UNIVERZITA PARDUBICE
Fakulta elektrotechniky a informatiky

Mobilní aplikace pro vyhledání nejbližší recyklační stanice

Roman Holomek

Bakalářská práce 2012

Univerzita Pardubice
Fakulta elektrotechniky a informatiky
Akademický rok: **2011/2012**

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení:	**Roman Holomek**
Osobní číslo:	**I08053**
Studijní program:	**B2646 Informační technologie**
Studijní obor:	**Informační technologie**
Název tématu:	**Mobilní aplikace pro vyhledání nejbližší recyklační stanice**
Zadávající katedra:	**Katedra informačních technologií**

Zásady pro vypracování:

Anotace:
Cílem práce je návrh a realizace mobilní aplikace nad platformou Android pro vyhledání a zobrazení nejbližší cesty k recyklační stanici.
Teoretická část:
V této části budou zmíněny algoritmy pro hledání nejkratší cesty a potřebné datové struktury. Dále bude popsán operační systém Android a vhodná vývojová prostředí pro jeho programování. Bude zde uvedena technologie XML a popsány důležité vlastnosti jazyka Java.
Implementační část:
V této části bude pomocí vybraných technologií navržena a realizována mobilní aplikace, která bude schopná na základě zadaných lokalit z Google Maps a pomocí GPS vyhledat a zobrazit nejkratší cestu k recyklačnímu středisku nebo sběrnému místu.

Rozsah grafických prací:

Rozsah pracovní zprávy:

Forma zpracování bakalářské práce: **tištěná/elektronická**

Seznam odborné literatury:

Wallace J. Android Apps for Absolute Beginners. Apress, ISBN 978-1-4302-3446-3,2011.
Mednieks Z., Dornin L. Programming Android. OReilly, ISBN13: 978-0596521479, 2010.

Vedoucí bakalářské práce: **Ing. Zdeněk Šilar**
Katedra informačních technologií

Datum zadání bakalářské práce: **16. prosince 2011**
Termín odevzdání bakalářské práce: **11. května 2012**

L.S.

prof. Ing. Simeon Karamazov, Dr.
děkan

Ing. Lukáš Čegan, Ph.D.
vedoucí katedry

V Pardubicích dne 30. března 2012

Prohlášení autora

Prohlašuji, že jsem tuto práci vypracoval samostatně. Veškeré literární prameny a informace, které jsem v práci využil, jsou uvedeny v seznamu použité literatury.

Byl jsem seznámen s tím, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorský zákon, zejména se skutečností, že Univerzita Pardubice má právo na uzavření licenční smlouvy o užití této práce jako školního díla podle § 60 odst. 1 autorského zákona, a s tím, že pokud dojde k užití této práce mnou nebo bude poskytnuta licence o užití jinému subjektu, je Univerzita Pardubice oprávněna ode mne požadovat přiměřený příspěvek na úhradu nákladů, které na vytvoření díla vynaložila, a to podle okolností až do jejich skutečné výše.

Souhlasím s prezenčním zpřístupněním své práce v Univerzitní knihovně.

V Pardubicích dne 14. 4. 2012 Roman Holomek

Poděkování

Na prvním místě bych rád poděkoval mému vedoucímu této práce, Ing. Zdeňku Šilarovi, který byl po celou dobu velice vstřícný a ochotný. Dále bych chtěl poděkovat firmě Tvář Webu s.r.o., která mi poskytla potřebná data, design pro celou aplikaci a všichni její zaměstnanci byli nanejvýše vstřícní k jakékoliv konzultaci o této práci.

Anotace

Tato bakalářská práce se zabývá problematikou programování pro platformu Android. Jsou zde popsána vývojová prostředí, jejich vlastnosti, výhody a nevýhody, dále také srovnání s ostatními platformami pro chytré telefony a výhody a nevýhody tohoto programování. Jako ukázka je k práci naprogramována funkční aplikace, která je určena k vyhledávání nejbližších recyklačních míst podle GPS ve vašem zařízení. Aplikace využívá knihovny Google Maps a je naprogramovaná v programovacích jazycích Java a XML.

Klíčová slova

Android, mobilní aplikace, recyklování

Title

Repak – Android application for searching the shortest way to recycling stations

Annotation

This thesis is about Android programming. Development environments are described here, their properties, advantages and disadvantages, comparison with other platforms for smart phones and advantages and disadvantages of this programming. As example is programmed working application, which is used to searching nearest recycling places by GPS in your device. Application uses Google Maps libraries and is programmed in Java and XML.

Keywords

Android, mobile application, recycling

Obsah

Seznam obrázků

1 Úvod

V dnešní mobilní době zažívá obrovský rozmach svět chytrých telefonů. Za posledních pár let šla technologie v této oblasti velice dopředu a dnes už můžete pomocí mobilního telefonu dělat téměř cokoliv od správy emailů, natáčení HD videí, focení 3D fotek, sledování televizního vysílání až po využití telefonu jako GPS navigace, či hraní plnohodnotných 3D her s vysokým rozlišením. Jejich využití je opravdu velice široké a z toho důvodu samozřejmě také roste zájem o aplikace pro tyto platformy.

Když už je řeč o platformách, tak na nynějším trhu mají největší podíl dvě a to iOS od společnosti Apple, který je navržen pouze pro produkty této společnosti, jako jsou například iPhone či iPad, a Android od společnosti Google, který běží v největší míře na přístrojích HTC, Samsung nebo Sony Ericsson. Dále se na trhu objevují i jiné platformy, ale již v daleko menší míře, jako například PalmOS, Symbian OS a Windows Phone.

Pro vypracování této bakalářské práce jsem si po zvážení vybral aplikaci pro Android platformu, dle mého názoru má momentálně nejlépe nakročeno k tomu, aby tento trh časem ovládla. Soudím tak hlavně kvůli její flexibilitě a rychlosti vývoje. Android má velikou výhodu v tom, že pracuje téměř na jakémkoliv zařízení, takže jeho rozšiřování jde velice rychle kupředu.

Jako ukázka vývoje pro Android jsem naprogramoval aplikaci Repak, která souží k vyhledávání nejbližších recyklačních míst a dokáže k nim zobrazit nejkratší cestu. Tato aplikace byla mojí první aplikací pro tuto platformu vůbec, takže s ní bylo pro mě spojeno i rozhodování mezi vývojovými prostředími a kompletní studování vývoje pro Android od základů.

2 Android

2.1 Historie

Společnost Android Inc. byla založena v roce 2003 v Kalifornii ve městě Palo Alto čtveřicí zakladatelů Richem Minerem, Andym Rubinem, Chrisem Whitem a Nickem Searsem. V srpnu 2005 jí odkoupila společnost Google Inc.

Po odkoupení společností Google Inc. začal Andy Rubin se svým týmem pracovat na vývoji operačního systému pro mobilní telefony a ostatní mobilní zařízení na bázi Linuxu. V prvopočátcích to vypadalo, že se společnost Google snaží pouze o vytvoření vlastního mobilního telefonu. V roce 2005 bylo vytvořeno uskupení Open Handset Alliance, toto uskupení zahrnovalo společnosti, které se zabývaly výrobou mobilních telefonů, čipů a mobilních aplikací jako jsou například HTC, Intel, LG, Google, Qualcomm, Samsung a dalších 26 společností. Toto uskupení bylo založeno, aby vyvinulo otevřený standard pro mobilní zařízení. Hned po vytvoření toto uskupení oznámilo svůj první produkt, platformu Android postavenou na jádře Linux verze 2.6. Také bylo oznámeno, že tato platforma bude v budoucnu použitelná na tisících různých mobilních zařízeních. Tyden po tomto oznámení byl pro vývojáře vydán první Android SDK pod licencí open-source.[3]

Prvního komerční telefon s Androidem 1.0 přišel na trh v USA v roce 2008 a byl to telefon T-Mobile G1 od HTC. Poté následovaly další zařízení a v roce 2009 už jich bylo více než 20. V roce 2009 byla také vydána nová verze Android 1.5 s označením Cupcake, která podporovala řadu nových funkcí jako upload videí na Youtube nebo významně rychlejší vyhledávání GPS polohy s pomocí AGPS. V ten samý rok byly vydány ještě hned další dvě verze – Android 1.6 Donut, s funkcí převodu textu na řeč a integrací fotoaparátu a kamery, a Android 2.0 Eclair, který výrazně zlepšil podporu synchronizace a znatelně vylepšil webový prohlížeč. V roce 2010 byla vydáná verze Android 2.2 Froyo s novým typem widgetů na domovské obrazovce, možností využívat telefon jako Wi-Fi hotspot a podporou vícejazyčných klávesnic, dále v tomto roce byla vydána verze Android 2.3 Gingerbread, která obsahuje výrazné změny v uživatelském prostředí zaměřené na zjednodušení a zrychlení celého systému.

Po Android verzi 2 přišla v roce 2011 verze Android 3.0 Honeycomb, tato nová verze byla navržená hlavně pro tablety a zařízení s větší obrazovkou, měla přepracovaný multitasking, systém notifikací, jinou způsob modifikací domovské obrazovky a widgetů.

Koncem roku 2011 byla vydána zatím poslední verze Android 4.0 Ice Cream Sandwich, Android s touto verzí udělal jeden veliký krok, a to, že spojil svět mobilů se světem tabletů.

2.2 Charakteristika

Tato platforma funguje na systému takzvaných „aktivit“, tyto aktivity jsou na sobě naprosto nezávislé a spočívá v nich několik obrovských výhod Androidu. Hlavní výhodou je, že se můžete z jedné aktivity kdykoliv přepnou na libovolnou aktivitu jak v rámci vaší aplikace, tak i do jiné kdekoliv v systému. Při přepnutí je následující aktivitě poslán takzvaný „intent“. V rámci tohoto „intentu“ můžete předávat mezi aktivitami různé parametry a navíc může obsahovat i informace o animacích pře přepínání a spoustu dalších volitelných parametrů.

