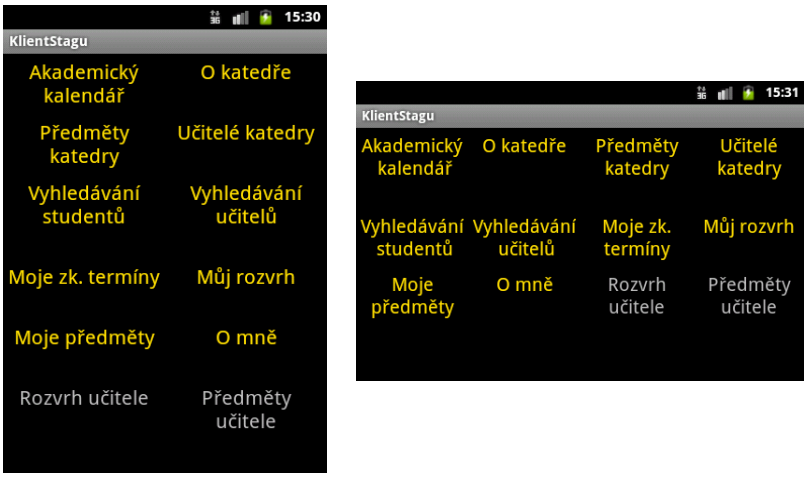
```
public boolean onOptionsItemSelected(MenuItem item) {
 switch (item.getItemId()) {
  case R.id.options:
   SharedPreferences prefs =
   PreferenceManager.getDefaultSharedPreferences(this);
   global.stagservices = prefs.getString(KEY_UNIVERSITY, "");
   global.faculty = prefs.getString(KEY_FACULTY, "");
   global.department = prefs.getString(KEY_DEPARTMENT, "");
   startActivity(new Intent(this, EditPreferences.class));
   return true;
  case R.id.exit:
   finish();
   return true;
  case R.id.help:
   startActivity(new Intent(this, Help.class));
   return true;
  case R.id.info:
   ...
   return true;
  default:
   return super.onOptionsItemSelected(item);
 }
}
```

Toto samozřejmě není jediný možný způsob jak vytvořit menu, ale je podle mého názoru nejpřehlednější. Podrobněji se porovnání dvou různých přístupů k programovaní GUI věnuji v kapitole 4.2.

4.4.2. GridView

GridView umožňuje dvou-dimenzionální zobrazení položek v rolovatelné mřížce. Položky mřížky jsou načítány opět prostřednictvím adaptéru, který je ale mnohem jednodušší, než tomu bylo u ListView 4.3.4. GridView je navíc tak flexibilní, že umožňuje i automatické přepočítání sloupců a řádků tak, aby při překlopení mobilního zařízení na šířku nezobrazoval pořád stejnou mřižku, ale efektivněji využil celou plochu obrazovky. Toto chování se definuje v XML layoutu a jedná se o parametr `android:numColumns="auto_fit"`. Vaše Activity by mohla vypadat například takto:

```
public class HlavniMenuActivity extends Activity
 implements AdapterView.OnItemClickListener {
 String[] items={"Akademický kalendář", "O katedře", "Předměty katedry",
   "Učitelé katedry", "Vyhledávání studentů", "Vyhledávání učitelů",
   "Moje zk. termíny", "Můj rozvrh", "Moje předměty", "O mně",
```



Obrázek 14. Ukázka dvou-dimezionálního zobrazení mřížkou - GridView

```
   "Rozvrh učitele", "Předměty učitele"};
 @Override
 public void onCreate(Bundle savedInstanceState) {
  super.onCreate(savedInstanceState);
  setContentView(R.layout.main_grid);
  GridView g=(GridView) findViewById(R.id.grid);
  g.setAdapter(new GridViewAdapter(this,
    android.R.layout.simple_list_item_1, items));
  g.setOnItemClickListener(this);
 }
}
```

Všimněte si v definici třídy vaší Activity je *implements AdapterView.OnItemClickListener*, které u jiných Activity nebývá. Toto nám umožňuje zapracovat obsluhu události *onItemClick* a zmiňovaného adaptéru přímo do těla Activity. V těle metody *onCreate* nastavíme metodou *setContentView* název našeho layoutu main_grid.xml (umístěného v adresáři *res/layout*), jehož jednoduchý kód je následující:

```
<?xml version="1.0" encoding="utf-8"?>
<GridView xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/grid"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:columnWidth="105dp"
    android:numColumns="auto_fit"
    android:stretchMode="columnWidth"
    android:verticalSpacing="35dp"
```

```
    android:horizontalSpacing="10dp">
</GridView>
```

