

Západočeská univerzita v Plzni
Fakulta aplikovaných věd
Katedra informatiky a výpočetní techniky

Diplomová práce

Rozvrh hodin pro mobilní zařízení

Plzeň, 2012

Veronika Dudová

Prohlášení

Prohlašuji, že jsem diplomovou práci vypracovala samostatně a výhradně s použitím citovaných pramenů.

V Plzni dne 10. května 2012

Veronika Dudová

Abstrakt

Tato diplomová práce popisuje vytvoření mobilní aplikace zobrazující rozvrh hodin pro studenty ZČU. První část dokumentu se zabývá popisem nejrozšířenějších platforem pro mobilní zařízení a výběrem vhodného systému pro vytvoření aplikace. V druhé části je popsán způsob vývoje aplikací pro vybraný systém (Android). Další část popisuje vytvoření samotné aplikace, její použití a funkce. V poslední části je popsáno publikování aplikace na Google Play, včetně hodnocení a statistik, dále je uveden přehled verzí a možnosti dalších rozšíření.

Abstract

This master thesis describes the creation of mobile application showing the timetable for students of UWB. The first part of the document deals with the description of the most widely used platform for mobile devices and selecting the appropriate system for creating application. The second section describes how to develop applications for the selected system (Android). The next section describes the creation of the application itself, its use and functions. The last section describes publishing the application on Google Play, including user ratings and statistics, as well as an overview of versions and the possibility of further extensions.

Poděkování

Chtěla bych poděkovat vedoucímu práce doc. Ing. Pavlu Heroutovi, Ph.D. za cenné rady a připomínky.

Můj dík patří i rodině a přítelovi za podporu při studiu a poskytnutí potřebného zázemí.

Obsah

1	Úvod	1
2	Platformy mobilních zařízení	2
2.1	Srovnání OS podle rozšířenosti	2
2.2	iOS – Apple	4
2.3	BlackBerry – RIM	5
2.4	Windows Phone – Microsoft	6
2.5	Symbian OS – Symbian Foundation	7
2.6	Android – Google	8
2.7	Výběr platformy pro diplomovou práci	9
3	Vytváření aplikací pro Android	10
3.1	Nastavení	12
3.1.1	Systémové nároky	12
3.1.2	Instalace a nastavení SDK	12
3.1.3	Instalace a nastavení ADT pluginu	14
3.1.4	Vytvoření virtuálních zařízení	15
3.2	Vývoj aplikací	16
3.2.1	Android projekt	16
3.2.2	Sestavení a spuštění aplikace	20
3.2.3	Aktivity	22
3.2.4	Služby	26
3.2.5	Uživatelské rozhraní	27
3.2.6	Uživatelské rozhraní – komponenty	30
3.3	Ladění a testování	37
3.3.1	Android Debug Bridge	37
3.3.2	Dalvik Debug Monitor Server	38
3.3.3	Logování	41
3.3.4	Android Lint	42
3.3.5	Testování	42
3.4	Publikování	43
4	Aplikace Rozvrh	44
4.1	Struktura aplikace	44
4.1.1	AndroidManifest.xml	45
4.2	Data aplikace	46
4.2.1	Získání dat	46
4.2.2	Zpracování dat	48
4.2.3	Ukládání a přístup k datům	49
4.2.4	Servisní vrstva	53
4.3	Vzhled aplikace	53
4.3.1	Rozvrh	54
4.3.2	Předmět, učitel a student	55
4.3.3	Menu	57
4.3.4	Vzhled podle verze	58
4.4	Funkce aplikace	59

4.4.1	Přidání rozvrhu	59
4.4.2	Vyhledání rozvrhu podle jména studenta	61
4.4.3	Zobrazení rozvrhu	62
4.4.4	Zobrazení rozvrhu – kombinované studium	63
4.4.5	Detail předmětů, učitelů a studentů	63
4.4.6	Nastavení aplikace	65
4.4.7	Přehled změn	69
4.4.8	Gesta	70
4.4.9	Widget	73
4.5	Testování aplikace	77
4.5.1	Android JUnit	77
4.5.2	Robotium	78
4.5.3	Testovací projekt	79
4.5.4	Spuštění testů	80
4.5.5	Databáze pro testování	80
4.5.6	Testování servisních tříd	82
4.5.7	Testování uživatelského rozhraní	82
4.5.8	Testovací zařízení	84
5	Nasazení aplikace – publikování na Google Play	86
5.1	Přípravení aplikace	86
5.2	Vývojářská konzole	86
5.2.1	Hodnocení a komentáře	88
5.2.2	Statistiky	88
5.3	Verze aplikace	90
5.4	Možnosti rozšíření	92
6	Závěr	93
	Seznam zkratk	94
	Reference	95
A	AndroidManifest.xml	97
B	Uživatelský manuál	99

1 Úvod

V současné době stále více studentů používá „chytrý“ telefon s přístupem na internet k synchronizaci a správě kontaktů, dokumentů, kalendáře apod. Kromě zmíněných aktivit běžný student každodenně využívá rozvrh hodin. Bohužel na malých mobilních zařízeních (mobilní telefon, PDA) chybí aplikace, která by on-line a volitelně off-line zobrazovala rozvrh hodin, detaily jednotlivých rozvrhových akcí apod. V současné době je v zásadě možné rozvrh hodin zobrazit on-line pomocí webového prohlížeče a off-line pomocí PDF dokumentu. Vedle těchto hlavních možností již existují ještě další aplikace umožňující zobrazit rozvrh. Všechny uvedené způsoby mají své nevýhody z pohledu uživatele malých mobilních zařízení.

Nevýhodami webového prohlížeče je nutnost být připojen k síti (on-line), dále ne zcela optimální vzhled portálových www stránek v mobilním prohlížeči, nutnost postupného proklikávání několika stránek a v neposlední řadě samotná rychlost načítání. Použití PDF dokumentu také není příliš vhodné, neboť je potřeba mít v mobilním zařízení PDF čtečku, přičemž rychlost jejího spouštění není příliš velká. Navíc je PDF dokument prohlížen off-line, což znamená, že při změně rozvrhové akce je nutno stáhnout aktuální verzi PDF dokumentu. To představuje opět nutnost několika postupných kroků, stejně jako v případě použití webového prohlížeče. V současné době existuje několik málo poměrně neznámých aplikací použitelných pro zobrazení informace o rozvrhu. Jejich společná zásadní nevýhoda je, že dokáží zobrazit informace o jedné rozvrhové akci. To znamená, že uživatel musí postupně (jednorázově) zadat údaje o všech rozvrhových akcích, což není příliš uživatelsky přívětivé chování.

Vzniká tak požadavek na vytvoření aplikace pro mobilní telefon, která by po zadání pouze studentova osobního čísla získala on-line ze serveru portal.zcu.cz potřebná data celého rozvrhu a přehledně je zobrazila. Tato data je vhodné ukládat, aby pro pozdější opětovné zobrazení nebyl potřeba přístup k internetu, tj. bylo možné okamžité off-line zobrazení. Samozřejmě, že v případě změny jakékoliv rozvrhové akce je vždy možno znovu jednoduše načíst aktuální verzi rozvrhu.

2 Platformy mobilních zařízení

Platformy pro mobilní zařízení se stále rozvíjejí a v dnešní době existuje několik velmi rozšířených operačních systémů pro tzv. chytré telefony. Jedná se o telefony s funkcemi jako je gps navigace, přístup na internet (wifi, 3G) a s ním spjaté služby jako je např. správu emailů. Vzhledem ke snižující se ceně těchto telefonů se v poslední době velmi masově rozšířily, proto jsem se rozhodla vytvořit aplikaci pro některý operační systém využívaný na chytrých mobilních telefonech. Neuvažovala jsem tedy vůbec o vytváření aplikace pro dnes již zastaralou platformu Java ME, která má omezené možnosti a rozsah využití.

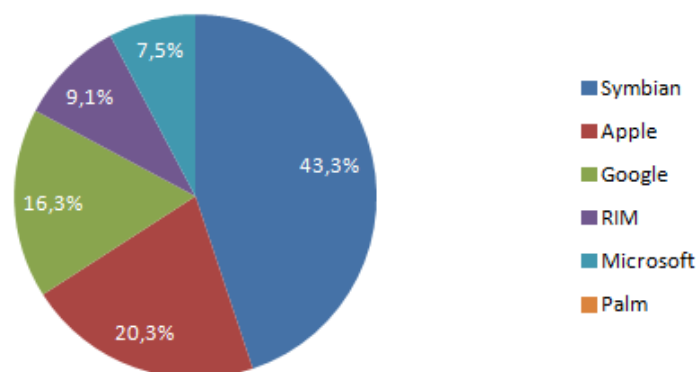
Dnes je na trhu pět nejrozšířenějších operačních systémů pro chytré mobilní telefony. Jedná se o iOS firmy Apple, BlackBerry OS firmy RIM, Windows Phone od firmy Microsoft, Symbian od Symbian Foundation a Android od firmy Google. V této kapitole srovnám systémy podle rozšířenosti, popíši každý ze systémů, jeho výhody a nevýhody z pohledu vybrání cílové budoucí platformy a na konci kapitoly uvedu jaký operační systém jsem vybrala a proč.

2.1 Srovnání OS podle rozšířenosti

Je velký rozdíl mezi Evropou a Spojenými Státy v tom, jaké platformy pro chytré telefony se nejvíce používají. V Evropě vévodí Symbian se skoro polovičním podílem na trhu, oproti tomu v US jsou nejrozšířenější platformy od Google, RIM a Apple. Na grafu 2.1 je vidět podíl platform pro evropský trh, na grafu 2.2 pro americký trh. Data jsou získána ze stránky [1], kde zdrojem dat je agentura comScore.

Platformy chytrých telefonů v Evropě

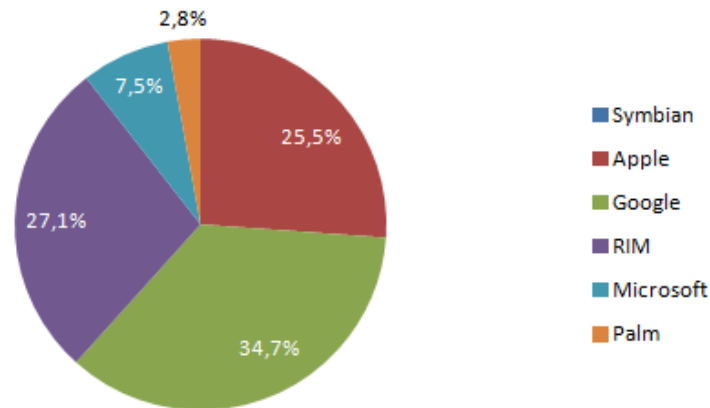
zdroj: comScore MobiLens, březen 2011



Obrázek 2.1: Srovnání podílu platform chytrých telefonů v Evropě

Platformy chytrých telefonů v U.S.

zdroj: comScore MobiLens, březen 2011



Obrázek 2.2: Srovnání podílu platforem chytrých telefonů v USA

Zajímavé je také provnat růst mobilních zařízení s platformou od Google od července 2010 do července 2011 [2] (tabulka 2.3). V této tabulce je vidět, že operační systém Android již předstihl iOS (Google vs. Apple). Dá se předpokládat, že růst, který zaznamenal Google, bude i nadále trvat na úkor ostatních platforem.

Top Smartphone Platforms in EU5 by Share of Smartphone Users* 3 Month Average Ending July 2011 vs. July 2010 Total EU5 (DE, FR, IT, ES and UK) Mobile Subscribers, Age 13+ Source: comScore MobiLens			
Smartphone Platform	Share (%) of EU5 Smartphone Users		
	Jul-10	Jul-11	Point Change
Total Smartphone Users	100.00%	100.0%	0.0
Symbian	53.9%	37.8%	-16.1
Google	6.0%	22.3%	16.2
Apple	19.0%	20.3%	1.2
RIM	8.0%	9.4%	1.5
Microsoft	11.5%	6.7%	-4.8

*MobiLens measures users above the age of 13 and reports on only primary handset usage.

Obrázek 2.3: Změny v čase na podílu platforem chytrých telefonů v Evropě

2.2 iOS – Apple



iOS je operační systém od firmy Apple, který se používá v telefonech iPhone, ale i v dalších mobilních zařízeních této firmy, jak jsou iPod Touch a iPad. První verze tohoto operačního systému byla představena 9. ledna 2009 a dnes je dodáván ve verzi 5.

V operačním systému iOS lze spouštět aplikace vytvořené v jazyku C či nadstavbě Objective-C. API pro tvorbu aplikací se nazývá Cocoa Touch, přičemž velmi dobře zpracovaný návod pro programování s tímto API viz [3]. Vyvíjet pro tuto platformu lze pouze v aplikaci XCode, což je vývojové prostředí od firmy Apple, které je dostupné pouze pro operační systém Mac OS X. To znemožňuje vývoj na Windows či Linux.

Vytvořenou aplikaci můžeme instalovat na telefon, jen pokud je podepsána certifikátem vydaným společností Apple. Certifikát lze získat tak, že si založíme konto u Apple na stránce developer.apple.com/programs/ios, počkáme na schválení a zaplatíme poplatek 99 dolarů. Tento poplatek je nutné platit každý rok, pokud chceme vydávat vlastní aplikace. Poté již můžeme naši aplikaci publikovat v AppStore, kde si ji mohou ostatní majitelé zařízení iPhone a iPad stáhnout a používat.

Výhody:

- více homogenní prostředí, není tolik verzí OS a existuje pouze velmi omezený počet různých zařízení, vždy od firmy Apple
- programovací jazyk C, snažší portace aplikací
- na iOS se prodává více aplikací než na ostatní platformy

Nevýhody:

- vývoj jen pod Mac OS, tzn. nutnost mít počítač od Apple
- cena ročního poplatku za možnost vydávat aplikace v AppStore
- vyšší cena iPhoneu → rozšířenost zařízení v ČR
- Apple nemusí aplikaci povolit, což může být velký potenciální problém

2.3 BlackBerry – RIM



BlackBerry OS je operační systém vyvinutý kanadskou firmou Research In Motion (RIM). Tento operační systém se používá v telefonech BlackBerry. Jeho hlavní výhodou je, že na rozdíl od telefonů s jinými operačními systémy má zabudovanou podporu firemních emailů. Díky tomu jsou tyto telefony hojně využívány jako firemní telefony, obzvláště jsou oblíbené ve Spojených státech.

Jádro operačního systému je založeno na Javě, a pro programování aplikací pro tento OS je k dispozici příslušné API. Vývoj tedy není omezen jen na určitý operační systém jako je tomu u Apple. Dokonce s novou verzí Widget SDK již vývojáři nejsou limitováni jen na Javu, protože tato verze podporuje i webové technologie HTML, CSS, JavaScript a nástroj do příkazové řádky, který umožňuje převést webové aplikace do Java aplikací.

Nově vznikl také operační systém BlackBerry PlayBook OS, který je využíván v tabletu PlayBook. Pro tento operační systém existují nástroje, které umožňují převést aplikace naprogramované pro Android na aplikace právě pro BlackBerry PlayBook OS. Je možné i online otestovat Android aplikaci, nako-lik je či není kompatibilní s BlackBerry systémem. Součástí pluginu Android Development Tools (ADT) je možnost konvertovat aplikaci přímo z vývoje-ového prostředí Eclipse. BlackBerry Playbook si poradí i s aplikacemi napsanými v C/C++ a má podporu pro Adobe AIR, což je běhové prostředí pro nasazení standalone aplikací [4].

Pro publikování vytvořené aplikace je nutné vytvořit účet na BlackBerry App World, asociovat jej s PayPal účtem a počkat na schválení aplikace od RIM. Nejsou však potřeba žádné registrační poplatky.

Výhody:

- možnost použít aplikace napsané v C/C++, HTML5, Java, Java Android, Adobe AIR
- díky podpoře více programovacích jazyků si programátor může vybrat pro něj ten nevhodnější
- publikování zdarma

Nevýhody:

- telefony značky BlackBerry jsou u nás málo rozšířené

2.4 Windows Phone – Microsoft



Windows Phone je mobilní operační systém vyvíjený firmou Microsoft. Nahradil svého předchůdce Windows Mobile v únoru roku 2010, kdy byl představen jako Windows Phone 7 a byla vydána poslední verze Windows Mobile 6.5.5. Tyto dva operační systémy patří sice do stejné rodiny operačních systémů Windows CE, ale nejsou kompatibilní. Modernější Windows Phone je využíván v různých telefonech od firem HTC, Dell, Samsung a LG. Na začátku roku 2011 bylo oznámeno, že systém Windows Phone bude využíván jako primární operační systém v telefonech Nokia.

Pro Windows Phone již nelze použít nativní vývoj v jazyce C++, jako tomu bylo u předchozí verze. Aplikace se vyvíjejí pomocí platformy Silverlight [5] a v XNA frameworku [6]. Obě tyto technologie jsou postavené nad .NET Frameworkem a programovat se dá pomocí jazyků C#, Visual Basic .NET a F#. Programovacím prostředím je Visual Studio, přičemž postačuje verze, která je zdarma. Potřebné nástroje jsou tedy zdarma, jedinou podmínkou je použití OS Windows Vista nebo Windows 7.

Místo, odkud lze stahovat a kupovat aplikace pro Windows Phone, se nazývá Marketplace. Abychom zde mohli publikovat vlastní aplikaci, je nutné se zaregistrovat jako vývojář na webu App Hub [7] a zaplatit poplatek 99 dolarů (ročně). Pro studenty je registrace zdarma v rámci programu DreamSpark. Hotovou a otestovanou aplikaci je dále potřeba odeslat do certifikačního procesu. Zde je aplikace ověřována ze strany firmy Microsoft a pokud splňuje podmínky pro publikování je certifikována, podepsána a může být zveřejněna na Marketplace. Požadavky na aplikaci jsou jak na obsah (používání vulgárních výrazů, ochranných známek apod.), tak na technické řešení (využití zdrojů, jazyková lokalizace apod.) [8]. Pro otestování, zda aplikace projde certifikací pro publikování na Marketplace, slouží nástroj Marketplace Test Kit [9].

Výhody:

- student může vytvářet a publikovat aplikace zdarma
- poměrně snadné vytváření výkonných 2D a 3D her pomocí XNA
- .NET technologie – možno aplikaci použít např. v konzoli Xbox 360

Nevýhody:

- poměrně malé zastoupení na evropském trhu
- není možný nativní vývoj v jazyce C++ (nelze spustit aplikace pro Windows Mobile)

2.5 Symbian OS – Symbian Foundation



Operační systém Symbian od Symbian Foundation (dříve Symbian Ltd.) je systém využívaný především v telefonech Nokia. Zatím poslední verzí je Symbian Belle používaný ve třech telefonech Nokia. Společnost Nokia ale 11. února 2011 oznámila přechod k platformě Windows Phone 7, vývoj Symbianu převezme poradenská firma Accenture.

Začátkem roku 2010 byl tento operační systém uvolněn pod Eclipse Public License a zdrojové kódy celého systému si tak může kdokoliv stáhnout a upravit.

Pro vývojáře nabízí Symbian vývoj nativních aplikací pro mobilní telefon v C/C++, kde se jako vývojové prostředí většinou používá Carbide.c++ [10]. V novějších verzích Symbianu je využívá Qt framework [11], který je jednodušší na práci. Jako vývojové prostředí je využíván Qt Creator. Pro převod aplikací napsaných pro iOS nebo Android je k dispozici Application Programming Interface (API) mapování, kde lze zjistit, jaké funkce jednoho systému odpovídají funkcím v Qt, což může portaci velmi usnadnit.

Pro publikování aplikace v Nokia Ovi Store je nutná registrace jako vydavatel, kde registrační poplatek činí 1 euro. Po nastavení názvu aplikace, ikony atd. je nutné poslat aplikaci k posouzení (Quality Assurance review), které trvá čtyři až šest dní. Po schválení je aplikace přístupná v Nokia Store.

Výhody:

- otevřený operační systém
- nástroje pro portaci do Qt
- nízký registrační poplatek pro publikování aplikací

Nevýhody:

- nejistá budoucnost – již nebude v telefonech Nokia
- málo zařízení s novější verzí systému

2.6 Android – Google



Android je operační systém založený na jádře Linuxu. Je vyvíjen Open Handset Alliance, která je vedena firmou Google. Zdrojové kódy Androidu jsou zveřejněny jako open-source pod Apache License. Ve třetí čtvrtině roku 2010 se Android stal nejlépe prodávající se platformou pro chytré telefony a v říjnu roku 2011 dosáhl hranice 700 tisíc aktivací telefonů Android každý den. Tento operační systém můžeme nalézt na telefonech HTC, Samsung, LG, Motorola, Sony Ericsson a dalších. Nejrozšířenější verzí je 2.3.x, nejnovější verzí je 4.0.x, která spojuje verze určené pro mobilní telefony a tablety.

Aplikace pro Android se programují v jazyce Java. K dispozici je Android SDK, které poskytuje nástroje a API potřebné k vyvíjení aplikací pro Android platformu. Hlavní vývojové prostředí je Eclipse, do kterého existuje ADT plugin, který rozšiřuje možnosti prostředí pro vytváření nových Android projektů, UI aplikace, podepisování aplikací atd. Veškeré nástroje jsou dostupné zdarma na [12], kde se nacházejí i různé návody, tutoriály, videa a dokumentace k API. Komunita vývojářů pro Android je velmi široká, což pomáhá při získávání nových znalostí a řešení nejrůznějších problémů.

Instalace aplikací není vázána jen na jeden online obchod jako je tomu u AppStore Applu, ale je možné instalovat aplikace z internetových stránek třetích stran, z oficiálního Android Marketu nebo z jiných online obchodů. Pro publikování vlastní aplikace na Android Market je nutné se zaregistrovat jako vývojář a zaplatit jednorázový poplatek 25 dolarů. Poté už můžeme nahrát a publikovat vlastní aplikaci, která se během několika minut objeví na Android Marketu dostupná ke stažení. Schvalovací proces jako u obchodů pro jiné platformy zde tedy není. Koncem roku 2011 zavedl Google bezpečnostní nástroj Bouncer, kterým jsou aplikace zpětně kontrolovány. Tento nástroj hledá stopy škodlivého softwaru či spywaru, srovnává i nové verze aplikací s dřívějšími, aby vyhledal malware, viry a spamy.

Výhody:

- velká komunita vývojářů
- publikování ihned, bez čekání na schválení
- jen jednorázový poplatek pro publikování aplikací
- mobilní telefony se systémem Android u nás velmi rozšířené

Nevýhody:

- nutnost brát v úvahu různé verze OS i hardware zařízení

2.7 Výběr platformy pro diplomovou práci

Po prozkoumání všech nejrozšířenějších platform pro chytré telefony bylo nutné se rozhodnout, kterou z nich vybrat jako cílovou platformu pro vytvoření aplikace zobrazující rozvrh hodin pro studenty ZČU. Určila jsem si tedy hlavní kritéria, která by měla platforma (operační systém) splňovat.

Kritéria pro výběr platformy:

1. Musí být na telefonech, které jsou u nás rozšířené a cenově dostupné studentům.
2. Platforma nesmí být zastaralá a její další vývoj a použití v budoucnu nesmí být nejisté.
3. Publikování hotové aplikace v oficiálních online obchodech nesmí být příliš drahé.
4. Možnost vydat aplikaci mimo oficiální online obchody výhodou.
5. Vývoj aplikací by neměl být omezen jen na jeden operační systém.
6. Dostatečně velká komunita vývojářů.
7. Možnost použít hotovou aplikaci (s určitými změnami) i na jiných platformách výhodou.

V následující tabulce (Tabulka 1) je vidět srovnání platform podle toho, jak splňují či nespĺňují daná kritéria. Každý sloupec odpovídá kritériu stejného čísla, zelená barva znamená jeho splnění, žlutá částečné splnění a červená nespĺnění.

Platforma	1	2	3	4	5	6	7
iOS	červená	zelená	červená	červená	červená	zelená	žlutá
BlackBerry	červená	červená	zelená	zelená	zelená	žlutá	zelená
Windows Phone	žlutá	zelená	červená	červená	červená	zelená	žlutá
Symbian	zelená	červená	zelená	zelená	žlutá	žlutá	žlutá
Android	zelená	zelená	zelená	zelená	zelená	zelená	žlutá

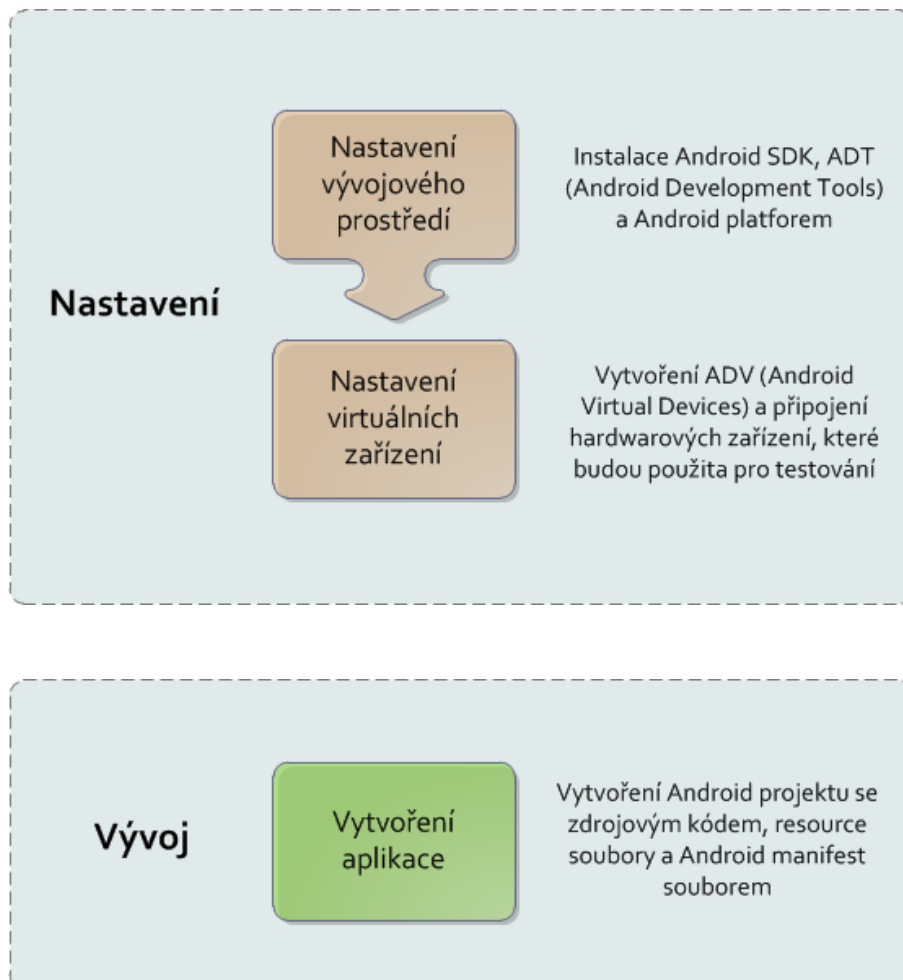
Tabulka 1: *Splnění kritérií platform pro použití v diplomové práci*

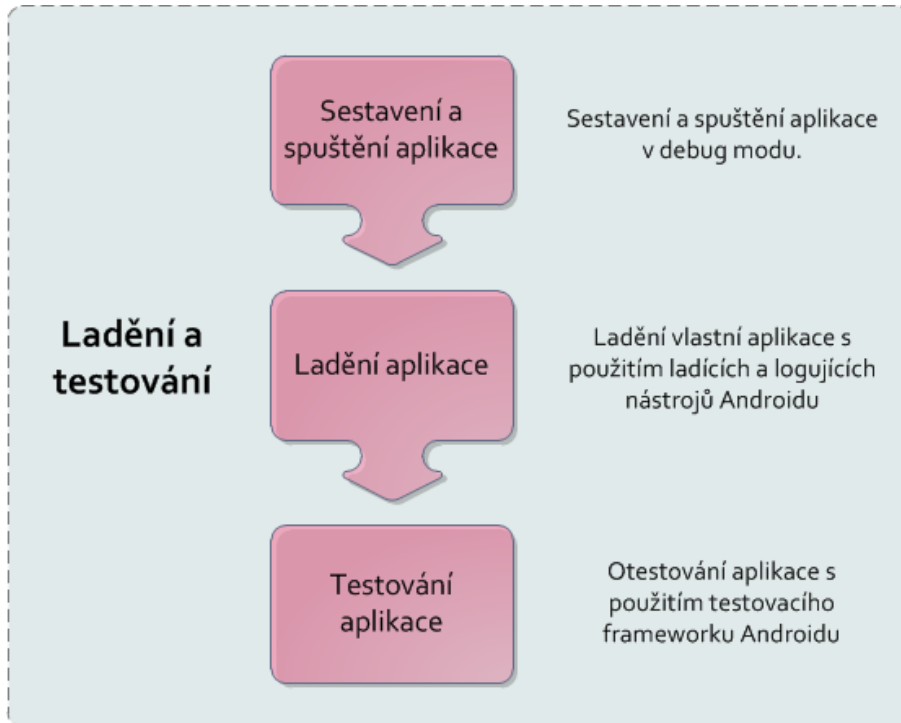
Z tabulky je vidět, že nejvíce stanovených kritérií splňuje platforma Android. Tato moderní platforma od Google v poslední době zaznamenala obrovský růst a ve Spojených státech v prodeji předstihla i Apple s iOS. Z těchto důvodů jsem jako cílovou platformu pro vytvoření aplikace zvolila Android.

3 Vytváření aplikací pro Android

Vývojový proces pro vytvoření Android aplikace je rozdělen na několik logických částí, které jsou schématicky znázorněny na následujících diagramech. Pro jejich vytvoření byly použity informace ze stránky [12] z části *Dev Guide - Introduction*.

Každá z těchto částí bude postupně podrobněji popsána a vysvětlena v samostatných kapitolách.





3.1 Nastavení

Vývoj aplikací pro Android usnadňuje skupina nástrojů poskytovaná s SDK. Tyto nástroje můžeme používat přímo z příkazové řádky nebo je využívat ve vývojovém prostředí Eclipse pomocí pluginu ADT (Android Development Tools). Vývoj v prostředí Eclipse je preferovaným způsobem, protože můžeme z prostředí rovnou využívat nástroje potřebné pro vývoj aplikace. Existuje také podpora pro prostředí IntelliJ IDEA. Já jsem ve své práci používala prostředí Eclipse, proto dále budu popisovat nastavení a práci v tomto prostředí.

Kompletní popis nastavení, odkud jsem čerpala informace, je v sekci SDK na stránce [12].

3.1.1 Systémové nároky

Pro vývoj aplikací pro Android s použitím Android SDK je nejprve nutné zkontrolovat, zda splňujeme systémové a softwarové nároky.