2.3 Programování pro android

Než programátor začne programovat pro Android, musí se rozhodnout, v jakém programovacím jazyce bude aplikace vyvíjet. Možností je buďto jazyk C, nebo Java. Jazyk C má výhodu v tom, že aplikace napsané v něm mohou pracovat o něco rychleji, ale rozdíl mezi C a Javou by se mohl projevit až při programování náročných 3D aplikací nebo takových, kde je zapotřebí velice rychlý, přímý přístup k paměti. Na toto omezení, které je asi jedinou nevýhodu Javy, moje aplikace zcela jistě nenarazí, takže jsem si bez většího přemýšlení vybral právě jazyk Java a tím se i bude nadále tato práce zabývat.

Pro zprovoznění Androidu na našem systému musíme mít tedy nainstalovanou pokud možno aktuální verzi JDK, což je zkratka pro Java Development Kit, pro správný chod Javy. K tomu musíme dále nainstalovat Software Development Kit, který je pro vývoj androidních aplikací nezbytný a navíc nám poskytne řadu velice užitečných utilit, některé z nich si dále přiblížíme.

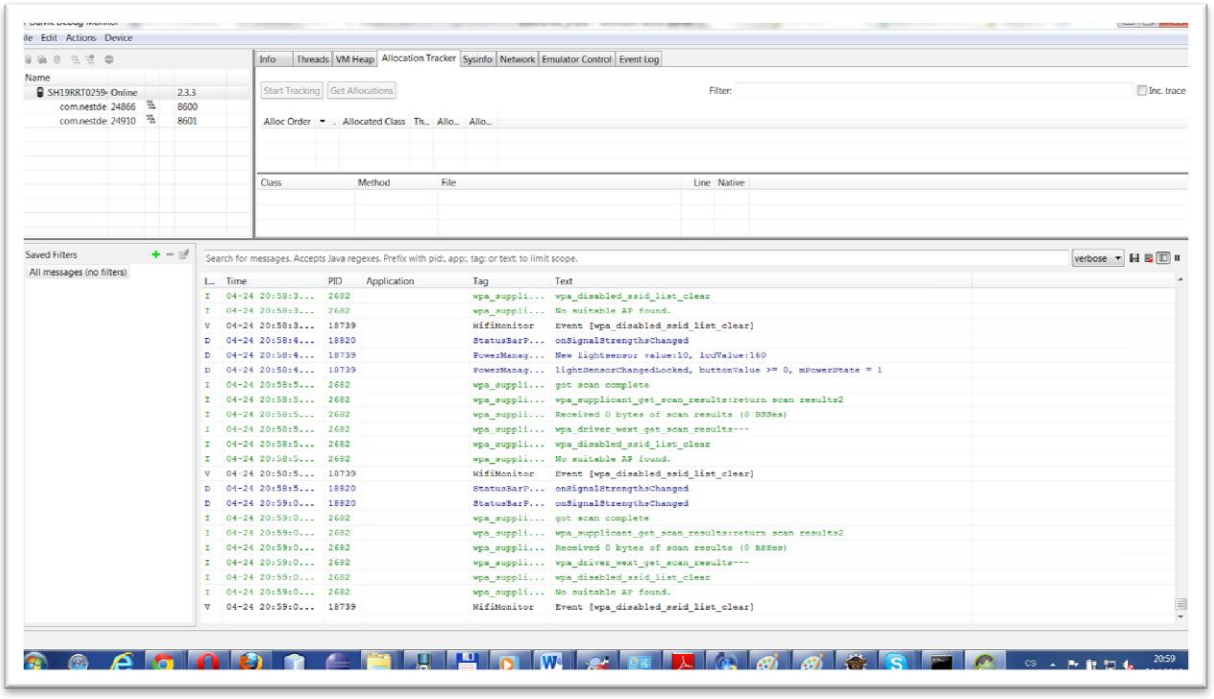
2.3.1 Dalvik Debug Monitor Server (DDMS)

Tento, pro vývojáře aplikací pro Android, naprosto nezbytný program, nám umožňuje sledovat téměř vše, co se na našem zařízení se systémem Android děje. Díky vestavěnému modulu LogCat, můžeme sledovat veškeré procesy, které probíhají na připojeném zařízení v reálném čase. Umožňuje nám také filtrovací funkce, díky kterým můžeme na výstupu sledovat například jen události, které probíhají přímo v rámci naší aplikace. Na obrázku obr.1 můžete vidět ukázku prostředí DDMS i s výstupy z připojeného zařízení.

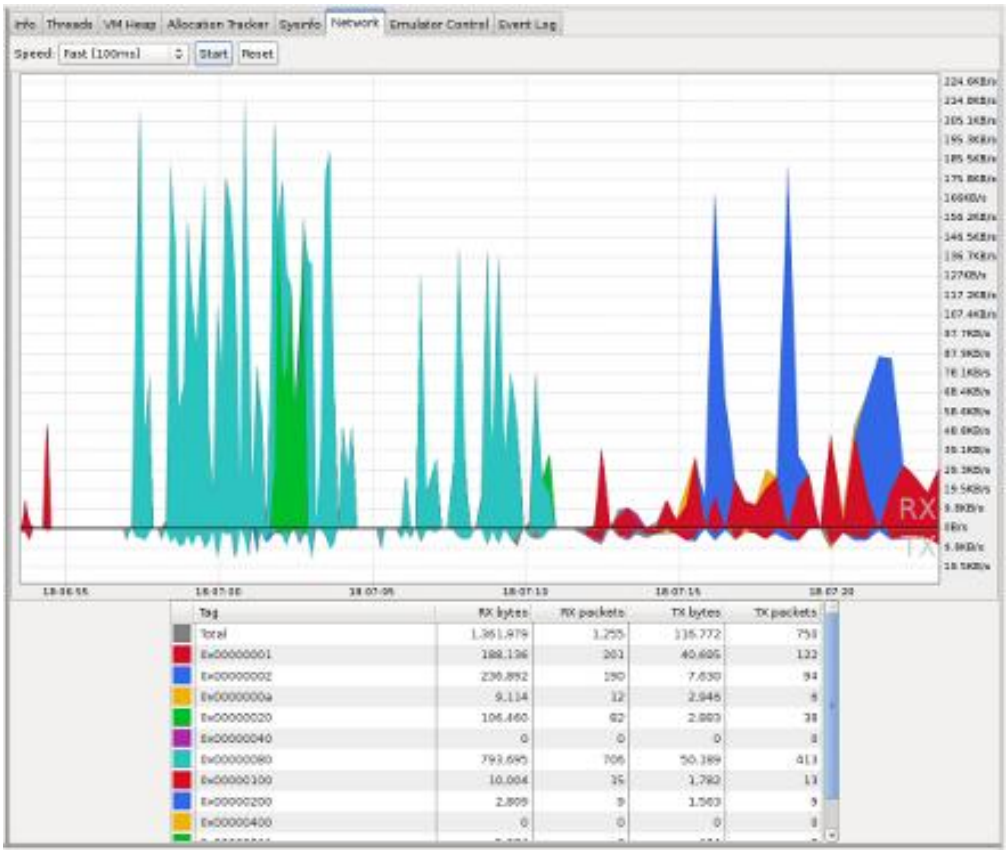
DDMS nabízí velikou škálu dalších funkcí, dokáže nasimulovat libovolnou polohu pro ladění lokalizačních funkcí, dokáže pořídit obrázek obrazovky z připojeného zařízení, prohlížení alokované paměti, vytvořených objektů, kontrolu paralelně běžících vláken nebo také sledování síťové komunikace. Ukázku průběhu pozorování síťové komunikace můžete vidět na obrázku obr.2 níže.

2.3.2 9-patch images

Další velice užitečná utilita je program pro dělání takzvaných 9-patch obrázků. Jedná se o speciální formát PNG souboru, díky němuž můžeme v aplikaci jedno tlačítko použít s různými velikostmi a délkami stran bez deformace. Jedná se v podstatě o to, že v transparentním PNG obrázku nakreslíme černou čárou o síle jeden pixel na každé straně tlačítka a tím vyznačíme plochy, které se mají roztahovat. Zbytek obrázku zůstane nedotčený, díky čemuž se nám nezdeformují rohy u zaoblených tlačítek, i když je natáhneme pouze v jednom směru. K vytváření takových souborů slouží nástroj draw9patch, který nám nabízí androidní SDK. Po vytvoření takového obrázku se tento musí uložit ve formátu jmeno_souboru.9.png a Android sám pozná, že se jedná o obrázek typu 9-patch.



