TECHNICKÁ UNIVERZITA V LIBERCI
Fakulta mechatroniky, informatiky a mezioborových studií

Studijní program: B2646 – Informační technologie
Studijní obor: 1802R007 – Informační technologie

Aplikace pro sledování nákladů provozu automobilu pro OS Android

Application for monitoring of vehicle costs for OS Android

Bakalářská práce

Autor: **Leoš Dostál**
Vedoucí práce: Ing. Přemysl Svoboda
Konzultant: Ing. Tomáš Martinec, PhD.

V Liberci 15. 5. 20012

Vzhledem ke správnému číslování stránek nebude tato strana obsahem BP a bude nahrazena oficiálním zadáním.

Prohlášení

Byl(a) jsem seznámen(a) s tím, že na mou bakalářskou práci se plně vztahuje zákon č. 121/2000 Sb., o právu autorském, zejména § 60 – školní dílo.

Beru na vědomí, že Technická univerzita v Liberci (TUL) nezasahuje do mých autorských práv užitím mé bakalářské práce pro vnitřní potřebu TUL.

Užiji-li bakalářskou práci nebo poskytnu-li licenci k jejímu využití, jsem si vědom povinnosti informovat o této skutečnosti TUL; v tomto případě má TUL právo ode mne požadovat úhradu nákladů, které vynaložila na vytvoření díla, až do jejich skutečné výše.

Bakalářskou práci jsem vypracoval(a) samostatně s použitím uvedené literatury a na základě konzultací s vedoucím bakalářské práce a konzultantem.

Datum

Podpis

Poděkování

Na tomto místě bych rád poděkoval mému vedoucímu bakalářské práce Ing. Přemyslu Svobodovi za vedení mé práce. Jeho rady a poznatky byly velice užitečné a pomohly mi k realizaci mé práce. V neposlední řadě bych také rád poděkoval rodině za umožnění studia na vysoké škole a za jejich podporu při studiu.

Abstrakt

Tato práce by měla seznámit čtenáře s operačním systémem Android, který je využíván převážně v mobilních zařízeních. Práce seznamuje se základními rysy a architekturou tohoto operačního sytému. Dále práce seznamuje s vývojovým balíčkem Android SDK a vývojovým prostředím Eclipse. Čtenář je seznámen se základní adresářovou strukturou Android projektu, se strukturou Android aplikace a s vývojem aplikace ve vývojovém prostředí Eclipse. Čtenář bude tedy schopen začít vytvářet a testovat projekty pro tento OS.

Druhou částí práce je návrh vlastního programu, který umožní kompletní správu provozních výdajů automobilu. Aplikace by měla být určena pro širokou veřejnost a umožňovat kompletní správu provozních výdajů více automobilů. Výdaje by měly být řazeny do sekcí pro lepší pořádek a třídění záznamů. Aplikace by měla umět počítat statistické údaje. Také by měla umožnit zálohu a obnovu uživatelských dat.

Poslední částí je implementace navržené aplikace a její otestování na reálných zařízeních. Tato část seznámí uživatele s konkrétní implementací navržené aplikace. Nastíní tedy čtenáři, jak vypadá konkrétní implementace programu pro tento OS.

Klíčová slova

OS Android, Android SDK, mobilní zařízení, provozní výdaje automobilu

Abstract

This work should introduce the reader to OS Android, which is mainly used in mobile devices. This work introduces the basic features and architecture of this operating system. Further the Android SDK and develop environment Eclipse are introduced. The reader is familiar with the basic directory structure of the Android project, structure of Android application and with development of Android application in develop environment Eclipse. The reader will be able to begin to create and test own projects for this OS.

The second part of this work is about design of own application, which will be able to store all operating costs of the car. The application should be available for general public and provide complete monitoring costs of more cars. Costs should be divided into sections for better order and sorting of records. Application should be able to calculate some statistics. It should also be able to backup and restore user's data.

The last part is the implementation of the proposed application and it's testing on real devices. This part introduces the reader to specific implementation of the proposed application. So it should outline the readers, how a particular implementation of the application looks like for this OS.