Podporované operační systémy:

- Windows XP (32-bit), Vista (32- nebo 64-bit), nebo Windows 7 (32- nebo 64-bit)
- Mac OS X 10.5.8 nebo novější (pouze x86)
- Linux (glibc 2.7 nebo novější)

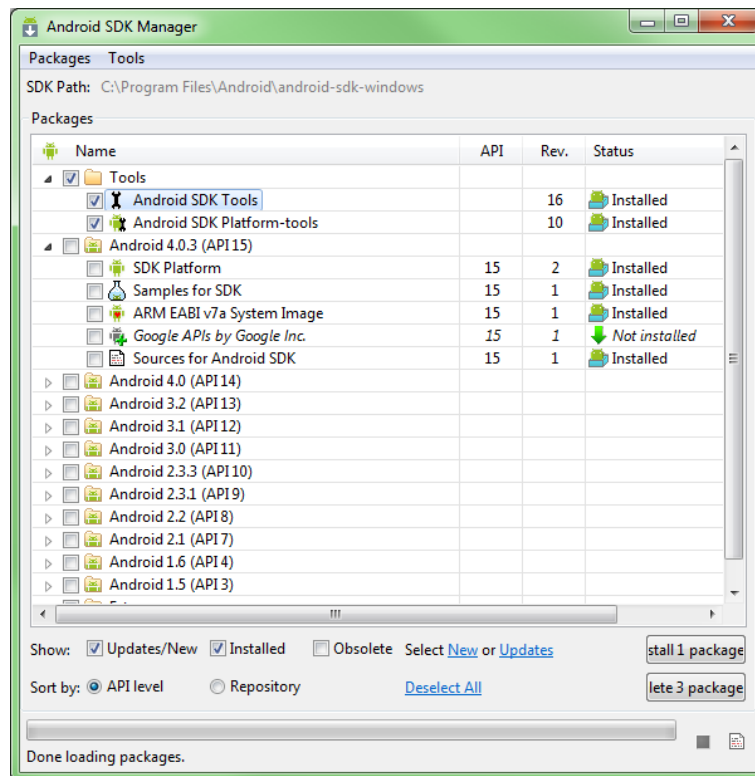
Podporované vývojové prostředí je Eclipse IDE od verze 3.6 (Helios). Doporučené balíčky jsou Eclipse IDE for Java Developers, Eclipse Classic nebo Eclipse IDE for Java EE Developers. Dále je nutné mít nainstalovanou Javu jako JDK 6 a doporučený je také plugin Android Development Tools (ADT) do prostředí Eclipse.

3.1.2 Instalace a nastavení SDK

Abychom mohli začít používat Android SDK, je nutné stáhnout a nainstalovat balíček SDK, který obsahuje základní SDK nástroje, pomocí kterých je možno stáhnout další SDK komponenty, jako např. Android platformy. Ze stránky [12] stáhneme SDK pro náš operační systém. U systému Windows si můžeme vybrat z archívu nebo instalačního souboru, což je doporučovaná možnost. Při instalaci je zkontrolováno, zda je v systému nainstalováno potřebné JDK.

Samotná instalace startovacího balíčku SDK pro programování Android aplikací nestačí. Je nutné nejprve stáhnout platformy, pod které chceme naši aplikaci vyvíjet. Spustíme tedy nástroj Android SDK Manager, který je součástí

startovacího balíčku. Program zobrazuje SDK komponenty, které jsou k dispozici ke stažení z repositářů (viz obr. 3.4). Přes toto grafické rozhraní tak můžeme instalovat i aktualizovat potřebné komponenty pro vývoj, které lze rozdělit do několika kategorií.



Obrázek 3.4: Obrazovka programu Android SDK Manager

SDK Tools

Nástroje pro ladění a testování aplikace a další pomocné nástroje. Tyto nástroje jsou nainstalovány společně se startovacím balíčkem a jsou zde pravidelně aktualizovány, proto je vhodné jednou za čas spustit tento Manager a aktualizace zkontrolovat. Nástroje se nachází v adresáři `<sdk>/tools/`.

SDK Platform-tools

Platformně závislé nástroje pro vývoj a ladění aplikace. Podporují nejnovější funkce pro Android platformu. Nachází se v adresáři `<sdk>/platform-tools/`.

Android platformy

SDK platforma je k dispozici pro každou vydanou Android platformu nasazenou na zařízení. Každá komponenta SDK obsahuje plně kompatibilní Android knihovnu, bitovou kopii systému, ukázkový kód a vzhledy pro emulátor.

USB ovladače

Obsahuje soubory ovladače, které můžeme nainstalovat na systém Windows. Aplikaci pak můžeme spustit a ladit na reálném zařízení.

Ukázky (Samples)

Obsahují ukázkové kódy a aplikace pro každou Android platformu.

Dokumentace

Obsahuje lokální kopii aktuální dokumentace pro API.

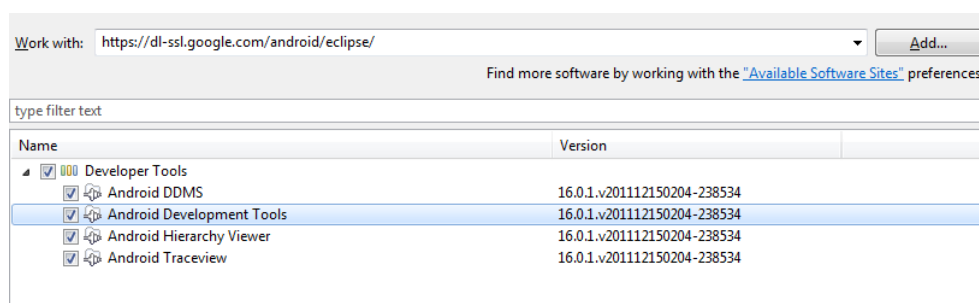
V okně vybereme, pro jaké verze operačního systému chceme stáhnout příslušné SDK a ukázky programů a nainstalujeme příslušné balíky. Při vývoji aplikace se tak rozhodneme, jakou nejnižší verzi systému budeme podporovat. Pokud např. budeme chtít využívat nových vlastností, které přinesl Android 2.2, a vybereme tuto platformu jako cílovou, bude naše aplikace spustitelná i na všech vyšších systémech, ale na 2.1 a nižších ne. Proto jsem se rozhodla vytvářet aplikaci pod verzí 1.6, což je verze, která byla v prvních Android telefonech prodávaných u nás. Díky tomu aplikaci neomezím jen na uživatele s novější verzí operačního systému ovšem na úkor využívání nových funkcí, které přišly s vyššími verzemi.

3.1.3 Instalace a nastavení ADT pluginu

Android Development Tools (ADT) je plugin pro Eclipse IDE, který je navržen tak, aby poskytl výkonné a integrované prostředí, ve kterém lze vytvářet Android aplikace.

ADT rozšiřuje prostředí Eclipse o možnost vytvářet nové Android projekty, uživatelské rozhraní aplikací, přidávat komponenty založené na Android Framework API, ladit aplikace pomocí Android SDK nástrojů a dokonce exportovat podepsané (nebo nepodepsané) .apk soubory k distribuci aplikace.

ADT plugin je kompatibilní s Eclipse od verze 3.6 Helios. Pro jeho stažení a instalaci využijeme nástroj Update Manager, který je součástí Eclipse. Spustíme jej přes Help -> Install New Software..., kde jako URL zdroje zadáme adresu <https://dl-ssl.google.com/android/eclipse/>. Zobrazí se dostupné nástroje, kde jedním z nich je i ADT (viz obr. 3.5). Doporučuji nainstalovat i ostatní nástroje, které se hodí pro ladění a testování aplikace.



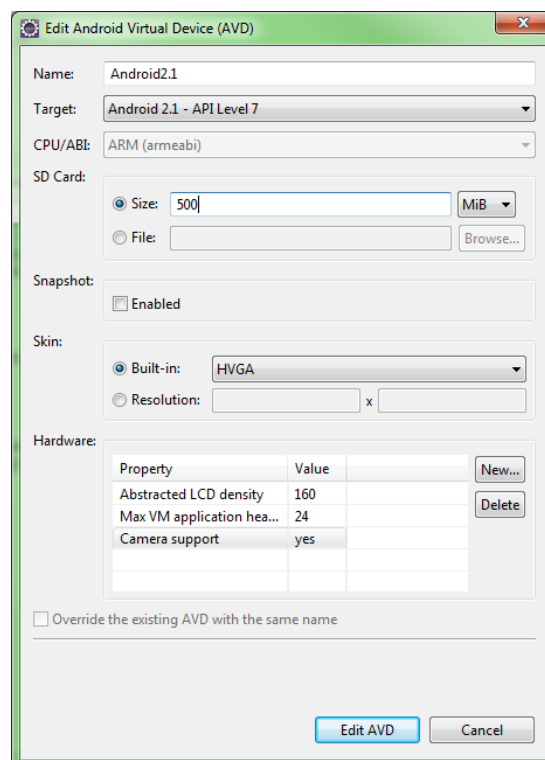
Obrázek 3.5: Instalace ADT pluginu ve vývojovém prostředí Eclipse

Posledním krokem je nastavení ADT pluginu (Windows -> Preferences..., záložka Android), aby ukazoval na adresář s nainstalovaným Android SDK.

3.1.4 Vytvoření virtuálních zařízení

Pro vývoj, ladění a testování aplikace je nutné mít nějaké zařízení, na kterém budeme aplikaci spouštět. Zařízením nemusí být přímo fyzické zařízení, ale můžeme využít i Android emulátor. Pro jeho nastavení se využívá tzv. Android Virtual Device (AVD). Ten umožňuje nastavit hardwarové i softwarové možnosti zařízení.

AVD se vytváří přes grafický AVD Manager, který můžeme spustit jako samostatnou aplikaci nebo i přímo z Eclipse. Zde můžeme vytvářet libovolné množství AVD s různými konfiguracemi. Kromě nastavení cílové platformy, která má být spuštěna při emulaci, lze nastavit i velikost SD karty, rozlišení obrazovky, velikost paměti atd. (obr. 3.6). Je proto výhodné vytvořit více AVD s nastaveními, jaké předpokládáme, že budou mít uživatelé používající naši aplikaci. Při spuštění aplikace v Eclipse se zobrazí seznam všech nastavených AVD,



Obrázek 3.6: Nastavení Android Virtual Device (AVD)

kde se určí, na kterém z nich se má aplikace spustit. Virtuální zařízení můžeme spustit i samostatně přímo přes AVD manager.

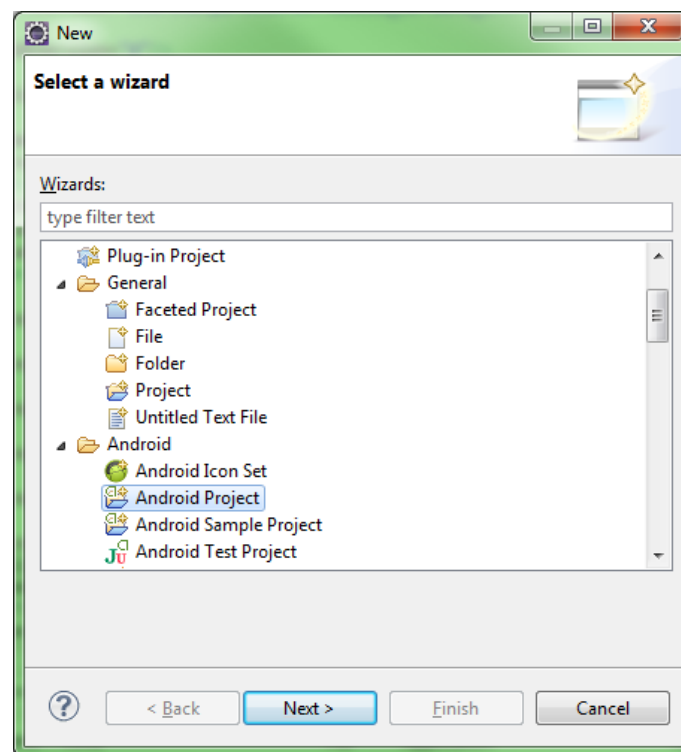
3.2 Vývoj aplikací

V této kapitole bude popsána struktura Android projektu a jednotlivé části, ze kterých je složena Android aplikace. Informace byly čerpány z [12] a publikací [13] a [14].

3.2.1 Android projekt

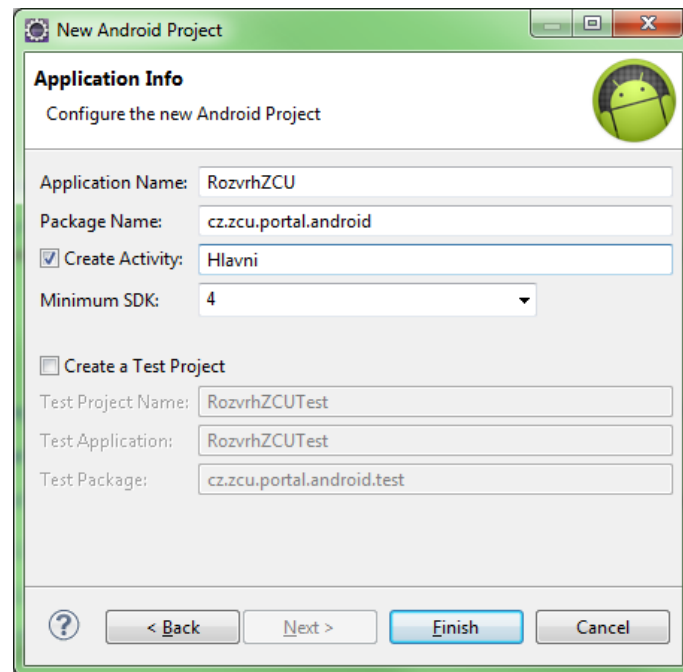
Po nainstalování Android SDK, začneme vytváření aplikace založením Android projektu. Tento projekt můžeme vytvořit buď pomocí příkazové řádky a nástroje *android* nebo přímo v prostředí Eclipse. Nadále budu popisovat jen práci v prostředí Eclipse, kde je vytváření projektů jednodušší a intuitivnější. Postupu pro vytvoření projektu přes příkazovou řádku můžete nalézt na <http://developer.android.com/guide/developing/projects/projects-commandline.html>.

V Eclipse vytvoříme projekt pomocí tzv. wizardu. Zvolíme **File > New > Project**, kde pod záložkou **Android** vybereme **Android Project** (viz obr. 3.7).



Obrázek 3.7: Spuštění wizardu pro vytvoření Android projektu

Následuje série nastavení, kde postupně zadáme název projektu, cílovou platformu, název aplikace, balíku a v neposlední řadě minimální verzi SDK, pro kterou bude naše aplikace dostupná (viz obr. 3.8). Je zde také možnost vytvořit současně i testovací projekt.



Obrázek 3.8: *Wizard pro vytvoření Android projektu*

Tímto způsobem je vytvořen nový Android projekt. Ten není prázdný, ale je zde připravena struktura pro zdroje a jsou zde již některé vygenerované součásti budoucí aplikace (obr. 3.9). Nyní popíšeme jednotlivé součásti projektu, ze kterých se Android aplikace skládá.

src/

Obsahuje aktivity, což jsou třídy popisující obrazovky aplikace (viz podrobně 3.2.3), a další zdrojové kódy aplikace. Cesta ke zdrojovým kódům je určena podle nastaveného balíku při vytváření projektu.

gen/

Obsahuje Java soubory, které jsou automaticky generovány pluginem ADT. Obsahuje vždy soubor `R.java`, kde jsou identifikátory všech prostředků jako jsou řetězce a layouty.

assets/

Tento adresář je ve výchozím stavu prázdný. Slouží jako úložiště statických souborů, které chceme přibalit k aplikaci. Například při vývoji her se zde ukládají textury a další herní data.

bin/

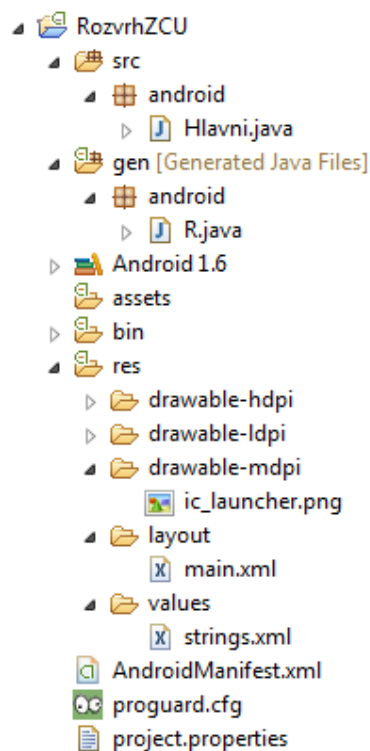
Po zkompileování aplikace je zde uložen výsledný `.apk` soubor.

res/

Umístění, kde se uchovávají prostředky (resources), jako jsou ikony, návrhy GUI,

popis menu, hodnoty řetězců a další. Prostředky by měly být vždy umístěny ve specifických podadresářích, např. XML soubory definující zobrazení patří do složky `layout`, bitmapové soubory do složky `drawable` apod.

Dále je možné vytvářet i alternativní prostředky přidáním kvalifikátorů. Například kvalifikátor `en` bude udávat, že dané prostředky mají být použity, pokud je lokalizace nastavena na angličtinu. Do složky `values-en` tak budeme ukládat hodnoty (texty, pole apod.) v anglickém jazyce. Existuje řada dalších kvalifikátorů, např. `port` a `land` udávající orientaci displeje, při které se mají dané prostředky použít. Kvalifikátory `hdpi`, `ldpi` a `mdpi` udávají jemnost rozlišení obrazovky přístroje. Více informací o zdrojích viz <http://developer.android.com/guide/topics/resources/>.



Obrázek 3.9: *Struktura nově vytvořeného Android projektu*

`libs/`

Tato složka se nevytvoří automaticky, ale v praxi se často používá pro uchování knihoven třetích stran, které jsou potřeba pro běh naší aplikace.

`AndroidManifest.xml`

XML soubor, který popisuje aplikaci, její komponenty (aktivity, služby), požadovaná práva a mnoho dalšího. Každá aplikace musí v kořenovém adresáři obsahovat tento soubor. Obsahuje základní informace o aplikaci, které musí operační

systém mít předtím, než bude moci aplikaci spustit. Jak vypadá manifest po vytvoření projektu je vidět na následujícím výpisu.

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

    <uses-sdk android:minSdkVersion="4" />

    <application
        android:icon="@drawable/ic_launcher"
        android:label="@string/app_name" >
        <activity
            android:name=".Hlavni"
            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>
```

Kořenem je element `manifest`, který obsahuje definici jmenného prostoru, název balíku aplikace, kód verze a název verze. Poslední dva údaje jsou povinné pro publikování aplikace a při každém vydání nové verze je nutné změnit minimálně atribut `versionCode`. Jedná se o atribut, který uvádí interní číslo verze, kde čím větší přirozené číslo, tím novější verze.

Povinným elementem, který udává minimální, cílovou a maximální verzi systému, na kterém půjde aplikace spustit, je element `uses-sdk`. Velmi důležité je uvést minimální verzi (atribut `minSdkVersion`), naopak udávat maximální verzi se nedoporučuje. Jako cílovou verzi bychom měli uvést maximální verzi, na které jsme aplikaci otestovali.

Pod elementem `application` se nachází jednotlivé `activity`. Pomocí elementu `intent-filter` se udávají podmínky, za kterých se má aktivita zobrazit. Kromě aktivit může element `application` obsahovat i služby (`service`), přijímače (`receiver`) a poskytovatele obsahu (`provider`).

Pokud chceme, aby naše aplikace využívala některé vlastnosti zařízení, jako je např. určení zeměpisné polohy, přístup k internetu apod., musíme si vyžádat oprávnění. Uživatel nainstalováním naší aplikace souhlasí s udělením uvedených práv. Práva se označují pomocí elementu `uses-permission`. Zmíněná práva by v manifestu vypadala následovně:

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

Kompletní popis a struktura souboru viz <http://developer.android.com/guide/topics/manifest/manifest-intro.html>.

3.2.2 Sestavení a spuštění aplikace

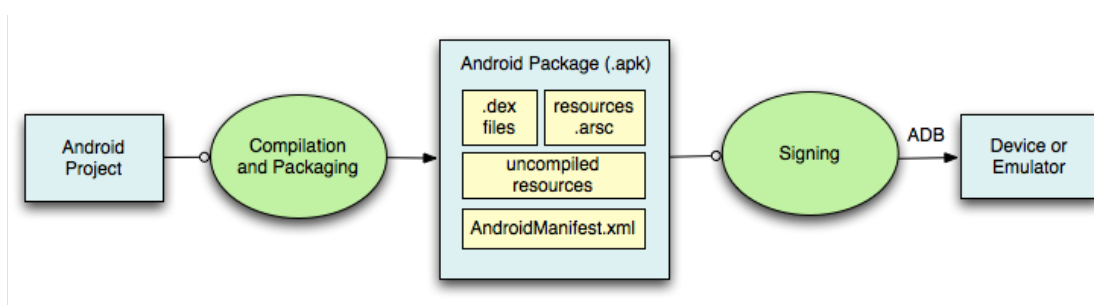
Následující informace a obrázky jsou převzaty a upraveny z [12] ze sekce Dev Guide -> Building and Running.

Android projekty jsou při sestavení zabaleny do souboru `.apk`, což je kontejner pro binární aplikaci. Obsahuje všechny informace potřebné pro běh aplikace (na fyzickém zařízení i na emulátoru), jako jsou zkompileované `.dex` soubory (`.class` soubory konvertované do Dalvik byte kódu), binární verzi souboru `AndroidManifest.xml`, zkompileované a nekompileované prostředky (resources).

Při vyvíjení aplikace v Eclipse plugin ADT postupně sestavuje projekt tak, jak měníme a ukládáme zdrojový kód (autobuild). Soubor `.apk` lze pak nalézt v adresáři `bin` projektu.

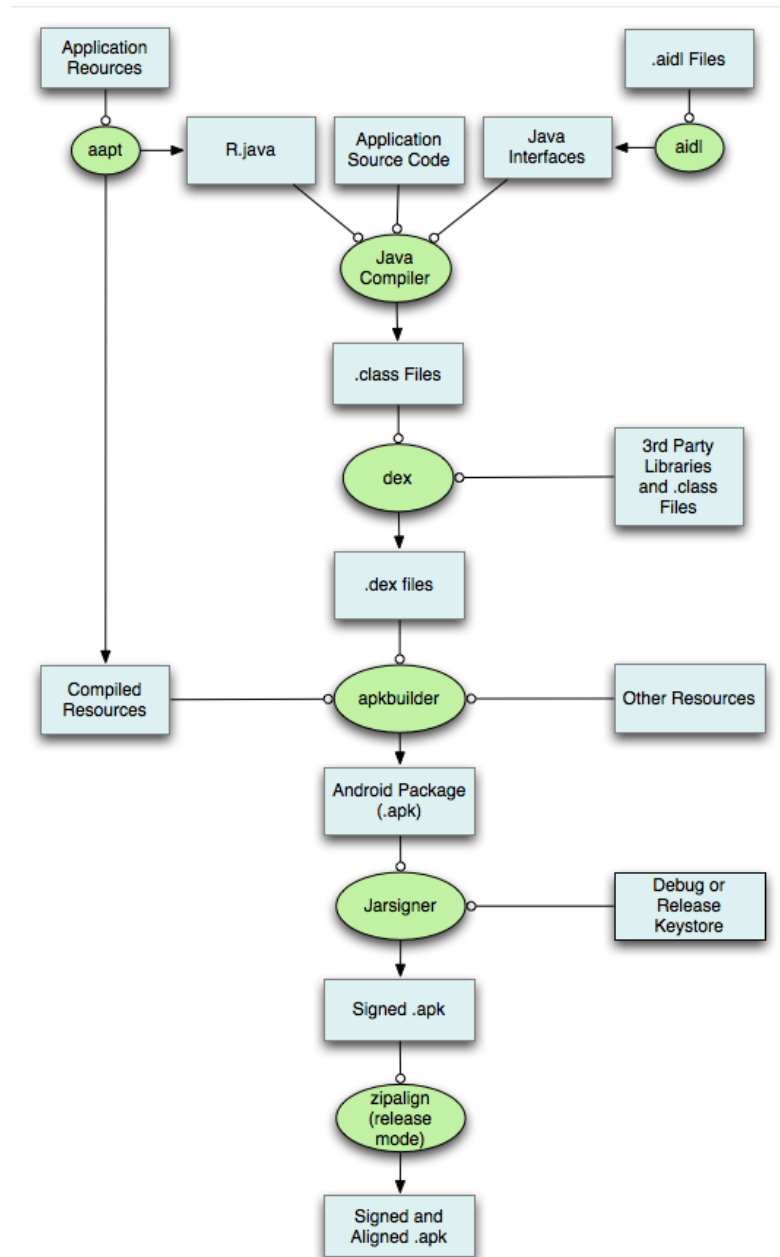
Chceme-li spustit aplikaci v emulátoru nebo na zařízení, aplikace musí být podepsána s použitím debugovacího nebo release módu. Pro ladění a testování je vhodné podepsat aplikaci v debug módu, protože nástroje na sestavení použijí debugovací klíč se známým heslem, takže heslo nemusíme zadávat při každém sestavení aplikace. Pokud chceme aplikaci vydat a publikovat na Android Market, je nutné ji podepsat v release módu s použitím vlastního privátního klíče. Více informací o podepisování aplikace viz kapitola 5.

Následující diagram (obr. 3.10) zobrazuje komponenty, které se podílejí na sestavení a spuštění aplikace.



Obrázek 3.10: Sestavení a spuštění aplikace

Při detailním pohledu vidíme různé nástroje a procesy, které se zapojují do sestavení aplikace (obr. 3.11).



Obrázek 3.11: Sestavení aplikace

Typické sestavení aplikace zahrnuje:

- *Android Asset Packaging Tool* (aapt) zkompile soubory s prostředky, jako jsou `AndroidManifest.xml` a XML soubory pro aktivity. Dále vygeneruje soubor `R.java`, takže můžeme na prostředky odkazovat v Java kódu.
- Nástroj *aidl* zkonvertuje všechny `.aidl` soubory rozhraní do Java rozhraní.
- Veškerý Java kód je zkompilován do `.class` souborů.
- Nástroj *dex* zkonvertuje `.class` soubory do Dalvik byte kódu. Jakékoliv další knihovny třetích stran a `.class` soubory jsou také zkonvertovány.
- Veškeré nekompilovatelné prostředky (např. obrázky), kompilované prostředky a `.dex` soubory jsou poslány do nástroje *apkbuilder*, aby byly zabaleny do souboru `.apk`.
- Jakmile je vytvořen `.apk` soubor, je nutné jej podepsat debug nebo release klíčem.
- Nakonec, pokud byla aplikace podepsána v release módu, je nutné „zarovnat“ `.apk` soubor pomocí nástroje *zipalign* za účelem optimalizace využití paměti při běhu na zařízení.

3.2.3 Aktivity

Informace k této kapitole byly získány z [12] ze sekce `Dev Guide` -> `Activities`.

Aktivity jsou komponenty aplikace, které nabízejí obrazovky pro interakce uživatele. Každá aktivita má přiřazený pohled (`View`), který popisuje obrazovku uživatelského rozhraní, přes kterou uživatel komunikuje s aplikací. Jedna aplikace je obvykle složená z více aktivit, které jsou k sobě volně vázány. Jedna aktivita je označena jako hlavní a je to ta, která se spustí při prvotním spuštění aplikace. Každá aktivita může spustit další aktivitu, která je vložena na vrchol zásobníku, přičemž aktuální aktivita je pozastavena. K vrácení se k předchozí aktivitě může sloužit tlačítko `Zpět`. Pokud je v zásobníku je jen jedna aktivita a stiskneme tlačítko `Zpět`, tak obvykle dojde k ukončení celé aplikace.

Vytvoření aktivity

Aktivitu vytvoříme jako třídu oddělenou od třídy `Activity`. Ve třídě implementujeme metody, které volá systém, když aktivita prochází stavy svého životního cyklu. Mezi nejdůležitější metody patří metoda `onCreate()`, která je volána, když je aktivita vytvořena, a měla by obsahovat základní komponenty aktivity a také definici rozvržení pro uživatelské rozhraní. V metodě `onPause()`

by mělo dojít k uložení změn, které by měly být persistentní, protože volání této metody indikuje, že uživatel opouští aktivitu. Ostatní metody spjaté s životním cyklem aktivity budou popsány dále.

Uživatelské rozhraní pro aktivitu zajišťuje hierarchie pohledů, což jsou objekty odvozené od třídy `View`. Android nabízí řadu připravených pohledů, které lze použít pro tvorbu rozvržení použít.

Spuštění aktivity

Aktivitu můžeme spustit pomocí volání metody `startActivity()`, které se předá intent (záměr), který popisuje aktivitu, kterou chceme spustit. Intent může specifikovat konkrétní aktivitu nebo popisovat typ akce, kterou chceme spustit (systém vybere příslušnou aktivitu, která může být i z jiné aplikace). Intent může obsahovat i data, která jsou potřeba pro spuštění aktivity.