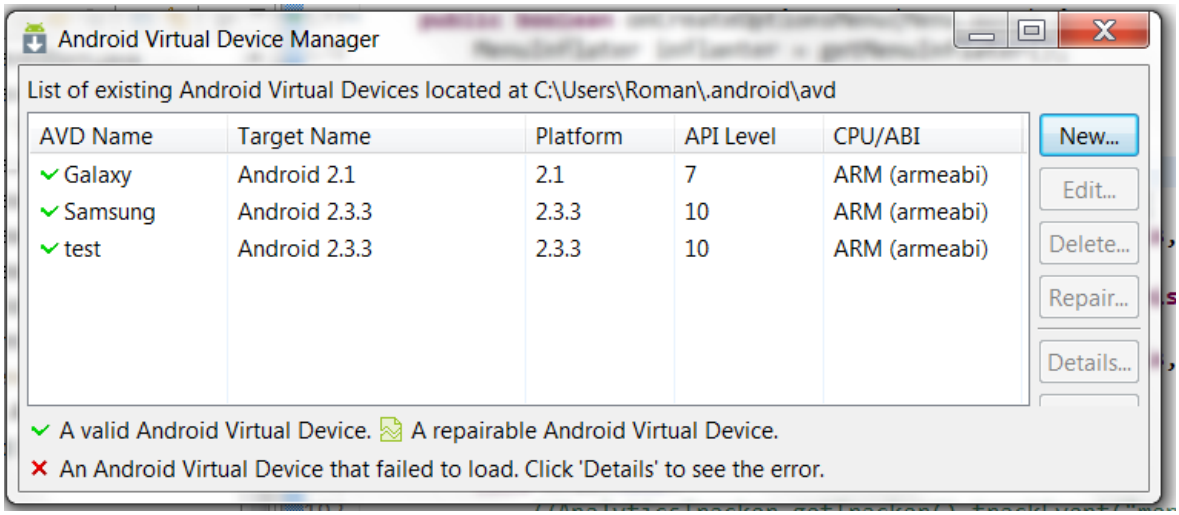
Obrázek 1 – Ukázka prostředí DDMS



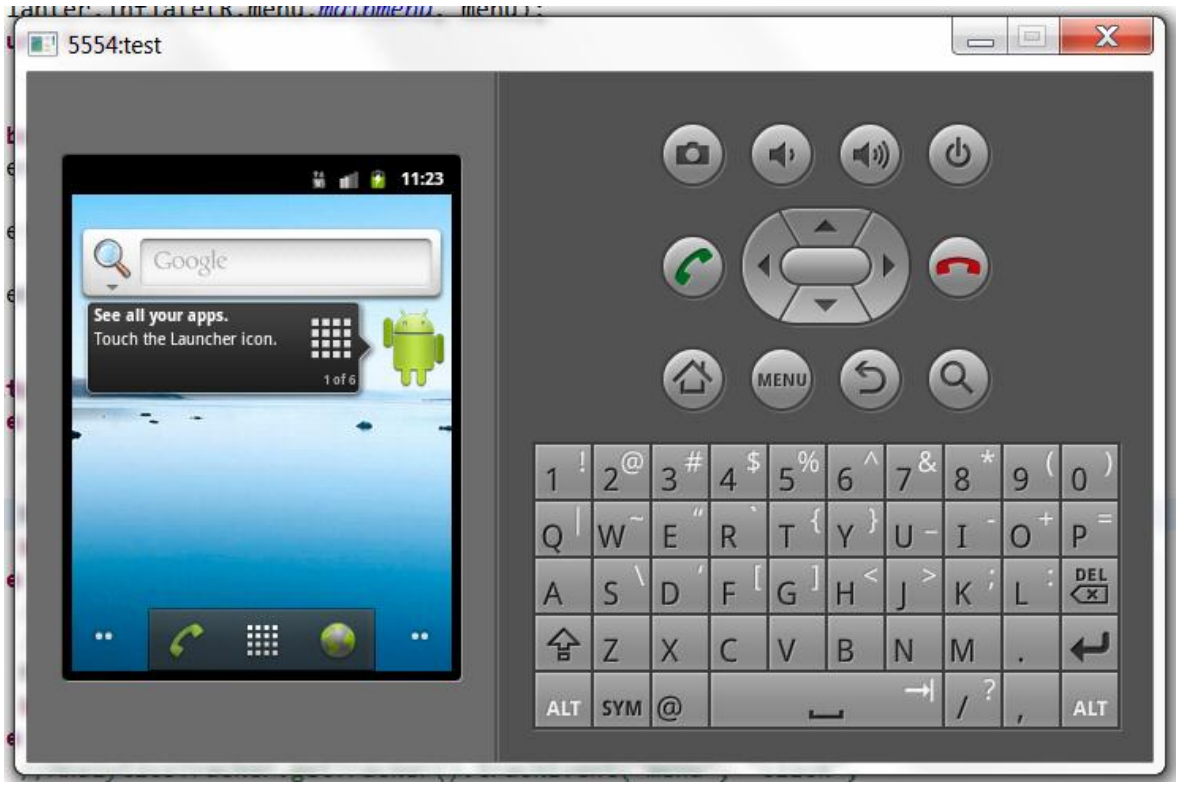
Obrázek 2 – Ukázka sledování síťové komunikace v DDSM

2.4 Emulátory

Pokud chce někdo vyvíjet aplikace opravdu profesionálně, neobejde se bez použití emulátorů. Emulátory dokážou nasimulovat jak hardware, tak i software libovolného zařízení. V případě Androidu jsou emulátory nejdůležitější pro testování aplikací s různými typy obrazovek, v tomto má programování pro Android velikou nevýhodu oproti programování aplikací pro iPhone. Zatímco iPhone je jedno zařízení s jedním rozlišením, popřípadě se dvěma když vezmeme v potaz i nový iPhone 4S, který má rozlišení dvojnásobné, Android podporuje celou škálu zařízení s velkým rozptylem nejrůznějších rozlišení, což je v rámci programování univerzální aplikace celkem veliký problém. Emulátor pro systém Android se nazývá Android Virtual Device nebo také ve zkratce AVD. K jeho spuštění a nastavení se používá AVD Manager. V něm si můžeme nastavit požadované rozlišení a typ obrazovky, kapacitu paměťové karty, verzi Androidu, kterou zařízení simuluje a další. Na obrázku obr.3 můžete vidět AVD Manager a na obrázku obr.4 je zobrazeno samotné AVD.



Obrázek 3 – AVD Mannager



Obrázek 4 – Ukázka AVD

2.5 Vývojová prostředí

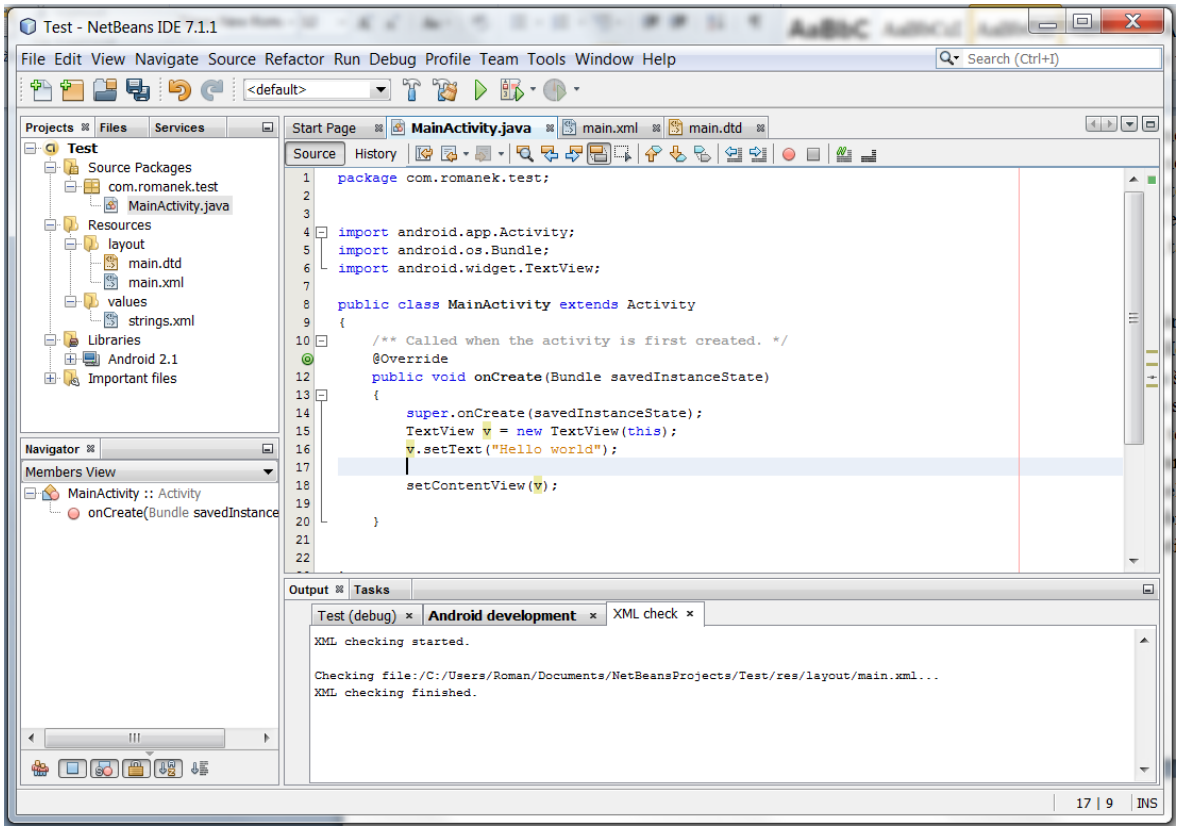
Dalším nezbytným krokem, který musí vývojář udělat, než začne aplikace pro Android vyvíjet, je výběr vývojového prostředí. Tento krok je velice důležitý pro další práci, protože kvalitní vývojové prostředí vám může usnadnit veliké množství práce, jako

například automaticky nabízet predikci kódu, generovat funkce, nové třídy, spravovat importy knihoven a spoustu dalších.

2.5.1 NetBeans IDE

Netbeans IDE je skvělé vývojové prostředí od firmy Oracle pro programovací jazyk Java, C++, PHP a další. Nabízí veliké množství funkcí a možností, včetně možnosti doinstalovat plug-in pro vývoj aplikací pro Android. Bohužel je tento plug-in neoficiální a není podporovaný společností Google, z čehož vyplývá, že pokud se rozhodnete pro vývoj v tomto prostředí, čeká vás celá řada problémů. Hlavním problémem je, že kompletní vývojářská podpora od Googlu pro Android s prostředím NetBeans nepočítá, takže pokud se dostanete do úzkých, tak si budete velice často muset poradit sami.

Instalace plug-inu se mi z nějakého, pro mě dosud neznámého, problému podařila úspěšně až na třetí pokus. Po nainstalování jsem si zkusil klasický program „Hello world“ a po buildu aplikace se nic nestalo, takto jsem to zkoušel několikrát a až asi na šestý pokus se mi aplikace rozeběhla na mém připojeném mobilním zařízení. Tímto jsem si ověřil, že v prostředí NetBeans skutečně aplikace pro Android vyvíjel lze. Po chvíli testování se ukázalo, že toto vývojové prostředí pracuje vcelku rychle, ale na druhou stranu není pro Android příliš vybaveno, absence GUI pro stavění XML elementů je vcelku citelný nedostatek, také jsem se nikde nedohledal vestavěného AVD Manageru. Na obrázku obr.5 můžete vidět ukázku kódu androidní aplikace napsané v tomto vývojovém prostředí.