Dále v těle Activity definujeme GridView a propojíme s názvem *grid*, abysme mohli nastavit adaptér. Nastavení adaptéru (*GridViewAdapter*) s třemi argumenty, kde definujeme třetí - items - pole s řetězci, které chceme pomocí GridView zobrazit. Tento adaptér musíme doprogramovat a vložit ho do těla naší Activity. Adaptér je, ve srovnání s vlastními adaptéry pro ListView, jednodušší a může vypadat takto:

```
private class GridViewAdapter extends ArrayAdapter<CharSequence> {
 Context ctxt;
 GridViewAdapter(Context ctxt, int resource,String[] items) {
  super(ctxt, resource, items);
  this.ctxt=ctxt;
 }
 public View getView(int position, View convertView,ViewGroup parent) {
  TextView label=(TextView)convertView;
   if (convertView==null) {
    convertView=new TextView(ctxt);
    label=(TextView)convertView;
   }
   label.setText(items[position]);
   label.setTextSize(20);
   label.setGravity(Gravity.CENTER_HORIZONTAL);
   //set color for students
   if (position < 10)
    label.setTextColor(getResources().getColor(R.color.upolYellow));
   return(convertView);
 }
}
```

Nakonec ještě musíme obsloužit událost *onItemClick*, aby byly položky GridView aktivní a mohly například spouštět další Activity. Tato obsluha není složitá a vypadá takto:

```
public void onItemClick(AdapterView<?> parent, View v,int position, long id) {
 switch (position) {
 case 0:
  //akademicky kalendar
  i = new Intent(HlavniMenuActivity.this, AcademicYear.class);
  i.putExtra("requestCode", 0);
  startActivityForResult(i,0);
  break;
```

```
 case 1:
  ...
}
```

V tomto případě se po stisku první položky zobrazení GridView (*Akademický kalendář*) spustí další Activity s názvem *AcademicYear*. Indexy jsou počítány z levého horního rohu mřížky do prava po řádcích (plynou jako běžný text).

4.4.3. WebView

Obrázek 15. Ukázka prvku WebView

WebView je zobrazení, které využívá interní webový prohlížeč - renderovací engine WebKit. Engine obsahuje metody pro navigaci vpřed a vzad v historii, zvětšování nebo zmenšování a další. WebKit vyžaduje přidání oprávnění *INTERNET* do souboru AndroidManifest.xml, čehož dosáhneme vložením tohoto elementu:

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

Ve své aplikaci WebView využívám k zobrazení uživatelské příručky - nápovědy (viz obrázek 15.), která je celá vytvořená v HTML a uložená v adresáři */res/raw*. Pro správnou funkci WebView je opět nutný XML layout a programový kód ve vaší Activity.
Kód Activity Help (obsahuje i metodu readTextFromResource, která zpracovává HTML kód):

```
@Override
```

```
public void onCreate(Bundle savedInstanceState) {
 super.onCreate(savedInstanceState);
 setContentView(R.layout.help);
 WebView webview = (WebView) findViewById(R.id.webview_help);
 webview.loadDataWithBaseURL("file:///android_asset/",
   readTextFromResource(R.raw.help), "text/html", "utf-8", "");
}
private String readTextFromResource(int resourceID) {
  InputStream raw = getResources().openRawResource(resourceID);
  ByteArrayOutputStream stream = new ByteArrayOutputStream();
  int i;
  try {
   i = raw.read();
   while (i != -1) {
    stream.write(i);
    i = raw.read();
   }
  raw.close();
  }
  catch (IOException e) {
   ...
  }
  return stream.toString();
}
```

Jak vidíme v definici proměnné *webview*, máme zde definováno propojení s elementem Webview jmenujícího se *webview_help*, který je umístěn v XML layoutu s názvem *help* (jak vidíme v definici metody setContentView).
XML layout souboru help.xml je tento:

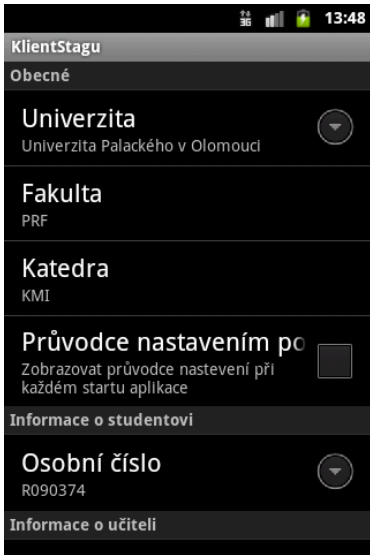
```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:orientation="vertical">

    <WebView android:id="@+id/webview_help"
        android:layout_width="fill_parent"
        android:layout_height="fill_parent">
</WebView>
</LinearLayout>
```

Tyto definice stačí na to, aby bylo zobrazení WebView funkční - načítalo soubor help.html z adresáře */res/raw* a zobrazilo jej.