Keywords

OS Android, Android SDK, mobile devices, operating costs of car

Obsah

Prohlášení ... 3
Poděkování ... 4
Abstrakt ... 5
Klíčová slova ... 5
Abstract ... 6
Keywords ... 6
Seznam obrázků ... 8
Seznam ukázek kódů ... 8
Seznam zkratek ... 9
1 Úvod ... 10
2 Teoretická část ... 11
2.1. Co je Android ... 11
2.2. Historie ... 12
2.3. Architektura ... 13
2.4. Struktura aplikace ... 14
2.5. Vývojové prostředí ... 16
2.6. XML návrhy ... 21
2.7. Soubor R.java ... 23
2.8. Nabídky menu ... 23
2.9. Notifikace ... 25
3 Praktická část ... 26
3.1. Návrh aplikace ... 26
3.2. Vlastní implementace ... 26
3.3. Zpracování záznamů ... 31
3.3.1. Databáze a tabulky ... 31
3.3.2. Zpracování dat v objektech třídy Cursor ... 36
3.3.3. Kontroly zadávaných údajů ... 36
3.3.4. Výpočet spotřeby ... 38
3.3.5. Výpočty statistik ... 40
Závěr ... 42
Seznam použité literatury ... 43
Seznam příloh ... 44
Návod k obsluze ... 46
Obsah CD-ROM ... 53

Seznam obrázků

Obrázek 1: Hlavní obrazovka OS Android 4.0 12
Obrázek 2: Struktura OS Android 13
Obrázek 3: Životní cyklus aktivity 15
Obrázek 4: Adresářová struktura projektu ve vývojové prostředí Eclipse 17
Obrázek 5: Správce a vytváření virtuálního zařízení 18
Obrázek 6: Průběh překladu Android projektu 20
Obrázek 7: Spuštěná první aplikace na Virtuálním zařízení 21
Obrázek 8: Příklady kontextového menu aplikací 24
Obrázek 9: Příklady notifikací 25
Obrázek 10: Schéma aktivit aplikace 27
Obrázek 11: Aplikace – Úvodní aktivity 28
Obrázek 12: Aplikace – sekce Tankování 29
Obrázek 13: Aplikace – sekce Údržba a Ostatní 29
Obrázek 14: Aplikace – sekce Statistiky 30
Obrázek 15: Aplikace – další aktivity 31
Obrázek 16: Databázové tabulky aplikace 32

Seznam ukázek kódů

Ukázka kódu 1: Nová aktivita 19
Ukázka kódu 2: AndroidManifext.xml 20
Ukázka kódu 3: XML soubor s rozvržením aktivity 22
Ukázka kódu 4: Soubor R.java 23
Ukázka kódu 5: XML soubor s rozvržením menu 24
Ukázka kódu 6: Textový řetězec s SQL příkazem jedné tabulky 34
Ukázka kódu 7: Metoda pro přidání vozidla do databáze 34
Ukázka kódu 8: Příkaz pro úpravu záznamu v databázi 35
Ukázka kódu 9: Metoda pro smazání záznamu z databáze 35
Ukázka kódu 10: Metoda pro výběr záznamu z databáze 35
Ukázka kódu 11: Zpravování objektu třídy Cursor 36
Ukázka kódu 12: Metoda kontrolující chronologii záznamů 37
Ukázka kódu 13: Kontrola budoucího data 38

Seznam zkratek

OS – Operační systém

HW – Hardware

SDK – Software Development Kit

PDA – Personal Digital Assistant

RAM – Random-Access Memory

SD – Secure Digital

3D – 3 Dimensions

DVM – Dalvik Virtual Machine

SMS – Short Message Service

ADT – Android Development Tool

GPS – Global Positioning System

USB – Universal Serial Bus

ARM – Advance RISC Machine

APK – Android application Package file

XML – Extensible Markup Language

DEX – Dalvik Executable Format

HTML – HyperText Markup Language

CSS – Cascading Style Sheets

SQL – Structured Query Language

SPZ – Státní poznávací značka

1 Úvod

Hlavním úkolem této práce je návrh aplikace pro správu provozních výdajů automobilu. Práce čerpá z předešlého bakalářského projektu [1], kde byla provedena rešerše již existujících programů a webových portálů umožňujících nějakým způsobem správu výdajů automobilu. Z rešerše vychází hlavně návrh vlastností a funkcí aplikace. Tyto programy byly určené pro různé operační systémy. Vlastní aplikace by měla samozřejmě umět uchovávat údaje o tankování automobilu, to by mělo být hlavní funkcí programu. Dále je také vhodné uchovávat další výdaje, jako jsou servis či nákup doplňků. Samozřejmostí by také mělo být zobrazení statistik z uchovávaných dat. Aplikace by měla počítat průměrnou spotřebu automobilu, což se hodí pro majitele starších vozidel, které nejsou vybaveny palubním počítačem, který by toto obstarával.

Dalším úkolem je implementace a otestování této aplikace, zde přichází na řadu výběr platformy. Zvolena byla mobilní platforma, konkrétně operační systém (OS) Android. Mobilní platforma byla zvolena, protože většina lidí má mobilní zařízení neustále u sebe, tím pádem je umožněno přidávat a mít neustálý přehled o výdajích automobilu.