Obrázek 5 – Vývojové prostředí NetBeans

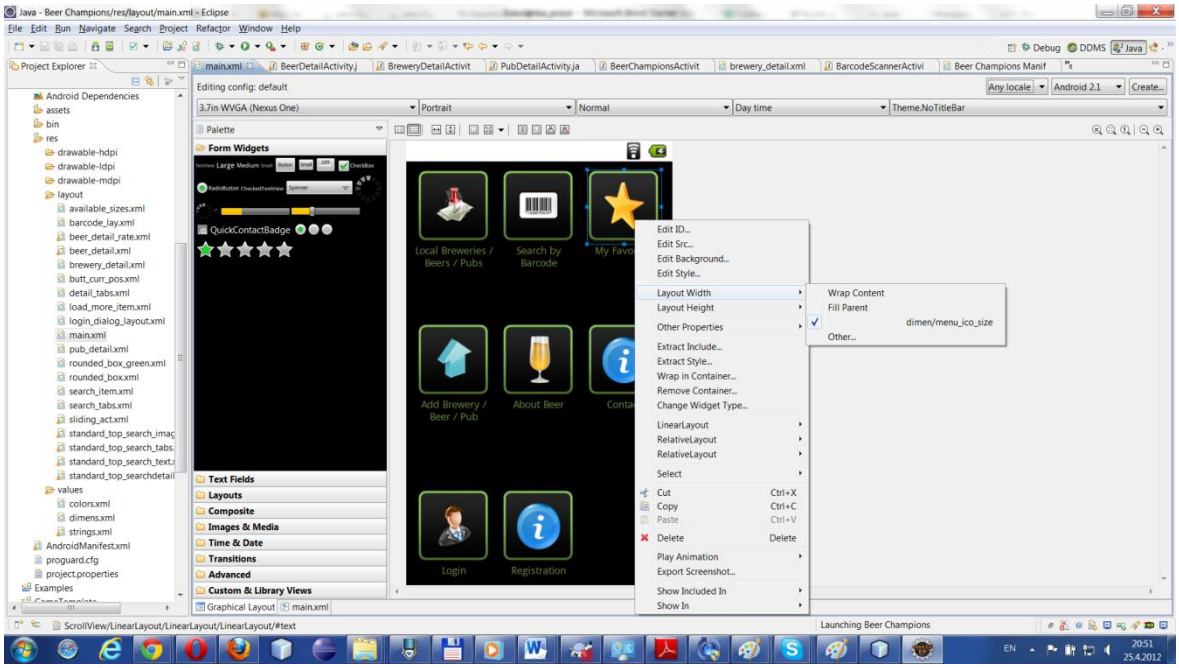
2.5.2 Eclipse

Dalším vývojovým prostředím je Eclipse. Opět se jedná o vývojové prostředí zejména pro programovací jazyk Java, ale oproti již zmiňovanému prostředí NetBeans má toto vývojové prostředí jednu velikou výhodu v tom, že je plně jako jediné podporováno společností Google pro programování aplikací pro Android.

Instalace Androidu do tohoto prostředí spočívá opět v instalaci plug-inu, v případě Eclipsu se tento plug-in nazývá ADT, neboli Android Development Tools. Instalace v mém případě proběhla naprosto bez problému. Dále je potřeba si doinstalovat AVD Manager a udělat pár menších nastavení, v čemž také nebyl nejmenší problém. Po instalaci stačí jen založit nový projekt pro Android a spustit ho, Eclipse hned po buildu aplikace vyhledá všechny zařízení s Androidem připojeným k vašemu počítači a nabídne vám je v seznamu, pokud žádné nenalezne, tak vám umožní přejít do AVD Manageru a zde virtuální zařízení nastavit a spustit. Už v tomto jsem hned v prvních chvílích viděl veliké výhody oproti výše probíranému prostředí NetBeans. Součástí ADT je také DDMS, takže můžeme hned po instalaci zařízení také aplikaci ladit.

Velikou výhodou je GUI, neboli grafické prostředí, pro stavění struktury každé aktivity v XML. Toto GUI pracuje vcelku spolehlivě, i když po nějakém čase programování pro Android od něj většina vývojářů upustí a radši si píše XML kód ručně,

protože při skládání složitějších struktur začíná být v tomto velice nepřehledné a také nedostačující. Ukázku tohoto prostředí můžete vidět na obrázku obr.6.



Obrázek 6 – Grafické prostředí v Eclipsu

2.5.3 Ostatní vývojová prostředí

Jiná vývojová prostředí než dvě výše zmíněná prakticky nestojí za zmínku. Aplikace pro Android se dají samozřejmě psát i v poznámkovém bloku a kompilovat přes příkazovou řádku, ale v praxi je takovéto programování nepoužitelné a nedokážu si představit jediný rozumný důvod k tomu, aby se vývojář ubíral touto cestou vývoje.

2.5.4 Zhodnocení vývojových prostředí

Jak už bylo napsáno výše, při programování pro Android má vývojář prakticky jen dvě možnosti, v kterém vývojovém prostředí bude vyvíjet, a těmi jsou Eclipse a NetBeans. Já se rozhodl jednoznačně pro Eclipse, důvodů bylo hned několik. Téměř všechny fóra a tutoriály, které jsem navštívil, se odkazují na programování v Eclipsu, je to nejspíše hlavně dáno tím, že samotná společnost Google toto prostředí podporuje. Oproti NetBeans je sice o něco pomalejší, ale pohodlností programování, ladícími funkcemi a grafickým prostředím pro skládání XML to zcela napravuje. Celá tato práce je tedy vyvíjena ve vývojovém prostředí Eclipse.

2.6 Výhody a nevýhody

Na závěr této kapitoly bych rád popsal výhody a nevýhody Androidu oproti největšímu konkurentu, čímž je iOS pro iPhone. Pro mě největší výhoda je to, že pro Android je možnost programování v jazyce Java, aplikace pro iOS se programují v jazyce Objective-C, který je podle mě až zbytečně nepřehledný a složitý na programování, i když tohle může být určitě věcí názoru, pokud někdo preferuje jazyk C, může pro něj být toto výhoda spíše pro iOS. Android má oproti svému konkurentovi celkem výhodu v psaní layoutů ve formátu XML, kterým můžete ovlivňovat kterýkoliv element ve vaší aktivitě, toto zaručuje velikou flexibilitu při stavění aplikací, máte veliké možnosti ve stylování jakýchkoliv komponent ať už systémových nebo vámi vytvořených. Oproti tomu má iOS určitou nevýhodu, protože má řadu systémových komponent, jako například přepínače tabů, defaultně nastavených a je jen málo možností, jak je změnit do takové podoby, kterou byste zrovna potřebovali. Systému iOS se na druhou stranu nemůže odepřít to, že oproti Androdu ještě pořád funguje o něco plynuleji a s menší odezvou, toto je určitě do značné míry ovlivněno hardwarem daného zařízení, ale i když jsem porovnával práci na iPhonu a novém HTC, tak zde pořád znatelný rozdíl je.

3 Aplikace Repak

3.1 Seznámení s aplikací

Repak je mobilní aplikace pro vyhledávání nejbližších recyklačních míst, jejich zobrazení na mapě a vykreslení nejbližší cesty k nim. Celá aplikace je spravována přes webové prostředí a to umožňuje aplikaci dynamicky měnit bez potřeby nových aktualizací v Android Marketu. Veškerá data jsou poskytnuta od společnosti Tvář Webu s.r.o., která tato data používá pro spravování webových stránek jedné nejmenované irské firmy. Data jsou pro mě zpřístupněna přes několik skriptů na serveru, na které potom posílá aplikace požadavek na server s požadovanými parametry a jako odpověď dostává XML s daty.

3.1.1 Parametry vyhledávání

Aplikace umožňuje zadávání různých parametrů pro vyhledávání, tyto parametry potom ovlivní výsledek vyhledávání recyklačních míst. Parametry jsou zde zpřístupněny tyto:

- Typ recyklačního místa (Facility type) – zde jsou 3 druhy míst spojených s recyklací: Sběrné místo (Bring Bank), Civilní Recyklační Místo (Civic Amenity Centre), Recyklační Středisko (Recycling Centre)
- Lokalita (Local Authority Area) – jelikož data jsou poskytnuta od irské společnosti, tak zde lze vybrat pouze irské lokace
- Klíčové slovo (Keyword) – standardní klíčové slovo
- Materiál (Material) – výběr materiálku k recyklaci, po výběru nějakých materiálů budou ve výsledku hledání pouze místa, kde přijímají dané druhy materiálů
-

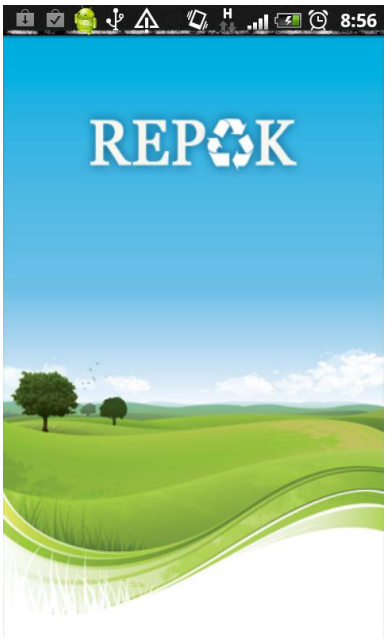
3.2 Struktura

Jak už bylo popsáno výše, aplikace je postavená tak, aby ji bylo možné z velké části spravovat pomocí webového rozhraní. Veškerá data jsou uchována v databázi na serveru a při spuštění aplikace se dotáže na příslušný skript na daném serveru, skript potom jako odpověď vrátí data v XML formátu a ty jsou dále v aplikaci rozparsována a uložena pro další použití. Dotazů na server je v aplikaci hned několik. Hned na úvodní obrazovce se stáhnou data s nastavením, ty obsahují veškeré možnosti pro filtry vyhledávání, dodatečné články apod. Při každém vyhledávání se posílá na server dotaz, který obsahuje parametry s námi zvolenými hodnotami filtrů a jako odpověď získáme výsledný seznam recyklačních míst.