4.4.4. Preferences

Preferences jsou ve spojení s Shared Preferences jednou z možností jak ukládat data v Androidu. Preferences jsou pro každou Activity zvlášť, zatímco Shared Preferences jsou dostupné pro celou aplikaci. Nejvíce se používají jako prostředek k nastavení chování programované aplikace a ukládání uživatelských nastavení. Pokud bysme si chtěli uživatelskou obrazovku s nastaveními (obrázek 16.) vytvořit, musíme si pro to založit novou Activity. Tato vaše nová Activity ovšem nebude potomkem Activity, ale PreferenceActivity. Navíc můžeme ještě přidat *implements OnSharedPreferenceChangeListener*, pokud chceme, aby se zadané hodnoty nejenom zapsaly do Shared Preferences, ale i aktualizovalo zobrazení na obrazovce (pokud například máme hodnoty uvedené v druhém řádku konkrétního nastavení) - k tomu je ale potřebné si doprogramovat obsluhu. V tomto přikladu se, v rámci zjednodušení, bez této vymoženosti obejdeme. Kód



Obrázek 16. Ukázka Preferences

Activity EdiPreferences:

```
import android.os.Bundle;
import android.preference.Preference;
import android.preference.PreferenceActivity

public class EditPreferences extends PreferenceActivity {
 @Override
 public void onCreate(Bundle savedInstanceState) {
  super.onCreate(savedInstanceState);
  addPreferencesFromResource(R.xml.preferences);
  ...
```

```
}
```

Tento kód ve spojení s XML layoutem v příloze B. vygeneruje uživatelská nastavení podle XML souboru, a výsledkem je obrazovka na obrázku 16. Výhodou Shared Preferences je jednoduchá implementace a dostupnost všech uložených prostředků napříč celou aplikací a to i po vypnutí a opětovném spuštění aplikace. Je to dáno tím, že se jednotlivá nastavení ukládají do interního úložiště v podobě XML souboru obsahujícího všechna naše nastavení.

4.5. Zpracování dat z IS/STAG

Protože veškeré zprávy z IS/STAG jsou ve formátu XML, viz kód v kapitole 2.3., je nutné jej zpracovat a uložit data do takové formy s kterou můžeme v programovém kódu aplikace (Activity) pracovat. Extrakci dat z XML předvedu na jednoduchém XML s následujícími daty harmonogramu akademického roku:

```
<ns1:getHarmonogramRokuResponse>
 <harmonogramRoku>
  <harmonogramItem>
   <datumOd>10.9.2011</datumOd>
   <popis>Začátek ak. roku</popis>
  </harmonogramItem>
  <harmonogramItem>
   <datumOd>10.9.2011</datumOd>
   <popis>Zacíná: Letní zkouškové období</popis>
  </harmonogramItem>
 </harmonogramRoku>
</ns1:getHarmonogramRokuResponse>
```

Zpracovávání a stahování dat z IS/STAG řeším privátní metodou *DownloadXML*. Se zpracováváním XML mě pomohl článek *Connecting to the Web: I/O Programming in Android* [7], řešící zpracovávání RSS kódů. V následujícím kódu je stahování XML zajištěno metodou OpenHttpConnection, která vrací InputStream. Tento je následně zpracováván a jsou k tomu využívány podtřídy *javax.xml.parsers* - konkrétně DocumentBuilder, DocumentBuilderFactory a ParserConfigurationException.