```
Intent intent = new Intent(this, Nastaveni.class);
startActivity(intent);

Intent intent = new Intent(Intent.ACTION_SEND);
intent.putExtra(Intent.EXTRA_EMAIL, adresyPrijemcu);
startActivity(intent);
```

Pokud chceme získat výsledek ze spuštěné aktivity, voláme aktivitu pomocí metody `startActivityForResult()`. Pro obdržení výsledku je nutné implementovat metodu `onActivityResult()`, která je zavolána, pokud je spuštěná aktivita ukončena.

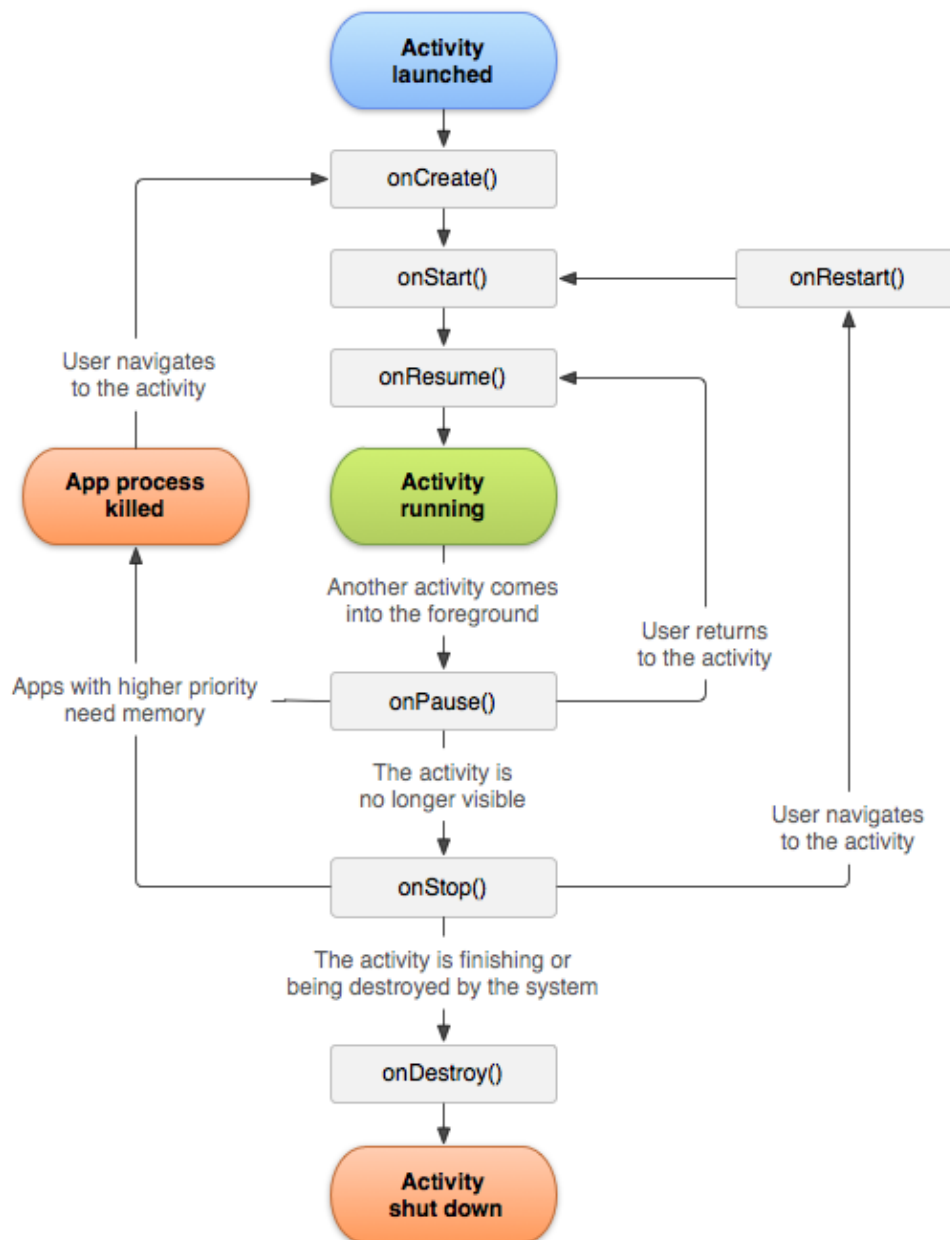
Ukončení aktivity

Aktivita se ukončuje voláním metody `finish()`. Tuto metodu není potřeba explicitně volat pro naše aktivity, ale měla by se používat jen v případech, kdy nechceme, aby se uživatel do této aktivity vrátil.

Životní cyklus

Pochopení životního cyklu aktivit a implementace daných metod je pro programování Android aplikací klíčové. Abychom reagovali na změny stavu aktivity, přepíšeme dané metody událostí. Aktivita se může nacházet v jednom ze tří stavů: *obnovena (běžící)*, *pozastavena* a *zastavena*. Aktivita ve stavu *pozastavena* nebo *zastavena* může být z paměti odstraněna. Pokud je aktivita následně otevřena, je nutné ji vytvořit znovu od začátku.

Na obrázku 3.12 (převzat z [12]) je vidět životní cyklus aktivity. V tabulce 2 jsou popsány jednotlivé metody životního cyklu.



Obrázek 3.12: Životní cyklus aktivity

Metoda	Popis	Následuje
<code>onCreate()</code>	Metoda se volá vždy, když je aktivita poprvé vytvořena. V této metodě by se mělo provést veškeré statické nastavení, jako je inicializace uživatelského rozhraní, načtení dat, nastavení listenerů apod. Metodě je předán objekt obsahující předchozí stav aktivity (pokud existuje), který může být využit pro obnovu aktivity.	<code>onStart()</code>
<code>onRestart()</code>	Tato metoda je volána, když byla aktivita zastavena a restartuje se.	<code>onStart()</code>
<code>onStart()</code>	Metoda je volána, když se aktivita zobrazí uživateli.	<code>onResume()</code> nebo <code>onStop()</code>
<code>onResume()</code>	Metoda se volá těsně před tím, než je aktivita přesunuta do popředí, např. po prvním spuštění, restartu, uvolnění dialogového okna apod. V této metodě by mělo docházet k obnově uživatelského rozhraní.	<code>onPause()</code>
<code>onPause()</code>	Zavolána, pokud je aktivita přesunuta na pozadí. Zde by se měly uložit změny, které proběhly a mají být zachovány a měly by být uvolněny prostředky, které jsou v držení.	<code>onResume()</code> nebo <code>onStop()</code>
<code>onStop()</code>	Metoda se volá ve chvíli, kdy se má aktivita zastavit.	<code>onRestart()</code> nebo <code>onDestroy()</code>
<code>onDestroy()</code>	Zavolána při ukončování aktivity, buď tím, že se zavolala metoda <code>finish()</code> , nebo proto, že systém potřebuje více operační paměti a aktivitu ukončí.	

Tabulka 2: *Metody životního cyklu aktivity*

3.2.4 Služby

Služba je součástí aplikace, která může provádět dlouhotrvající operace na pozadí, přičemž neposkytuje uživatelské rozhraní. Jiná součást aplikace (např. aktivita) může spustit službu a ta bude poté pokračovat na pozadí, i když uživatel přepne do jiné aplikace. Služby se mohou spouštět manuálně (pomocí `startService()`) nebo jsou spuštěny, když se nějaká aktivita pokusí ke službě připojit prostřednictvím meziprocesové komunikace (IPC). Pro služby platí, že v paměti může být spuštěna vždy jen jedna její instance.

Vytvoření služby

Psaní služby se podobá psaní aktivity. Vytvoříme třídu oddělenou od třídy `Service` a přepíšeme některé z metod životního cyklu služby. Služby mají stejně jako aktivity metody `onCreate()`, `onStart()` a `onDestroy()`. Další důležité metody zpětného volání jsou metody `onStartCommand()` (při spuštění služby) a `onBind()` (při připojení jiné komponenty).

Spuštění a ukončení služby

Jak již bylo řečeno, z aktivity spustíme službu metodou `startService()`, které předáme instanci třídy `Intent` specifikující službu, kterou chceme spustit. Systém Android poté zavolá v dané službě metodu `onStartCommand()` a předá jí `Intent`. Pokud služba zatím neběží, je před její metodou `onStartCommand()` nejprve zavolána metoda `onCreate()`.

```
Intent intent = new Intent(this, UpdateWidgetService.class);
startService(intent);
```

Služba se může sama ukončit voláním metody `stopSelf()` nebo ji může ukončit jiná aktivita voláním `stopService()`, kde je nutné předat odkaz na stejnou instanci `Intent`, jako při jejím spuštění. Pokud byla služba připojena ke klientské aplikaci pomocí `bindService()`, je ukončena až když všechny klientské aplikace zavolají `unbindService()`. Systém násilně ukončuje službu pouze v případě nedostatku paměti, kdy musí obnovit systémové prostředky pro aktivitu, která je v popředí.

Služby v manifestu

Služby, stejně jako aktivity, musí být uvedeny v manifestu aplikace v elementu `application`.

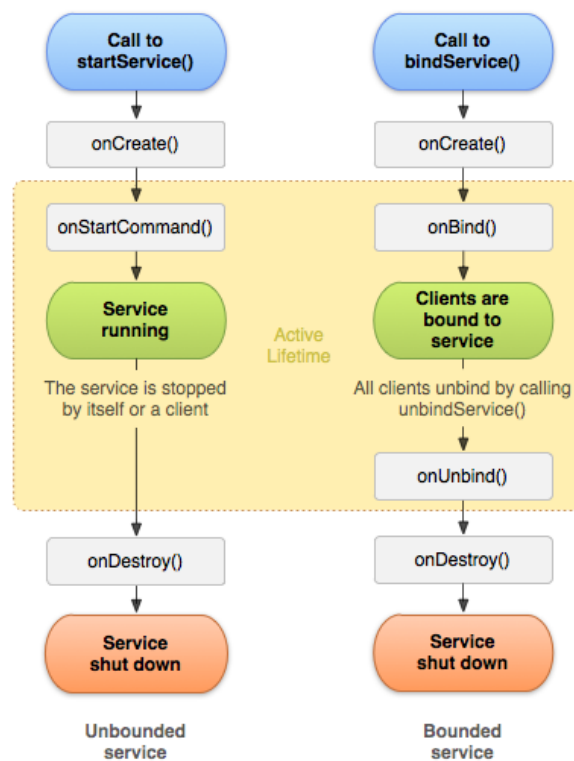
```
<application ... >
  <service android:name=".services.UpdateWidgetService" />
</application>
```


Jediným povinným atributem je název služby, což je plně kvalifikované jméno třídy, nebo lze použít i relativní cestu od definovaného balíku. Dále můžeme definovat i oprávnění, která jsou potřebná pro běh dané služby.

Životní cyklus

Životní cyklus služeb je jednodušší než životní cyklus aktivit. Nicméně je důležité věnovat pozornost tomu, jak se služby vytvářejí a ukončují, protože služby mohou běžet na pozadí bez vědomí uživatele. Špatně navržená služba může zbytečně zabírat systémové prostředky a vybit baterii přístroje.

Podle způsobu spuštění služby existují dva životní cykly. Jeden pro službu spuštěnou pomocí `startService()` a druhý, kdy došlo k připojení klientské aplikace pomocí `bindService()`. Na obrázku 3.13 (převzat z [12]) jsou vidět oba dva životní cykly služby.



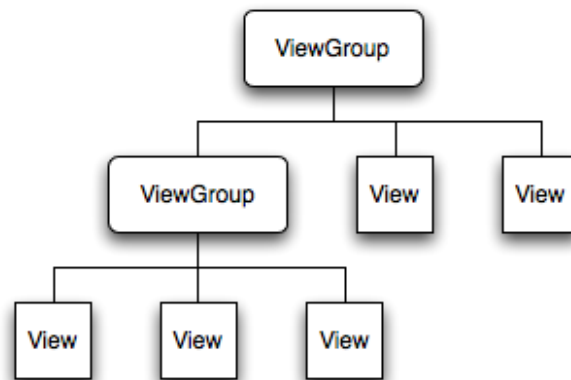
Obrázek 3.13: Životní cykly služby

3.2.5 Uživatelské rozhraní

V Android aplikacích je uživatelské rozhraní postaveno na objektech typu `View` a `ViewGroup`. `View` objekty slouží jako základ pro podtřídy nazývané „widgety“, které poskytují plně implementované UI objekty jako jsou textová pole a tlačítka. `ViewGroup` je základem pro podtřídy nazývané „layouty“ (nebo také ná-

vrhy), které nabízejí různé druhy návrhů architektury, jako je lineární, tabulkový nebo relativní.

Uživatelské rozhraní aktivit se definuje jako hierarchie objektů `ViewGroup` a `View`. Hierarchie je ukázána na následujícím diagramu (obr. 3.14) a může být tak jednoduchá či složitá, jak je potřeba.



Obrázek 3.14: *Strom hierarchie objektů uživatelského rozhraní*

Aktivitě se UI nastavuje pomocí metody `setContentView()`, které předáme odkaz na objekt, který je kořenem stromu. Objekty je možné vytvářet přímo v Java kódu, ale běžnější způsob je použít XML soubor s návrhem uživatelského rozhraní. Každý element v XML je buď `View` nebo `ViewGroup` objekt (nebo jejich potomek). Názvy elementů odpovídají názvům objektů, takže např. element `<TextView>` vytvoří v UI objekt `TextView` a element `<LinearLayout>` vytvoří objekt `LinearLayout`, což je potomek `ViewGroup`.

Použití deklarativního GUI pomocí XML přináší výhody, jako je lepší oddělení prezentační vrstvy aplikace od kódu, který řídí její chování. Popisy uživatelského rozhraní jsou pro aplikační kód externím zdrojem, proto je lze upravovat, aniž by bylo nutné upravovat zdrojový kód a sestavovat celou aplikaci. Také je možné vytvářet návrhy pro různá rozlišení obrazovky a různé jazyky. Pokud je potřeba vytvářet uživatelské rozhraní dynamicky, lze použít částečně definované návrhy a dodefinovat je za běhu aplikace, nebo vytvořit kompletní objekty přímo v kódu.

XML soubor s návrhem uživatelského rozhraní se v projektu umísťuje do `res/layout` a jeho načtení je zajištěno pomocí třídy `R.java`. Pokud chceme návrh použít jako vzhled aktivity, musíme jej předat metodě `setContentView()` volané z `onCreate()`.

```
public void onCreate(Bundle savedInstanceState) {  
    super.onCreate(savedInstanceState);  
    setContentView(R.layout.mu_j_navrh);  
}
```

Následující kód ukazuje XML návrh uložený v souboru `muj_navrh.xml`.

```
<?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" >
    <TextView android:id="@+id/text"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Text s~popisem" />
    <Button android:id="@+id/button"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/btn_potvrdit" />
</LinearLayout>
```

Kořenovým elementem je `LinearLayout`, který obsahuje prvek pro text a tlačítko. Každý element může obsahovat různé atributy, které definují např. výšku, šířku, text apod. Nastavení šířky či výšky na `fill_parent` způsobí, že je prvek roztáhnut na velikost své nadřazené skupiny prvků. Naopak s hodnotou `wrap_content` se prvek chová tak, aby maximálně obemknul svůj obsah. Šířku i výšku můžeme nastavit napevno pomocí `android:width` a `android:height`, ale není to doporučováno, protože může nastat problém u zařízeních s rozdílným rozlišením displeje. Uvedený `LinearLayout` vkládá prvky vedle sebe vertikálně či horizontálně (podle nastavení). Existují i další layouts, např. `RelativeLayout` umisťuje prvky na určité místo vzhledem k pozici jiného prvku.

Každý element (objekt) může obsahovat atribut `id`, který slouží pro unikátní identifikaci daného objektu. Identifikátor je celé číslo, ale můžeme využít zástupný text (např. `@+id/text`), podle kterého pak můžeme daný objekt nalézt a použít ve zdrojovém kódu. Symbol `@` (zavináč) na začátku řetězce indikuje, že XML parser by měl parsovat zbytek textu a identifikovat jej jako ID prostředku. Symbol `+` (plus) znamená, že se jedná o nový název prostředku a bude automaticky vytvořen a přidán do souboru `R.java`. Získání instance tlačítka z návrhu a změna textu se provede následovně:

```
Button tlacitko = (Button) findViewById(R.id.button);
tlacitko.setText(getResources().getString(R.string.btn_zrusit));
```

Text prvků můžeme zadat přímo nebo jako `@string/tlacitko_text`. Zápis znamená, že název tlačítka má být brán ze souboru `res/values/string.xml`. Obsah tohoto souboru může vypadat např. takto:

```
<resources>
    <string name="btn_potvrdit">Povrdit</string>
    <string name="btn_zrusit">Zrušit</string>
</resources>
```

3.2.6 Uživatelské rozhraní – komponenty

V této kapitole budou popsány komponenty, které pomáhají dotvářet uživatelské rozhraní. Těmito komponentami jsou menu, dialogy a notifikace.

Menu

Menu jsou běžné komponenty uživatelského rozhraní používané v mnoha typech aplikací. V Android aplikacích rozlišujeme několik typů menu. Jedná se o menu nabídek, kontextové a vyskakovací (popup) menu.

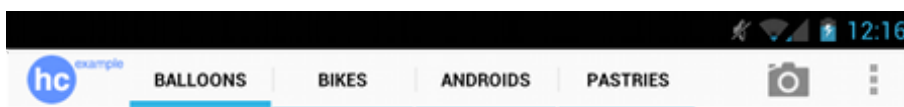
Všechny typy menu se mohou definovat pomocí XML souboru umístěného v `res/menu/`. Toto menu pak můžeme ve zdrojovém kódu načíst jako objekt `Menu`. Ukázka XML souboru `main_menu.xml` následuje.

```
<?xml version="1.0" encoding="utf-8"?>
<menu xmlns:android="http://schemas.android.com/apk/res/android">
  <item android:id="@+id/new_item"
        android:icon="@drawable/ic_menu_add"
        android:title="@string/pridej_polozku">
    <item android:id="@+id/nastaveni"
          android:icon="@drawable/ic_menu_preferences"
          android:title="@string/nastaveni" />
  </item>
</menu>
```

Každá položka menu obsahuje identifikátor, ikonu vzhledu a popis. Do menu je možné vkládat i podmenu, a to přidáním elementů `menu` a příslušných `item` pod první element `menu`.

Nejběžnějším typem menu, který by měl být v každé aplikaci, je tzv. menu nabídek. V tomto menu by měly být základní akce a další možnosti spjaté se současnou aktivitou nebo celou aplikací, jako např. odkaz na aktivitu s nastavením aplikace.

Při vývoji aplikace Android 3.0 a vyšší jsou položky menu přístupné přes plovoucí lištu akcí tzv. *action bar* (obr. 3.15). Do verze Android 2.3.x je menu vyvoláno stisknutím hardwarového *Menu* tlačítka a zobrazuje se v dolní části obrazovky (obr. 3.16).



Obrázek 3.15: Menu nabídek – Android 3.0 a vyšší



Obrázek 3.16: Menu nabídek – Android 2.3.x a nižší

Abychom specifikovali, jaký XML soubor popisující menu se má v aktivitě použít, je nutné přepsat metodu `onCreateOptionsMenu()`.

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

Když uživatel vybere určitou položku z menu, je nutné specifikovat akci, která se má provést. To provedeme přepsáním metody `onOptionsItemSelected()`.

```
@Override
public boolean onOptionsItemSelected(MenuItem item) {
    switch (item.getItemId()) {
        case R.id.new_item:
            newItem();
            return true;
        case R.id.nastaveni:
            startActivity(new Intent(this, Nastaveni.class));
            return true;
        default:
            return super.onOptionsItemSelected(item);
    }
}
```

Kontextové menu se využívá při interakci s určitou položkou uživatelského rozhraní. Vyvolání kontextového menu se nejčastěji provádí delším přidržením

prstu (stylusu) nad určitou položkou. Obdobně jako u předchozího typu menu i zde jsou metody pro vytvoření menu a reakci na výběr položky. Jedná se o metody `onCreateContextMenu()` a `onContextItemSelected()`. Obdobné je i vyskakovací menu, které je ale dostupné až od API úrovně 11.

Dialogy

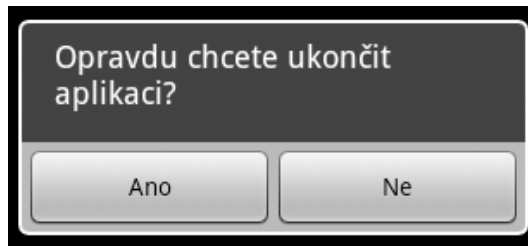
Dialog je obvykle okno, které se objeví před aktuální aktivitou. Tato aktivita ztrácí fokus a dialog reaguje na veškerou interakci s uživatelem. Dialogy se nejčastěji používají pro oznámení nějaké zprávy uživateli a k provádění krátkých úkolů, které se přímo vztahují k průběhu aplikace (jako např. ukazatel průběhu nebo přihlašovací dialog).

Dialog je v Android aplikacích tvořen pomocí třídy `Dialog` nebo jeho potomků. Každý dialog je vždy vytvořen a zobrazen jako součást aktivity a měl by se vytvářet v rámci metody zpětného volání `onCreateDialog(int)`. Pro zobrazení dialogu poté v aktivitě voláme metodu `showDialog(int)`, které předáme celé číslo identifikující dialog, který chceme zobrazit. Pokud chceme mít v aktivitě více různých dialogů je vhodné v metodě `onCreateDialog(int)` použít příkaz `switch` a podle vstupního parametru se rozhodnout, který dialog má být zobrazen. Dialog zrušíme zavoláním metody `dismiss()`.

Nejpoužívanějším typem dialogu je `AlertDialog`, který je rozšířením třídy `Dialog`. Dialog umožňuje zobrazit nadpis, text nějaké zprávy, jedno až tři tlačítka a seznam položek pro výběr (viz obr. 3.18). K vytvoření a nastavení dialogu se používá podtřída `AlertDialog.Builder`. Na následující výpisu je vidět nastavení zprávy a tlačítka pro potvrzení akce (zde ukončení aplikace) a pro zrušení dialogu. Obrázek 3.17 ukazuje jak vypadá dialog v aplikaci.

```
AlertDialog.Builder builder = new AlertDialog.Builder(this);
builder.setMessage("Opravdu chcete ukončit aplikaci?")
    .setCancelable(false)
    .setPositiveButton("Ano", new DialogInterface.OnClickListener() {
        public void onClick(DialogInterface dialog, int id) {
            Hlavni.this.finish();
        }
    })
    .setNegativeButton("Ne", new DialogInterface.OnClickListener() {
        public void onClick(DialogInterface dialog, int id) {
            dialog.cancel();
        }
    });
AlertDialog alert = builder.create();
```

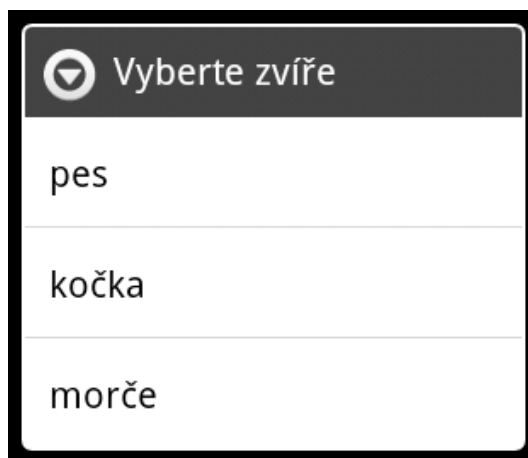
Na dalším výpisu je vidět vytvoření dialogu pro zobrazení seznamu vybíratelných položek. Po kliknutí na položku bude vypsán její název pomocí třídy `Toas` (viz dále Notifikace). Spuštěný dialog v samotné aplikaci je vidět na obrázku 3.18.



Obrázek 3.17: Ukázka dialogu – tlačítka

```
final CharSequence[] items = { "pes", "kočka", "morče" };

AlertDialog.Builder builder = new AlertDialog.Builder(this);
builder.setTitle("Vyberte zvíře");
builder.setItems(items, new DialogInterface.OnClickListener() {
    public void onClick(final DialogInterface dialog, final int item) {
        Toast.makeText(getApplicationContext(), items[item], Toast.
            LENGTH_SHORT).show();
    }
});
AlertDialog alert = builder.create();
```



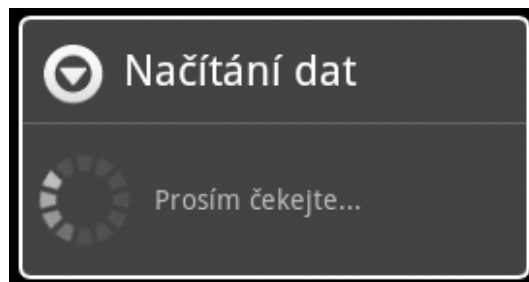
Obrázek 3.18: Ukázka dialogu – seznam položek

Pokud bychom u předchozího příkladu chtěli zobrazit seznam pro výběr více položek (*checkbox*) nebo jedné položky (*radio button*), použili bychom místo metody `setItems()` metodu `setMultiChoiceItems()`, respektive metodu `setSingleChoiceItems()`.

Pro potřeby zobrazení animace určitého postupu (např. při načítání nebo stahování dat) se používá `ProgressDialog`, potomek třídy `AlertDialog`. Ten umožňuje zobrazovat postup pomocí otáčejícího se kolečka, pokud postup není přesně určen, nebo pomocí tzv. *progress bar*. Dialog také může obsahovat tlačítka, např. pro zrušení probíhajícího stahování.

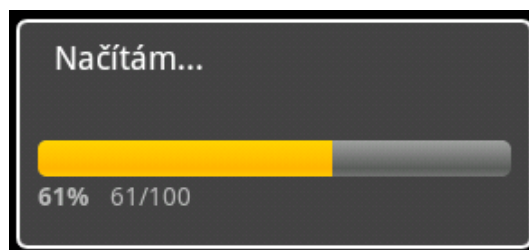
Jednoduchý `ProgressDialog` se spustí voláním `ProgressDialog.show()` jako je ukázáno na výpisu. Prvním parametrem je kontext aplikace, následuje titulek, zpráva a posledním parametrem je určení, zda je postup neurčitý (hodnota `true`). Ukázka dialogu opět následuje (obr. 3.19).

```
ProgressDialog dialog = ProgressDialog.show(Hlavní.this, "Načítání dat",  
    "Prosím čekejte...", true);
```



Obrázek 3.19: Ukázka dialogu – načítání

Vytvoření dialogu, který by zobrazoval postup např. pomocí procent, je jednoduché. Tou obtížnější částí je jeho obsluha, která zahrnuje vytvoření obslužného vlákna, které do původního vlákna posílá informace o postupu. Jak takový dialog vypadá, je vidět na obrázku 3.20. Ukázku vytvoření a spuštění takového dialogu můžete nalézt na <http://developer.android.com/guide/topics/ui/dialogs.html>. Na této stránce naleznete i postup, jak nastavit vlastní design pro dialogy, pokud se vám výchozí vzhled dialogů nelíbí nebo nehodí.



Obrázek 3.20: Ukázka dialogu – progress bar

Notifikace

Při běhu aplikace mohou nastat situace, kdy je potřeba uživateli oznámit, že nastala určitá událost. Některá oznámení či upozornění vyžadují reakci uživatele, jiné nikoliv. Například pokud došlo k doručení SMS zprávy, můžeme tuto skutečnost oznámit uživateli prostřednictvím malé zprávy, která se objeví na displeji a po chvíli sama zmizí. Pokud aplikace běží na pozadí a je potřeba interakce od uživatele, mělo by být vytvořeno oznámení, přes které může uživatel s aplikací komunikovat. Další možností je při provádění určité činnosti (např. stahování souboru) zobrazit dialog s postupem práce (viz předchozí část).

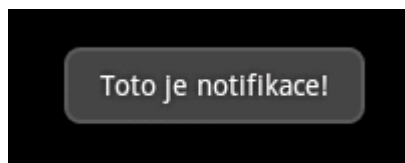
Existují tedy tři typy oznámení, jedním je krátká zpráva tzv. *Toast*, druhý je oznámení ve stavovém řádku a poslední je vyvolání dialogového okna.

Toast notifikace je krátká zpráva, která se objeví přes obrazovku a po krátké době sama zmizí. Zabírá jen tolik místa, kolik je potřeba pro zobrazení textu zprávy. S tímto typem notifikace nemůže uživatel provádět žádné interakce. *Toast* notifikaci lze vytvářet v aktivitě nebo ve službě; pokud je vytvořena v rámci služby, zobrazí se přes aktuální spuštěnou aktivitu.

Inicializace tohoto oznámení se provádí pomocí metody `makeText()` třídy `Toast`. Tato metoda má tři parametry: aplikační kontext, text zprávy a dobu trvání zprávy. Zobrazení oznámení se provede zavoláním metody `show()` na takto vytvořený objekt.

```
Toast toast = Toast.makeText(getApplicationContext(), "Toto je notifikace!",  
    Toast.LENGTH_SHORT);  
toast.show();
```

Jak vypadá takováto notifikace na obrazovce zařízení je vidět na obrázku 3.21.



Obrázek 3.21: *Toast notifikace*

Standardně se notifikace zobrazuje uprostřed spodní části obrazovky. Pozici můžeme změnit zavoláním metody `setGravity(int, int, int)`, kde prvním parametrem je konstanta pro zarovnání (tzv. `Gravity`), následuje offset od osy X a od osy Y. Pokud chceme posunout pozici vpravo, zvýšíme hodnotu druhého parametru, chceme-li posun směrem dolů, zvýšíme hodnotu posledního parametru. Nastavení zobrazení notifikace v pravém horním rohu vypadá následovně:

```
toast.setGravity(Gravity.TOP|Gravity.RIGHT, 0, 0);
```

Oznámení ve stavovém řádku oproti této notifikaci přidá ikonu do systémového stavového řádku (může u ní zobrazit krátký text) a zprávu do oznamovacího okna. Pokud uživatel klikne na zobrazenou notifikaci, systém spustí `Intent`, který je k této notifikaci přiřazen (obvykle se jedná o aktivitu). Součástí notifikace může být i upozornění pro uživatele pomocí zvuku, vibrace nebo blikání LED diody přístroje.

Oznámení do stavového řádku by se mělo používat vždy v případě, když služba běžící na pozadí potřebuje upozornit uživatele o určité události, která

vyžaduje odpověď. Služby na pozadí by nikdy neměly spouštět aktivity samy od sebe, aby upozornily uživatele.

Notifikaci můžeme opět vytvořit v aktivitě nebo ve službě, častější použití je ve službě běžící na pozadí. K vytvoření této notifikace se používá třídy `Notification` a `NotificationManager`. Pomocí instance třídy `Notification` se definuje nastavení oznámení, jako je ikona, text a další nastavení jako např. zvuk pro přehrání. Parametry konstruktoru jsou ikona, text do stavového řádku a čas, kdy notifikaci vyvolat. Dále nastavíme titulek a text v oznamovací oblasti a aktivitu pro spuštění.

```
Notification notification = new Notification(R.drawable.icon, "Ahoj!", System
    .currentTimeMillis());
Intent notificationIntent = new Intent(this, Hlavni.class);
PendingIntent contentIntent = PendingIntent.getActivity(this, 0,
    notificationIntent, 0);
notification.setLatestEventInfo(getApplicationContext(), "Titulek notifikace"
    , "Text notifikace", contentIntent);
```

`NotificationManager` je systémová služba, která se stará o veškeré notifikace stavového řádku. K manažeru se nepřístupuje přímo, ale musíme získat jeho referenci pomocí `getSystemService()` a poté mu můžeme předat oznámení pomocí metody `notify()`. Této metodě kromě samotného objektu notifikace předáme i námi nastavené unikátní id. Notifikaci z oznamovacího okna lze smazat zavoláním metody `cancel(int)`, kterému předáme její id.