3.2.1 Obrazovky

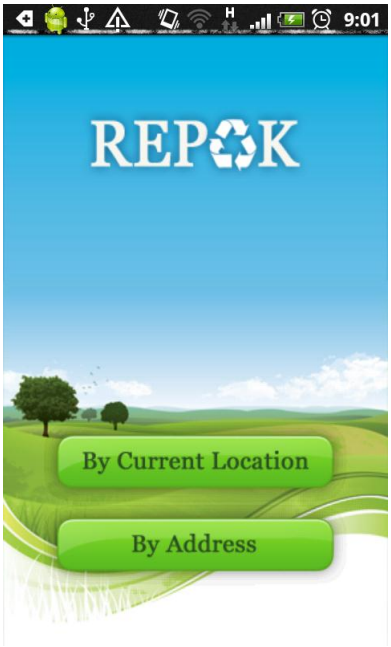
Na platformě Android se jednotlivá obrazovka nazývá „aktivita“. Tyto aktivity jsou na sobě naprosto nezávislé, ale při přechodu mezi nimi si mohou předávat volitelné parametry. V mojí aplikaci je použito 10 následujících aktivit:

- Splashscreen – Tato aktivita se zobrazí při prvním spuštění aplikace, obsahuje pouze obrázek s logem a po nastaveném čase, v našem případě 3 vteřiny, se sama přepne na následující aktivitu. Během této doby se zatím na pozadí aplikace stahují potřebná data a zahájí se vyhledávání současné polohy.



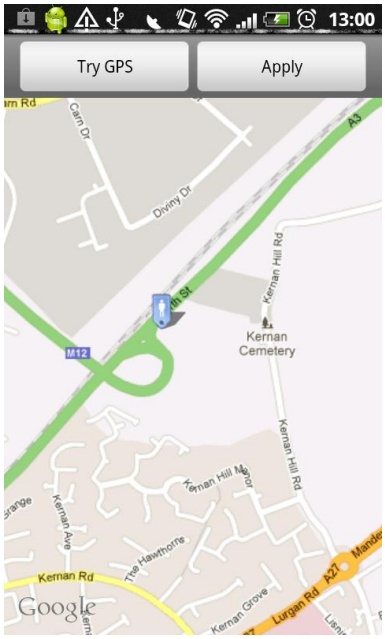
Obrázek 7 – Splashscreen

- Home screen – Aktivita s úvodním menu, jsou zde dvě možnosti a to buď rychlé vyhledávání nejbližších recyklačních stanic (By Current Location), nebo přejití na upřesnění vyhledávání (By Address).



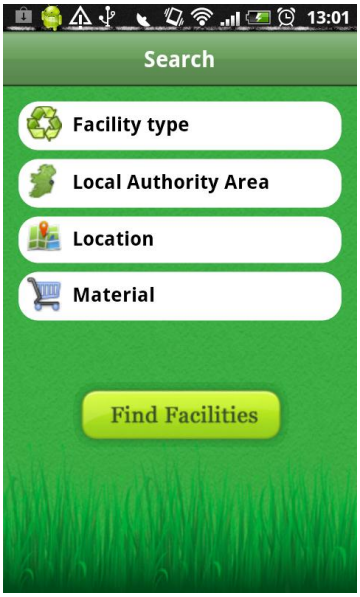
Obrázek 8 – Home screen

- Location screen – Pokud nemáte zapnutou GPS, tak se Vás aplikace na Home screen zeptá prostřednictvím dialogu, jestli nechcete nastavit svojí lokaci ručně, v případě že ano, se ocitnete na této aktivitě, zde je mapa, na které se po kliku vyobrazí ikonka, která ukazuje na vámi zvolenou lokaci, po dalším kliku na jiné místo se ikonka přesune, pokud jste se zvolenou lokací spokojení, tak stačí pro uložení kliknout na tlačítko „Apply“. Pokud máte GPS zapnutou a pouze jste v době zapnutí aplikace neměli signál, můžete zvolit možnost „Try GPS“ a to Vás přesune zpět na Home screen, kde se zobrazí dialog s vyčkávací animací, který po získání lokace z GPS zmizí.



Obrázek 9 – Location screen

- Search screen – Na této aktivitě se nacházejí výše zmíněné filtry, všechny, až na hledání podle klíčového slova, mohou zahrnovat buď žádnou (v tom případě se berou v potaz všechny možnosti) nebo libovolný počet možností.



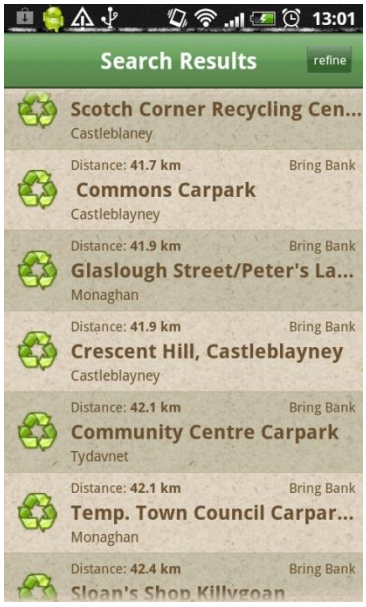
Obrázek 10 – Search screen

- Search options screen – Po kliknutí na jakýkoliv filtr na Search screen přejdete na tuto aktivitu (mimo klíčového slova, to je pouze textové editační políčko, které nikam

nepřepíná), zde můžete označit požadované možnosti, které chcete zahrnout do vyhledávání a po stisku tlačítka „save“ se vámi vybrané možnosti uloží.

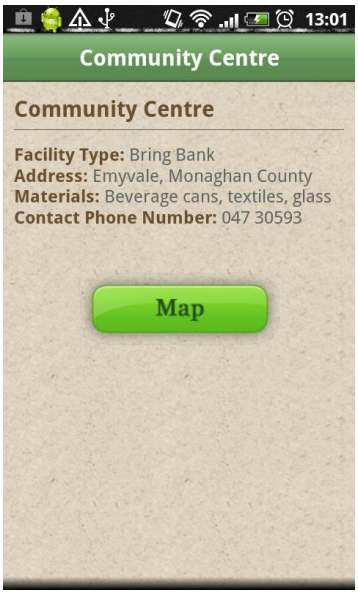
Obrázek 11 – Options screen

- Search result screen – Aktivita obsahující skrolovací seznam všech nalezených recyklačních míst, počet výledků hledání je omezen na 15, ale po stisku posledního políčka v seznamu „Load more …“ se počet výsledků vždy o 15 zvětší, pokud jsou zobrazeny všechny výsledky, které je nám server schopný poskytnout, tak toto poslední tlačítko zmizí.



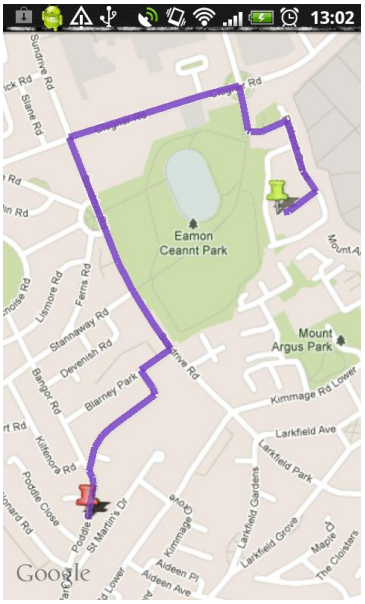
Obrázek 12 – Search result screen

- Search detail screen – Po kliknutí na libovolný výsledek ze seznamu na Search result screen se přesunete na tuto aktivitu, kde se nachází detail recyklační stanice, kde můžete vidět veškeré dostupné podrobnosti. Dále se tu také nachází tlačítko „Map“, které Vás přesune na mapu.



Obrázek 13 – Search detail screen

- Map detail screen – Zde se na mapě vyobrazí pozice vybrané stanice a pokud aplikace lokalizovala vaší pozici, tak i ikonku s vámi a vykreslí se nejkratší cesta mezi těmito body.

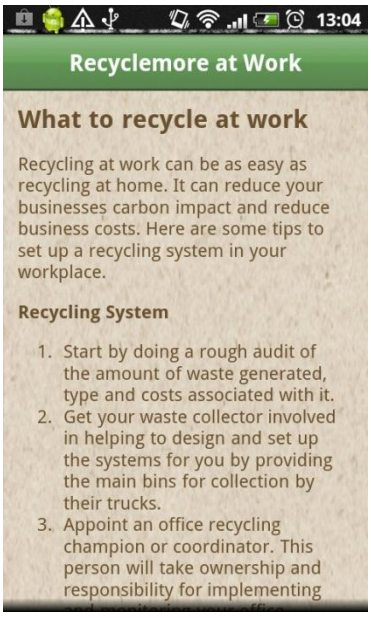


Obrázek 14 – Map screen

- Articles list screen – Tato aktivita obsahuje seznam dostupných článků, tyto články je možné spravovat z webového rozhraní.

Obrázek 15 – Articles list screen

- Article detail screen – Samotný celý článek z Articles list aktivity.