Důvodem zvolení OS Android pro tuto práci byla jeho rozšířenost mezi uživateli. OS Android totiž najdeme i ve velice levných, a tím pádem zajímavých zařízeních pro běžné uživatele, kteří nevyžadují nejnovější a nejvýkonnější chytré telefony, které se objeví na trhu. Konkurenčními systémy v této oblasti jsou iOS od společnosti Apple a WindowsPhone 7 od společnosti Microsoft. iOS je možné nalézt pouze v zařízeních iPhone, které jsou relativně drahé a nabízejí v prodeji pouze jedno zařízení, které má vždy stejný design a hardwarovou (HW) specifikaci. Konkurence od Microsoftu v podobě WindowsPhone 7 není ještě veliká, protože se jedná o nový systém, který se od předchozí generace zásadně liší, tím pádem není možná přenositelnost aplikací mezi starou verzí a novou. Také není na trhu takový podíl zařízení jako s OS Android.

Práce tedy seznamuje se základy OS Android, vývojem a strukturou aplikací pro tento operační systém. Toto zahrnuje seznámení se s vývojovým balíčkem Android Software Development Kit (SDK) a vývojovým prostředím Eclipse. Pro testování aplikací je použit emulátor virtuálního Android zařízení, dále také reálná fyzická zařízení.

2 Teoretická část

2.1. Co je Android

Jedná se o open source software pro mobilní zařízení (mobilní telefony, tablety, Personal Digital Assistant (PDA), navigace, v dnešní době se jedná i o některá vozidla a televize). Operační systém je nyní vyvíjen společností Google a je založený na Linuxovém jádru. Snahou systému je svižný běh a uživatelská přívětivost na rozmanitých zařízeních, které disponují různě výkonným hardwarem a různě silnou baterií.

Operační systém Android je primárně využíván pro mobilní telefony, v posledních letech se však hodně rozšířil na tablety, s jejichž podporou se dříve nepočítalo. Původní verze Androidu byly určeny pro mobilní telefony, avšak výrobci využívali i tyto verze ve svých tabletech. To se však změnilo s příchodem verze 3.1, která byla určena pouze pro tablety a verze 2.3.x určená pouze pro mobilní telefony byla používána i pro tablety. V roce 2011 však společnost Google představila novou verzi 4.0, která je určena jak pro mobilní telefony, tak pro tablety. Tento krok je důležitý hlavně pro vývoj aplikací, protože aplikace napsané pro mobilní telefon s verzí 4.0 budou úplně stejně fungovat i na tabletu s Androidem 4.0.

Důvodem, proč zvolit pro vývoj aplikace právě OS Android, je jeho budoucí potenciál a jeho rychlý rozvoj. Jedná se o celkem nový OS, který se velice rychle rozšířil mezi uživatele. Výhody proti počítačům jsou zřejmé, mobilní zařízení má již většina populace stále u sebe, protože tato zařízení jsou malá a v dnešní době již velice výkonná. Dnešní běžná zařízení jsou vybavena jedno jádrovým procesorem o frekvenci 1GHz a operační pamětí Random-Access Memory (RAM) o velikosti 512MB, což je v porovnání s fyzickou velikostí a hmotností dostačující HW základ pro běh aplikací i náročných her. Na trhu jsou k dispozici i mnohem výkonnější zařízení, standardem se začínají stávat dvou jádrové procesory o taktu 1–1,5GHz s 1GB RAM pamětí. Na trhu se již objevil i sériově vyráběný *smartphone* se čtyř jádrovým procesorem o frekvenci 1,5GHz. Tyto HW specifikace se pomalu začínají přibližovat výkonu dnešních notebooků.

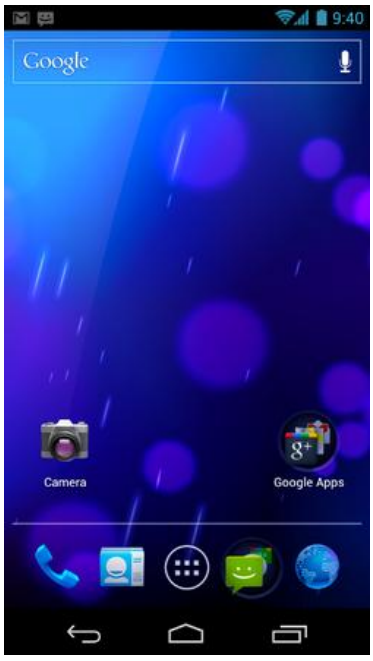
Nevýhodou mobilních zařízení je absence HW klávesnice u většiny dostupných zařízení. Dále je také limitující velikost displeje a kapacita baterie, které poskytnou

energii dnešním zařízením na cca jeden den. Vše je jen o kompromisu mezi velikostí, výdrží a výkonem.

Oficiální cestou pro získání aplikací je obchod Google Play (dříve AndroidMarket). Aplikace je zde možné stahovat zdarma, nebo za peníze. Bohužel placené aplikace lze často nalézt i na různých fórech volně ke stažení.