```
final int HELLO_ID = 1;

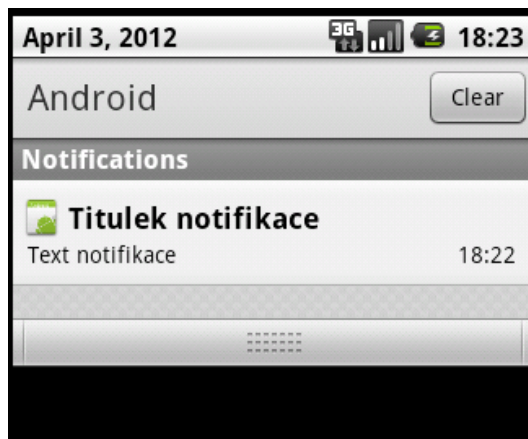
NotificationManager mNotificationManager = (NotificationManager)
    getSystemService(Context.NOTIFICATION_SERVICE);
mNotificationManager.notify(HELLO_ID, notification);
```

Jak vypadá takto vytvořené oznámení na stavovém řádku je vidět na obrázku 3.22, zpráva v oznamovacím okně je na obrázku 3.23.



Obrázek 3.22: Oznámení ve stavovém řádku

Pokud by nám výchozí vzhled notifikace nevyhovoval, můžeme oznámení upravit použitím vlastního XML layoutu. Více informací o notifikacích viz <http://developer.android.com/guide/topics/ui/notifiers/notifications.html>.



Obrázek 3.23: Zpráva v oznamovacím okně

3.3 Ladění a testování

Android SDK poskytuje nástroje pro ladění a testování aplikace. Součástí prostředí Eclipse je JDWP-compliant debugger, který umožňuje krokování kódu, zobrazení hodnot proměnných či pozastavení běžící aplikace. Kromě tohoto debuggeru se při ladění Android aplikací typicky používají nástroje Android Debug Bridge (adb) a Dalvik Debug Monitor Server (DDMS). Ladit a testovat můžeme aplikaci buď na fyzickém zařízení nebo na emulátoru (AVD).

3.3.1 Android Debug Bridge

Android Debug Bridge (adb) je univerzální nástroj pro příkazovou řádku, který nám umožňuje komunikovat s emulátorem nebo s připojeným Android zařízením. Jedná se o program typu klient-server, který zahrnuje tři komponenty:

- Klient, který běží na vývojovém stroji. Klienta lze spustit z shellu pomocí příkazu `adb`. Ostatní Android nástroje, jako je např. DDMS nebo ADT plugin, také vytvářejí adb klienty.
- Server, který běží jako proces na pozadí na vývojovém stroji. Server spravuje komunikaci mezi klientem a adb démonem běžícím na emulátoru nebo zařízení.
- Démon, který běží jako proces na pozadí na každém emulátoru nebo zařízení.

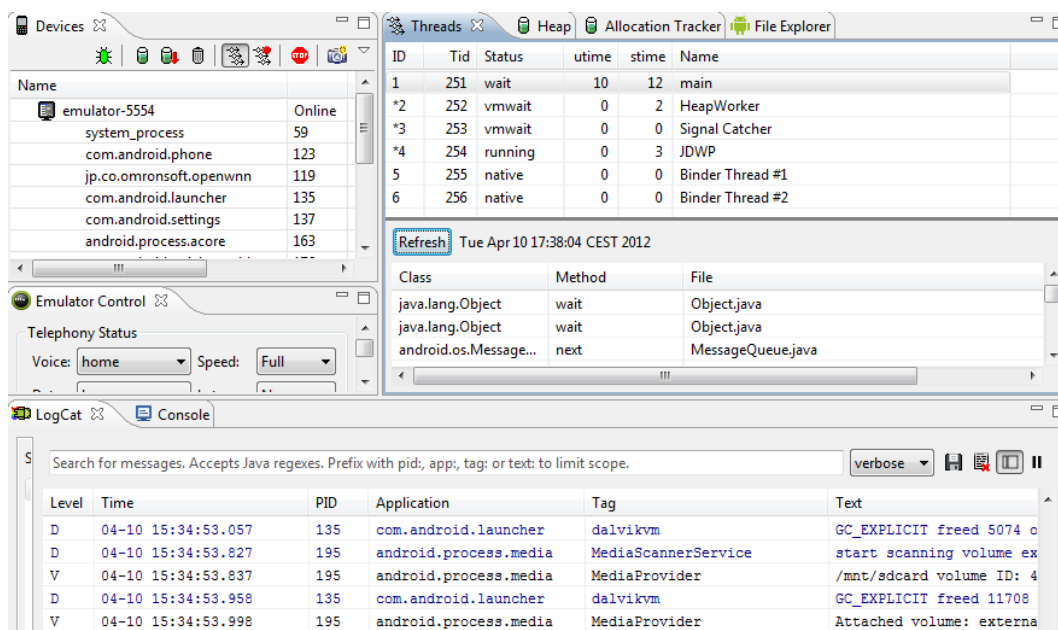
Pomocí příkazu `adb devices` si můžeme nechat vypsat všechna připojená zařízení. Ta jsou popsána identifikátorem ve tvaru `<typ>-<port>` (emulátor by vypadal např. jako `emulator-5554`) a stavem, který může být `offline`,

v případě že zařízení neodpovídá, nebo `device`, pokud je zařízení připojeno k adb serveru.

Tento nástroj umožňuje instalovat aplikace na zařízení, přenášet soubory, restartovat zařízení, nechat si vypsat logování a mnoho dalšího. Více informací o adb viz <http://developer.android.com/guide/developing/tools/adb.html>

3.3.2 Dalvik Debug Monitor Server

DDMS je grafické rozhraní, které komunikuje se zařízeními pomocí adb. Tento nástroj poskytuje informace o vláknech, haldě, procesech, souborech a dále umožňuje zobrazit logování, pořizovat snímky obrazovky a mnoho dalšího. DDMS je integrováno v prostředí Eclipse a spustí se přes **Window > Open Perspective -> Other... > DDMS**. Na obrázku 3.24 je vidět, jak vypadá toto rozhraní.



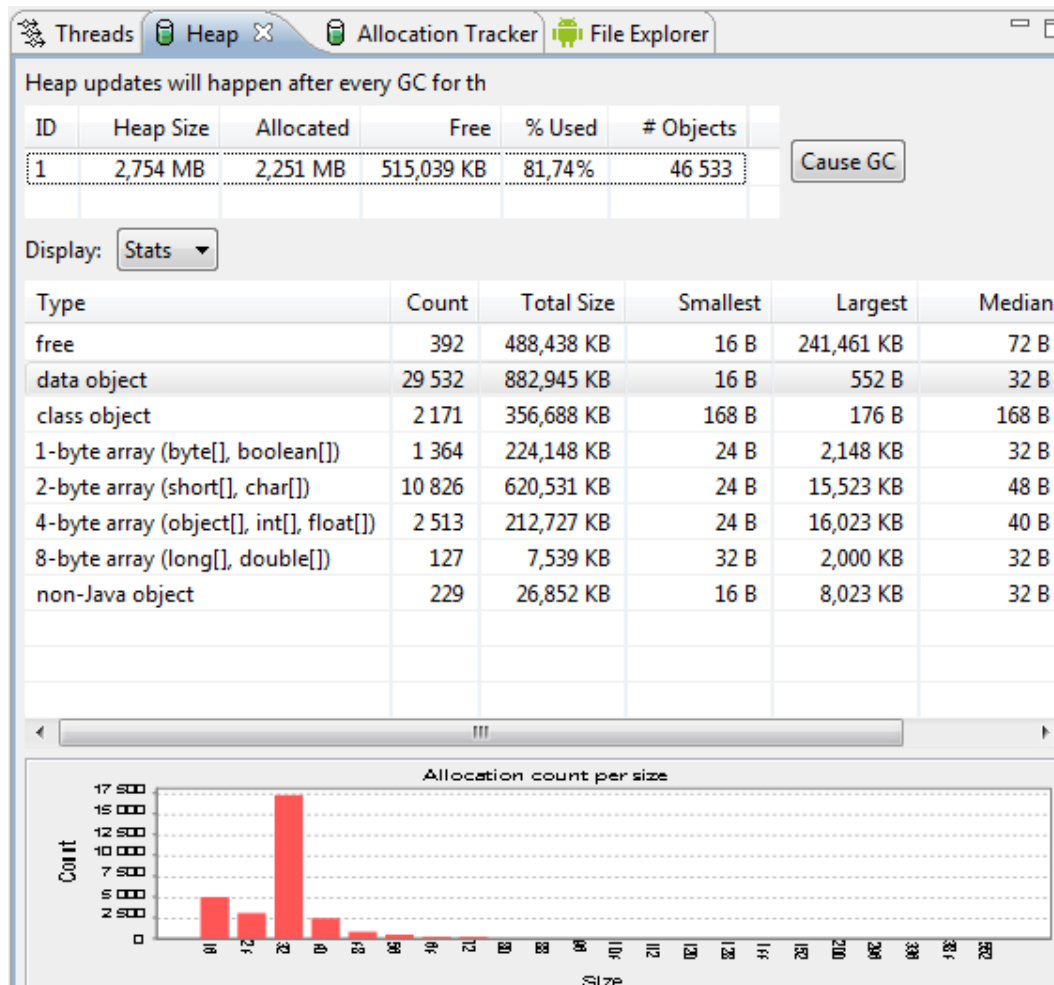
Obrázek 3.24: Dalvik Debug Monitor Server

Nyní popíši nejdůležitější funkce DDMS. Kromě těchto funkcí nástroj nabízí i zobrazení logu (viz samostatně kapitola 3.3.3), sledování metrik metod, jako je počet volání, čas provádění atd.

Využití haldy pro proces

DDMS umožňuje zobrazit, kolik paměti používá konkrétní proces. Můžeme tak sledovat využití haldy v daném okamžiku běhu aplikace a odhalit tak případné problémy (obr. 3.25).

Pro zobrazení využití haldy vybereme na kartě *Devices* proces, který nás zajímá a klikneme na tlačítko *Update Heap*. Na kartě *Heap* klikneme na *Cause GC*, abychom vyvolali uvolnění paměti (*garbage collection*). Po dokončení operace se zobrazí skupina typů objektů a velikost paměti, která byla alokována pro každý typ. Pro obnovení dat klikneme opět na *Cause GC*. Kliknutím na typ objektu je vidět graf zobrazující počet objektů alokovaných pro konkrétní velikost paměti v bajtech.



Obrázek 3.25: DDMS - využití haldy

Sledování alokace paměti objektům

Další funkcí, kterou DDMS poskytuje, je sledování objektů, kterým je alokována paměť, a možnost vidět, jaké třídy a vlákna tyto objekty alokují (obr. 3.26). Funkce nám umožňuje v reálném čase sledovat, jaké objekty jsou vytvářeny při provádění určitých akcí v aplikaci. Tyto informace jsou důležité pro posouzení využití paměti, a tak lze odhalit nežádoucí dopad na výkon aplikace.

Pro sledování alokace paměti klikneme na kartě *Devices* na proces, který nás zajímá, a na kartě *Allocation Tracker* klikneme na tlačítko *Start Tracking*. Klik-

nutím na tlačítko *Get Allocations* získáme seznam objektů, který byl alokován od té doby, kdy jsme naposledy na toto tlačítko klikli. Pro zastavení sledování klikneme na tlačítko *Stop Tracking*. Kliknutím na konkrétní řádek v seznamu jsou vidět detailní informace, jako například metoda a číslo řádku, kde došlo k alokaci objektu.

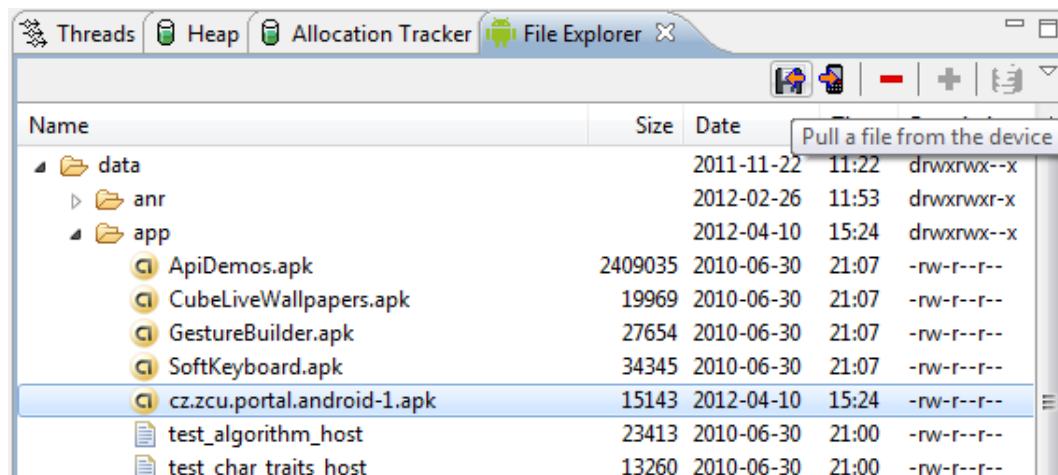
A...	.	Allocated Class	Th...	Allocated in	Allocated in
10	656	float[]	5	android.view.MotionEvent	<init>
12	88	android.view.MotionEvent	5	android.view.MotionEvent	obtain
9	80	long[]	5	android.view.MotionEvent	<init>
7	56	android.view.KeyEvent	5	android.view.KeyEvent\$1	createFromParcel
6	56	android.view.KeyEvent	5	android.view.KeyEvent\$1	createFromParcel
5	56	android.view.KeyEvent	5	android.view.KeyEvent\$1	createFromParcel
4	56	android.view.KeyEvent	5	android.view.KeyEvent\$1	createFromParcel
11	36	int[]	5	android.view.MotionEvent	<init>

Class	Method	File	Line	Native
android.view.MotionEvent	obtain	MotionEvent.java	260	false
android.view.MotionEvent	access\$000	MotionEvent.java	29	false
android.view.MotionEvent\$1	createFromParcel	MotionEvent.java	1204	false
android.view.MotionEvent\$1	createFromParcel	MotionEvent.java	1209	false
android.view.IWindow\$Stub	onTransact	IWindow.java	119	false
android.os.Binder	execTransact	Binder.java	288	false
dalvik.system.NativeStart	run	NativeStart.java	-2	true

Obrázek 3.26: DDMS - sledování alokace paměti

Práce se souborovým systémem

Na kartě *File Explorer* v DDMS máme možnost pracovat se soubory zařízení nebo emulátoru. Soubory můžeme prohlížet, kopírovat i mazat, jednoduše tak můžeme zkoumat soubory vytvořené naší aplikací nebo přenášet soubory do/z zařízení (obr. 3.27).



Obrázek 3.27: DDMS - souborový systém

3.3.3 Logování

Logovací systém Androida poskytuje mechanismus pro shromažďování a zobrazování ladících informací systému. Log obsahuje systémové zprávy, které obsahují i výpisy chyby (*stack trace*), když zařízení či emulátor vyhodí chybu, a zprávy, které jsme napsali do naší aplikace pomocí třídy `Log`. O zobrazení logu v reálném čase se stará nástroj `LogCat`, který lze spustit před `adb` nebo `DDMS`.

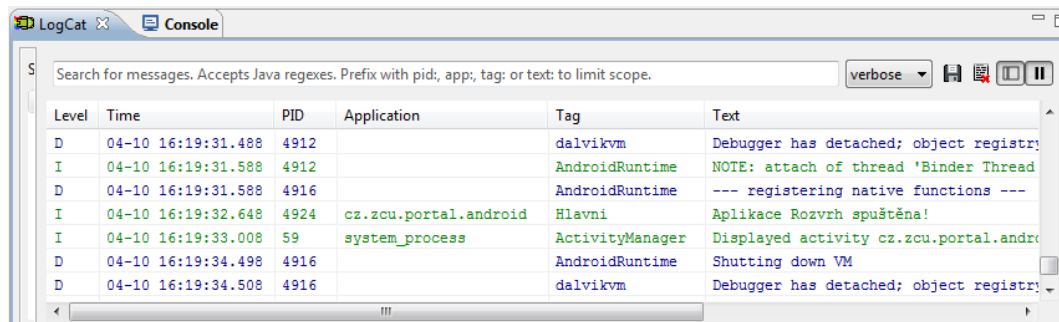
Pro logování v aplikaci se používá třída `Log` a její statické metody, které odpovídají jednotlivým úrovním logování dle závažnosti zprávy.

- `v(String, String)` – verbose (podrobný)
- `d(String, String)` – debug (ladění)
- `i(String, String)` – info
- `w(String, String)` – warning (varování)
- `e(String, String)` – error (chyba)

Prvním parametrem metod je označení odkud se loguje, tzv. tag, druhým parametrem je samotná zpráva. Příklad použití informativní zprávy ve zdrojovém kódu:

```
Log.i("Hlavni", "Aplikace Rozvrh spuštěna!");
```

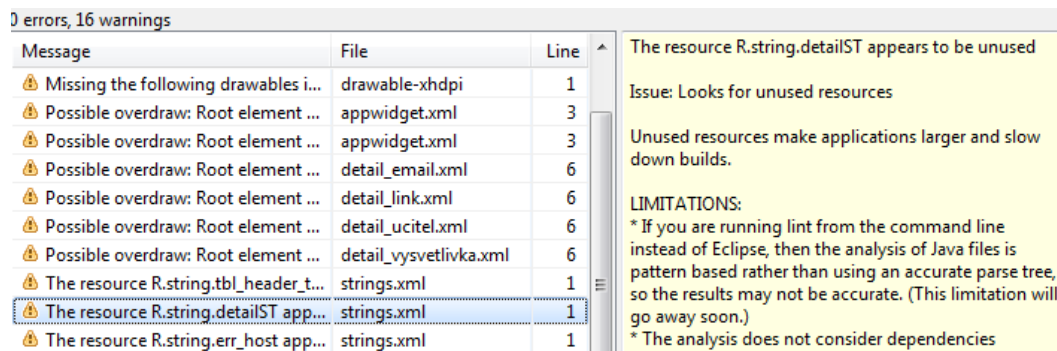
Následující obrázek (3.28) zobrazuje předchozí zprávu v `LogCatu`.



Obrázek 3.28: LogCat

3.3.4 Android Lint

Android Lint je nástroj představený v ADT 16 umožňující skenovat zdrojový kód a prostředky (*resources*) Android projektu pro nalezení potenciálních chyb. Nástroj je přístupný jak z příkazové řádky, tak jako integrace do prostředí Eclipse. Mezi typy chyb, které nástroj nalezne, patří chybějící překlady, problémy s layouty, nepoužité prostředky, chyby v manifestu, texty napsané přímo ve zdrojovém kódu a mnoho dalšího. Ke každé nalezené chybě je zobrazen popis vysvětlující danou chybu a možnost nápravy. Ukázka zobrazení nalezených chyb a varování pomocí nástroje Lint viz obr. 3.29. Více informací o nástroji Lint lze nalézt na adrese <http://tools.android.com/tips/lint>.



Obrázek 3.29: Ukázka nástroje Android Lint

3.3.5 Testování

Android vývojové prostředí zahrnuje testovací framework, který pomáhá otestovat všechny aspekty aplikace. Testování pro Android je postaveno na JUnit testech. Můžeme tak využít obyčejné JUnit testy, které nevolají Android API, nebo použít rozšíření pro testování Android komponent. Základem Android JUnit testů je třída `AndroidTestCase`, od které dědí specifitější testovací třídy jako např. `ServiceTestCase`.

Dále existují nástroje, které umožňují jít v testování ještě dále. Pomocí testovacího frameworku *Robotium* [15] můžeme otestovat aplikaci tak, jako by jí procházel samotný uživatel. Jedná se o způsob testování, jako se používá u webových aplikací pomocí nástroje *Selenium*.

Více o testování bude popsáno v kapitole zaměřené na otestování vlastní aplikace (kapitola 4.5).

3.4 Publikování

Po fázi vytvoření a otestování aplikace zbývá aplikaci publikovat, aby byla přístupná ostatním uživatelům. Aplikaci je pro publikování nejprve nutno připravit. Příprava zahrnuje vytvoření release verze aplikace a její podepsání s použitím privátního klíče.

Aplikaci můžeme publikovat na webových stránkách, zasláním emailem, ale nejlepší způsob, jak aplikaci dostat k co nejvíce lidem, je publikovat aplikaci v online obchodě Google Play. Pro publikování na Google Play je nutné se nejprve registrovat jako vývojář a uhradit jednorázový poplatek 25 dolarů. Po registraci již můžeme přes vývojářskou konzoli (<https://play.google.com/apps/publish>) nahrávat vlastní aplikace.

Podrobný popis vytvoření release verze aplikace a publikování na Google Play bude popsán dále v rámci publikování vlastní aplikace (kapitola 5).

4 Aplikace Rozvrh

V této kapitole popíši mnou vytvořenou aplikaci pro zobrazení rozvrhu hodin studentů ZČU. Jako minimální verze Androidu byla použita verze 1.6, protože jsem chtěla, aby aplikaci mohlo používat co nejvíce studentů. Aplikace je přístupná na Google Play a v současné době jí má nainstalovanou přibližně 1000 uživatelů. Při psaní aplikace jsem se snažila dodržovat zásady dle [16]. Uživatelský manuál k aplikaci je k dispozici v příloze B.

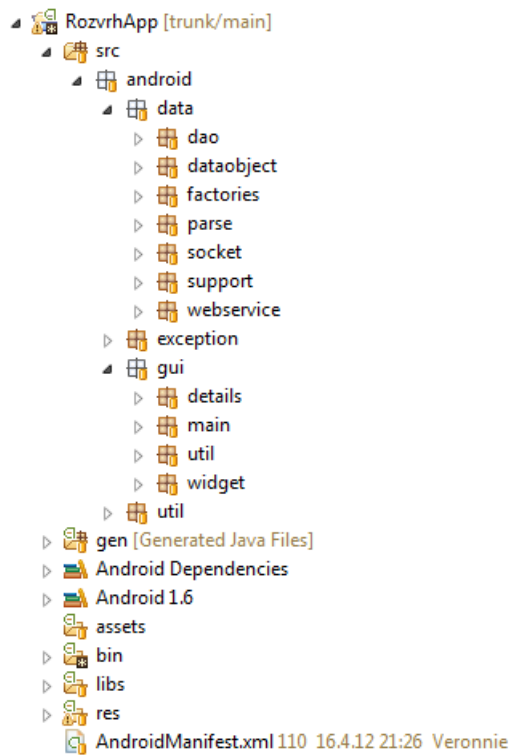
Základními funkcemi aplikace jsou:

- přidání libovolného počtu rozvrhů zadáním osobního čísla nebo vyhledáním pomocí jména
- přehledné zobrazení rozvrhu, detailu předmětů a učitelů
- aktualizace a mazání rozvrhů
- změna jazyka aplikace
- přepínání rozvrhů pomocí gest
- widget na plochu

4.1 Struktura aplikace

Aplikace je rozdělena na tři logické celky, které jsou v aplikaci zastoupeny následujícími balíky. Struktura aplikace je vidět na obrázku 4.30.

1. `cz.zcu.portal.android.data`
Tento balík obsahuje veškerou práci s daty, jako jejich získání pomocí webových služeb, jejich zpracování, ukládání a přístup k těmto datům. Tato část je vytvářena tak, aby byla co nejvíce odstíněna od Android SDK a mohla tak být s menšími obměnami použita i v jiné aplikaci.
Jediná součást Androidu, která se musí v třídách pracujících s databází použít, je kontext dané aplikace, přes který se přistupuje k databázi. Pokud bychom chtěli používat jinou databázi (např. v jiné aplikaci), stačilo by vytvořit novou implementaci DAO (Data Access Objec) objektů a tovarny na tyto objekty.
2. `cz.zcu.portal.android.gui`
Tato část aplikace obsahuje všechny součásti závislé na Android SDK. Nalézají se zde všechny aktivity, služby, a další součásti vzhledu aplikace.
3. `cz.zcu.portal.android.util`
Balík pro obecné součásti aplikace, je zde třída pro kódování Base64.



Obrázek 4.30: *Struktura aplikace Rozvrh*

4.1.1 AndroidManifest.xml

Manifest je základem aplikace, jsou zde uvedeny všechny aktivity z balíku `gui` a další podstatné informace o aplikaci. Balík, použitý jako jednoznačný identifikátor aplikace, se jmenuje `cz.zcu.portal.android`. Poslední vydaná verze aplikace má kódové označení 14 a název verze, který se zobrazuje uživatelům je 1.2.0. Dále jsou zde uvedena práva pro přístup k internetu a právo na čtení stavu sítě. Potřeba přístupu na internet je kvůli stažení dat rozvrhu, čtení stavu sítě se používá k oznámení uživateli, pokud by se snažil stáhnout data bez zapnuté sítě.

Minimální verze SDK byla použita verze 4 (Android 1.6), cílovou verzí je verze 10 (Android 2.3.3). Jako cílovou verzi jsem neoznačila poslední verzi 15 (Android 4.0.3), protože se od ostatních verzí značně liší a pro správné zobrazení by byly potřeba větší zásahy do aplikace. Díky tomu, že tato verze není označena jako cílová, provede systém sám potřebné úpravy, aby aplikace v zařízení s vyšší verzí vypadala a chovala se podle očekávání.

Uvedené nastavení vypadá v manifestu následovně.

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="cz.zcu.portal.android"
```

```
    android:versionCode="14"
    android:versionName="1.2.0">

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

<uses-sdk android:minSdkVersion="4" android:targetSdkVersion="10"/>
...

```

Dále se zde nachází i nastavení týkající se widgetu aplikace, více viz 4.4.9. Kompletní manifest je uveden v příloze A.

4.2 Data aplikace

Aplikace pracuje s daty studentů, učitelů, předmětů a rozvrhových akcí. V této kapitole bude popsáno, jak jsou data získávána, ukládána a jak je s nimi manipulováno.

4.2.1 Získání dat

Webové služby

Data rozvrhových akcí, informací o předmětech a učitelích jsou získávána pomocí webových služeb nad IS/STAG [17]. Poskytovány jsou dva standardy webových služeb, SOAP (Simple Object Access Protocol) a REST (Representation State Transfer). Ve své aplikaci využívám služeb podle REST, protože potřebuji získávat jen samotná data. Pomocí URL adresy lze tak získat potřebná data v různých formátech, jako je např. XML nebo CSV.

REST adresa pro získání rozvrhových akcí studenta vypadá následovně: `http://stag-ws.zcu.cz/ws/services/rest/rozvrhy/getRozvrhByStudent?osCislo=X`, kde za `X` dosadíme osobní číslo studenta. Výchozím výstupním formátem je XML, změna formátu na CSV se provede přidáním parametru `outputFormat=csv`. Každá URL adresa je tedy tvořena čtyřmi částmi:

1. `http://stag-ws.zcu.cz/ws/services/` – základní URL webových služeb nad IS/STAG
2. `rest/rozvrhy/` – adresa konkrétní služby
3. `getRozvrhByStudent` – název metody
4. `?osCislo=X&outputFormat=csv` – parametry metody

Ve své aplikaci využívám následující webové služby:

- /rozvrhy/getRozvrhByStudent
- /student/getStudentInfo (vyžaduje přihlášení)
- /student/najdiStudentyPodleJmena (vyžaduje přihlášení)
- /ucitel/getUcitelInfo
- /predmety/getPredmetInfo

Na základě podnětu Úřadu na ochranu osobních údajů byl omezen přístup k rozvrhům studentů nepřihlášeným osobám. Proto určité služby lze volat jen přes zabezpečenou vrstvu HTTPS a klient se musí ověřit standardním mechanismem HTTP BASIC, a to pomocí Orion nebo STAG jména a hesla. Detailní popis procesu přihlášení lze nalézt na [17] v sekci Technické informace.

Formát dat

Jak bylo řečeno výše, webové služby umožňují získávat data v různých formátech. Vzhledem k tomu, že data budou stahována z mobilního telefonu, je nejdůležitějším kritériem velikost stahovaného souboru. Snahou je tedy co nejvíce omezit objem přenesených dat. Z tohoto důvodu jsem vybrala formát CSV, jehož velikost byla na testovaných datech až čtyřikrát menší, než v případě XML. Vzhledem k charakteru přenášených dat (málo strukturovaná data), by použití XML nepřinášelo žádné výhody.

Získání dat v aplikaci

Jak již bylo řečeno, pro získání dat pro potřeby aplikace jsou použity webové služby. Třídy pracující s těmito webovými službami se nacházejí v datové části aplikace v balíku `cz.zcu.portal.android.data.webservice`. Přístup k jednotlivým třídám je zprostředkován pomocí továrny `WebServiceFactory`. Třídy pro webové služby se jmenují podle adresy konkrétní služby, se kterou pracují. V balíku jsou tedy obsažené metody `PredmetWebService`, `RozvrhWebService` apod.

Každá třída pracující s webovou službou musí dědit od abstraktní třídy `AbstractWebService`. Tato třída zprostředkovává veškerou síťovou komunikaci pomocí tříd z balíku `org.apache.http`. Nejprve je nastaveno bezpečné spojení pomocí HTTPS a poté vytvořena instance třídy `HttpGet`, které je předána URL adresa webové služby. Pokud je pro získání dat potřebná autorizace, je nastavena v hlavičce pomocí HTTP BASIC, jak je ukázáno na následujícím výpisu.

```
HttpGet getMethod = new HttpGet(url);
if (autorizace != null) {
    getMethod.setHeader("Authorization", "Basic " + autorizace);
}
```

Pro zakódování přihlašovacích údajů je možné použít třídu `Base64` z balíku `android.util`, ale bohužel až od API úrovně 8 (Android 2.2). Z tohoto důvodu jsem použila volně dostupnou třídu, kterou vytvořil Rober W. Harder [18].

Odpověď serveru je poté kontrolována a podle stavového kódu je buď vypsaná chyba, nebo vrácen výsledný obsah odpovědi, který je dále zpracováván (viz 4.2.2).

Balík `socket` obsahuje třídy a metody pro síťovou komunikaci pomocí vytvoření objektu `javax.net.ssl.SSLSocket`. Tento způsob byl používán v prvotním vývoji aplikace, ale pro svoji nespolehlivost při přenosu byl nahrazen výše uvedeným způsobem. Informace o síťové komunikaci jsem čerpala z [21].

4.2.2 Zpracování dat

Příchozí data o rozvrhových akcích, studentech a učitelích jsou ve formátu CSV, ukázka dat rozvrhových akcí je vidět na obrázku 4.31. Jako oddělovač jednotlivých sloupců je použit středník a data každého sloupce jsou v uvozovkách. Prvním řádkem je hlavička, která udává názvy jednotlivých sloupců. Všechny příchozí údaje nejsou pro potřeby aplikace důležité, z každého řádku jsou získána jen potřebná data a je vytvořen odpovídající objekt. Každému typu dat (učitel, student, akce) tedy odpovídá datová třída a následně tabulka v databázi. Datové třídy se nacházejí v balíku `data.dataobject`.

roakIdno	nazev	katedra	predmet	ucitIdno	ucitIdno.ujmeno.uci	prijmeni.uci	titulPred.	titulZa.uci	platn
202494	Formálníj KIV	FJP	FJP	59289	59289	Richard Lipka	Ing.		A
202627	Přenos da KIV	PD	PD	17911	17911	Martin Šimek	Ing.	Ph.D.	A
203764	Teorie gra KMA	TGD1	TGD1	17655	17655	Zdeněk Ryjáček	Prof. RNDr.	DrSc.	A
203769	Teorie gra KMA	TGD1	TGD1	55184	55184	Jakub Teska	RNDr.	Ph.D.	A
202490	Formálníj KIV	FJP	FJP	17183	17183	Karel Ježek	Doc. Ing.	CSc.	A
202661	Programo KIV	PIA	PIA	224144	224144	Jan Tichava	Ing.		A

Obrázek 4.31: Ukázka CSV souboru s daty rozvrhových akcí

O parsování dat se starají třídy z balíku `data.parse`, kde každá třída odpovídá jednomu typu dat. V prvotní verzi aplikace bylo použito velmi jednoduché parsování, kde se řádek s hlavičkou ignoroval, jednotlivé řádky byly rozděleny pomocí středníků na pole řetězců a jednotlivé údaje byly identifikovány indexem. Bylo tedy napevno určeno, kde se jaký údaj má nacházet, např. index 0 na obrázku 4.31 odpovídal identifikátoru rozvrhové akce (`roakIdno`). Toto řešení sice bylo funkční, ale nešťastně zvolené s ohledem na možnou změnu vstupních dat. Vložení nového sloupce zapříčiní špatnou interpretaci dat, chybějící sloupec dokonce chybu aplikace.

Později jsem zavedla nový způsob parsování, kde je nejprve načten první řádek s hlavičkou, a názvy odpovídající potřebným údajům jsou namapovány

na čísla sloupců (indexy). Pomocí klauzule `switch - case` je pak odpovídající údaj správně interpretován. Aby bylo možné `switch - case` využít pro názvy sloupců, je použit výčtový typ, viz následující ukázka z třídy `RozvrhCSV`.

```
public enum Hlavicka {
    roakIdno, nazev, katedra, predmet, ... , NOVALUE;
    public static Hlavicka toHlavicka(final String str) {
        try {
            return valueOf(str);
        } catch (Exception ex) {
            return NOVALUE;
        }
    }
}

public static List<DORozvrhovaAkce> zpracujData(final BufferedReader bfr) {
    List<DORozvrhovaAkce> listAkci = new ArrayList<DORozvrhovaAkce>();
    Map<Integer, String> mapa = getHeader(radka); // <pozice, hodnota>
    String radka = bfr.readLine();
    while ((radka = bfr.readLine()) != null) {
        String[] hodnoty = radka.split(";");
        DORozvrhovaAkce akce = new DORozvrhovaAkce();
        for (int i = 0; i < hodnoty.length; i++) {
            String hodnota = hodnoty[i];
            String sloupec = mapa.get(i);
            if (sloupec != null) {
                switch (Hlavicka.toHlavicka(sloupec)) {
                    case ucitIdno:
                        ucitel.setUcitIdno(hodnota);
                        break;
                    case jmeno:
                        ucitel.setJmeno(hodnota);
                        break;
                    ...
                }
            }
            listAkci.add(akce);
        }
    }
    return listAkci;
}
```

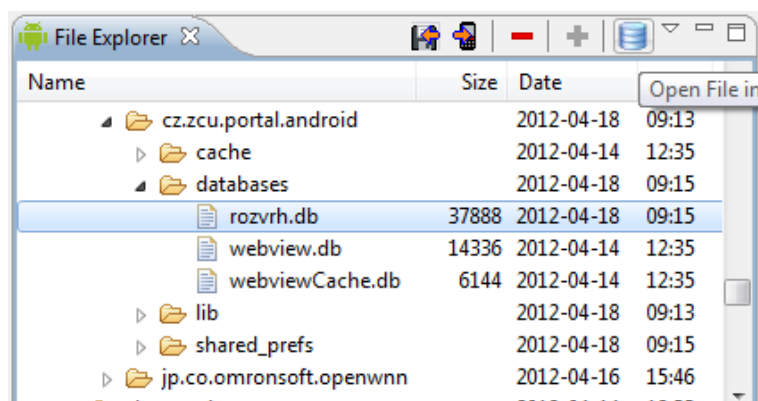
Řešení není tedy závislé na pořadí sloupců a je potřeba jen znát názvy sloupců s daty, která chceme ukládat.