Obrázek 16 – Article detail screen

3.2.2 Uchovávání dat

V aplikaci je důležité, aby po získání dat ze serveru byly tyto data uchovaná napříč všemi aktivitami. Pro tento účel byla vytvořena třída DataStorage.class, která uchovává řadu statických proměnných a metod, díky kterým pak je možné k těmto hodnotám pohodlně přistupovat z libovolné aktivity. Mezi nejdůležitější data patří nastavení, které se stahuje jedinkrát při spuštění aplikace, tento XML soubor s nastavením obsahuje následující informace:

- Seznam filtrů – Všechno možnosti, podle kterých se dá filtrovat výsledek vyhledávání s informace o každém z nich (titulek, klíč, typ, URL adresa příslušného obrázku).
- Seznam jednotlivých položek ke každému filtru – zde je pro každou položku uchován titulek a klíč, případně popis položky.
- Seznam všech článků – Data o všech článcích (titulek, klíč, URL obrázku)

3.3 Spojení se serverem

Pro spojení se serverem jsem naprogramoval statickou metodu *loadData()* ve třídě *DataStorage.class*, která vyžaduje dva parametry – *URL* z jaké mají být data stažena a *String* který indikuje klíč pro uložená data. Vždy, když je v libovolné aktivitě potřeba stáhnout nějaká data, tak je nadeklarována lokální funkce *loadData()*, ve které se vytvoří instance *java.net.URL* a zavolá se již výše zmíněná statická metoda z *DataStorage.class*. Zde můžete vidět celou tuto metodu:

```
public static boolean loadData(URL url, String key) {
	if(xml_library == null)
		xml_library = new HashMap<String, Document>();
	try {
		DocumentBuilderFactory dbf =
		DocumentBuilderFactory.newInstance();
		dbf.setNamespaceAware(true);
		DocumentBuilder db = dbf.newDocumentBuilder();

		Document doc = db.parse(new InputSource(url.openStream()));

		xml_library.put(key, doc);
		xml_library.get(key).getDocumentElement().normalize();

	} catch (Exception e) {
		return false;
	}
	return false;
}
```

Jak je vidět ve zdrojovém kódu výše, při zavolání metody se zkontroluje, zda je nadefinovaná statická proměnná *xml_library* třídy *DataStorage.class*, pokud ne, tak se

nadefinuje na prázdnou hash mapu. Hash mapa byla zvolená s parametry typu *String* a *org.w3c.dom.Document*, protože je do ní potřeba ukládat textové klíče pro lepší orientaci a stažené XML dokumenty, na což výborně poslouží *org.w3c.dom.Document.* Dále se zde vytvoří instance *DocumentBuilderFactory*, která slouží ke správnému zformátování stažených dat při vytváření instance *DocumentBuilder*. Díky instanci *DocumentBuilder* můžeme poté lehce stáhnout data ze serveru, která se nám přímo zformátují do požadovaného formátu *Document.* Následuje už jen uložení dat do naší knihovny *xml_library.* Poslední krok v této funkci je zavolání metody *normalize()*, díky níž se zajistí, že v dokumentu budeme mít validní XML.

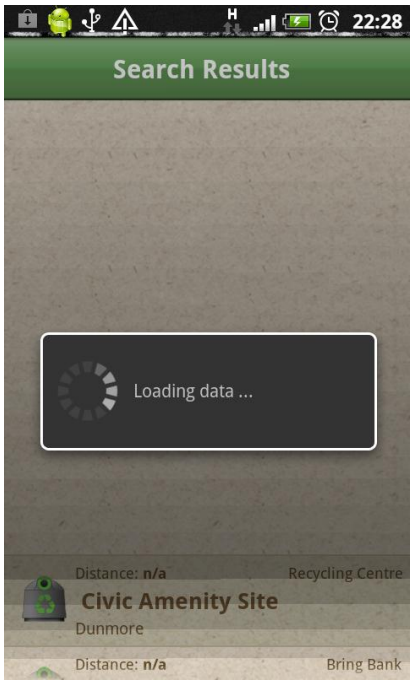
3.3.1 Asynchronní procesy

Asynchronní procesy se používají v případě, že v aplikaci potřebujeme vykonávat nějaké věci na pozadí, například stahování dat ze serveru nebo aktualizování současné polohy. Právě pro tyto dvě věci se v mojí aplikaci asynchronní procesy používají. K vytvoření asynchronního procesu je v aplikaci naprogramovaná třída AsyncLoadXMLFeed, která dědí androidní třídu AsyncTask. Tato zděděná třída je v Androidu určená právě k takovýmto úkonům. V následujícím kódu je znázorněna struktura této třídy:

```
private class AsyncLoadXMLFeed extends AsyncTask<Void, Void, Void>{
	ProgressDialog dialog;

	protected void onPreExecute(){

		dialog = ProgressDialog.show(SearchResultActivity.this, "","Loading
		data ...", true);

	}

	@Override
	protected Void doInBackground(Void... voids){

		DataStorage.loadData(url, "test_search");
		return null;
	}

	@Override
	protected void onPostExecute(Void params){
		fillList();
		if(dialog != null)
			dialog.dismiss();
	}
}
```

V metodě onPreExecute() je napsán kód, který se má vykonat před spuštěním procesu na pozadí. V případě aplikace Repak jsem si zvolil zobrazení nahrávacího dialogu, protože tato třída je určena čekání na stažení dat ze serveru. V metodě doInBackground()

je potom kód, který probíhá na pozadí, zde se zavolá statická metoda loadData() z třídy DataStorage, která je určena ke stahování dat. Jakmile se kód provede, zavolá se metoda onPostExecute(), ve které se běh procesu vrátí z pozadí zpět do hlavního vlákna, zavolá se metoda fillList(), která stažená data zpracuje a zobrazí na obrazovce a zruší se dialog.



Obrázek 17 – Nahrávací dialog při asynchronním procesu

3.3.2 XML

Knihovny pro parsování XML, které jsem nalezl dostupné na Android, se ukázaly jako značně nedostatečné, proto jsem se rozhodl naprogramovat vlastní knihovnu, postavenou tak, abych s ní mohl dále parsovat libovolné XML soubory. Knihovnu jsem pojmenovat *XmlDocumentNestDesign* a poskytuje následující metody:

- *public static Node getChildByName(Document document, String name)* – Pomocí této metody prvního potomka se shodným jménem jako parametr *name.* Vrací instanci typu *org.w3c.dom.Node*, což je jeden prvek XML souboru, který v sobě uchovává všechny své potomky.

- *public static NodeList getChildListByName(Node element, String name)* – Tato metoda nám seznam všech potomků se jménem shodným s parametrem *name*. Vrací *org.w3c.dom.NodeList*, což je pole prvků typu *org.w3c.dom.Node.*

- *public static String getFirstTagValueByName(Node element, String name)* - Díky této metodě získáme hodnotu prvního elementu shodného s parametrem *name*. Vrací hodnotu v textové podobě typu String.

Díky výše uvedené knihovně můžeme potom lehce rozparsovat jakýkoliv XML dokument a získat z něho potřebná data. Na obrázku Obr. 2 můžete vidět ukázku XML dat, která vrací server.

```
▼<option>
   <title>Facility type</title>
   <name>facility_type</name>
   <type separator=",">multiple</type>
   <hint>Facility type</hint>
   <image_url>ico_trash.png</image_url>
  ▼<items>
   ▼<item>
      <title>Bring Bank</title>
      <value>1</value>
      <image_url>ico_bring_bank.png</image_url>
     ▼<detail_info>
        <link_type>alert</link_type>
        <title>Bring Bank</title>
       ▼<description>
          Unmanned(acessible all hours), small range of material (typically 3 colours Glass and Cans)
        </description>
      </detail_info>
    </item>

   ...
```

Obrázek 18 – Ukázka XML získaného ze serveru

Jazyk XML se v aplikaci nepoužívá pouze k přijímání dat, velikou výhodou programování pro Android je, že se v něm dá napsat celý vzhled aplikace. Pro tento účel jsou v aplikaci vytvořeny adresáře pro různé druhy souborů určujících vzhled, všechny se nacházejí v adresáři „res“.

Hlavním adresářem je adresář „layout“, zde jsou uloženy XML soubory, které určují rozmístění prvků v aktivitách, kód v takovémto souboru může vypadat například takto:

```
<?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:background="@drawable/mainback"
    android:gravity="center_horizontal|bottom"
    android:orientation="vertical"
    android:paddingBottom="50px" >

    <TextView
        android:id="@+id/headText"
        android:layout_width="wrap_content"
        android:layout_height="40px"
        android:text="Your recycling facility search"
        android:textColor="@android:color/primary_text_light"
        android:textSize="20px" />

    <ImageButton
        android:id="@+id/curr_loc_butt"
        android:layout_width="wrap_content"
```

```
            android:layout_height="wrap_content"
            android:background="@null"
            android:src="@drawable/butt_curr_pos" />

        <ImageButton
            android:id="@+id/address_butt"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:background="@null"
            android:src="@drawable/butt_by_address" />

</LinearLayout>
```

V ukázce kódu výše je vidět struktura XML souboru pro zobrazení Home screen, na něm se pokusím popsat princip stavění layoutů pro aktivity. Základ celého souboru je element „LinearLayout“. Layouty obecně nám slouží k jednoduššímu rozmístění jejich podřízených elementů. Každý layout se hodí na jiné rozmístění.

LinearLayout slouží k prostému řazení prvků za sebe, podle nastavení buďto vertikálně, nebo horizontálně, prvky v něm se nemůžou nijak překrývat a není možné prvky v něm relativně pozicovat, ale přesto je tento layout nejpoužívanější, hlavně nejspíše pro jeho jednoduchost.

Dalším velmi používaným layoutem je RelativeLayout, v něm se mohou prvky překrývat, mohou být relativně pozicovány jak vůči samotnému layoutu, tak sami vůči sobě. Veliká výhoda je právě v možnosti překrývání prvků, díky tomu je možné používat různé pokročilejší elementy, které mohou například vyjíždět nad ostatní obsah aktivity. Nevýhoda tohoto layoutu je v tom, že se musíme o každý prvek starat, což nám v případě prvního zmiňovaného z veliké míry odpadá.

Android nabízí i další layouty, jako třeba TabLayout, AbsoluteLayout nebo FrameLayout, ale ty už se zdaleka nepoužívají v takové míře jako předchozí dva zmiňované. Mohou být velice užitečné, ale spíše ve výjimečných situacích, proto se o nich dále nebudu podrobněji rozepisovat.

Veškeré layouty jsou určeny výhradně k tomu, aby zapouzdřily buď další layouty, nebo samotné jednoduché prvky, mezi ty patří například ImageButton nebo TextView, které můžete vidět v kódu výše. Tyto jednoduché prvky jsou určeny k zobrazení nejakého určitého obsahu. Může to být například text, nebo obrázek, tabulka, video nebo webová stránka. Android těchto prvků nabízí velice dlouhý seznam, takže při stylování aplikace můžete být opravdu velice kreativní.