2.2. Historie

Původní projekt byl snahou společnosti Android Inc., která byla založena v Kalifornii v roce 2003. Google Inc. odkoupil tuto společnost v roce 2005 a udělal z ní svou dceřinou společnost. Vývojový tým Google stojí právě za zrozením operačního systému Android, který byl uveden na trh v roce 2007 jako otevřená mobilní platforma, která běží na jádře Linux verze 2.6. V tento rok byl také vydán první vývojový balíček Android SDK umožňující vývoj aplikací pro tuto novou platformu. První telefon využívající tuto platformu byl uveden v roce 2008 ve Spojených státech amerických. O výrobu telefonu se postarala společnost HTC. Telefon byl uveden v České republice v roce 2009. Od této doby roste počet uživatelů Androidu až na dnešních cca 250 milionů aktivací systému. [2]

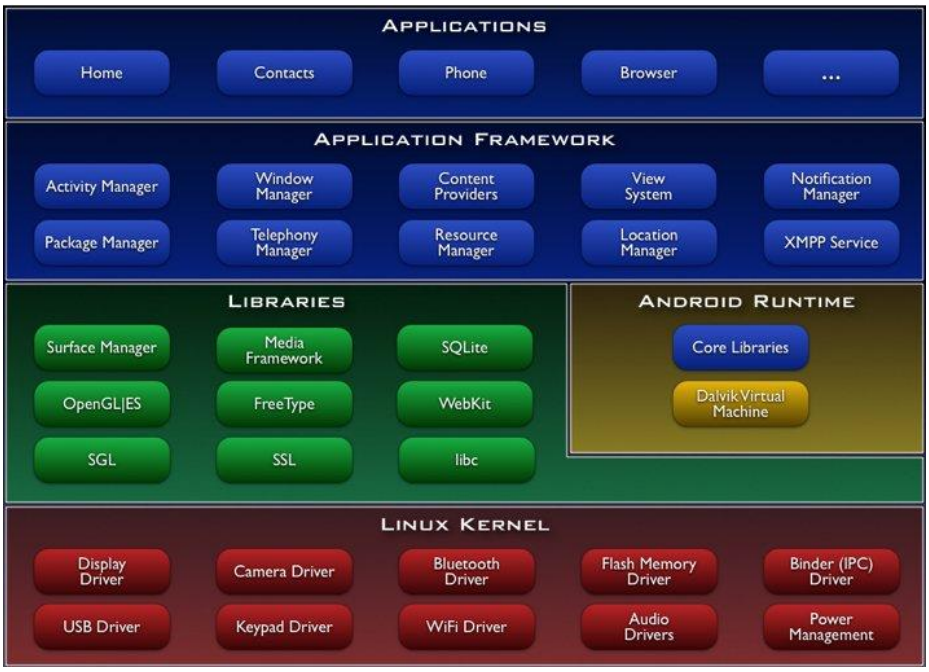


Obrázek 1: Hlavní obrazovka OS Android 4.0 [2]

2.3. Architektura

OS je založen na Linuxovém jádře, tedy i architektura se Linuxu podobá. Android tedy umožňuje běh více aplikací najednou, avšak není možné mít zobrazeno více aplikací na displeji najednou. Vždy je aktivní pouze jedna aplikace přes celou obrazovku a je možné mezi dalšími jednoduše přepínat. Jediným omezením je operační paměť, která je však dobře optimalizována a spravována. Tedy, když je potřeba paměť pro novou aplikaci nebo aktivitu zpracovávající příchozí hovor, operační systém uvolní paměť. Toto probíhá například nuceným ukončením nějaké jiné aplikace, která již delší dobu nebyla aktivní. Další vlastností, která vychází z Linuxu, jsou minimální práva pro aplikace. Toto je důležitý bezpečnostní prvek, je takto ochráněna stabilita systému a citlivé uživatelské informace. Každá aplikace při instalaci žádá uživatele o povolení pro různé funkce, například zápis a čtení Secure Digital (SD) karty, přístup k internetu, přístup ke kontaktům a další. [3]

Operační systém je rozčleněn na následujících 5 základních vrstev.