4.2.3 Ukládání a přístup k datům

Veškerá data o studentech, učitelích, předmětech a rozvrhových akcích jsou ukládána v Android databázi SQLite. Data jsou do databáze ukládána, aby zobrazení rozvrhu mohlo probíhat offline a nebylo nutné znovu stahovat stejná

data. Aktualizace dat probíhá vždy jen na vyžádání uživatele, který aktualizaci může spustit v nastavení aplikace (viz dále). Nemůže tedy dojít k samovolnému stahování dat, o kterém by uživatel nebyl informován.

Android SDK poskytuje plnou podporu pro práci s databází SQLite. Přístup k databázi mají všechny třídy aplikace, přičemž databáze není viditelná mimo aplikaci. Pokud bychom chtěli prohlížet obsah databáze přímo v Eclipse, stačí nainstalovat nástroj *CellObject SQLite & XML Browser* [20], který se integruje přímo do DDMS. Kromě prohlížení databáze umožňuje prohlížet i XML soubory s nastavením aplikace (viz dále). Pro zobrazení databáze stačí v DDMS přejít na kartu *File Explorer*, vybrat v emulátoru naši databázi umístěnou v `data/data/cz.zcu.portal.android/databases` a kliknout na tlačítko – viz obrázek 4.32.



Obrázek 4.32: Otevření databáze v *CellObject SQLite Browser*

Poté je zobrazena karta *CellObject SQLite Browser*, kde je vidět jak struktura databáze (název tabulky, schéma), tak její obsah (viz obrázek 4.33). Data jsou přístupná jen ke čtení.

url	katedra	pocetKr...	pozadavky	predmet	id
https://portal.zcu.cz/wps/myportal/predmety/kiv/ppr	KIV	6	Zápočet: Sam...	PPR	1
https://portal.zcu.cz/wps/myportal/predmety/kiv/vsp	KIV	6	Požadavky: Z...	VSP	2
https://portal.zcu.cz/wps/myportal/predmety/kiv/zim	KIV	4	Zápočet - sam...	ZIM	3
https://portal.zcu.cz/wps/myportal/predmety/kiv/op...	KIV	5	Vypracování a ...	OPSWI	4
https://portal.zcu.cz/wps/myportal/predmety/kiv/pds	KIV	6	Zpracování ref...	PDS	5

Obrázek 4.33: Ukázka obsahu databáze – tabulka předmětů

K vytvoření databáze a k její správě slouží třída *SQLiteOpenHelper*, od které oddědíme vlastní třídu a přepíšeme metody `onCreate()`, `onUpgrade()`, `onOpen` atd. K zadávání dotazů slouží metoda `query`, které můžeme předat SQL

příkaz. Pro složitější dotazy můžeme použít `SQLiteQueryBuilder` (viz dále). Každý dotaz vrací objekt `Cursor`, který obsahuje nalezené řádky.

Objektově relační mapování (ORM)

Pro usnadnění práce s daty a databází jsem se rozhodla použít objektově relační mapování. Podmínkou pro výběr knihovny byla její jednoduchost, velikost, celková nenáročnost na výkon systému a podpora `SQLite` databáze. Tyto podmínky splňovala knihovna `ORMLite`, jejímž tvůrcem je Gray Watson [19]. Jak již název vypovídá, knihovna poskytuje jednoduchý způsob pro persistenci Java objektů v `SQL` databázi s důrazem na nízkou režii a složitost.

Pro označení třídy jako persistentní je využíváno anotací. Třída je označena anotací s názvem tabulky, ve které bude objekt uložen a anotován bude i každý parametr třídy, který bude odpovídat sloupci v tabulce. Následuje ukázka ze třídy `DORozvrhovaAkce` obsahující data o jedné rozvrhové akci.

```
@DatabaseTable(tableName = "akceRo")
public class DORozvrhovaAkce implements DOIrozvrhovaAkce {

    @DatabaseField(id = true)
    private String roakIdno;

    @DatabaseField
    private String nazev;

    @DatabaseField
    private String katedra;

    @DatabaseField(dataType = DataType.DATE_LONG)
    private Date datumOd;

    ...
}
```

Odpovídající tabulka se tedy bude jmenovat `akceRo`, identifikátorem bude parametr `roakIdno`, ostatní parametry budou sloupce tabulky. Zde bych ještě chtěla upozornit na ukládání data, kde je nutné nastavit typ na `DATE_LONG`, aby tak byl datum ukládán jako počet milisekund.

Všechny datové objekty z balíku `dataobject` jsou takto anotovány a v databázi tak existují odpovídající tabulky. Protože mezi tabulkou `akceRo` a tabulkou studentů `student` je vazba M:N, bylo nutné vytvořit propojující tabulku `akceStudent`. Ta obsahuje ID rozvrhové akce a studenta jako cizí klíče, primární klíč je automaticky generován. Ukázka takové anotace:

```
@DatabaseTable(tableName = "akceStudent")
public class DORozvrhovaAkceStudent implements DOIrozvrhovaAkceStudent {

    @DatabaseField(generatedId = true)
```

```

private int id;

@DatabaseField(foreign = true, columnName = RA_ID_FIELD_NAME)
private DORozvrhovaAkce ra;

@DatabaseField(foreign = true, columnName = STUDENT_ID_FIELD_NAME)
private DOSTudent student;
...
}

```

OrmLiteSqliteOpenHelper

Ke správě databáze je nutné vytvořit vlastní třídu, která bude dědit od třídy `OrmLiteSqliteOpenHelper`. Tato třída umožňuje vytvářet a měnit databázi a také poskytuje *Data Access Object* (DAO) třídy. Metody, které je nutné implementovat, jsou `onCreate(SQLiteDatabase db, ConnectionSource cs)` a `onUpgrade(SQLiteDatabase db, ConnectionSource cs, int old, int new)`. Metoda `onCreate()` vytváří databázi, když je aplikace poprvé nainstalována a metoda `onUpgrade()` se stará o úpravy v databázi (např. přidání sloupce), které se provedou při změně verze aplikace.

Tato třída obsahuje i metody pro získání DAO tříd. Ty se získávají pomocí statické metody `createDao()` ORMLite třídy `DaoManager`, které se jako parametr předá třída, pro kterou se má DAO vytvořit. DAO poskytují základní CRUD operace a nástroj pro vytváření dotazů (`QueryBuilder`). Ukázka metody pro získání DAO ve třídě `DatabaseHelper` následuje.

```

public Dao<DORozvrhovaAkce, String> getRozvrhovaAkceDao() throws
    SQLException {
    return DaoManager.createDao(getConnectionSource(), DORozvrhovaAkce.class);
}

```

Vlastní DAO třídy

Protože základní operace, které poskytují DAO třídy nejsou dostačující, vytvořila jsem vlastní obalující DAO třídy, které obsahují veškeré potřebné dotazy a příkazy do databáze. Třídy poskytované třídou `DatabaseHelper` označuji jako *Dao*, vlastní obalující třídy jako *DAO*.

Veškerý přístup k objektům z databáze tak probíhá přes tyto DAO třídy. Ty jsou uloženy v balíku `data.dao` a dědí od třídy `MainDAO`. Tato třída poskytuje metody k získání kontextu aplikace a k přístupu ke třídě `DatabaseHelper` pomocí `OpenHelperManager`, tak jak je ukázáno na výpisu.

```

DatabaseHelper databaseHelper = OpenHelperManager.getHelper(context,
    DatabaseHelper.class);

```

Pro vytváření dotazů do databáze využívám v DAO třídách `QueryBuilder`, který umožňuje tvorbu dotazu postupným zřetězováním metod. Na následujícím

výpisu je ukázáno jeho použití při získávání objektu `DOPredmet` na základě katedry a zkratky předmětu. Dotaz vrací seznam nalezených předmětů, který v tomto případě musí být jednoprvkový.

```
// ziskani query builderu z Dao
QueryBuilder<DOPredmet, String> queryBuilder = getPredmetDao().queryBuilder
    ();
// katedra a zkratka predmetu musi odpovidat parametru
queryBuilder.where().eq(DOIPredmet.KATEDRA_FIELD_NAME, katedra).and().eq(
    DOIPredmet.PREDMET_FIELD_NAME, predmetZkr);
// priprava sql prikazu
PreparedQuery<DOPredmet> preparedQuery = queryBuilder.prepare();
// dotaz na vsechny predmety odpovidajici podmince
List<DOPredmet> predmety = getPredmetDao().query(preparedQuery);
```

4.2.4 Servisní vrstva

Mezi vrstvou, která přímo přistupuje do databáze (DAO třídy), a vrstvou pro zobrazení dat (aktivity) jsem vytvořila mezivrstvu, kterou tvoří servisní třídy z balíku `data.support`. Metody těchto tříd jsou volány z aktivit a služeb aplikace a provádějí různé kontroly dat a volání do samotné datové vrstvy.

Názvy tříd jsou opět odvozeny od objektů, se kterými pracují. Nacházejí se zde proto třídy `StudentiSupport`, `RozvrhoveAkceSupport` apod. Konkrétní zajímavé funkce budou popsány v rámci popisu funkcí aplikace (viz kapitola 4.4).

4.3 Vzhled aplikace

Třídy, které souvisí se vzhledem aplikace a zobrazením dat, se nacházejí v balíku `cz.zcu.portal.android.gui`. Jedná se hlavně o aktivity, tvořící uživatelské rozhraní aplikace. Všechny hlavní aktivity se nacházejí v podbalíku `main`, dílčí aktivity pro zobrazení detailů předmětů, studentů a učitelů jsou v balíku `details`. V balíku `utils` se nacházejí pomocné třídy pro vzhled dialogu, popisu změn, stahování dat apod. Posledním balíkem je balík `widget`, který obsahuje třídy pro tvorbu a aktualizaci widgetu (viz 4.4.9).

Součástí jednotlivých obrazovek aplikace jsou popsány jak pomocí XML (umístěné v `res/layout`), tak i přímo ve zdrojovém kódu. Popis pomocí XML jsem použila pro části statické (např. obrazovka pro hledání rozvrhu podle osobního čísla) a dílčí (např. řádek v detailu předmětu). Vytvoření prvků uživatelského rozhraní pomocí zdrojového kódu bylo použito u dynamických částí, jako jsou jednotlivé rozvrhové akce tvořící rozvrh.

V aplikaci jsem věnovala zvýšenou pozornost tomu, aby byla dobře použitelná jak v zobrazení na výšku, tak na šířku. Nyní budou popsány základní prvky uživatelského rozhraní.

4.3.1 Rozvrh

Rozvrh studenta obsahuje následující části:

- titulek s osobním číslem studenta – obsahuje případně i jeho jméno a příjmení, pokud uživatel zadal v nastavení aplikace přihlašovací údaje
- hlavičku s číslem hodiny a časem
- řádku obsahující:
 - zkratku označující den
 - rozvrhové akce v daný den a čas
- popis jaký je aktuální týden (sudý nebo lichý)
- seznam předmětů bez uvedeného času

Všechny popsané části jsou vidět na obrázku 4.34.



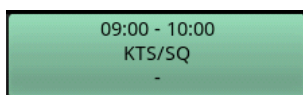
Obrázek 4.34: Ukázka zobrazení rozvrhu v aplikaci

Barvy jednotlivých rozvrhových akcí jsou nastaveny stejně jako na Portále ZČU, šedá tedy zobrazuje přednášku, tmavší zelená cvičení a světle zelená seminář. U akcí, které jsou jen sudý či lichý týden je tato skutečnost označena

v pravém dolním rohu (obr. 4.35). U akcí, které nejsou podle přesně stanoveného času daného časovou řadou ZČU (typicky tělocviky), je čas uveden v horní části prvku (obr. 4.36).



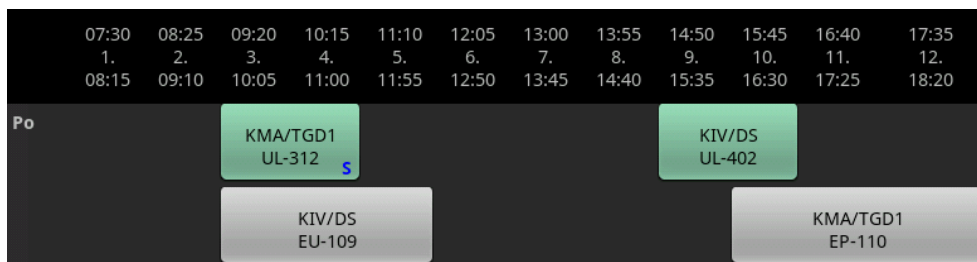
Obrázek 4.35: Ukázka předmětu v sudém týdnu



Obrázek 4.36: Ukázka předmětu mimo časovou řadu

O zobrazení rozvrhu se stará aktivita `RozvrhShow`. Odpovídajícím hlavním layoutem je `rozvrh.xml`, kde jednotlivé jeho části jsou nejprve obaleny `ScrollView` pro možnost posouvání zobrazeného rozvrhu, poté jsou obaleny `android.gesture.GestureOverlayView` pro zachytávání gest (více viz 4.4.8). Základem layoutu je `TableLayout`, který je prázdný a jednotlivé řádky jsou přidávány dynamicky v kódu. Dále zde jsou `TextView` pro zobrazení typu týdne a textu, pokud v celém rozvrhu nejsou žádné rozvrhové akce.

Řádky (`TableRow`) obsahují `TextView` se zkratkou dne a případně rozvrhové akce na určené pozici. Rozvrhová akce je tvořena tlačítkem (objekt `Button`) s textem tvořícím jeho popis. Pokud dojde k překrývání akcí, je nalezen nevhodnější volný řádek, pokud volný řádek není, je vytvořen nový se stejnou barvou pozadí a bez označení dne (obr. 4.37).



Obrázek 4.37: Ukázka překrývajících se rozvrhových akcí

Ostatní dílčí layouty pro zobrazení rozvrhu jsou pro odlišení označeny `rozvrh_`, např. `rozvrh_hlavicka.xml`.

4.3.2 Předmět, učitel a student

Detail o konkrétní rozvrhové akci i o samotném předmětu lze vyvolat kliknutím na tlačítko s danou rozvrhovou akcí v rozvrhu. Detail obsahuje název předmětu, status, místnost, semestr atd (obr. 4.38).



Předmět: KIV/PPR	
Předmět	Paralelní programo
Statut	A
Místnost	UU-407
Semestr	Zimní
Druh	Přednáška
Týden	Jiný
Den	Úterý
Začátek	13:55
Konec	17:25
Datum od	20.9.2011
Datum do	13.12.2011
Vyučující	Tomáš Koutný
Courseware *	https://portal.zcu.c
* funguje jen u předmětů, které mají stránku	
Kredity	6
Požadavky	Zápočet: Samostat (bodové hodnocen třeba 25 bodů. Mez stránky v předmět

Obrázek 4.38: Ukázka detailu předmětu

Údaje obsahují i odkaz na předmět na Courseware, kdy kliknutím na tento odkaz je spuštěn webový prohlížeč zařízení s uvedenou URL adresou. Dalším klikatelným údajem je jméno vyučujícího, které po kliknutí zobrazí detail učitele (obr. 4.39). Zde je důležitým údajem emailová adresa, která po kliknutí umožní poslat email na tuto adresu z námi zvolené aplikace poštovního klienta na zařízení.

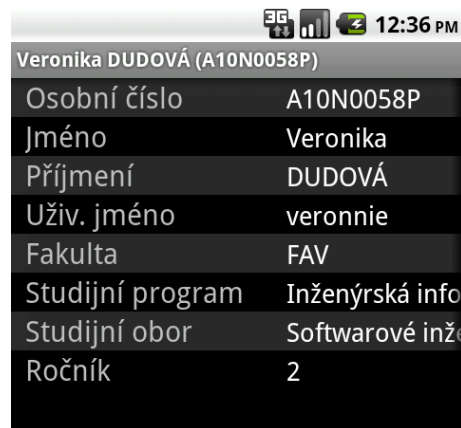


Tomáš Koutný	
Jméno	Tomáš
Příjmení	Koutný
Katedra	KIV
Email	txkoutny@kiv.zcu.cz

Obrázek 4.39: Ukázka detailu vyučujícího

Detail studenta je možno zobrazit přes menu aplikace (viz kapitola 4.3.3). Co je obsahem této obrazovky je vidět na následujícím obrázku 4.40.

Pro zobrazení těchto obrazovek slouží aktivity z balíku `gui.details`, konkrétně `PredmetShow`, `UcitelShow` a `StudentShow`. Základním layoutem pro zobrazení je `detaily.xml`, který obsahuje `ScrollView` pro možnost posuvu



Veronika DUDOVÁ (A10N0058P)	
Osobní číslo	A10N0058P
Jméno	Veronika
Příjmení	DUDOVÁ
Uživ. jméno	veronnie
Fakulta	FAV
Studijní program	Inženýrská info
Studijní obor	Softwarové inž
Ročník	2

Obrázek 4.40: Ukázka detailu studenta

a prázdný `TableLayout` připravený k naplnění daty. Naplnění probíhá opět v aktivitách, protože se jedná o dynamická data načtená z databáze. Pro vytváření řádků tabulky jsou používány metody vlastní třídy `DetailCreator`, které jsou volány z aktivit. Dále jsou používány dílčí layouty, které jsou pro přehlednost označeny `detail_`. Jedná se např. o layout `detail_email.xml`, který slouží pro zobrazení emailu. Pro označení, že se jedná o email a chceme provést příslušné akce po kliknutí na něj, se používá atribut `android:autoLink`. Ukázka viz následující zápis.

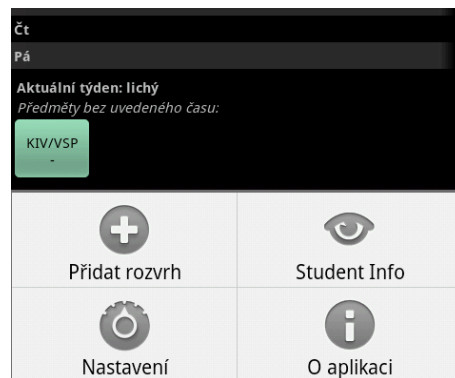
```
<TextView
    android:id="@+id/emailHodnota"
    android:textSize="18sp"
    android:clickable="true"
    android:autoLink="email"
/>
```

Pokud bychom chtěli, aby se po kliknutí na text spustil webový prohlížeč, použili bychom zápis `android:autoLink="web"`.

4.3.3 Menu

Dalším výrazným prvkem uživatelského rozhraní je menu. Menu se spustí na obrazovce s rozvrhem po kliknutí na hardwarové tlačítko přístroje pro to určené. Menu obsahuje položku pro přidání rozvrhu, pro nastavení aplikace a pro zobrazení informace o aplikaci. Pokud má uživatel vyplněny přihlašovací údaje v nastavení aplikace, pak se zobrazí i položka pro detail studenta (obr. 4.41).

Vzhled menu je určen layoutem `menu/rozvrh_menu.xml`, který je nastaven v metodě `onOptionsItemSelected()` ve třídě `RozvrhShow`. V metodě pro přípravu menu (`onPrepareOptionsMenu()`) je zkontrolováno, zda existují údaje uživatele a pokud ano, tak je zobrazena příslušná položka v menu. Metoda

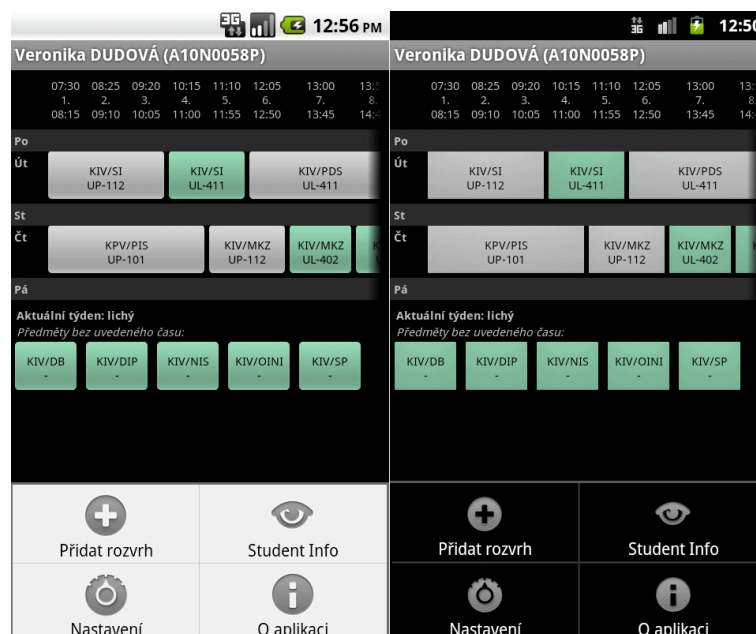


Obrázek 4.41: Menu aplikace

onOptionsItemSelected() poté slouží k odchytávání kliknutí na položku menu a ke spuštění příslušných aktivit.

4.3.4 Vzhled podle verze

Samotný vzhled aplikace je také ovlivněn použitou verzí Androidu. Všechny použité obrázky aplikace jsou pro verzi emulátoru Android 2.2. Pro verzi emulátoru Android 2.3.1 (API 9), již vypadá aplikace trochu jinak. Je to z toho důvodu, že jsou použity základní GUI prvky a záleží na jejich implementaci v daném systému. Porovnání obrazovky s rozvrhem a se zobrazeným menu na Android 2.2 a Android 2.3.1 viz obrázek 4.42.



Obrázek 4.42: Porovnání vzhledu aplikace, vlevo Android 2.2, vpravo Android 2.3.1

U fyzických zařízení s Android 2.3.x, které mají vlastní grafickou nadstavbu (všechny kromě Nexus řady) ale aplikace vypadá jako pro Android 2.2. Pokud bych chtěla přesně určit, jak má aplikace vypadat, musela bych layouty a ikony uložit do složek podle verze, např. složka `drawable-hdpi-v9` by obsahovalo ikony, které se mají použít od API verze 9. Já ale nechávám vzhled na samotném systému, aby aplikace dodržovala „look & feel“ dané grafické nadstavby a tak i ostatních aplikací.

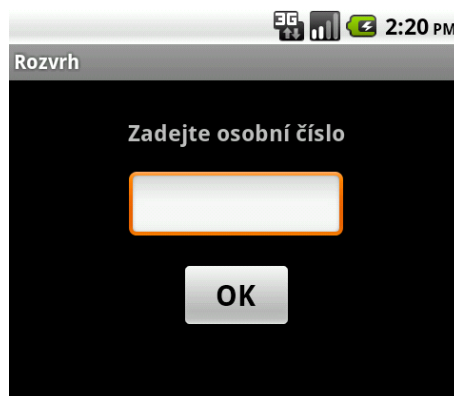
4.4 Funkce aplikace

V této kapitole budou popsány jednotlivé funkce aplikace.

4.4.1 Přidání rozvrhu

Základní funkcí aplikace je možnost přidávat rozvrhy studentů ZČU. Pokud spustíme aplikaci, je volána aktivita `Hlavni`, která zkontroluje, zda je již v aplikaci nějaké osobní číslo nastaveno (viz 4.4.6). Pokud ano, pak spustí aktivitu pro zobrazení rozvrhu daného osobního čísla (viz 4.4.3), pokud ne, tak je spuštěna aktivita `OsGet`, která se stará právě o přidání rozvrhu. Tuto aktivitu je možno spustit i kliknutím na položku menu *Přidat rozvrh*.

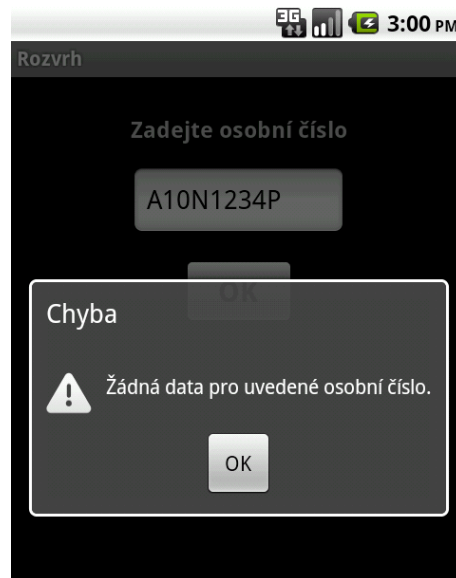
Obrazovka aktivity je velmi jednoduchá, obsahuje jen pole pro zadání osobního čísla studenta, jehož rozvrh chceme stáhnout, a potvrzovací tlačítko (obr. 4.43).



Obrázek 4.43: *Obrazovka pro zadání osobního čísla*

Po kliknutí na potvrzovací tlačítko je provedena kontrola, zda pole není prázdné. Pokud ano, je zobrazen chybový dialog, jinak je z balíku `gui.util` zavolána třída `DataDownloader`. Tato třída slouží ke stahování potřebných dat v samostatném vlákně, kde v hlavním vlákně běží dialog oznamující, že jsou data načítána. Třída `DataDownloader` obsahuje objekt `Handler`, který slouží

k posílání zpráv. V tomto případě je objektu poslána z vlákna zpráva o výsledku stahování. `Handler` nejprve zruší běžící dialog a poté podle čísla zprávy rozezná, zda bylo stahování v pořádku, či zda došlo k chybě. V případě chyby je zobrazen vlastní chybový dialog – `ErrorDialog` (obr. 4.44). V případě úspěšného stažení dat je poslána zpráva handleru z volající aktivity. O používání vláken v Android jsem čerpala z [22].



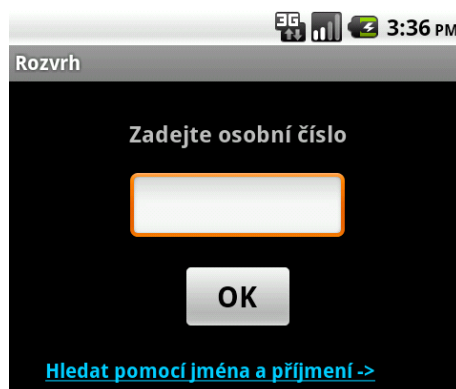
Obrázek 4.44: Jeden z možných chybových dialogů při získávání dat studenta

Stahování dat spočívá ve volání metody ze servisní vrstvy, která využívá metody tříd pracujících s webovými službami. Jedná se o metodu `saveData(String)` ze třídy `RozvrhSupport`. V metodě je nejprve zavolána služba pro získání rozvrhu (`getRozvrhByStudent`), která vrací seznam rozvrhových akcí jako objektů `DORozvrhovaAkce`. Poté je pro každý předmět (unikátní kombinace katedry a názvu rozvrhové akce) volána služba pro získání detailu předmětů (`getPredmetInfo`) a jsou tak vytvořeny objekty `DOPredmet`. Na základě rozvrhových akcí je poté pro každé id učitele volána služba `getUcitelInfo` a vytvořeny objekty `DOUcitel`. Pokud má uživatel správně vyplněny přihlašovací údaje, jsou staženy i detailní informace o studentovi (`getStudentInfo`) a vytvořen objekt `DOStudent`. Vytvořené objekty jsou pak uloženy do databáze.

Po dokončení stahování dat je v popředí opět aktivita `OsGet`. Ta po úspěšném stažení dat spustí aktivitu `RozvrhShow`, která zobrazí stažený rozvrh. Aktivitě musí být předáno osobní číslo studenta jako parametr, podle kterého bude moci volat pomocné třídy pro získávání dat o rozvrhu studenta z databáze.

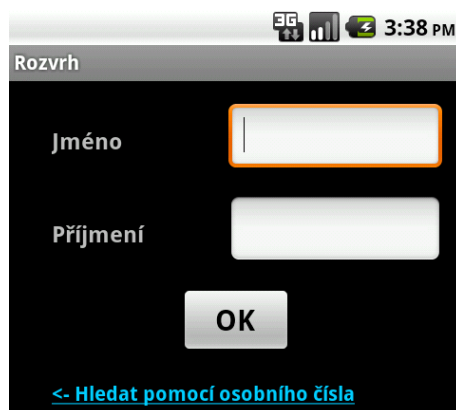
4.4.2 Vyhledání rozvrhu podle jména studenta

Pokud má uživatel v nastavení vyplněny Orion přihlašovací údaje, může vyhledávat další studenty podle jména a příjmení. Při přidávání rozvrhu je nabídnuta možnost využít této funkce (obr. 4.45).



Obrázek 4.45: *Obrazovka pro zadání osobního čísla – uživatel s přihlašovacími údaji*

Po kliknutí na text *Hledat pomocí jména a příjmení* je zobrazena obrazovka pro zadání jména a příjmení (obr. 4.46). Vyhledávat je možné i podle samotného jména či příjmení a pro hledání platí stejná pravidla jako při použití stejné funkce na Portále ZČU.



Obrázek 4.46: *Obrazovka pro hledání studentů podle jména*

O vytvoření obrazovky se stará aktivita `OsGet`, stejně jako v případě přidávání rozvrhu podle osobního čísla. Po kliknutí na potvrzovací tlačítko je volána aktivita `VysledkyHledani` a té je předáno jméno a příjmení, které uživatel zadal.

Aktivita volá třídu `DataDownloader`, která tentokrát slouží pro získání údajů o nalezených studentech voláním metody `getStudentsByNameFromNet(String prijmeni, String jmeno)` ze třídy `RozvrhSupport`. Opět je volána metoda

pracující s webovými službami, nyní se jedná o službu `najdiStudentyPodleJmena`. Vrácen je seznam nalezených studentů naplněný objekty `D0Student` a aktivita je o výsledku stahování informována pomocí handleru.

Nalezená data studentů jsou zobrazena pomocí `ExpandableListView`, který umožňuje vytvářet seznamy umožňující na rozkliknutí zobrazit další informace. Z dat studentů je vytvořen seznam, kde je vidět jméno, příjmení a osobní číslo a další údaje na rozkliknutí. Výsledek hledání pak může vypadat např. jako je na obrázku 4.47.



Obrázek 4.47: *Obrazovka s výsledkem hledání studentek s jménem Veronika a příjmením začínajícím na D*

Kliknutím na tlačítko se symbolem plus (+) dojde ke stažení dat o rozvrhu, stejně jako v případě zadání osobního čísla, jen s tím rozdílem, že údaje o studentovi již není nutné znovu stahovat. Po úspěšném stažení a uložení dat je opět spuštěna aktivita zobrazující rozvrh.

4.4.3 Zobrazení rozvrhu

O zobrazení a práci s rozvrhem se stará aktivita `RozvrhShow`, které je v bundlu předáno osobní číslo a semestr, pro který chceme data zobrazit. Pokud semestr nebyl předán (první zobrazení rozvrhu), je nastaven podle aktuálního měsíce v roce. Informace o osobním čísle a semestru je uložena do sdílených předvoleb

(objekt `SharedPreferences`), které jsou jakýmsi nastavením aplikace, které zůstává zachováno i po vypnutí aplikace a lze k nim kdykoliv přistupovat.

Data pro zobrazení rozvrhu jsou získána z metody `getRozvrhAkceVyplnene()` ze třídy `RozvrhoveAkceSupport`, které se jako parametr předá osobní číslo a semestr. V databázi je nalezen odpovídající seznam rozvrhových akcí, seřadí se podle dne a času a doplní se prázdné údaje čísla hodiny začátku a konce akce. Jako příklad lze uvést tělocviky, které časově neodpovídají používané časové řadě a nemají uvedena čísla hodin. U takovýchto akcí je proveden přepočít, aby se akce v časové ose rozvrhu co nejlépe zobrazovaly.

Dále je také zjištěno nejnižší a nejvyšší číslo hodiny začátku akce v celém týdnu, aby nebyly zbytečně zobrazeny prázdné sloupce a velikost displeje tak byla co nejlépe využita. Pokud tedy student bude nejdříve začínat až např. šestou vyučovací hodinou, nebudou mu předchozí hodiny zobrazeny ani v hlavičce rozvrhu, obdobně s nejpozdější hodinou.