Vlastnosti jednotlivých prvků se upravují pomocí atributů, těmito atributy se dá nastavit prakticky všechno, co nás napadne jako velikost prvku, jeho pozadí, styl, identifikátor, barva a velikost písma atd.

3.4 Google Map

Společnost Google poskytuje pro vývojáře softwaru pro chytré mobilní telefony vlastní knihovnu Google map, která ovšem není součástí standardních androidích knihoven. Proto je aplikace Google mapy trochu složitější a vyžaduje několik kroků k úspěšnému zprovoznění.

3.4.1 Podmínky pro správnou funkčnost

Jelikož tato knihovna není součástí standardních knihoven Androidu, tak její užití musíme zmínit v tzv. „AndroidManifest.xml“ souboru, což se soubor s veškerým systémovým nastavením pro celou androidí aplikaci. Jako jeden z potomků elementu <application> tedy musí přibýt element, který aplikaci oznámí použití příslušné knihovny, jako v následujícím kódu.

```
<uses-library
    android:name="com.google.android.maps"
    android:required="true" >
</uses-library>
```

Dalším krokem je získání aplikačního klíče od Googlu, s ním jsem měl celkem problémy, sám moc nechápu, proč Google tento postup nezjednoduší. Na oficiálních stránkách je několikastránkový návod, jak tento klíč získat, já zde jen informativně popíši základní kroky pro demonstraci.

Prvním krokem je, že si najdeme debug.keystore v našem SDK, zde si klíč musíme vytvořit pomocí příkazové řádky, příkazy pro to jsou ve výše zmiňovaném návodu. Poté musíme z klíče získat kód v MD5, který na stránkách Google vložíme do překladače a ten nám vrátí hexadecimální klíč, jenž je potřeba vložit ke každému prvku s mapou. Tyto klíče jsou potřeba dva, jeden pro testování a druhý pro vypublikovanou aplikace. Problémy byly hlavně se získáním klíče, podařilo se mi to až na třetí pokus, protože Google nechtěl klíč, který mi byl vygenerován, přijmout.

Po správném nastavení knihoven a získání aplikačního klíče by nám teoreticky už nic nemělo překážet v tom, abychom vytvořili aktivitu, která bude obsahovat MapView, které mapu dokáže zobrazit, jako jeden z atributů tohoto elementu je právě výše zmiňovaný aplikační klíč, který musíte doplnit. Toto je MapView pro zobrazení mapy v aplikaci Repak:

```
<com.google.android.maps.MapView
    android:id="@+id/mapView"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:apiKey="0ah3I3v92nMve7vhE-Erq_pkllR2CfoMR1MiUlA"
    android:clickable="true"
    android:enabled="true" />
```

Pokud vše provedete správně, měla by se vám po přesunutí na tuto aktivitu zobrazit mapa. Pomocí různých dalších nastavení si můžeme mapu upravit k obrazu svému, mezi nejčastější nastavení patří vycentrování mapy k určitému bodu, nastavení výchozího přiblížení nebo třeba zobrazení, v jakém se mapa bude zobrazovat (např. satelitní pohled nebo zvýraznění silnic).

3.4.2 Získání lokace

Android podporuje získávání lokace ze tří různých zdrojů, pomocí GPRS, WiFi nebo GPS.

Jako první je získání současní lokace pomocí GPRS. V tomto případě se lokace dopočítává z přijímání signálu ze tří různých vysílačů, pro funkci navigace je to velice nepřesná metoda. Přesnost se uvádí kolem 1 – 1.5 kilometru, takže pokud byste chtěli například hledat nejkratší cestu, tak by vaše zařízení mohlo vyhodnotit vaší polohu na úplně jiné silnici a v tu chvíli je pro vás vypočítaná cesta naprosto bezcenná.

Další možnost je získávání lokace pomocí Wi-Fi, tento způsob také není nikterak přesný, lokace se určuje podle IP adresy směrovače, takže nikdy nezjistíte přímo svojí polohu, ale polohu přístupového bodu poskytovatele internetu, což znamená, že přesnost může být i několik kilometrů.

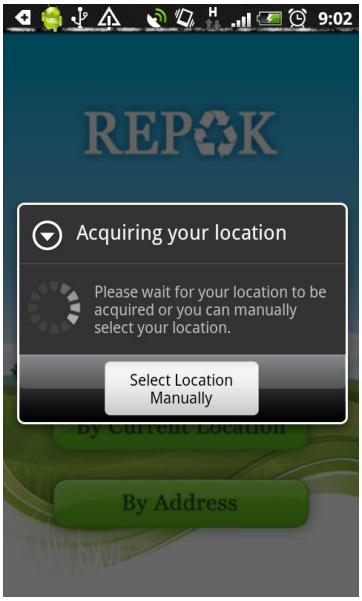
Poslední možností je využití GPS, pro získávání přesné lokace je to určitě nejlepší možnost, i když má několik nevýhod. Signál GPS neprojde železnou střechou, takže uvnitř budovy vám neposkytne v podstatě žádnou polohu. Další nevýhodou jsou veliké energetické nároky, pokud bude GPS na zařízení puštěná, můžete počítat s velice rychlým úbytkem baterie.

GPS

Z popisu výše uvedených možností je jasné, že pro aplikaci Repak, z důvodu vykreslování nejkratší cesty, je použitelná pouze varianta s GPS. Při zapnutí aplikace zkontroluje, zda je GPS v zařízení zapnutá, v případě, že není, se zobrazí dialog, který nabídne jako možnost přepnout se pomocí systémové funkce do nastavení bezdrátových funkcí. Tato funkce vypadá v Androidu takto:

```
startActivity(new
Intent(android.provider.Settings.ACTION_LOCATION_SOURCE_SETTINGS));
```

Pokud aplikace zjistí, že GPS je zapnutá, zobrazí se dialog s animací vyhledávání, během této doby se na pozadí začne v intervalu 2 sekundy systém dotazovat GPS na polohu, dokud GPS nevrátí validní data. Pokud se nacházíte v místě, kde GPS nemá signám, můžete toto ukončit tlačítkem zpět nebo si vybrat vaší aktuální polohu ručně, to bude popsáno později. Na obrázku obr.19 můžete vidět dialog s vyhledáváním GPS.



Obrázek 19 – Dialog s průběhem získávání polohy

Manualní zadávání

Pokud nechcete zapínat GPS nebo nemáte potřebný signál, můžete v menu na Home screen kliknout na položku „My location“ a aplikace se přesune na aplikaci s mapou. Mapa je vycentrována na Irsko, protože odtamtud také pochází databáze. Zde můžete klikem na obrazovku označit libovolné místo a na něm se objeví ikonka s panáčkem, tato ikonka označuje místo, které bude po stisku tlačítka „Apply“ nastaveno jako vaše pozice pro vyhledávání.

Další možností, jak se dostat z této aktivity je, že se opět pokusíte dále získat pozici pomocí GPS. Po stisku tlačítka „Try GPS“ se přesunete zpět na Home screen, kde se opět zobrazí dialog s vyhledáváním vaší polohy do té doby, než GPS dostane signál.

3.4.3 Vykreslování cesty

Vykreslování cesty na mapě byl při programování této aplikace jeden z největších problémů, ač jsem čekal, že toto bude v rámci knihovny Google map hračkou. Google map API bohužel vykreslování cesty nepodporuje, samozřejmě můžete aplikaci odkázat na aplikaci Google Map s tím, že jí jako parametr pošlete polohu recyklačního místa, kam se chcete dostat a zde si zvolíte vyhledání trasy, ale při použití tohoto způsobu je nutnost spokojit se s tím, co nabízí cizí aplikace a my nemůžeme ovlivnit vzhled, ikonky a atd. Jako další nevýhoda je to, že pokud si vybereme polohu ručně, tak nejsme schopni této aplikaci poslat současně dva body, takže cestu mezi recyklačním střediskem a námi zvolenou polohou stejně nezískáme.

Právě kvůli výše popsaným nevýhodám jsem se rozhodl cestu vykreslit ručně. Jediný způsob, který jsem našel, jak tohoto dosáhnou, je poslat request na klasické Google

mapy a zpracovat získaná data. Pro sestavení URL pro Google mapy jsem v aplikaci napsal následující metodu:

```
public static String getUrl(double fromLat, double fromLon, double toLat,
            double toLon) {// connect to map web service
      StringBuffer urlString = new StringBuffer();
      urlString.append("http://maps.google.com/maps?f=d&hl=en");
      urlString.append("&saddr=");// from
      urlString.append(Double.toString(fromLat));
      urlString.append(",");
      urlString.append(Double.toString(fromLon));
      urlString.append("&daddr=");// to
      urlString.append(Double.toString(toLat));
      urlString.append(",");
      urlString.append(Double.toString(toLon));
      urlString.append("&ie=UTF8&0&om=0&output=kml");

      String ggg = urlString.toString();
      return urlString.toString();
}
```

Jako parametry jsou vyžadovány zeměpisná šířka a výška dvou bodů, mezi kterými chceme získat nejkratší cestu. Google neumí přijmout jako parametry přímo tyto data, umí pouze najít cestu mezi dvěma adresami. To se dá obejít tak, že místo adres dáte jako vyhledávací řetězec do parametru &saddr a &daddr čárkami oddělenou zeměpisnou šířku a zeměpisnou výšku. Jako další parametry jsou například kódování nebo formát výstupních dat. V našem případě nám Google vrátí data ve formátu KML, což je aplikace metajazyka XML určená k publikaci a distribuci geografických dat. V tomto souboru je obsaženo obrovské množství dat, jako například, vzdálenost, popis cesty, styl spojové čáry, druh použitých ikonek a tak dále. Nás ale bude zajímat pouze výčet jednotlivých souřadnic, což jsou pro nás body, kde se nějakým způsobem mění směr jízdy.

S přijatými daty se pracuje obdobně jako s parsováním XML souboru, na internetu jsem dohledal nejjednodušší způsob zpracování dat od Google map a aplikoval jsem ho na mojí aplikaci. Princip je v podstatě v tom, že si nalezneme element se souřadnicemi a ty poté uložíme do dvourozměrného pole typu double.