Obrázek 2: Struktura OS Android [3]

- *Linux kernel*: Obstarává veškerou správu hardwaru zařízení a umožňuje tak vyšším vrstvám jeho použití. Jádro se stará např. o správu paměti a o procesy.
- *Libraries*: Jsou to knihovny napsané v jazyce C/C++, které využijí hlavně vývojáři prostřednictvím rozhraní *Application framework* při vývoji svých

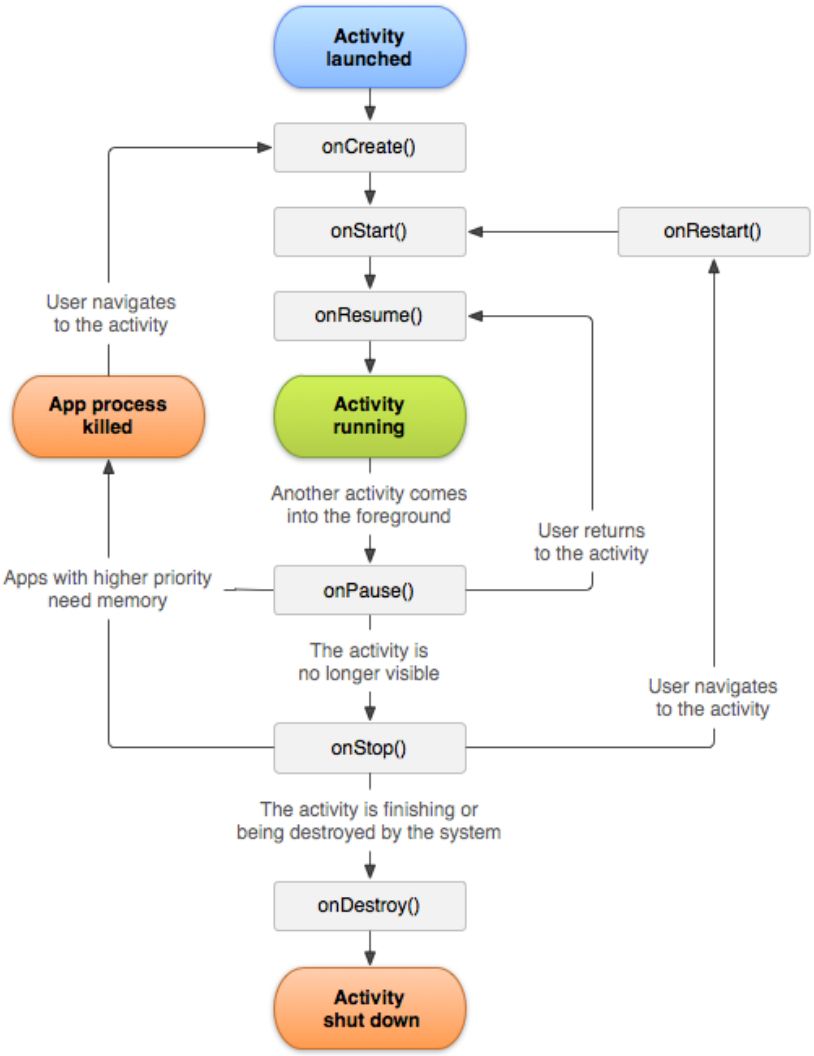
aplikací. Jsou zde např. knihovny pro správu multimediálních souborů, pro práci s SQLite databází a pro vykreslování 3 Dimensions (3D) grafiky.

- *Android Runtime:* Což je v podstatě speciální virtuální stroj, který se jmenuje Dalvik Virtual Machine (DVM). Díky této vrstvě jsou aplikace vyvíjeny v jazyce Java, protože DVM obsahuje základní Java knihovny, které jsou podobné platformě Java SE.
- *Application framework:* Umožňuje přístup ke službám. Tyto služby jsou využitelné při vývoji aplikací, umožňují totiž přístup k datům z jiných aplikací.
- *Applications*: Jedná se o základní aplikace, jinými slovy veškeré aplikace pro systém Android. Aplikace jsou buď před-instalované výrobcem v zařízení, nebo je možné je stahovat v obchodě Google Play.

2.4. Struktura aplikace

Každá aplikace pro tento operační systém se skládá z několika hlavních komponent. [4] Komponentami jsou:

- Aktivity: Jedná se v podstatě o uživatelské prostředí aplikace. Jsou analogií k oknům programů pro počítače. Jedna aktivita odpovídá jedné obrazovce aplikace. Aplikace se tedy skládá běžně z několika aktivit, mezi kterými je přepínáno. Jedna aktivita slouží například k zobrazení Short Message Service (SMS) a další aktivita slouží pro její psaní. Aktivity si také mohou mezi sebou vyměňovat data a reagovat na své stavy. Aktivity je také možné používat i mezi aplikacemi. Je tedy možné využít aktivitu pro výběr kontaktu v jakékoli jiné aplikaci, protože OS tuto aktivitu již obsahuje. Není ji tedy nutné vytvářet znovu, stačí ji pouze zavolat a vybrat pomocí ní kontakt. O správu aktivit se stará *Activity Manager*, který řídí celý životní cyklus aktivit v aplikaci. Aktivity jsou zde uloženy v zásobníku, kde na vrcholu je aktuálně zobrazovaná aktivita. Životní cyklus aplikace lépe popisuje Obrázek 3.