Definujeme si základní element (harmonogramItem) a pak procházíme XML a hledáme vždy další element harmonogramItem a v cyklu for načítáme obě datové položky a ukládáme si je pomocí objektu sr. Výpis metody *DownloadXML* s komentářem v těle:

```
private ArrayList<TwoRowListItem> DownloadXML(String URL) {
 InputStream in = null;
 ArrayList<TwoRowListItem> items = new ArrayList<TwoRowListItem>();
```

```
 try {
  in = global.OpenHttpConnection(URL,this);
  Document doc = null;
  DocumentBuilderFactory dbf = DocumentBuilderFactory.newInstance();
  DocumentBuilder db;
   try {
    db = dbf.newDocumentBuilder();
    doc = db.parse(in);
   } catch (Exception e) {
     ...
   }
   doc.getDocumentElement().normalize();
   //tady se definuje nadřazený element
   NodeList itemNodes = doc.getElementsByTagName("harmonogramItem");
   TwoRowListItem sr;
   for (int i = 0; i < itemNodes.getLength(); i++) {
    Node itemNode = itemNodes.item(i);
    if (itemNode.getNodeType() == Node.ELEMENT_NODE) {
     sr = new TwoRowListItem();
     Element itemElement = (Element) itemNode;

     //extrakce popisu z elementu harmonogramItem a uložení do objektu sr
     NodeList nameNodes = (itemElement).getElementsByTagName("popis");
     Element nameElement = (Element) nameNodes.item(0);
     NodeList txtNodes = ((Node) nameElement).getChildNodes();
     sr.setTxtBig(((Node) txtNodes.item(0)).getNodeValue());

     //extrakce datumu z elementu harmonogramItem a uložení do objektu sr
     NodeList nameAbrNodes = (itemElement).getElementsByTagName("datumOd");
     Element nameAbrElement = (Element) nameAbrNodes.item(0);
     NodeList txtNameAbrNodes = ((Node) nameAbrElement).getChildNodes();
     sr.setTxtSmall(((Node) txtNameAbrNodes.item(0)).getNodeValue());
     items.add(sr);
    }
   }
 } catch (IOException e1) {
     ...
 }
 return items;
}
```

Metoda *DownloadXML* vrací ArrayList, který definuji jako argument svému adaptéru, jenž ho pak patřičně zobrazí.

Závěr

Tvorba aplikací pro operační systém Android není na první pohled jednoduchá, protože vývojář musí mít hned na začátku poměrně dost znalostí - navíc různého druhu. Nicméně pokud překonáte tyto počáteční nároky, otevře se vám najednou netušené spektrum možností k využití nabitých znalostí. Co se týká aplikace Klient Stagu pro Android, kterou jsem předváděl svým spolužákům a z jejich reakcí mohu říct, že si uživatele jistě najde. Uživatelům, stejně jako mě, přišly nejzajímavější funkce Moje zkouškové termíny, Můj rozvrh a možnost napsat e-mail přimo z mobilního zařízení konkrétnímu učiteli bez znalosti jeho emailové adresy.

Conclusions

Creating applications for operating system Android is not that easy as seems to be, because developer has to have wide range of knowledge in different kinds. However if you pass initial demands, it would open to you unexpected posibilities to use knowledge in Android development. Application Stag client for Android I showed my schoolmates and I can say that it would definitely have its users. Users like the most My exam terms, My schedule and ability to send an e-mail to a teacher directly from mobile device, without exact e-mail address knowledge.

Reference

[1] Mark L. Murphy. *Android 2 – Průvodce programováním mobilních aplikací.* Computer Press, Brno, 2011.

[2] Sharon Zakhour. *Java - Výukový kurz.* Computer Press, Brno, 2007.

[3] Tomáš Kypta. *Vyvíjíme pro Android*, *Seriál Vyvíjíme pro Android na serveru Svět Androida*

[4] Google. *Android Developers*

[5] Dokumentace webových služeb Univerzity Palackého v Olomouci. *Webové služby nad IS/STAG Univerzity Palackého v Olomouci*

[6] Dokumentace webových služeb Západočeské univerzity v Plzni. *Webové služby nad IS/STAG Západočeské univerzity v Plzni*

[7] Wei-Meng Lee. *Connecting to the Web: I/O Programming in Android*. Článek na Internetu, 2010.