Samotná cesta se poté vykreslí na mapu pomocí třídy MapOverlay, která dědí z com.google.android.maps.Overlay. Tato třída projede všechny získané body a postupně vykreslí čáry mezi sousedními. Toto se prování v tomto cyklu for:

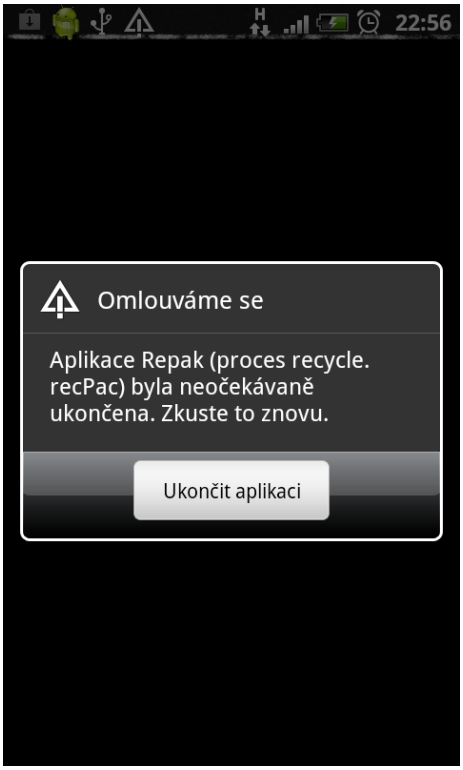
```
for (int i = 0; i < mPoints.size(); i++) {
      Point point = new Point();
      mv.getProjection().toPixels(mPoints.get(i), point);
      x2 = point.x;
      y2 = point.y;
      if (i > 0) {
            canvas.drawLine(x1, y1, x2, y2, paint);
      }
      x1 = x2;
      y1 = y2;
}
```

Proměnná mv je zde instance MapView, která je v Androidu určena k zobrazování mapy. Pomocí této instance dokážeme naše zeměpisné souřadnice převést na pixelové souřadnice na obrazovce našeho zařízení. Dále se díky instanci canvas, která je typu Canvas, vykreslí daná čára na obrazovku. Po spojení všech bodů se nám na mapě zobrazí křivka značící nejkratší cestu v podání Google map.

Rychlost zobrazení cesty záleží na počtu bodů, které dostaneme jako data od Google map, na krátkou vzdálenost, do stovek kilometrů, by se cesta měla vykreslit téměř okamžitě. V aplikaci Repak je trochu problém, že data o všech recyklačních místech pocházejí z Irska, proto když zvolíme, že naší polohu chceme určit podle GPS v našem zařízení, tak při zobrazení mapy můžeme sledovat značnou prodlevu mezi zobrazení mapy a vykreslení cesty. Je to dáno velikou vzdáleností mezi Českou republikou a Irskem. Toto byl také jeden z důvodů, proč jsem se rozhodl do aplikace zakomponovat možnost určit lokaci manuálně.

3.5 Testování a ladění

Testování a ladění je vždy ta nejzdlouhavější část vývoje, ale pro správný chod aplikace je to nezbytné. Testování spočívá hlavně v tom, že se snažíme nasimulovat pokud možno všechny možné situace a zkoumáme, jak na ně aplikace reaguje. Pokud v nějakém momentě spadne nebo se zachová nestandardně, přijde na řadu ladění. Problém je, že při pádu aplikace Android zobrazí dialog s informací, že aplikace spadla a ukončí ji, takže nemáme tušení, z jakého důvodu se to stalo. Pokud máme zařízení, na kterém aplikaci testujeme, připojené k počítači, můžeme nastalou chybu odchytit pomocí debuggeru ve vývojovém prostředí a DDMS, díky tomu zjistíme přesný řádek a povahu chyby a můžeme jí velice často lehce odstranit.



Obrázek 20 – Dialog zobrazený Androidem při pádu aplikace

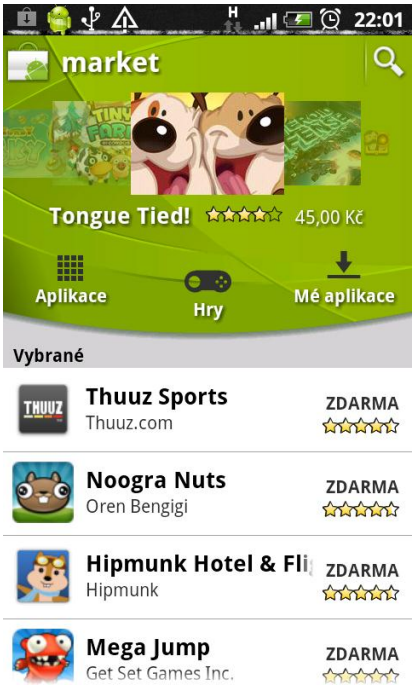
Potíže nastanou v případě, že pro testování není možné být k počítači připojení. U aplikace Repak tato situace nastala při testování funkce GPS. K tomu, abych ověřil správnou funkčnost GPS, bylo potřeba se dostat pod širé nebe a pro testování správných reakcí na změnu polohy bylo potřeba se se zařízením hýbat, takže připojení k počítači v tuto chvíli nebylo prakticky možné.

Pro odchytávání chyb při této činnosti, jsem na internetu dohledal knihovnu s třídou ExceptionHandler, na tuto třídu stačí v aktivitě, kde chceme chyby odchytávat, zavolat metodu register(), která má za jeden z parametrů URL, na které se odešle report při případném pádu aplikace.

4 Android Market

Největší přístupným úložištěm aplikací pro Android je Google Market (V současné době už je přejmenován na Google Play). V tomto úložišti se nacházejí aplikace jak zdarma, tak i placené. Aplikace jsou rozděleny do různých kategorií, které se pak dělí ještě na menší podkategorie. K tomu, abyste se dostali do Android Marketu, musíte mít vytvořený účet od Google, který je zdarma. Stahování aplikací probíhá v nejvíce případech prostřednictvím aplikace v mobilu, kde si je vyberete aplikaci a stisknete pouze instalovat, poté se aplikace stáhne a sama nainstaluje.

Než se tento proces spustí, tak musíte ještě povolit všechny oprávnění, které aplikace požaduje. Tyto oprávnění mohou být občas velice zrádné, pokud neznámé aplikaci povolíte prohlížení SMS zpráv nebo přístup na SD kartu, může si aplikace dělat s vašimi daty vše, co se jí zlíbí, proto je dobré si před stažením aplikace přečíst alespoň nějaké poznámky k ní a zvážit, zda přístupy, které vyžaduje, jsou adekvátní k tomu, co aplikace má skutečně vykonávat.



Obrázek 21 – Ukázka prostředí Android Marketu

4.1 Publikování

Na Android Market může dávat aplikace prakticky kdokoliv, jediná podmínka je založení účtu a zaplacení vstupního poplatku, který činí 25 euro. Po zprovoznění účtu můžete začít nahrávat aplikace. Aplikace, kterou zde chcete publikovat, musí být spojena s unikátním klíčem, který k ní můžete připojit ve vývojovém prostředí. Klíč je potřeba

nejdříve vytvořit v úložišti keystore, při vytváření musíte zadat pár informací o klíči, jako dobu platnosti, název, vlastníka nebo třeba zemi původu. Tento klíč je pro aplikaci velice důležitý, díky němu se po nahrání může aplikace updatovat a Market automaticky každého uživatele, který si aplikaci stáhnul, upozorní na novou aktualizaci, kterou lze provést v rámci Marketu bez nové instalace.

Po spojení aplikace s aplikačním klíčem nám nic nebrání aplikaci nahrát. O každé aplikaci můžete doplnit informační informace, připojit k ní kontaktní informace a nahrát její screenshoty a další věci.

Když je aplikace na Marketu nahrána, tak zde můžeme sledovat, kolik lidí si aplikaci stáhlo a jak uživatelé aplikaci hodnotí. Další velice užitečnou funkcí je sledování pádů aplikace, pokud někomu z nějakého důvodu aplikace spadne, systém uživateli nabídne možnost zaslat report s údaji o chybě, která v aplikaci nastala, je už jen na uživateli jestli tento report odešle nebo ne, ale pokud ano, tak se právě tady zobrazí.

5 Závěr

Systém Android má dle mého názoru největší budoucnost v oblasti operačních systémů pro chytré telefony a to hlavně díky jeho flexibilitě a lehké rozšiřitelnosti. Programování pro tuto platformu je vcelku příjemné hlavně díky programovacímu jazyku Java a stylování v XML.

Aplikace Repak byla moje první aplikace pro tuto platformu, začal jsem na ní pracovat přibližně půl roku před odevzdání této bakalářské práce. Za tu dobu jsem už udělal dalších pár aplikací, a když se podívám na tuto aplikaci zpětně, tak už teď bych spoustu věcí změnil nebo naprogramoval od začátku jinak. Má určitě několik slabých stránek, například není ošetřená proti telefonům s výsuvnou klávesnicí a může mít problém při náhlém výpadku internetu, ale jinak aplikace pracuje celkem slušně. Těžko odhadnout jak moc by byla využitelná praxi, dle mého názoru by s jejím využitím neměl být žádný problém. Jediná věc, která by se musela změnit pro místní použití, je sehnat databázi recyklačních míst v České republice, kterou jsem bohužel nikde nesehnal.

Literatura

[1] JACKSON, Wallace. Android apps for absolute beginners. New York, NY: Distributed to the book trade by Springer Science Business Media, c2011, 328 s. ISBN 14-302-3446-6.

[2] ROGERS, Rick. Android application development. 1st ed. Sebastopol, Calif.: O'Reilly, c2009, 318 s. ISBN 05-965-2147-2.

[3] Wikipedia: Otevřená encyklopedie. Android [online]. 2012 [cit. 2012-05-11]. Dostupné z: http://en.wikipedia.org/wiki/Android_(operating_system)