Obrázek 3: Životní cyklus aktivity [5]

Každá aktivita se nachází v jednom ze čtyř následujících stavů:

- Aktivní: Aktivita je spuštěná a zobrazená na obrazovce.
- Pozastavená: Aktivita stále běží a je možné její část vidět na obrazovce, ale je překrytá nějakým upozorněním, které je aktivní, a s původní aktivitou nelze pracovat.
- Zastavená: Aktivita stále běží, ale není zobrazená, protože je spuštěná jiná aktivita, která byla spuštěná, nebo aktivovaná po ní.
- Mrtvá: Aktivita nebyla prozatím spuštěná, nebo byla ukončená.

Pro přechody mezi těmito stavy využívá OS takzvané *callback* metody, které jsou volány při přechodech mezi základními stavy. Povinností je

implementovat metodu *onCreate()*, protože bez ní by nebylo možné aktivitu spustit a zobrazit.

- metoda *onCreate():* Je volána při spuštění aktivity. Je tedy vhodná pro inicializaci, kterou je nutné provést pouze při startu aktivity.
- metoda *onStart():* Je také volána při spuštění aktivity, ale také při přechodu zastavené aktivity do popředí.
- metoda *onResume():* Je také volána při startu aktivity, také je volána při přechodu pozastavené aktivity do popředí.
- metoda *onPause():* Je volána při přechodu aktivní aktivity do pozastaveného stavu, tedy když je částečně překrytá jinou aktivitou či dialogem.
- metoda *onStop():* Je volána při přechodu aktivity do zastaveného stavu.
- metoda *onRestart():* Je volána když byla aktivita zastavena a je restartována.
- metoda *onDestroy():* Je volána při ukončování aktivity, může být ukončena systémem, nebo programově voláním metody *finish()*.

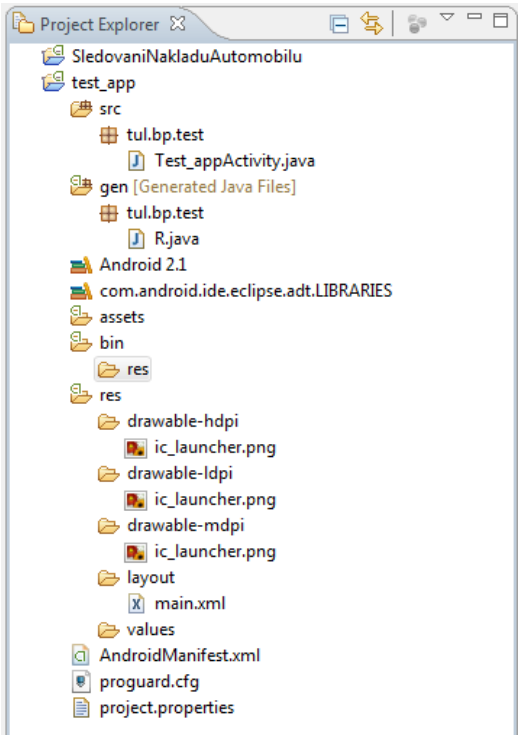
- Služby: Jedná se o procesy, které běží na pozadí a nemají žádné uživatelské prostředí. Využívají se k dlouhotrvajícím úkolům, jako například přehrávání hudby, stahování souborů z internetu nebo zachycení příchozího hovoru.
- Dodavatelé obsahu: Je to aplikační rozhraní pro přístup k datům ostatních aplikací, ale také i v rámci jedné aplikace umožňuje přístup k datům ostatním aktivitám aplikace. Vytvoření a nastavení dodavatelů obsahu umožní rozhodovat o tom, co a jak bude přístupné ostatním aktivitám. Pro ukládání dat se zde využívá souborů a SQLite databáze.
- Záměry: Jsou to systémová oznámení upozorňující aplikace na vznik událostí, jako je příchozí hovor, nízký stav baterie, dokončené stahování. Aplikace na ně potom může reagovat, například dialogem, či upozorněním na stavovém panelu.

2.5. Vývojové prostředí

Pro vývoj aplikací pro Android je potřeba Android SDK a vývojové prostředí. Vše je dostupné zdarma ze stránek www.developer.android.com, kde je také k dispozici celá dokumentace užitečná pro vývoj aplikací. Doporučeným vývojovým prostředím je Eclipse s modulem Android Development Tools (ADT). Po nainstalování Android SDK

a vývojového prostředí s ADT modulem, je nutná aktualizace vybraných knihoven, kterou je nutné zvolit podle toho, pro kterou verzi systému Android chceme vyvíjet. Zde je nutné podotknout, že výběr verze je důležitý. Když je zvolena určitá verze, pro kterou je aplikace vyvíjena, je možné tuto aplikaci spustit na systému novější verze. Bohužel není možné ji spustit ve verzi starší. [6]

Je to dáno tím, že nové verze se stále rozšiřují o nové knihovny, které tedy v předchozích verzích nejsou obsaženy, aplikace by tedy nemusely být funkční bez těchto knihoven. Je dobré tedy zvolit nejstarší možnou verzi vzhledem k potřebným knihovnám, dobré je také přihlížet na podíl jednotlivých verzí mezi uživateli. V roce 2011 byla nejrozšířenější verze 2.2 a druhou nejrozšířenější byla 2.3.x.