[8] Wikipedia. *The Free Encyklopedia*

A. Zdrojový kód ExamListBaseAdapter.java

```
package biz.pesek.klientstagu;
import java.util.ArrayList;

public class ExamListBaseAdapter extends BaseAdapter {
  private static ArrayList<Exam> examArrayList;
  private LayoutInflater mInflater;
  public ExamListBaseAdapter(Context context, ArrayList<Exam> results) {
    examArrayList = results;
    mInflater = LayoutInflater.from(context);
  }

  public int getCount() {
    return examArrayList.size();
  }

  public Object getItem(int position) {
    return examArrayList.get(position);
  }

  public long getItemId(int position) {
    return position;
  }

  public View getView(int position, View convertView, ViewGroup parent) {
    ViewHolder holder;
    if (convertView == null) {
        convertView = mInflater.inflate(R.layout.exam_list_row, null);
        holder = new ViewHolder();
        holder.tvPredmet = (TextView) convertView.findViewById(R.id.name);
        holder.tvJmeno = (TextView) convertView.findViewById(R.id.examiner);
        holder.tvDatum = (TextView) convertView.findViewById(R.id.date);
        holder.tvCasOd = (TextView) convertView.findViewById(R.id.time);
        holder.tvBudova = (TextView) convertView.findViewById(R.id.location);
        holder.tvPoznamka = (TextView) convertView.findViewById(R.id.note);
        convertView.setTag(holder);
    } else {
        holder = (ViewHolder) convertView.getTag();
    }
    holder.tvPredmet.setText(examArrayList.get(position).getPredmet());
    holder.tvJmeno.setText(examArrayList.get(position).getJmeno()
```

```
        +" "+examArrayList.get(position).getPrijmeni());
    holder.tvDatum.setText(examArrayList.get(position).getDatum());
    holder.tvCasOd.setText(examArrayList.get(position).getCasOd());
    holder.tvBudova.setText(examArrayList.get(position).getBudova()
        +"-"+examArrayList.get(position).getMistnost());
    if (examArrayList.get(position).getPoznamka().equals(""))
      holder.tvPoznamka.setVisibility(View.GONE);
    else {
      holder.tvPoznamka.setVisibility(View.VISIBLE);
      holder.tvPoznamka.setText(examArrayList.get(position).getPoznamka());
     }
     return convertView;
  }

  static class ViewHolder {
    TextView tvPredmet;
    TextView tvJmeno;
    TextView tvPrijmeni;
    TextView tvTitulPred;
    TextView tvTitulZa;
    TextView tvDatum;
    TextView tvBudova;
    TextView tvMistnost;
    TextView tvCasOd;
    TextView tvPoznamka;
  }
}
```

B. Zdrojový XML kód preferences.xml

```
<PreferenceScreen xmlns:android="http://schemas.android.com/apk/res/android">
    <PreferenceCategory android:title="Obecné" >
        <ListPreference
            android:dialogTitle="@string/choose_university"
            android:entries="@array/universities"
            android:entryValues="@array/stagservices"
            android:key="university"
            android:summary="@string/upol"
            android:title="@string/university" />
        <Preference
            android:key="faculty"
            android:title="@string/faculty" >
        </Preference>
        <Preference
            android:key="department"
            android:title="@string/department" >
        </Preference>
        <CheckBoxPreference
            android:key="startup_wizard"
            android:summary="@string/startup_wizard_description"
            android:title="@string/startup_wizard" />
    </PreferenceCategory>
    <PreferenceCategory android:title="@string/student_info" >
        <EditTextPreference
            android:dialogTitle="@string/enter_personal_id"
            android:key="personal_id"
            android:summary="@string/enter_personal_id"
            android:title="@string/personal_id" />
    </PreferenceCategory>
    <PreferenceCategory android:title="@string/teacher_info" >
        <EditTextPreference
            android:dialogTitle="@string/enter_teacher_id"
            android:key="ucitIdno"
            android:summary="@string/enter_teacher_id"
            android:title="@string/teachers_id" />
    </PreferenceCategory>
</PreferenceScreen>
```

C. Obsah přiloženého CD

V samotném závěru práce je uveden stručný popis obsahu přiloženého CD/DVD, tj. závazné adresářové struktury, důležitých souborů apod.

`bin/`
: Instalační balíček KlientStagu.apk vytvořené aplikace pro operační systém Android.

`doc/`
: Dokumentace práce ve formátu PDF, vytvořená dle závazného stylu KI PřF pro diplomové práce, včetně všech příloh, a všechny soubory nutné pro bezproblémové vygenerování PDF souboru dokumentace (v ZIP archivu), tj. zdrojový text dokumentace, vložené obrázky, apod.

`src/`
: Kompletní zdrojové texty programu KlientStagu pro Android se všemi potřebnými (převzatými) zdrojovými texty, knihovnami a dalšími soubory pro bezproblémové vytvoření spustitelných verzí programu.

`readme.txt`
: Instrukce pro instalaci a spuštění programu KlientStagu pro Android, včetně požadavků pro jeho provoz.

U veškerých odjinud převzatých materiálů obsažených na CD/DVD jejich zahrnutí dovolují podmínky pro jejich šíření nebo přiložený souhlas držitele copyrightu. Pro materiály, u kterých toto není splněno, je uveden jejich zdroj (webová adresa) v textu dokumentace práce nebo v souboru `readme.txt`.