Pro každý den v týdnu jsou pak ze seznamu vybrány odpovídající akce a ty zobrazeny v řádce jako tlačítka. Pokud dojde ke kolizi dvou či více akcí, je tato situace vyřešena přidáním nového řádku nebo nalezením volné pozice v již existujícím řádku. Každému tlačítku tvořícímu rozvrhovou akci je nastaveno ID dané akce. Toto ID je poté použito při kliknutí na akci, kde je ID předáno aktivitě pro zobrazení detailu předmětu (`PredmetShow`).

4.4.4 Zobrazení rozvrhu – kombinované studium

Trochu komplikovanější situace nastává u zobrazení rozvrhů pro kombinované studium. Takové rozvrhy obvykle vypadají tak, že jedna rozvrhová akce je jen v jeden konkrétní datum a čas a neopakuje se. U takovýchto akcí je potřeba zobrazit kromě standardních informací i datum konání. Jak vypadá takový rozvrh v aplikaci je vidět na obrázku 4.48.

Pro lepší přehlednost a větší komfort uživatele, jsem ještě vytvořila možnost nechat si zobrazit jen akce v aktuálním týdnu (obr. 4.49). Tyto akce jsou vráceny voláním metody `getRozvrhAkceVyplneneTyden()`, která pro nalezení akcí využívá dotazu do databáze na akce, které mají datum platnosti v určitém časovém rozmezí.

4.4.5 Detail předmětů, učitelů a studentů

Jak vypadají obrazovky pro zobrazení uvedených detailů bylo popsáno v 4.3.2. Zde bych nyní popsala, jakým způsobem jsou získávána data pro zobrazení. Každá obrazovka s detailem je složená ze dvou částí, z popisku a z hodnoty. Popisky

jsou uloženy v `values/arrays.xml` jako pole řetězců. Pro popisky jsem vytvořila i anglickou verzi, která je uložena v `values-en/arrays.xml`. Ukázka pole popisků z `arrays.xml` následuje.

```
<string-array name="STpopisky">
  <item>Osobní číslo</item>
  <item>Jméno</item>
  <item>Příjmení</item>
  <item>Uživ. jméno</item>
  <item>Fakulta</item>
  <item>Studijní program</item>
  <item>Studijní obor</item>
  <item>Ročník</item>
</string-array>
```

Tyto údaje jsou poté ve třídě `StudentShow` načteny následujícím způsobem.

```
String[] popisky = getResources().getStringArray(R.array.STpopisky);
```

Hodnoty objektu pro zobrazení (např. `DOStudent`), jsou získány voláním metody `toArray()` na daný objekt. Získané popisky i hodnoty jsou předány metodě `createDetails()` třídy `DetailsCreator`, která vytvoří příslušné řádky s dvojicí popisek – hodnota. Speciální řádky, jako klikací jméno učitele, který po kliknutí spustí příslušnou aktivitu s detailem, nebo řádek s emailem, jsou vytvářeny voláním dalších metod k tomu určených.

4.4.6 Nastavení aplikace

Na obrazovku s nastavením aplikace se dostaneme kliknutím na příslušnou položku v menu. V nastavení je možné změnit osobní číslo, pro které je zobrazen rozvrh, semestr a jazyk. Dále je zde možnost nastavit Orion nebo STAG přihlašovací údaje a také jsou zde akce pro aktualizaci nebo smazání vybraných rozvrhů. Obrazovku nastavení aplikace můžete vidět na obrázku 4.50.

O nastavení se stará aktivita `Nastaveni`, která dědí od `PreferenceActivity`. Díky tomu má aktivita přístup k vlastnímu nastavení aplikace, které se ukládá do XML do adresáře, který má aplikace k dispozici. Vytvoření obrazovky s nastavením spočívá ve vytvoření `PreferenceScreen`, která obsahuje různé předvolby jako např. `ListPreference` pro zobrazení seznamu pro výběr hodnoty, nebo `EditTextPreference` pro zadání textu. Tyto předvolby mohou být ještě rozděleny do kategorií `PreferenceCategory`. Jakmile uživatel změní některou z předvoleb, dojde k automatickému uložení hodnoty.



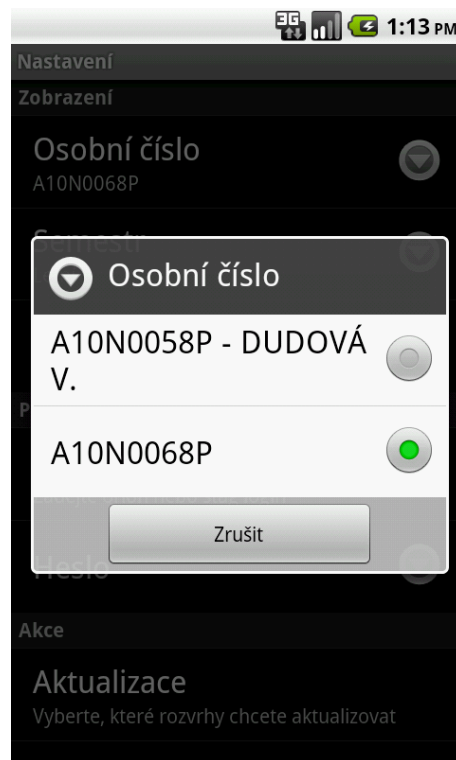
Obrázek 4.50: Obrazovka s nastavením aplikace

Jak vypadá nastavení `ListPreference` pro výběr osobního čísla je vidět na následujícím výpisu. Parametr označený jako *entries* je pole hodnot, které má být zobrazeno jako název, *entry value* označuje odpovídající hodnoty.

```
ListPreference vyberOs = new ListPreference(this);
vyberOs.setEntries(osobniCislaAJmena);
vyberOs.setEntryValues(osobniCisla);
vyberOs.setDialogTitle(R.string.nast_os);
vyberOs.setKey(getResources().getString(R.string.nast_os_key));
vyberOs.setTitle(R.string.nast_os);
vyberOs.setSummary(os);
```

V tomto případě pro zobrazení vkládám osobní číslo i s jménem uživatele (pokud je uloženo) a hodnotou je pouze osobní číslo. K takto uloženému osobnímu číslu pak mohu přistupovat kdekoliv v aplikaci přes uvedený klíč. Osobní číslo je takto načteno i při startu aplikace, aby došlo k zobrazení příslušného rozvrhu. Jak vypadá dialog pro zadání osobního čísla je vidět na obrázku 4.51.

Obdobně probíhá i nastavení `EditTextPreference` pro zadání přihlašovacích údajů. V případě hesla je z důvodu bezpečnosti nastaveno, aby se nezobrazoval přímo zadávaný text. Toto nastavení se provede následujícím způsobem, kde `prihlaseniHeslo` je příslušná `EditTextPreference`.



Obrázek 4.51: Nastavení osobního čísla

```
EditText myEditText = přihlasiHeslo.getEditText();
myEditText.setTransformationMethod(PasswordTransformationMethod.getInstance());
```

V aktivitě `Nastaveni` je metoda `onSharedPreferencesChanged()`, která je automaticky zavolána při změně jakékoliv předvolby. Pokud došlo ke změně osobního čísla nebo semestru je aktivita s nastavením ukončena a je zobrazen příslušný rozvrh. Při změně přihlašovacího jména či hesla je vytvořen autorizační řetězec pomocí Base64 a ten je uložen do vlastního nastavení aplikace. Chování v případě změny jazyka je popsáno dále.

Akce

Další možností, která není nastavením v pravém slova smyslu, je možnost aktualizovat či mazat rozvrhy. Po kliknutí na příslušnou předvolbu je zobrazen dialog se seznamem osobních čísel s možností zaškrtnout ty, které chceme aktualizovat (smazat) (viz obr. 4.52). Kliknutím na tlačítko OK dojde k příslušné akci.

Tato funkcionality je řešena pomocí vlastní třídy `CheckboxDialogMaker`, která vytvoří příslušný dialog. Po kliknutí na předvolbu je vytvořena instance této třídy, jsou jí předána osobní čísla a je zavolána příslušná metoda pro vytvoření a obsluhu dialogu, např. `createDeleteDialog()`. Metoda vytvoří po-



Obrázek 4.52: Dialog pro aktualizaci rozvrhů

ložky seznamu a určí akce, které se mají provést po kliknutí na tlačítka. Při aktualizaci je volán opět `DataDownloader` a dojde ke stažení a aktualizaci dat, při mazání je volána metoda ze servisní vrstvy, která se postará o smazání příslušných záznamů. Dále je ošetřeno, aby nemohlo dojít ke smazání aktuálně zobrazeného rozvrhu.

O úspěšném provedení akce je uživatel informován přes notifikaci `Toast`. V případě mazání je nutné znovu spustit aktivitu `Nastaveni`, aby došlo k znovunačtení osobních čísel a ta smazaná se tak již nezobrazovala.

Změna jazyka

Výchozím jazykem aplikace je čeština. Lokalizaci do anglického jazyka jsem provedla vytvořením příslušných anglických zdrojů s texty a poli (`string.xml` a `arrays.xml`), které jsem uložila do složky `values-en`. Jazyk aplikace je pak určen jazykem přístroje. Aby bylo možno přepínat jazyk v aplikaci bez závislosti na lokalizaci zařízení, bylo potřeba provést v aplikaci několik kontrol a nastavení.

V metodě `onSharedPreferencesChanged()` je po změně jazyka v nastavení aplikace zavolána statická metoda `updateLanguage()` třídy `RozvrhAplikace`. Tato třída dědí od `Application` a slouží tak ke správě stavu celé aplikace. V tomto případě je použita právě pro zajištění správného fungování při změně jazyka. Metoda `updateLanguage` zjistí jazyk, který je uložen v nastavení a porovná ho s jazykem konfigurace aplikace. Pokud došlo ke změně, nastaví nové `Locale` a aktualizuje konfiguraci, viz následující výpis.

```
Configuration config = ctx.getResources().getConfiguration();
String lang = PreferenceManager.getDefaultSharedPreferences(ctx).getString(
    ctx.getResources().getString(R.string.nast_locale_key), "cs");

// pokud doslo ke zmene locale musim nastavit na nove
if (!config.locale.getLanguage().equals(lang)) {
    Locale locale = new Locale(lang);
    Locale.setDefault(locale);
    config.locale = locale;
}
```

```
ctx.getResources().updateConfiguration(config, ctx.getResources().
    getDisplayMetrics());
}
```

Nastavení jazyka aplikace vypadá jednoduše, ale problém nastává při změně konfigurace, což je např. i změna orientace displeje. V tu chvíli je jazyk aplikace nastaven na jazyk přístroje. Z tohoto důvodu musí být ve třídě `RozvrhAplikace` metoda zachytávající změnu konfigurace, tedy `onConfigurationChanged()`, ve které bude opět zavolána metoda `updateLanguage()`. Pokud tedy má uživatel přístroj nastaven v anglickém jazyce a aplikaci chce používat v českém, musí být při každé změně jazyk přenastaven.

Data rozvrhových akcí jsou stažena v nastaveném jazyce. Pokud by uživatel změnil nastavený jazyk a chtěl by v tomto jazyce i data akcí, musel by rozvrh aktualizovat.

Dalším problémem s nastavením jazyka spočívá v tom, že po jeho změně přes nastavení zůstávají aktivity v původním jazyce. Změna se projeví až při znovuspuštění dané aktivity. Tento problém jsem vyřešila restartováním aplikace, které je provedeno tak, aby si jej uživatel ani nevšiml. Restart aplikace spočívá ve spuštění intentu (viz 3.2.3) hlavní aktivity a nastavení správného příznaku. Jakým způsobem je aplikace restartována, je vidět na následujícím výpisu.

```
Intent i = getPackageManager().getLaunchIntentForPackage(getPackageName());
i.addFlags(Intent.FLAG_ACTIVITY_CLEAR_TOP);
startActivity(i);
```

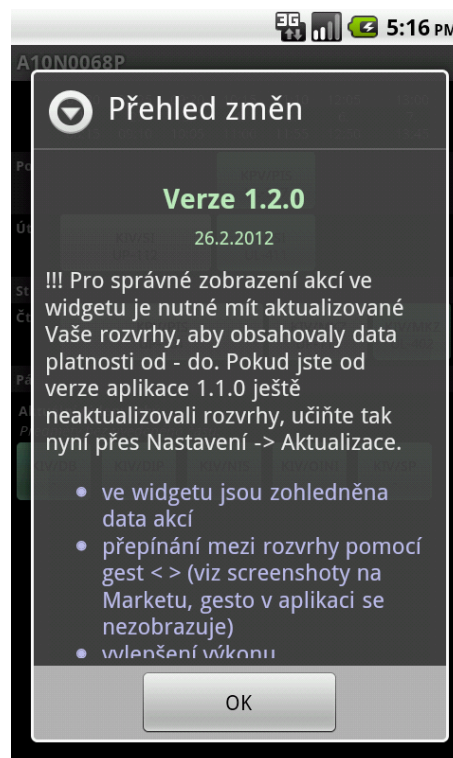
4.4.7 Přehled změn

Další funkcí, kterou jsem do aplikace zavedla, je zobrazení přehledu změn po spuštění nové verze aplikace. Uživatel tak okamžitě vidí, k jakým změnám v aplikaci došlo. Pro zobrazení změn je používána třída `ChangeLog` z balíku `gui.utils`, kterou vytvořil Karsen Priegnitz [23]. Třída poskytuje metody na zjištění prvního běhu aplikace v nové verzi i prvního běhu aplikace vůbec a umožňuje zobrazit příslušný seznam změn v přehledném dialogu.

Seznam změn je načítán ze souboru `res/raw/changelog.xml`, v případě anglické lokalizace se jedná o adresář `raw-en`. Textový soubor je vlastně jakési HTML, kde jsou uvedeny styly pro zobrazení a v samotném textu jsou použity speciální symboly pro odlišení typu textu. Tyto symboly jsou pak při zpracování třídou `ChangeLog` nahrazeny `div` tagy s příslušnou třídou pro odlišení stylu. Díky tomuto způsobu je psaní přehledu změn velmi jednoduché a rychlé.

Na následujícím kódu je vidět ukázka zápisu ze souboru `changelog.txt` a na obrázku 4.53 je vidět přehled změn přímo v aplikaci.

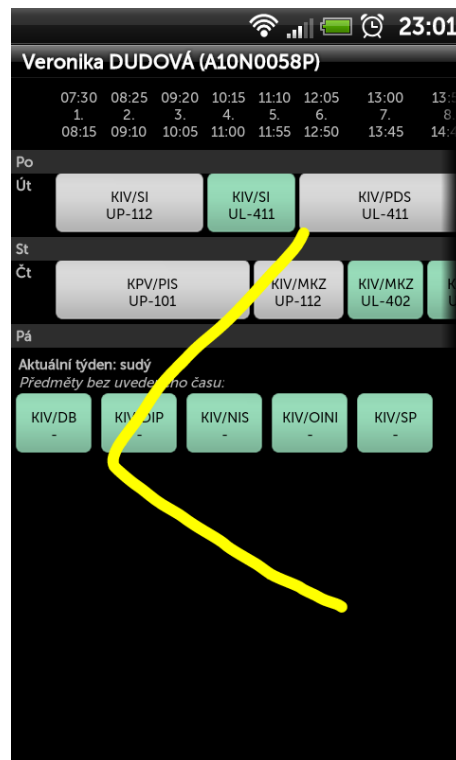
```
1.2.0
% Verze 1.2.0
_ 26.2.2012
! !!! Pro správné zobrazení akcí ve widgetu je nutné mít aktualizované Vaše
    rozvrhy, aby obsahovaly data platnosti od - do. Pokud jste od verze
    aplikace 1.1.0 ještě neaktualizovali rozvrhy, učiňte tak nyní přes
    Nastavení -> Aktualizace.
* ve widgetu jsou zohledněna data akcí
* přepínání mezi rozvrhy pomocí gest < > (viz screenshoty na Marketu,
    gesto v aplikaci se nezobrazuje)
* vylepšení výkonu
```



Obrázek 4.53: Ukázka přehledu změn po spuštění nové verze aplikace

4.4.8 Gesta

Pro snadné přepínání mezi více rozvrhy jsem přidala podporu pro gesta. Gesto je určitý symbol, který uživatel nakreslí prstem či stylusem na obrazovku zařízení. Ukázku, jaké gesto jsem vytvořila pro přepnutí rozvrhu, je vidět na obrázku 4.54. Barvu čáry jsem zde úmyslně nastavila na žlutou, aby bylo gesto viditelné, v aplikaci je barva nastavena na průhlednou.



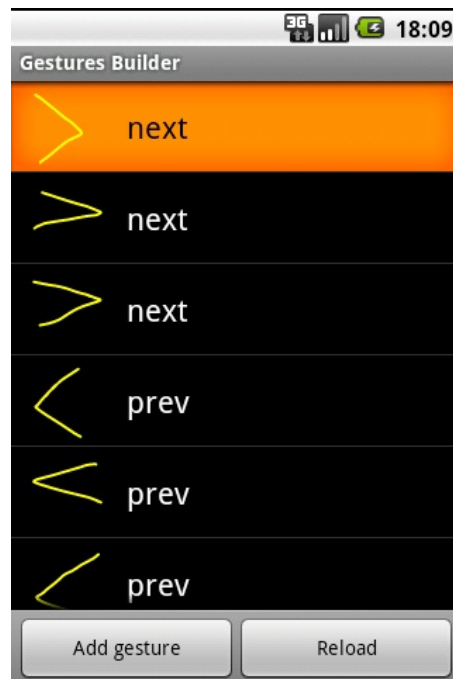
Obrázek 4.54: Gesto pro přepnutí na předchozí rozvrh

Aby bylo možné v aplikaci gesta využívat, je nejprve nutné daná gesta vytvořit. Jejich vytvoření probíhá na emulátoru v aplikaci **GesturesBuilder**. Pro fungování této aplikace je nutné mít u emulátoru nastavenou SD kartu, protože tam se budou ukládat vytvořená gesta. Každému gestu nastavíme název, který poté použijeme v aplikaci pro rozeznání daného gesta, a samotné gesto nakreslíme. Pro snazší a přesnější rozeznání gest je velmi vhodné jedno gesto nakreslit vícekrát pod stejným názvem. Dále je důležité vědět, že záleží i na směru kreslení gesta, tzn. stejné gesto na obrázku ale kresleno jednou shora a podruhé zdola nebude rozeznáno jako shodné. Na obrázku 4.55 jsou vidět mnou nastavená gesta v emulátoru.

Gesta se ukládají do `/mnt/sdcard/gestures` a je nutno je z emulátoru stáhnout a přidat k vlastní aplikaci. Stažení souboru s gesty se provede v DDMS přes souborový systém. Soubor s gesty je poté nutno umístit do `res/raw`.

V aplikaci používám gesta v aktivitě `RozvrhShow`, která implementuje rozhraní `OnGesturePerformedListener`. Načítání gest probíhá v metodě `onCreate()` následujícím způsobem.

```
GestureLibrary library = GestureLibraries.fromRawResource(this, R.raw.gestures);
library.load();
```



Obrázek 4.55: Nastavení gest v emulátoru

Po zjištění gesta je automaticky zavolána metoda `onGesturePerformed()`, která jako parametr dostane view, z kterého bylo gesto zachyceno, a samotné gesto jako objekt `Gesture`. S použitím načtené knihovny je zjištěno, jak moc se dané gesto shoduje s uloženými gesty. Míra podobnosti je zde zastoupena objektem `Prediction`, kde jsem musela nastavit určitou hodnotu, pro kterou budu dané gesto považovat za platné. Hodnotu jsem zvolila po otestování na emulátoru i fyzickém zařízení. Pokud je gesto platné, je zjištěno, o které se jedná, a podle toho je zobrazen buď následující rozvrh (gesto „uzavírací ostrá závorka“) nebo rozvrh předchozí gesto „otvírací ostrá závorka“. Při přepínání rozvrhu je tato skutečnost uživateli oznámena pomocí toast notifikace.

Dále je nutné v daném layoutu nastavit příslušné view. V layoutu `rozvrh.xml` je tedy jako kořenový element použit `android.gesture.GestureOverlayView`. Na následujícím výpisu je jeho nastavení, které bude dále popsáno.

```
<android.gesture.GestureOverlayView xmlns:android="http://schemas.android.
    com/apk/res/android"
    android:id="@+id/gesta"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:eventsInterceptionEnabled="false"
    android:gestureStrokeType="single"
    android:uncertainGestureColor="@android:color/transparent"
    android:gestureColor="@android:color/transparent"
    android:gestureStrokeLengthThreshold="250.0"
>
```

Důležité je nastavení šířky a výšky, které udává, že se gesta mají zachytávat na celé obrazovce. Atribut `eventsInterceptionEnabled` při nastavení na `true` udává, že má vrstva „ukrást“ události od svých potomků, jakmile je zjištěno, že je kresleno gesto. Toto nastavení znemožňuje rozumné posouvání obsahu pod ním, proto v tomto případě musí být nastaveno na `false`, aby se tak posouvání rozvrhu chovalo stejně jako bez vrstvy odchyťující gesta. Dále je zde nastaveno, že se bude jednat o gesto jedním tahem (`gestureStrokeType="single"`) a že kreslená čára nebude vidět. Poslední hodnota uvádí minimální délku čáry, aby mohla být rozeznána jako gesto. Tuto hodnotu jsem zvolila po testování na emulátoru i fyzickém zařízení.

4.4.9 Widget

Widget je určitá součást aplikace, která může být zobrazena na domovské obrazovce zařízení a dostávat pravidelné aktualizace. Widget obvykle slouží k zobrazení nejnovějších informací z aplikace nebo k přístupu k nejčastěji používaným funkcím. V systému Android jsou tak widgety na zobrazení hodin, počasí, hudebního přehrávače apod.

Aplikace Rozvrh se přímo nabízí k vytvoření widgetu, který by zobrazoval nejbližší rozvrhové akce. Uživatel by pak pouhým pohledem na telefon viděl, kdy a kde má další hodinu.

Aby bylo možno vytvořit widget aplikace, je potřeba vytvořit jeho layout, objekt popisující jeho metadata (`AppWidgetProviderInfo`), implementaci třídy `AppWidgetProvider`, která umožňuje měnit vzhled widgetu a reagovat na události spjaté s widgetem, případně i aktivitu pro konfiguraci widgetu.

Tyto jednotlivé součásti musí být popsány v manifestu. Ukázka manifestu aplikace Rozvrh je na následujícím výpisu. Poté budou jednotlivé součásti podrobněji popsány.

```
<activity android:name="cz.zcu.portal.android.gui.widget.
  RozvrhovaAkceWidgetConfigure">
  <intent-filter>
    <action android:name="android.appwidget.action.APPWIDGET_CONFIGURE"/>
  </intent-filter>
</activity>

<receiver android:name=".gui.widget.RozvrhovaAkceWidgetProvider" >
  <intent-filter>
    <action android:name="android.appwidget.action.APPWIDGET_UPDATE" />
  </intent-filter>
  <meta-data android:name="android.appwidget.provider"
    android:resource="@xml/appwidget_info" />
</receiver>
```

AppWidgetProviderInfo

Objekt `AppWidgetProviderInfo`, který obsahuje metadata pro widget, jako jsou jaký použít layout, frekvence aktualizace a jaká třída se stará o nastavení widgetu, je popsán XML souborem. Následující výpis zobrazuje použitý soubor `res/xml/appwidget_info.xml`.

```
<?xml version="1.0" encoding="utf-8"?>
<appwidget-provider xmlns:android="http://schemas.android.com/apk/res/
  android"
  android:minWidth="294dp"
  android:minHeight="72dp"
  android:updatePeriodMillis="0"
  android:configure="cz.zcu.portal.android.gui.widget.
    RozvrhovaAkceWidgetConfigure"
  android:initialLayout="@layout/appwidget">
</appwidget-provider>
```

Je vidět, že o konfiguraci se stará třída `RozvrhovaAkceWidgetConfigure`, jako layout je použit soubor `appwidget.xml` a frekvence aktualizace je nastavena na 0, což znamená, že widget se nebude aktualizovat. Toto nastavení je z toho důvodu, že jako nejmenší frekvenci aktualizace lze nastavit 30 minut a aktualizace není garantovaná. V případě zobrazení rozvrhových akcí je ale potřeba aktualizovat „když je třeba“, což nastane v případě, že určitá rozvrhová akce končí či začíná. Z tohoto důvodu jsem pro aktualizaci použila `AlarmManager` a službu, více viz dále.

Konfigurace widgetu

Aktivita s konfigurací widgetu nám umožňuje při jeho přidávání zvolit, pro jaké osobní číslo a semestr má widget zobrazovat data. Protože je možné na plochu přidávat více widgetů pro různé rozvrhy, přidala jsem pro jejich rozlišení možnost zvolit popisek widgetu. Výchozími hodnotami jsou poslední nastavené osobní číslo a semestr z aplikace. Jak vypadá nastavení widgetu je vidět na obrázku 4.56.

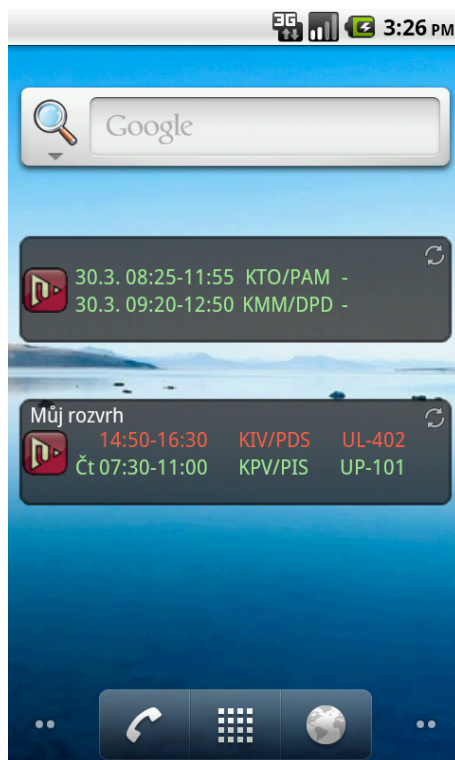


Obrázek 4.56: Nastavení widgetu

Aktivita `RozvrhovaAkceWidgetConfigure` dědí od `PreferenceActivity` a jednotlivé položky pro výběr nastavení jsou tvořeny stejně jako v případě aktivity `Nastaveni`. Pro každý vytvářený widget je zde do nastavení aplikace uloženo osobní číslo a semestr, kde klíč nastavení je `osobnicislo_idWidgetu` nebo `semestr_idWidgetu`. Tyto hodnoty je nutné mít uloženy, protože jsem implementovala funkci, aby se po kliknutí na widget spustila aplikace s daným rozvrhem (viz dále), a je tedy nutné zjistit nastavené hodnoty.

Vzhled widgetu

Vzhled widgetu udává layout `appwidget.xml`. Widget je formátu 4x1, což znamená, že zabírá na domovské obrazovce jeden celý řádek (čtyři sloupce). Jedná se o relativně komplikovaný layout, protože je potřeba potřebné informace umístit přehledně do poměrně malého prostoru. Pro tento layout jsem použila `RelativeLayout` a pozici jednotlivých prvků tak určuji relativně k rodiči nebo jinému prvku. Samotný layout se skládá z ikony rozvrhu, ze dvou rozvrhových akcí tvořených časem, názvem a místem akce, z ikonky na ruční aktualizaci rozvrhu a případně i popiskem. Ukázka widgetů je vidět na obrázku 4.57.



Obrázek 4.57: *Widgety zobrazující rozvrhové akce*

Červeně jsou zobrazeny právě probíhající akce, zeleně ty, které teprve nastanou. V případě, že akce bude probíhat daný den, tak není uvedena zkratka dne. U rozvrhů kombinovaného studia je uveden místo dne datum konání akce.

RozvrhovaAkceWidgetProvider a aktualizace obsahu widgetu

Tato třída dědí od `AppWidgetProvider` a slouží k zachytávání broadcast zpráv

týkajících se widgetu a k reakcím na tyto zprávy. Zprávami mohou být aktualizace widgetu, jeho smazání, odstranění posledního widgetu atd.

Jedinou metodu, kterou implementuji, je metoda `onUpdate()`, která je zavolána pokaždé, když je widget přidán na plochu. Obsluhu widgetů neprovádím v této třídě, jak je obvyklé, ale při zavolání metody `onUpdate()` spustím vlastní službu `UpdateWidgetService`, která se dále stará o aktualizace widgetů.

Použití vlastní služby je výhodné, protože widget je potřeba aktualizovat pokaždé za různý čas a služba provede potřebné aktualizace, vypočte, kdy se má znovu spustit a sama se ukončí. Takovéto chování služby šetří systémové prostředky a tím i baterii přístroje. Služba zjistí ID všech vlastních widgetů a pro každý z nich provádí potřebná nastavení.

Nejprve je nutné zjistit z nastavení jaké osobní číslo a semestr patří danému widgetu a pak je zavolána servisní metoda pro nalezení dvou následujících rozvrhových akcí. Touto metodou je `getSeznamAkciProWidget()` třídy `RozvrhoveAkceSupport`. Nalezení akcí není úplně triviální, protože se musí počítat s daty platnosti akce a s tím, že akce může být jen v lichý či sudý týden. U objektu `DOrozvrhoveAkce` jsou vytvořeny pomocné metody pro nalezení data začátku a konce nejbližšího konání akce. Nalezené akce jsou vráceny jako objekty `DOakceWidget`, které obsahují objekt rozvrhové akce a příznaky, zda akce probíhá právě teď či zda je v aktuální den. Dále obsahuje metodu na získání informací o rozvrhové akci v textové podobě vhodné pro zobrazení ve widgetu.

Služba dále provede následující změny zobrazení widgetu:

- nastaví popisek, pokud byl nastaven
- vytvoří intent a nastaví, aby se spustil po kliknutí na widget
- nastaví spuštění sebe sama po kliknutí na ikonku s aktualizací
- nalezené rozvrhové akce nastaví do příslušných view

Posledním úkonem služby je zjištění a nastavení, kdy se má znovu spustit. K zjištění času, za kolik minut má být služba znovu spuštěna, slouží metoda `getZaKolikMinutZměna()` ze třídy `RozvrhoveAkceSupport`. Této metodě je předán seznam nalezených akcí přes všechny widgety, ze kterých je zjištěno, za jak dlouho nastane začátek nebo konec nejbližší akce. Služba poté nastaví alarm, který za daný čas spustí tuto službu. Nastavení alarmu je vidět na následujícím výpisu.

```
AlarmManager alarmManager = (AlarmManager) context.getSystemService(Context.  
    ALARM_SERVICE);  
Calendar calendar = Calendar.getInstance();
```

```
calendar.add(Calendar.MINUTE, zmenaZaMinut);  
alarmManager.set(AlarmManager.RTC, calendar.getTimeInMillis(), sluzba);
```

Proměnná `zmenaZaMinut` je zjištěný čas, za kdy službu znovu spustit. Alarm je nastaven typu `RTC`, což znamená, že zařízení nebude probuzeno ze spánku, ale k vyvolání dojde, až když je zařízení vzbuzeno. Díky tomuto nastavení je šetřena baterie přístroje a nedochází ke zbytečným aktualizacím, které nejsou v danou chvíli potřeba. Nakonec se služba sama zastaví voláním `stopSelf()`.

4.5 Testování aplikace

Jak bylo uvedeno v kapitole 3.3.5, existuje více technik, jak otestovat Android aplikaci. Při testování aplikace Rozvrh jsem použila jak framework pro Android JUnit testy (postavené na JUnit 3), tak i rozšiřující nástroj Robotium [15]. Klasické JUnit testy nebo jejich rozšíření o podporu Android SDK jsem použila pro otestování technické kvality, tj. pro otestování tříd z balíku `data.support`, které slouží pro práci s daty. Testovací framework Robotium se naopak hodí pro otestování uživatelského rozhraní tak, jako by aplikaci používal uživatel. Tímto frameworkem jsou tedy otestovány aktivity z balíku `gui`.