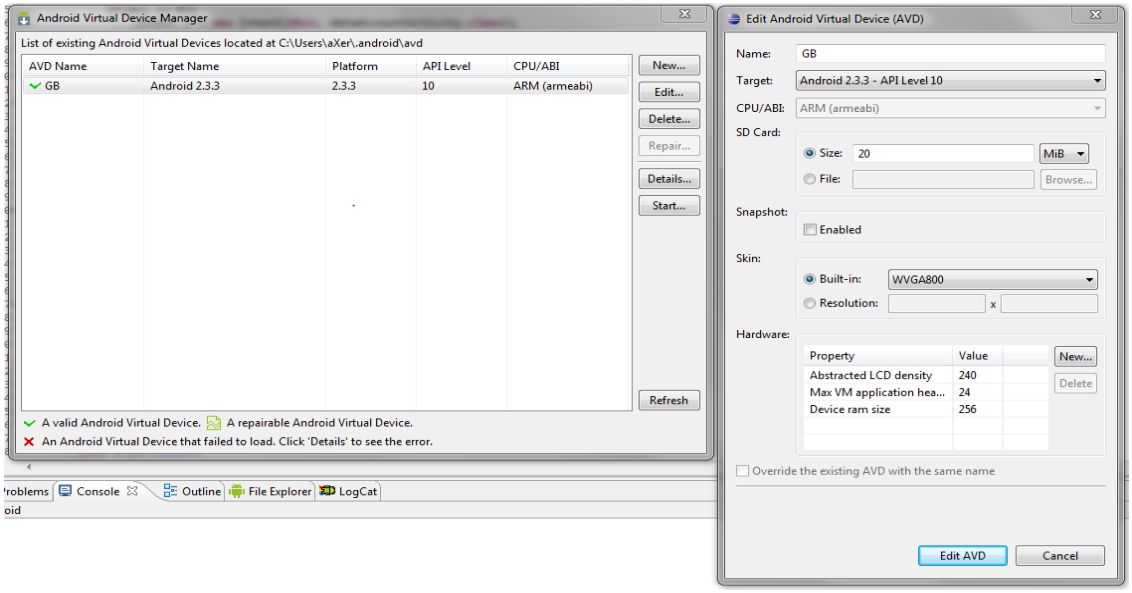
Obrázek 4: Adresářová struktura projektu ve vývojové prostředí Eclipse

Okno vývojové prostředí je rozděleno do čtyř částí. V horní části se nachází tlačítka pro práci s projekty, pro správu nástrojů Android SDK, pro spuštění projektu a další. V první části tohoto panelu je možné přepínat mezi náhledy. Pro vývoj je vhodný náhled s názvem *Java*. V ladícím režimu je naopak dobrý *Debug* pohled, kde je možné sledovat proměnné a stavy aplikace. Na levé straně je zobrazena struktura projektu a jednotlivé soubory, které jsou součástí projektu. Napravo se nachází editor, pod kterým je konzole, výpis chybových hlášek a také užitečný správce souborů na připojeném zařízení pro testování aplikací.

Po úspěšné instalaci a nastavení je nutné si vytvořit virtuální zařízení, na kterém bude možné spouštět vyvíjenou aplikaci. S balíkem Android SDK je nainstalován i manažer těchto zařízení. Manažera je možné nalézt v horní liště vývojového prostředí.

Je možné vytvořit více zařízení. U každého zařízení lze zvolit verzi operačního systému, rozlišení obrazovky a další hardware, který se běžně nachází u reálných zařízení, například fotoaparát, Global Positioning System (GPS) modul, polohovací čidlo, atd. Aplikace je také možné testovat na fyzickém zařízení, které musí být připojeno pomocí Universal Serial Bus (USB) kabelu k počítači a musí být nainstalovány příslušné ovladače pro komunikaci mezi zařízením a vývojovým prostředím.

Využití virtuálního zařízení je dostačující, jediným omezením je rychlost emulátoru. Jelikož je simulován procesor typu Advanced RISC Machine (ARM) na normálním počítači, odezva virtuálního zařízení je delší. Vhodné je určitě testovat aplikace na zařízeních s různým rozlišením, ať již na virtuálních či fyzických, tím je zaručeno dobré zobrazení a rozložení aplikace pro co největší počet zařízení.