4.5.1 Android JUnit

Jak již bylo řečeno, Android JUnit je rozšíření klasických JUnit testů a podporuje pouze psaní testů ve stylu JUnit 3. Android poskytuje několik tříd, které dědí od `TestCase` a `Assert` a poskytují specifické nastavovací metody (*setup* a *teardown*) a pomocné metody.

Základní třídou je `AndroidTestCase`, která poskytuje standardní metody `setUp()`, `tearDown()` a `assert` metody. Dále obsahuje metody pro testování práv (`permissions`). Rozšiřující třídou je `InstrumentationTestCase`, která se používá pokud potřebujeme využívat `instrumentation` metody (viz dále).

Dalšími třídami pro testování jsou specifické třídy podle typu komponenty. Existují tedy třídy pro testování aktivit, služeb a poskytovatelů obsahu. Pro testování aktivit se nejčastěji používají dvě třídy, a to `ActivityUnitTestCase` a `ActivityInstrumentationTestCase2`, které obě dědí od základní testovací třídy `InstrumentationTestCase`.

Android poskytuje i rozšiřující `assert` třídy `MoreAsserts` a `ViewAsserts`. První poskytuje metody např. pro kontrolu textu pomocí regulárního výrazu. Metody druhé třídy slouží k testování zarovnání objektů apod. v uživatelském rozhraní.

4.5.2 Robotium

Jedná se o nástroj, který se chová jako uživatel, vidí tedy to, co vidí uživatel, může klikat na prvky uživatelského rozhraní a dokonce přepínat mezi zobrazením na výšku a na šířku. Robotium rozšiřuje Android JUnit testy a je navržen tak, aby bylo psaní testů uživatelského rozhraní jednoduché, rychlé a efektivní.

Robotium je projekt vyvíjen pod Apache License 2.0. Projekt se neustále vyvíjí a aktuálně je k dispozici verze 3.2.1 vydaná 11.4.2012. Robotium je dostupné jako jar knihovna a pro její použití je tedy nutné tuto knihovnu přidat do adresáře libs testovacího projektu.

Veškerý přístup k testovacím funkcím frameworku Robotium je přes objekt Solo. Tento objekt je potřeba vytvořit v metodě setUp() a předat konstrukturu instanci objektu Instrumentation a aktivitu pomocí metody getActivity(). V samotných testovacích metodách pak můžeme volat metody objektu Solo. Mezi metody patří např.:

- clearEditText(int index)
- clickInList(int line)
- clickOnButton(String regex)
- clickOnMenuItem(String regex)
- enterText(int index, String text)
- searchText(String regex)
- getCurrentActivity()
- goBack(), goBackToActivity(String name)

Test uživatelského rozhraní, který můžeme napsat standardními Android JUnit testy, se dá mnohem jednodušeji napsat pomocí Robotium. Následuje ukázka otestování jednoduché kalkulačky, kde jsou dvě vstupní pole (EditText) a tlačítko (Button), které po kliknutí sečte dvě zadané hodnoty a výsledek zobrazí v prvním poli. Pomocí standardních testů by test vypadal následovně.

```
EditText cislo1 = (EditText) getActivity().findViewById(R.id.cislo1);
EditText cislo2 = (EditText) getActivity().findViewById(R.id.cislo2);
Button soucetButton = (Button) getActivity().findViewById(R.id.soucet_button);

// kliknuti na prvni pole a zadani cisel 1 a 5
TouchUtils.tapView(this, cislo1);
sendKeys(KeyEvent.KEYCODE_1, KeyEvent.KEYCODE_5);
// kliknuti na druhe pole a zadani cisla 5
```

```
TouchUtils.tapView(this, cislo2);
sendKeys(KeyEvent.KEYCODE_5);
// kliknuti na tlacitko
TouchUtils.tapView(this, soucetButton);

// ziskani vysledku a porovnani s ocekavanou hodnotou
String vysledek = cislo1.getText().toString();
String ocekavanyVysledek = "20.0";
assertEquals("Chybný součet", ocekavanyVysledek, vysledek);
```

Z výpisu je vidět, že programování takovýchto testů je velmi těžkopádné a musíme znát poměrně dost detailů ze samotné aplikace (jaká jsou view apod.). Přičemž z pohledu uživatele by otestování probíhalo v následujících krocích:

- Napiš do prvního pole 15.
- Napiš do druhého pole 5.
- Klikni na tlačítko *Sečti*.
- Zkontroluj, že výsledek je 20.

A právě takovýto způsob testování umožňuje Robotium. Celý test by vypadal následovně.

```
solo.enterText(0, "15");
solo.enterText(1, "5");
solo.clickOnButton("Sečti");
assertTrue(solo.searchEditText("20"));
```

Další výhodou Robotium je možnost otestovat i samotný apk soubor bez jeho zdrojových kódů. Robotium navíc poskytuje i snadnou integraci s Mavenem či Antem.

Použití frameworku Robotium pro testování aplikace Rozvrh bude ukázáno v kapitole 4.5.7.

4.5.3 Testovací projekt

Abychom mohli psát testy, je nejprve nutné vytvořit odpovídající testovací projekt. Tento projekt je nutné propojit s projektem (aplikací), který chceme testovat. V prostředí Eclipse zvolíme **File** → **New** → **Project...** Pod složkou **Android** zvolíme **Android Test Project**, čímž se spustí průvodce vytvořením projektu. Na první obrazovce vyplníme název projektu, na následující obrazovce vybereme, který projekt chceme testovat, a nakonec zvolíme verzi Androidu. Vytvoří se prázdný projekt, který obsahuje stejné složky jako při vytváření

klasického Android projektu (viz kapitola 3.2.1). Jak vypadá automaticky vytvořený manifest je vidět na následujícím výpisu.

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

  <uses-sdk android:minSdkVersion="4" />

  <instrumentation android:targetPackage="cz.zcu.portal.android" android:name
    ="android.test.InstrumentationTestRunner" />

  <application android:icon="@drawable/icon" android:label="@string/app_name"
    >
    <uses-library android:name="android.test.runner" />
  </application>
</manifest>
```

Zde bych chtěla poukázat na element `instrumentation`. Slovem *instrumentation* je označována sada kontrolních metod či „háčeků“ (*hooks*) v systému Android. Tyto metody umožňují kontrolu nad Android komponentami nezávisle na jejich životním cyklu. Také kontrolují jak Android načítá aplikace. Za normálních okolností běží komponenty v životním cyklu, který je určen systémem, a nemůžeme explicitně volat metody cyklu typu `onCreate()`. Pomocí Android *instrumentation* toto lze a je možné procházet životním cyklem komponent krok po kroku.

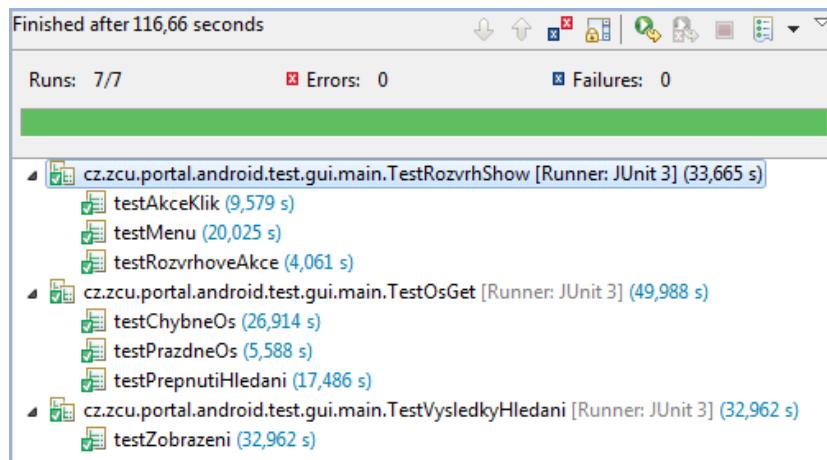
Po vytvoření projektu již můžeme vytvářet JUnit testy kliknutím pravého tlačítka myši na projekt a zvolením `File` → `New` → `JUnit Test Case`.

4.5.4 Spuštění testů

Pro spuštění testů klikneme pravým tlačítkem myši na testovací projekt a zvolíme `Run as` → `Android JUnit Test`. Testovací projekt bude nainstalován na emulátor, jako klasický Android projekt, dojde ke spuštění emulátoru a proběhnutí testů. Výsledek testů je pak v Eclipse zobrazen v záložce *JUnit* (viz obr. 4.58).

4.5.5 Databáze pro testování

Jednotlivé testy by měly být na sobě nezávislé, proto je vhodné mít vytvořenu testovací databázi. Tato databáze bude naplněna daty rozvrhů několika studentů a bude sloužit pro otestování funkčnosti aplikace. Vytvoření testovací da-



Obrázek 4.58: Ukázka výsledku testování

tabáze neprobíhá postupným vytvářením obsahu databáze, ale přehráním souboru s již naplněnou databází. Spustíme tedy aplikaci na emulátoru a zadáme několik osobních čísel, jejichž rozvrhy se budou pro testování hodit. Samotný soubor s databází poté získáme z emulátoru přes DDMS, viz kapitola 4.2.3. Tento soubor poté bude sloužit k přehrání databáze při testování na emulátoru či zařízení.

Soubor s databází umístíme do testovacího projektu do složky `assets`. K nahrání databáze slouží třída `DatabaseCreator`, která otevře databázi z `assets` jako vstupní proud a přepíše uloženou databázi aplikace, viz následující výpis.

```
// cesta k databazi, kterou chceme prepsat
String outFileName = DB_PATH + DB_NAME;

InputStream inputDB = myContext.getAssets().open(DB_NAME);
OutputStream outputDB = new FileOutputStream(outFileName);

byte[] buffer = new byte[1024];
int length;
while ((length = inputDB.read(buffer)) > 0) {
    outputDB.write(buffer, 0, length);
}

outputDB.flush();
outputDB.close();
inputDB.close();
```

Databáze je načtena v metodě `setUp()` každé testovací třídy. Jako kontext je předán `getInstrumentation().getContext()`, což je kontext testovacího projektu. Pokud je v testování potřeba kontext testovaného projektu, použije se metoda `getTargetContext()`, opět nad instancí třídy `Instrumentation`.

4.5.6 Testování servisních tříd

Pro testování tříd z balíku `support` jsem vytvořila testovací třídy dědicí od `InstrumentationTestCase`. I když se jedná o otestování servisních tříd, nelze použít obyčejný `TestCase`, protože je nutné získat kontext jak testovacího projektu, tak testované aplikace. Kontext testovacího projektu slouží k nahrání databáze, kontext testované aplikace slouží k vytvoření potřebných servisních tříd, které kontext využívají pro přístup do databáze.

Každá testovací třída je otestováním jedné servisní třídy. Názvy tříd a metod tak určují název testované třídy a jejích metod. Například metody ze třídy `PredmetySupport` testuje třída `TestPredmetySupport`. V metodě `setUp()` dochází k nahrání databáze a vytvoření instance servisní třídy, která je dále používána v testovacích metodách. Ukázka nastavení testovacího případu viz následující výpis.

```
@Override
protected void setUp() throws Exception {
    super.setUp();

    // nahrani databaze s daty
    DatabaseCreator dbCreator = new DatabaseCreator(getInstrumentation().
        getContext());
    dbCreator.createDataBase();

    this.predmetySupport = new PredmetySupport(getInstrumentation().
        getTargetContext());
}
```

Samotné testování metod pro uložení objektu spočívá v jeho vytvoření, zavolání metody pro uložení objektu a kontroly výsledku uložení. Obdobně jsou testovány i ostatní metody servisních tříd.

4.5.7 Testování uživatelského rozhraní

Třídy pro testování aktivit dědí od `ActivityInstrumentationTestCase2<T>`, kde objekt `T` je testovaná aktivita. Před samotným testováním je nutné zavolat metodu `getActivity()`. Tato metoda je součástí kontrolních metod a má na starosti vytvoření a spuštění samotné aktivity pro testování.

Pro testování aktivit jsem použila testovací framework Robotium a testování tedy probíhá pomocí objektu `Solo`, který je nutné inicializovat v metodě `setUp()` (viz dále).

Testovací třídy jsou uloženy v balíku `gui` a nachází se zde třídy pro otestování všech aktivit aplikace. Je tedy otestováno zobrazení rozvrhu, detailů (před-

mětů, učitelů, studentů), přidávání nového rozvrhu, hledání studentů podle jména a příjmení a nastavení aplikace. Detailněji bude nadále popsáno testování některých funkcí.

Testování zobrazení rozvrhu

Pro otestování aktivity `RozvrhShow` je nejprve nutné v metodě `setUp()` nastavit `bundle` se vstupními daty, který se předá aktivitě při spuštění. Ukázka jakým způsobem předat aktivitě data ještě před jejím spuštěním viz následující výpis.

```
Bundle bundle = new Bundle();
bundle.putString("os", OSOBNI_CISLO);
bundle.putString("semestr", SEMESTR);

Intent newIntent = new Intent();
newIntent.putExtras(bundle);
setActivityIntent(newIntent);

solo = new Solo(getInstrumentation(), getActivity());
```

Aktivitě je tedy předáno osobní číslo a semestr rozvrhu, který chceme otestovat. Před prvním zavoláním metody `getActivity()` je nutné zavolat metodu `setActivityIntent()` a nastavit tak intent s daty pro spuštění.

Na obrazovce s rozvrhem je testováno např. menu (`testMenu()`), zobrazení akcí (`testRozvrhoveAkce()`) či kliknutí na rozvrhovou akci (`testAkceKlik()`).

Otestování menu probíhá kliknutím na jednotlivé položky menu, které jsou identifikovány svým názvem. Po kliknutí na položku je zkontrolováno, zda byla spuštěna příslušná aktivita, dojde k otočení displeje a vrácení se zpět do aktivity zobrazující rozvrh. Ukázka jednoho prokliknutí menu viz výpis.

```
Activity a = getActivity();
solo.clickOnMenuItem(a.getString(R.string.nastaveni));
solo.assertCurrentActivity("Aktivita nastaveni", Nastaveni.class);
solo.setActivityOrientation(Solo.LANDSCAPE);
solo.goBack();
```

U zobrazení akcí je testováno, zda jsou v rozvrhu zobrazeny všechny akce, které patří danému studentovi v daný semestr. Seznam zobrazených akcí se získá jako `solo.getCurrentButtons()`, tedy zjistí se seznam tlačítek. Každé tlačítko reprezentuje jednu akci v rozvrhu a ID tlačítka odpovídá ID rozvrhové akce. Seznam akcí, které mají být zobrazeny, je zjištěn z databáze a dojde k porovnání ID každé akce.

Testování zobrazení detailu po kliknutí na rozvrhovou akci se provede voláním `solo.clickOnButton(0)` (kliknutí na první tlačítko) a zkontrolováním, zda se spustila správná aktivita.

Testování detailů předmětů, učitelů a studentů

Toto testování se týká aktivit `PredmetShow`, `UcitelShow` a `StudentShow`. Pro testování aktivit zobrazujících detailní údaje je nejprve potřeba předat aktivitám parametr s ID (předmětu, učitele, studenta), jehož detaily chceme zobrazit. Zobrazené hodnoty jsou testovány následujícím voláním.

```
assertEquals(solo.getText(6).getText(), "UP-115");
```

Nejprve je nalezeno `TextView` na dané pozici a poté je zkontrolováno, že se text shoduje s očekávaným. V případě předmětu je testováno i spuštění správné aktivity po kliknutí na jméno vyučujícího.

Testování přidávání rozvrhu

Obrazovka pro přidání rozvrhu (aktivita `OsGet`) obsahuje vstupní pole pro zadání osobního čísla, jehož rozvrh chceme zobrazit. Vstupní pole je otestováno na vyvolání chybového dialogu v případě prázdné hodnoty. V případě chybně zadaného osobního čísla (nejsou žádná odpovídající data) je také zobrazen chybový dialog. Test vyvolání dialogu se provádí následujícím způsobem:

```
solo.enterText(0, "abc");  
solo.clickOnButton("OK");  
solo.waitForView(getActivity().findViewById(R.id.error_dialog));  
assertTrue(solo.searchText(getActivity().getString(R.string.err_zadna_data))  
);  
solo.clickOnButton("OK");
```

Test probíhá tak, že je nejprve vložen text *abc* jako chybné osobní číslo, poté je kliknuto na tlačítko OK. Následně se čeká, než se zobrazí chybový dialog, konkrétně view, které je součástí dialogu. Nakonec je zkontrolováno, že dialog obsahuje správné chybové hlášení, a kliknutím na OK je dialog zavřen.

Dále je i otestována možnost přepnout ze zadávání osobního čísla na hledání podle jména a příjmení. Opět jsou zkontrolovány vstupní hodnoty a dále také spuštění aktivity `VysledkyHledani` zobrazující výsledky hledání. Aby bylo možno hledat výsledky, je do nastavení aplikace nahrán autorizační řetězec (zakódované přihlašovací jméno a heslo). Aktivita `VysledkyHledani` je otestována v samostatné třídě, kde je také nastavena autorizace a dále i zkontrolovány výsledky hledání a přidání rozvrhu prvního nalezeného studenta.

4.5.8 Testovací zařízení

Aplikaci jsem testovala na emulátoru i fyzických zařízeních s různými verzemi Androidu. V počátku vytváření aplikace jsem testovala převážně na verzích 1.6., 2.1 a 2.2, ale s postupem vydávání nových verzí Androidu jsem začala testovat i na zařízeních s verzemi 2.3.x a 4.0.x.

Fyzická zařízení, na kterých jsem testovala (v závorce jsou verze Androidu):

- T-Mobile G1 (1.6, 2.1, 2.2 – neoficiální)
- HTC Desire Z (2.2., 2.3.3, 4.0.3 – neoficiální)
- Samsung Galaxy S (2.2, 2.3.3, 4.0.4 – neoficiální)
- Samsung Galaxy Fit (2.2, 2.3.6)
- Samsung Galaxy 3 (2.1, 2.2)

5 Nasazení aplikace – publikování na Google Play

5.1 Přípravení aplikace

Pro publikování aplikace je nejprve nutno aplikaci připravit. Minimálním požadavkem je mít ikonu aplikace a kryptografický klíč. Aplikaci je nutno podepsat tímto vlastním privátním klíčem. Doba platnosti klíče musí být až po datu 22.9. 2033. Klíč je možné vygenerovat pomocí nástroje Keytool. Následuje ukázka příkazu Keytool pro vygenerování privátního klíče.

```
keytool -genkey -v -keystore muj-kluc.keystore -alias alias_name -keyalg RSA  
-keysize 2048 -validity 10000
```

Po spuštění tohoto příkazu budete vyzváni k zadání hesla, kterým bude keystore a klíč zaheslován. Platnost klíče se uvádí v počtech dnů a doporučeno je dávat 10000 a více.

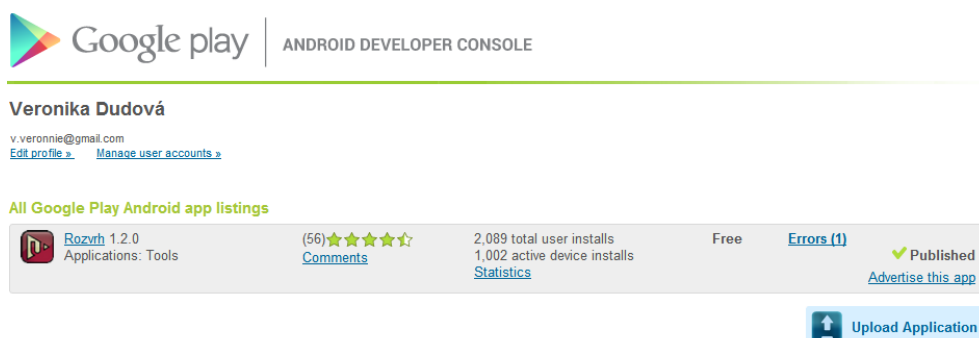
Pokud máme vygenerovaný klíč, můžeme aplikaci podepsat a provést zarovnání finálního apk balíku. Tyto akce lze provést z příkazové řádky, ale nejjednodušším způsobem je využít ADT plugin v Eclipse. V Eclipse klikneme pravým tlačítkem na daný Android projekt a zvolíme **Android tools** → **Export Signed Application Package...**. Objeví se průvodce exportem apk balíku, kde zadáme cestu k uloženému keystore a heslo, dále vybereme klíč a zadáme heslo k danému klíči. Nakonec zvolíme, kam chceme podepsaný apk balík uložit. Eclipse provede potřebné podepsání a zarovnání aplikace (viz 3.2.2), finální apk balík je pak připravený k publikaci na Google Play.

5.2 Vývojářská konzole

Aplikace na Google Play lze nahrávat jen z vývojářské konzole na adrese play.google.com/apps/publish. Zde je nutné se přihlásit Google účtem a pokud nejsme registrováni jako vývojáři, pak nejdříve provést registraci a zaplatit poplatek. Poté již máme přístup k publikování aplikací.

Jak vypadá vývojářská konzole, je vidět na obrázku 5.59. Je zde vidět seznam publikovaných aplikací a základní informace jako verze, hodnocení, základní statistika atd.

Po kliknutí na název aplikace se dostaneme na její detail, který můžeme upravovat.

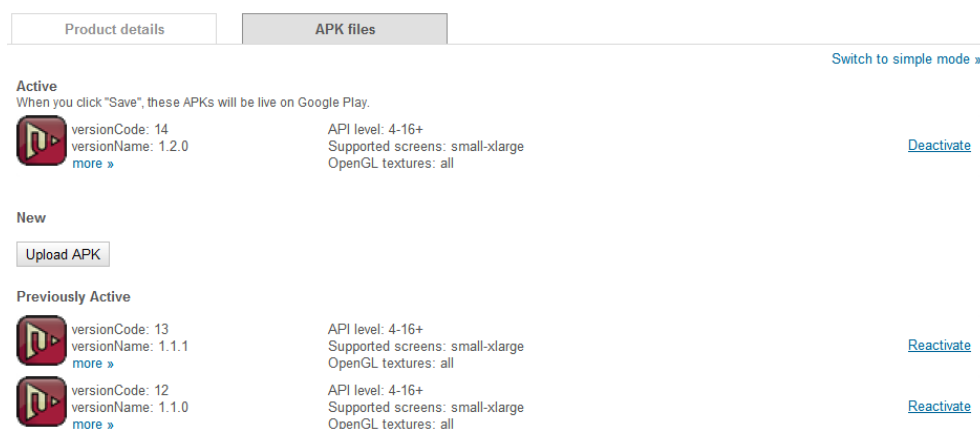


Obrázek 5.59: Android vývojářská konzole

Možnosti nastavení aplikace:

- screenshoty, ikona a další grafika
- jazyky v jakých budou informace o aplikaci
- název aplikace, popis, poslední změny – vše v uvedených jazycích
- typ aplikace a kategorie
- cena aplikace (nutno další nastavení)
- země, kterým bude aplikace přístupná
- údaje o vývojáři

Kromě detailu aplikace je zde i záložka s apk soubory. Zde jsou všechny nahrané verze a je zde možnost nahrávat nové, deaktivovat aktuální a reaktivovat starší (obr. 5.60).

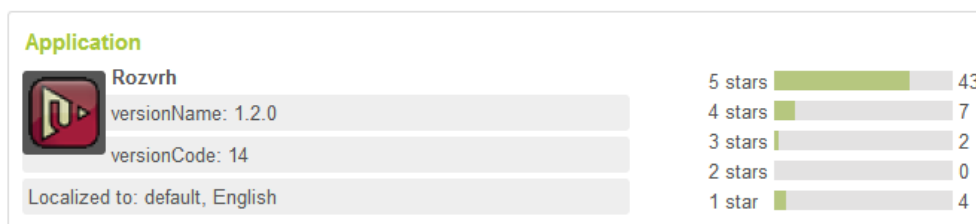


Obrázek 5.60: APK soubory ve vývojářské konzoli

5.2.1 Hodnocení a komentáře

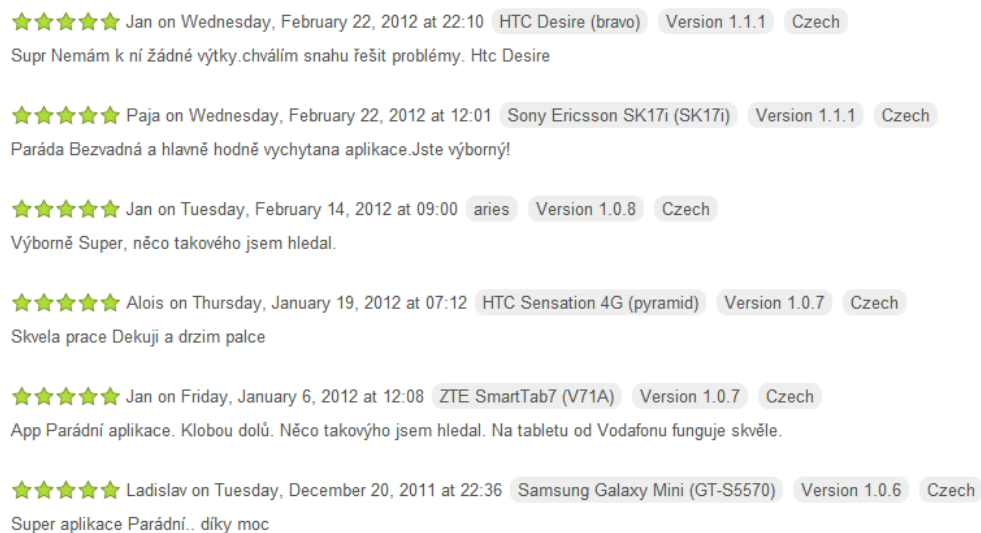
Uživatelé mají možnost ohodnotit aplikace na Google Play a napsat k nim komentáře. Vývojář tak dostane zpětnou vazbu, případné nedostatky může opravit a zvážit zapracování návrhů na vylepšení.

Hodnocení aplikace probíhá udělením hvězd, kde čím více hvězd, tím lepší hodnocení. Hodnocení aplikace Rozvrh je vidět na obrázku 5.61.



Obrázek 5.61: Hodnocení aplikace Rozvrh

K hodnocení mohou uživatelé přidat i vysvětlující komentáře. U komentářů je uveden datum, zařízení a verze aplikace, ke které se komentář vztahuje. Několik posledních komentářů je vidět na obrázku 5.62.



Obrázek 5.62: Komentáře k aplikaci

5.2.2 Statistiky

Vývojářská konzole poskytuje i podrobné statistiky týkající se používání aplikace. Je zde možno nechat si zobrazit statistiky za různá časová období. Poskytované statistiky:

- celkový počet zařízení, kde je aplikace nainstalována
- počet zařízení, na kterých byla aplikace nainstalována/odinstalována/aktualizována za poslední dny
- celkový počet uživatelů, kteří aplikaci nainstalovali
- počet uživatelů, kteří aplikace nainstalovali/odinstalovali/aktualizovali za poslední dny

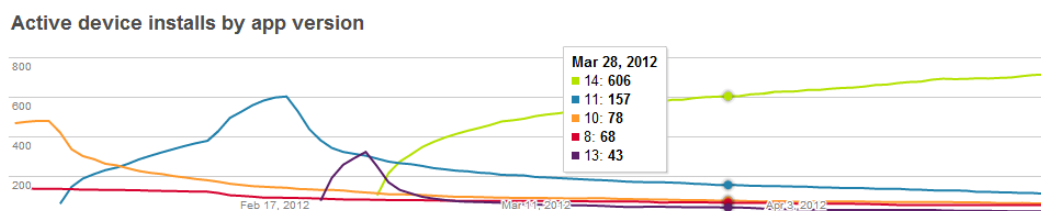
Všechny tyto statistiky lze ještě rozšířit o statistiky podle verze Androidu, druhu zařízení, země, jazyka, verze aplikace a operátora.

Na obrázku 5.63 je vidět statistika celkového počtu zařízení, kde byla aplikace nainstalována. Na grafu je vidět, že došlo ke dvěma větším nárůstům počtu nainstalování, a to od poloviny září 2011 a od poloviny února 2012, což je doba začátku zimního a letního semestru.



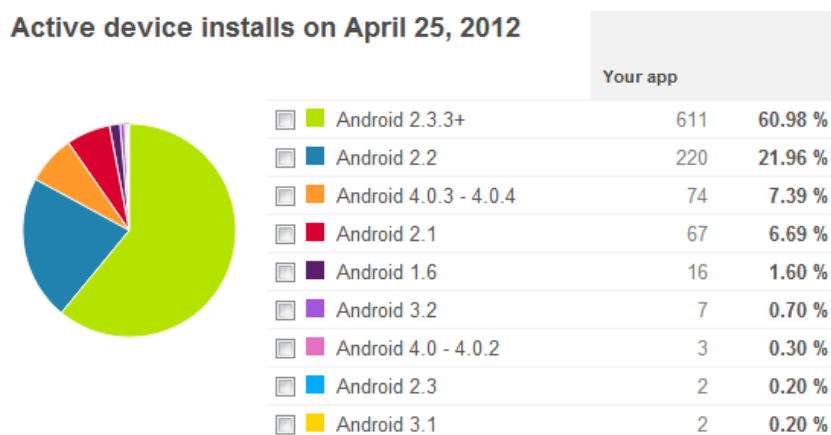
Obrázek 5.63: Statistika celkového počtu zařízení, kde byla aplikace nainstalována

Zajímavá je i statistika podle verze používané aplikace (obr. 5.64). Zde je dobře vidět, jak rychle uživatelé přecházejí na novější verzi a počty uživatelů, kteří používají jednotlivé verze.



Obrázek 5.64: Statistika celkového počtu zařízení, kde byla aplikace nainstalována, podle verze aplikace

Co se týče verze Androidu, na kterém aplikace běží, je z grafu 5.65 vidět, že nadpoloviční většinu má verze 2.3.3+. Verze 1.6, což je minimální verze, pro kterou aplikaci poskytuji, má zastoupení jen necelá dvě procenta.



Obrázek 5.65: Celkové počty zařízení s nainstalovanou aplikací podle verze systému

5.3 Verze aplikace

Verze	Kód	Datum	Popis změn
1.0	1	12.4.2011	Možnost přidávat rozvrhy zadáním osobního čísla. Změna rozvrhu a semestru přes nastavení.
1.0.1	2	19.4.2011	Lokalizace aplikace do angličtiny.
1.0.2	3	21.4.2011	Opravy chyb zobrazení rozvrhu.
1.0.3	4	22.4.2011	Vylepšeno zobrazení překrývajících se rozvrhů.
1.0.4	5	11.9.2011	Aktualizace a mazání rozvrhů přes menu. Přidány detaily o studentovi. Přidány detaily o učitelovi. Možnost zaslat vyučujícímu email (kliknutím na jeho emailovou adresu). Možnost vyhledat studenty podle jména a příjmení a uložení jejich rozvrhu.
1.0.5	6	18.9.2011	Přidán widget zobrazující dvě následující či aktuální rozvrhové akce. Aktualizace widgetu každých 5 minut. Přidáno označení akcí, které jsou jen v sudém/-lichém týdnu. Zobrazení zda je aktuální týden sudý nebo lichý.

Verze	Kód	Datum	Popis změn
1.0.5.1	7	18.9.2011	Oprava chybného zobrazení lichého/sudého týdne na některých zařízeních.
1.0.5.2	8	23.9.2011	Oprava chybného zobrazení lichého/sudého týdne na některých zařízeních.
1.0.6	9	27.11.2011	<p>Widget</p> <p>Aktualizace jen při změně rozvrhové akce.</p> <p>Přidáno nastavení widgetu, obsahuje osobní číslo, semestr a popis (nepovinný).</p> <p>Možno používat více widgetů najednou, po kliknutí na widget se zobrazí daný rozvrh.</p> <p>Oprava zobrazení lichých/sudých akcí z následujícího týdne.</p> <p>Aplikace</p> <p>Optimalizace a opravy chyb.</p> <p>Při zobrazení rozvrhu na šířku jeho roztažení přes celou šířku displeje.</p> <p>Možnost změny jazyka přes nastavení.</p>
1.0.7	10	17.12.2011	<p>Možnost v nastavení zadat orion login a heslo.</p> <p>Vyhledávání podle jména a příjmení studenta jen uživatelům s platnými přihlašovacími údaji.</p> <p>Jméno a příjmení studenta zobrazeno v hlavičce rozvrhu jen uživatelům s platnými přihlašovacími údaji.</p>
1.0.8	11	29.1.2012	<p>Oprava přepínání semestru.</p> <p>K detailu předmětu přidán odkaz na Courseware, počet kreditů a požadavky (nutná aktualizace rozvrhu).</p>
1.1.0	12	19.2.2012	<p>K detailu předmětu přidán datum akce.</p> <p>Podpora rozvrhu pro kombinované studium.</p> <p>Možnost zobrazit kompletní rozvrh nebo jen aktuální týden.</p> <p>Widget zobrazuje u budoucích akcí místo dne datum, pokud nejsou v aktuálním týdnu.</p>
1.1.1	13	21.2.2012	Opraveno zobrazování akcí ve widgetu.
1.2.0	14	26.2.2012	<p>Ve widgetu zohledněna data akcí.</p> <p>Přepínání mezi rozvrhy pomocí gest.</p> <p>Vylepšení výkonu.</p>

5.4 Možnosti rozšíření

Aplikaci již využívá poměrně hodně uživatelů a bylo by určitě vhodné aplikaci dále rozvíjet a rozšiřovat. Prvním rozšířením, které se nabízí, je přidání podpory pro další univerzity používající IS/STAG. Aplikace je vytvořena tak, aby bylo jednoduché využívat webové služby jiných univerzit, problém nastává při zobrazení rozvrhů s jinými časovými řadami. Bylo by tedy nutné vytvořit mechanismus pro načítání odpovídajících časových řad, které používají dané univerzity a vyřešit zobrazení více časových řad najednou.

Dalším zajímavým rozšířením je možnost přesouvat rozvrhové akce, tzn. možnost si vybrat na kterou akci (typicky cvičení) opravdu chodím. S tím souvisí i možnost vkládání vlastních akcí týkajících se studia, např. schůzky ohledně projektů apod.

Aplikace by šla dále rozšířit, aby sloužila i vyučujícím k zobrazení jejich rozvrhu.

6 Závěr

Seznámila jsem se s nejrozšířenějšími operačními systémy pro chytré telefony a vybrala systém Android, jako nejvhodnější pro realizaci aplikace rozvrhu hodin pro studenty ZČU. Dále jsem prostudovala a popsala postup vytváření aplikací pro Android a způsob použití webových služeb nad IS/STAG. Pomocí získaných znalostí jsem vytvořila aplikaci plně splňující zadání, která navíc obsahuje i další užitečné funkce nad rámec zadání. Aplikaci jsem během vývoje testovala jak na emulátoru, tak na několika fyzických zařízeních. Zadání diplomové práce tak bylo kompletně splněno.

Aplikace byla od poměrně ranného vývoje publikována na Google Play (dříve Android Market), postupně byly zapracovávány další funkce a aplikace byla vydávána v dalších verzích. Doposud těchto verzí bylo 14, od patchů opravujících chyby až po větší změny přidávající nové funkce. Postupem času si aplikace našla řadu příznivců a počet uživatelů se nyní pohybuje kolem 1000. Zpětná vazba od uživatelů mi dávala podněty k dalšímu rozšiřování aplikace. Na žádost uživatelů jsem vytvořila podporu pro rozvrhy studentů kombinovaného studia, která původně nebyla v plánu, díky mylnému přesvědčení, že by tito studenti aplikaci nevyužívali.

Podrobné informace o nasazení, počtech uživatelů, verzích atd. jsou uvedeny v kapitole 5.

Aplikace je velmi dobře použitelná jak pro zobrazení samotného rozvrhu, tak i pro zjištění detailů jednotlivých rozvrhových akcí nebo údajů o učitelích.

Protože aplikaci rozvrhu pod Portálem využívá několik dalších univerzit v ČR, lze předpokládat, že se vytvořená aplikace může rozšířit i za hranice ZČU.

Seznam zkratek

ADB	Android Debug Bridge – univerzální nástroj pro příkazovou řádku pro práci s Android zařízeními
ADT	Android Development Tools – plugin pro Eclipse IDE pro podporu programování pro Android
API	Application Programming Interface – rozhraní pro programování aplikací
APK	Application Package File – souborový formát Android aplikací
AVD	Android Virtual Devices – konfigurace virtuálního Android zařízení (emulátoru)
CSV	Comma-separated Values – jednoduchý souborový formát
DAO	Data Access Object – objekt pro přístup k datům
DDMS	Dalvik Debug Monitor Server – grafické rozhraní pro komunikaci s Android zařízením
IS/STAG	Informační systém pro administraci studijní agendy vysoké školy, univerzity nebo vyšší odborné školy.
JDK	Java Development Kit – soubor základních nástrojů pro vývoj Java aplikací
ORM	Objektově relační mapování
OS	Opravní systém
PDA	Personal Digital Assistant – malý kapesní počítač
REST	Representation State Transfer – rozhraní pro jednotný a snadný přístup ke zdrojům
SDK	Software Development Kit – sada vývojářských nástrojů
SOAP	Simple Object Access Protocol – protokol pro webové služby
UI	User Interface – uživatelské rozhraní
XML	Extensible Markup Language – rozšiřitelný značkovací jazyk

Reference

- [1] mpulp.mobi. *Srovnání prodeje mobilních telefonů podle platform* [online]. c2011 [cit. 15.1.2012]. Dostupné z <<http://mpulp.mobi/2011/05/smartphone-platforms-in-europe-vs-us/>>.
- [2] comScore. *Tisková zpráva srovnání prodeje mobilních telefonů v Evropě* [online]. c2012 [cit. 15.1.2012]. Dostupné z <http://www.comscore.com/Press_Events/Press_Releases/2011/9>.
- [3] The Pragmatic Studio. *iOS Developers Roadmap* [online]. c2012 [cit. 15.1.2012]. Dostupné z <<http://www.pragmaticstudio.com/iphone-roadmap>>.
- [4] Adobe. *Adobe AIR 3* [online]. c2012 [cit. 17.2.2012]. Dostupné z <<http://www.adobe.com/cz/products/air.html>>.
- [5] Microsoft. *Microsoft Silverlight* [online]. c2012 [cit. 17.2.2012]. Dostupné z <<http://www.microsoft.com/cze/web/silverlight/default.aspx>>.
- [6] Microsoft. *XNA Developer Center* [online]. c2012 [cit. 17.2.2012]. Dostupné z <<http://msdn.microsoft.com/cs-cz/xna>>.
- [7] App Hub. *App Hub* [online]. c2012 [cit. 17.2.2012]. Dostupné z <<http://create.msdn.com/en-US/>>.
- [8] Microsoft. *Application Certification Requirements for Windows Phone* [online]. c2012 [cit. 17.2.2012]. Dostupné z <[http://msdn.microsoft.com/en-us/library/hh184843\(v=vs.92\).aspx](http://msdn.microsoft.com/en-us/library/hh184843(v=vs.92).aspx)>.
- [9] Microsoft. *Windows Phone Marketplace Test Kit* [online]. c2012 [cit. 17.2.2012]. Dostupné z <[http://msdn.microsoft.com/en-us/library/hh394032\(v=vs.92\).aspx](http://msdn.microsoft.com/en-us/library/hh394032(v=vs.92).aspx)>.
- [10] Nokia. *Carbide.c++* [online]. c2012 [cit. 17.2.2012]. Dostupné z <http://www.developer.nokia.com/Resources/Tools_and_downloads/Other/Carbide.c++/>.
- [11] Nokia. *Qt* [online]. c2012 [cit. 17.2.2012]. Dostupné z <<http://www.developer.nokia.com/Develop/Qt/>>.
- [12] Google. *Android developers* [online]. c2012 [cit. 17.2.2012]. Dostupné z <<http://developer.android.com/index.html>>.
- [13] MURPHY, Mark L. *Android 2*. Computer Press, a.s., 2011. Vydání první. ISBN 978-80-251-3194-7.

-
- [14] MEDNIEKS, Zigurd. *Programming Android*. O'Reilly Media, 2011. ISBN 978-1449389697.
- [15] Robotium Developers. *Robotium* [online]. c2012 [cit. 11.2.2012]. Dostupné z <<http://code.google.com/p/robotium/>>.
- [16] MARTIN, Robert C. *Čistý kód*. Computer Press, a.s., 2009. ISBN 978-80-251-2285-3
- [17] VALENTA, Lukáš. *Webové služby IS/STAG* [online]. c2012 [cit. 14.4.2012]. Dostupné z <<https://stag-ws.tul.cz>>.
- [18] HARDER, Rober W. *Base64* [online]. c2012 [cit. 16.4.2012]. Dostupné z <<http://iharder.net/base64>>.
- [19] WATSON, Gray. *ORMLite* [online]. c2012 [cit. 17.4.2012]. Dostupné z <<http://ormlite.com>>.
- [20] CellObject. *CellObject SQLite Browser* [online]. c2012 [cit. 18.4.2012]. Dostupné z <<http://www.questoid.com/Tools/CellObjectSQLiteXMLBrowser.aspx>>.
- [21] SMITH, Dave. *Android Recipes: A Problem-Solution Approach*. Apress, 2011. ISBN 978-1430234135.
- [22] GARGENTA, Marko. *Learning Android*. O'Reilly Media, 2011. ISBN 978-1-4493-9050-1.
- [23] PRIEGNITZ, Karsen. *Android Change Log* [online]. c2012 [cit. 26.4.2012]. Dostupné z <<http://code.google.com/p/android-change-log/>>.

A AndroidManifest.xml

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="cz.zcu.portal.android"
    android:versionCode="14"
    android:versionName="1.2.0">

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

    <uses-sdk android:minSdkVersion="4" android:targetSdkVersion="10"/>

    <application android:name=".gui.main.RozvrhAplikace"
        android:icon="@drawable/icon"
        android:label="@string/app_name"
        android:configChanges="orientation|keyboardHidden|locale">

        <activity android:name=".gui.main.Hlavni"
            android:noHistory="true">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
                <category android:name="android.intent.category.HOME" />
            </intent-filter>
        </activity>

        <activity android:name=".gui.main.RozvrhShow" />

        <activity android:name=".gui.main.OsGet" />

        <activity android:name=".gui.main.Nastaveni"
            android:label="@string/nastaveni"
            android:launchMode="singleTop" />

        <activity android:name=".gui.details.PredmetShow" />

        <activity android:name=".gui.details.StudentShow" />

        <activity android:name=".gui.details.UcitelShow" />

        <activity android:name=".gui.main.OAplikaci" />

        <activity android:name=".gui.main.VysledkyHledani" />

        <activity android:name="cz.zcu.portal.android.gui.widget.
            RozvrhovaAkceWidgetConfigure"
            android:label="@string/app_name_widget">
            <intent-filter>
                <action android:name="android.appwidget.action.APPWIDGET_CONFIGURE"/>
            </intent-filter>
        </activity>
    </application>
</manifest>
```

```
<receiver android:name=".gui.widget.RozvrhovaAkceWidgetProvider" >
  <intent-filter>
    <action android:name="android.appwidget.action.APPWIDGET_UPDATE" />
  </intent-filter>
  <meta-data android:name="android.appwidget.provider"
    android:resource="@xml/appwidget_info" />
</receiver>

<service android:name=".gui.widget.UpdateWidgetService" />

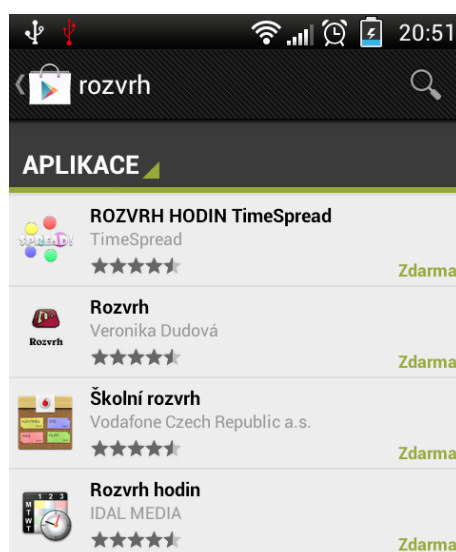
</application>
</manifest>
```


B Uživatelský manuál

Tento uživatelský manuál popisuje použití aplikace Rozvrh pro Android, která slouží studentům ZČU k zobrazení rozvrhu hodin a detailu jednotlivých předmětů.

Instalace

Nejjednodušší způsob, jak aplikaci nainstalovat, je pomocí aplikace Google Play. Spuštěte proto ve svém mobilním telefonu nebo tabletu tuto aplikaci a dejte vyhledat slovo „rozvrh“. Zobrazí se Vám výsledek hledání podobný obrázku 1.



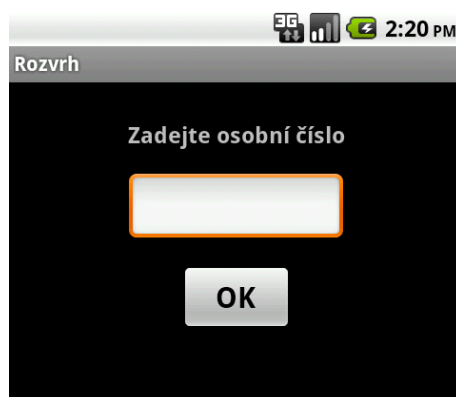
Obrázek 1: Výsledky hledání slova „rozvrh“ na Google Play

Zvolte aplikaci Rozvrh, pod kterou je podepsána autorka Veronika Dudová, a tu nainstalujte. Aplikace vyžaduje jen oprávnění pro síťovou komunikaci.

První spuštění

Po spuštění aplikace se Vám zobrazí přehled změn. Tento přehled se zobrazí vždy po spuštění nové verze aplikace a Vy tak můžete vidět, k jakým změnám v aplikaci došlo. Následně je zobrazena obrazovka pro zadání osobního čísla studenta (obr. 2), jehož rozvrh chcete mít k dispozici. Pro získání rozvrhu je nutné mít zapnuté připojení k internetu. Po zadání osobního čísla a kliknutí na tlačítko OK dojde ke stahování dat. Rychlost stahování je závislá na rychlosti připojení i aktuálnímu stavu využívaných webových služeb. Ke kompletnímu stažení data by ale mělo do jedné minuty. V případě chybně zadaného osobního čísla či chybě při datové komunikaci, je zobrazen chybový dialog.

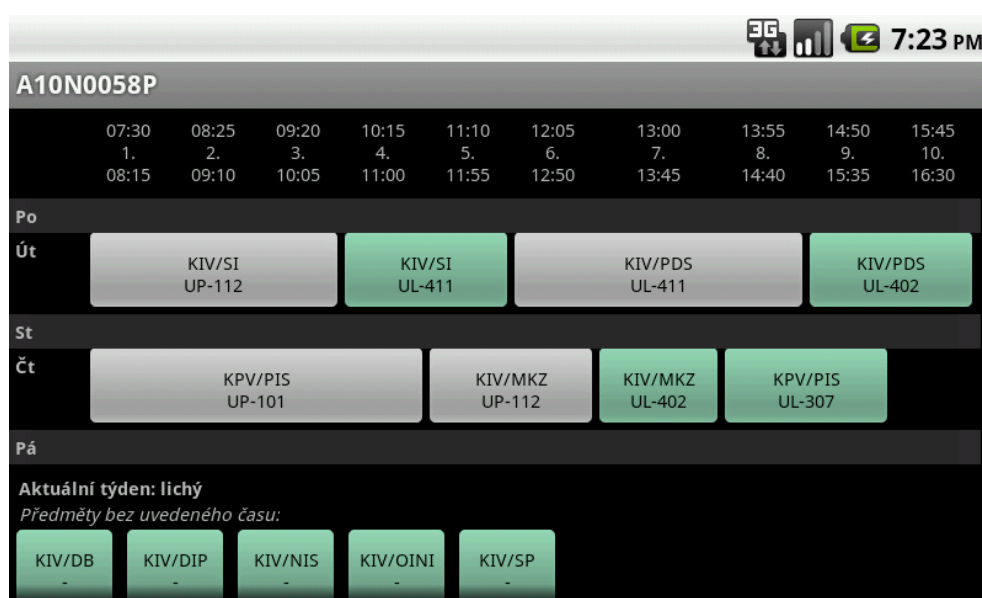
Po úspěšném stahování je zobrazena obrazovka s rozvrhem viz obr. 3.



Obrázek 2: Obrazovka pro zadání osobního čísla

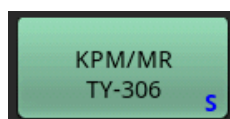
Rozvrh

Rozvrh obsahuje titulek s osobním číslem, hlavičku s číslem hodiny a časem, řádky s rozvrhovými akcemi, označení aktuálního týdne a seznam předmětů bez uvedeného času. Barvy jednotlivých rozvrhových akcí jsou nastaveny stejně jako

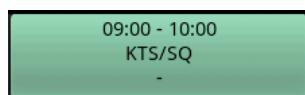


Obrázek 3: Ukázka zobrazení rozvrhu v aplikaci

na Portále ZČU, šedá tedy zobrazuje přednášku, tmavší zelená cvičení a světle zelená seminář. U akcí, které jsou jen sudý či lichý týden je tato skutečnost označena v pravém dolním rohu (obr. 4). U akcí, které nejsou podle přesně stanoveného času daného časovou řadou ZČU (typicky tělocviky) je čas uveden v horní části prvku (obr. 5).



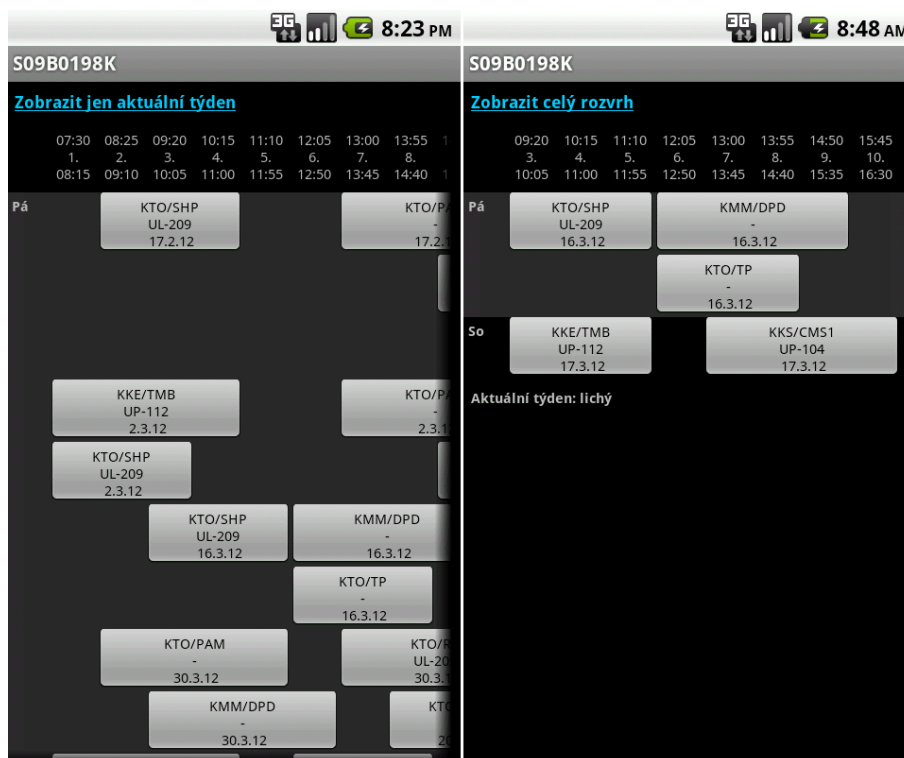
Obrázek 4: Ukázka předmětu v sudém týdnu



Obrázek 5: Ukázka předmětu mimo časovou řadu

Rozvrhy kombinovaného studia

U rozvrhů kombinovaného studia je u rozvrhových akcí zobrazen i datum konání a také je zde možnost přepínat mezi zobrazením celého rozvrhu nebo jen aktuálního týdne viz obr. 6.



Obrázek 6: Ukázka rozvrhu kombinovaného studia

Detail předmětů, učitelů a studentů

Aplikace umožňuje zobrazit i detailní informace o předmětech, učitelích a studentech. Detail o konkrétní rozvrhové akci i o samotném předmětu lze vyvolat kliknutím na tlačítko s danou rozvrhovou akcí v rozvrhu. Detail obsahuje název předmětu, status, místnost, semestr atd (obr. 7).

Údaje obsahují i odkaz na předmět na Courseware, kdy kliknutím na tento odkaz je spuštěn webový prohlížeč. Dalším klikatelným údajem je jméno vyuču-

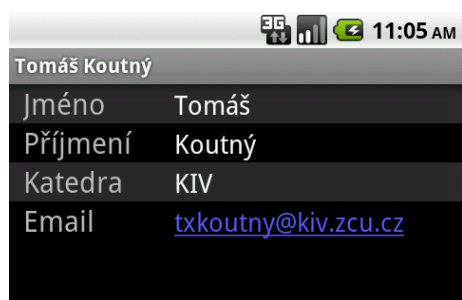


The screenshot shows a mobile application interface with a dark theme. At the top, there is a status bar with icons for signal strength, battery, and time (11:03 AM). Below the status bar, the text 'Předmět: KIV/PPR' is displayed. The main content is a list of course details, each on a separate line with a label on the left and a value on the right. The details include: Předmět (Paralelní programo), Statut (A), Místnost (UU-407), Semestr (Zimní), Druh (Přednáška), Týden (Jiný), Den (Úterý), Začátek (13:55), Konec (17:25), Datum od (20.9.2011), Datum do (13.12.2011), Vyučující (Tomáš Koutný), Courseware * (https://portal.zcu.c), a note (* funguje jen u předmětů, které mají stránku r), Kredity (6), and Požadavky (Zápočet: Samostatn, (bodové hodnocen, třeba 25 bodů. Mez, stránku v předmětů).

Předmět: KIV/PPR	
Předmět	Paralelní programo
Statut	A
Místnost	UU-407
Semestr	Zimní
Druh	Přednáška
Týden	Jiný
Den	Úterý
Začátek	13:55
Konec	17:25
Datum od	20.9.2011
Datum do	13.12.2011
Vyučující	Tomáš Koutný
Courseware *	https://portal.zcu.c
* funguje jen u předmětů, které mají stránku r	
Kredity	6
Požadavky	Zápočet: Samostatn (bodové hodnocen třeba 25 bodů. Mez stránku v předmětů

Obrázek 7: Ukázka detailu předmětu

jícího, které po kliknutí zobrazí detail učitele (obr. 8). Zde je důležitým údajem emailová adresa, která po kliknutí umožní poslat email na tuto adresu z námi zvolené aplikace poštovního klienta na zařízení.

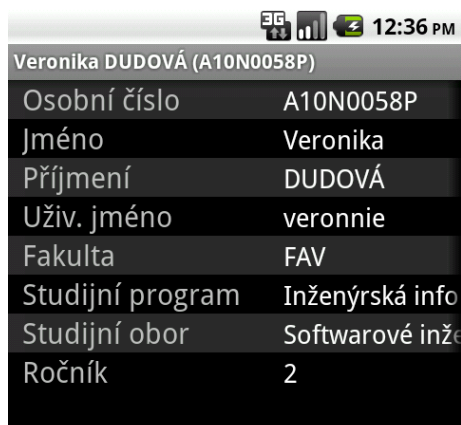


The screenshot shows a mobile application interface with a dark theme. At the top, there is a status bar with icons for signal strength, battery, and time (11:05 AM). Below the status bar, the text 'Tomáš Koutný' is displayed. The main content is a list of teacher details, each on a separate line with a label on the left and a value on the right. The details include: Jméno (Tomáš), Příjmení (Koutný), Katedra (KIV), and Email (txkoutny@kiv.zcu.cz).

Tomáš Koutný	
Jméno	Tomáš
Příjmení	Koutný
Katedra	KIV
Email	txkoutny@kiv.zcu.cz

Obrázek 8: Ukázka detailu vyučujícího

Detail studenta je možno zobrazit přes menu aplikace. Co je obsahem této obrazovky je vidět na následujícím obrázku 9.

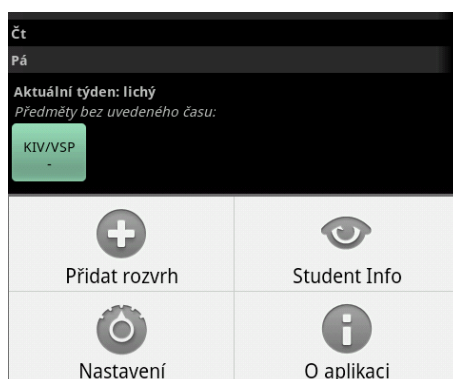


Veronika DUDOVÁ (A10N0058P)	
Osobní číslo	A10N0058P
Jméno	Veronika
Příjmení	DUDOVÁ
Uživ. jméno	veronnie
Fakulta	FAV
Studijní program	Inženýrská info
Studijní obor	Softwarové inž
Ročník	2

Obrázek 9: Ukázka detailu studenta

Menu aplikace

Menu obsahuje položku pro přidání dalšího rozvrhu, pro nastavení aplikace a pro zobrazení informací o aplikaci (obr. 10). Položka pro zobrazení informací o studentovi se zobrazí jen přihlášeným uživatelům (viz dále Nastavení).



Obrázek 10: Menu aplikace

Nastavení

Nastavení aplikace lze vyvolat přes příslušnou položku v menu. Ukázka obrazovky s nastavením viz obr. 11

Nastavení aplikace umožňuje:

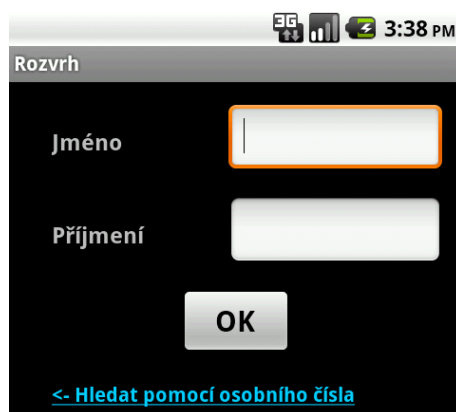
- přepínat mezi staženými rozvrhy studentů
- měnit semestr
- zvolit jazyk aplikace (čeština/angličtina)
- zadat Orion přihlašovací údaje pro hledání studentů podle jména a pro zobrazení detailu studentů
- aktualizovat a mazat rozvrhy



Obrázek 11: Obrazovka s nastavením aplikace

Hledání podle jména a příjmení

Pokud máte v nastavení vyplněny přihlašovací údaje, máte možnost vyhledávat další studenty podle jejich jména a příjmení. Při přidávání rozvrhu (položka *Přidat rozvrh* v menu) je nabídnuta možnost využít této funkce. Po kliknutí na text *Hledat pomocí jména a příjmení* je zobrazena obrazovka pro zadání jména a příjmení (obr. 12) studenta. Vyhledávat je možné i podle samotného jména či příjmení a pro hledání platí stejná pravidla jako při použití stejné funkce na Portále ZČU.



Obrázek 12: Obrazovka pro hledání studentů podle jména

Jak může vypadat výsledek hledání je vidět na obrázku 13. U každého nalezeného studenta je zobrazeno jméno, osobní číslo a další detailní údaje na rozkliknutí. Kliknutím na tlačítko se symbolem plus (+) dojde ke stažení dat o rozvrhu, stejně jako v případě zadání osobního čísla, a k jeho následnému zobrazení.



Obrázek 13: *Obrazovka s výsledkem hledání studentek s jménem Veronika a příjmením začínajícím na D*

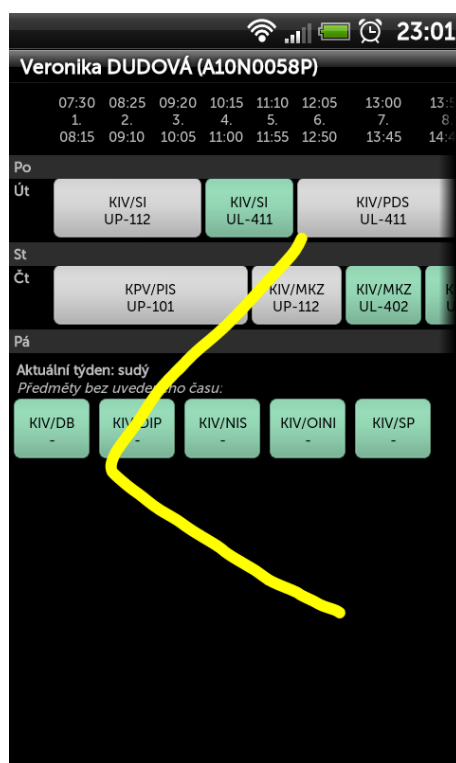
Gesta

Aplikace podporuje gesta pro snadné přepínání zobrazeného rozvrhu. Jedním tahem tak můžete přepínat mezi uloženými rozvrhy. Gesto má tvar levé či pravé ostré závorky, podle toho na který rozvrh se chceme přepnout. Ukázka gesta je na obrázku 14, kde je barva čáry úmyslně nastavena na žlutou, v samotné aplikaci je barva gesta průhledná.

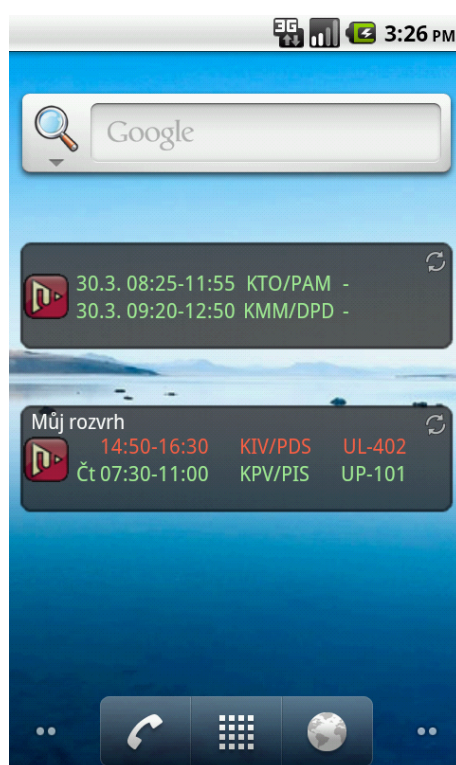
Widget

K aplikaci je také k dispozici widget, který lze umístit na domovskou obrazovku zařízení. Widget zobrazuje dvě nejbližší rozvrhové akce. Můžete tak pohyblivým pohledem na displej vidět, kdy a kde máte další hodinu. Widgetů můžete mít na ploše víc, ke každému můžete v rámci jeho nastavení přidat popisek. Ukázka widgetů viz obrázek 15.

Červeně jsou zobrazeny právě probíhající akce, zeleně ty, které teprve nastanou. V případě, že akce bude probíhat daný den, tak není uvedena zkratka dne. U rozvrhů kombinovaného studia je uveden místo dne datum konání akce.



Obrázek 14: Gesto pro přepnutí na předchozí rozvrh



Obrázek 15: Widgety zobrazující rozvrhové akce