Obrázek 5: Správce a vytváření virtuálního zařízení

Dalším krokem je již vytvoření samotného projektu, kde je nutné zadat název projektu, název balíčku a také verzi OS. Nachází se zde možnost vytvoření nové spouštěcí aktivity. Následně je vytvořen nový projekt a všechny náležité soubory a složky, které mají specifickou adresářovou strukturu.

```java
package test.newApp;

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

public class TestActivity extends Activity {
    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
    }
}
```

Ukázka kódu 1: Nová aktivita

Složky:

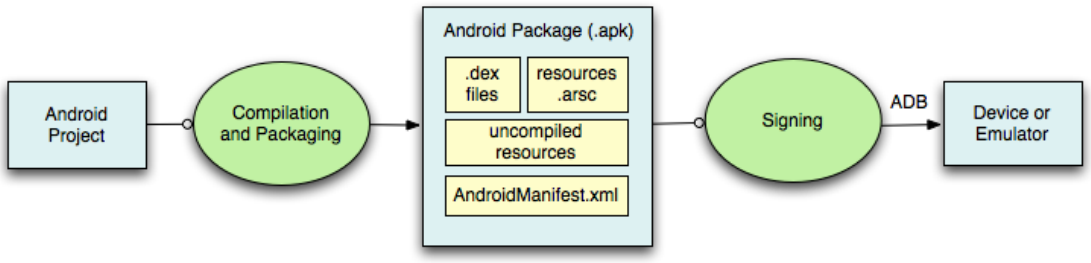
- *assets*: Složka pro statické soubory, které jsou následně přibaleny k aplikaci a je možné je využít v projektu.
- *bin*: Místo, kam se generuje přeložená aplikace, tedy Android application Package file (APK) soubor, který je možné instalovat na zařízení.
- *gen*: Místo, kam je překládán kód automaticky generovaného obsahu, například R.java.
- *libs*: Složka pro vkládání ostatních Java knihoven, které jsou potřeba v projektu.
- *res*: Složka určená pro další obsah potřebný v aplikaci, umísťují se zde grafické soubory, Extensible Markup Language (XML) soubory definující vzhled aktivit, také je zde možné ukládat textové konstanty.
    - *res/drawable*: Grafické soubory aplikace.
    - *res/layout*: XML soubory s grafickým rozvržením aktivit.
    - *res/menu*: XML soubory určující strukturu kontextových menu.
    - *res/values*: Řetězcové konstanty.
- *src*: Složka, kde se nacházejí soubory se zdrojovými kódy aplikace.
- *AndroidManifest.xml*: Dílčí soubor pro celou aplikaci, ve kterém jsou popsány komponenty a všechny součásti aplikace a důležité informace. Jsou zde například aktivity obsažené v projektu, potřebná oprávnění pro správný chod aplikace (zápis na SD kartu, přístup k uloženým kontaktům v zařízení, využití připojení k internetu, atd.), verze aplikace, určení, která aktivita je spouštěcí, nastavení ikony a názvu aplikace (jak bude zobrazena ve výběru aplikací po nainstalování), minimální verze OS, atd. Všechny tyto údaje jsou důležité pro

OS, na kterém bude aplikace instalována. Jsou ale důležité také pro zveřejnění aplikace v obchodě Google Play.

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="LD.BK.SledovaniNakladu"
    android:versionCode="1"
    android:versionName="1.0" >
    <uses-sdk android:minSdkVersion="4" />
    <uses-permission
             android:name="android.permission.WRITE_EXTERNAL_STORAGE"/>
    <application
        android:icon="@drawable/ic_launcher"
        android:label="CarCash" android:debuggable="true">
        <activity
            android:label="CarCash"
            android:name=".SledovaniNakladuAutomobiluActivity"
            <intent-filter >
                <action android:name="android.intent.action.MAIN" />
                <category
                      android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>
</manifest>
```

Ukázka kódu 2: AndroidManifext.xml

Výsledný projekt je zkompilován a zabalen do APK souboru, který obsahuje výsledné zdrojové soubory aktivit ve formátu Dalvik Executable Format (DEX). Ty jsou však nejprve překládány Java překladačem a následně kompilovány do DEX souboru, který lze spustit na virtuálním stroji DVM (každá aplikace má spuštěný vlastní virtuální stroj). Dále jsou obsaženy ostatní přibalené zdrojové soubory a *AndroidManifest.xml*. Výsledný APK soubor je následně podepsán. Poté je možné aplikaci instalovat na všech zařízeních, které mají vyšší verzi OS než je požadovaná minimální, která je uvedena v *AndroidManifest.xml*. [7]



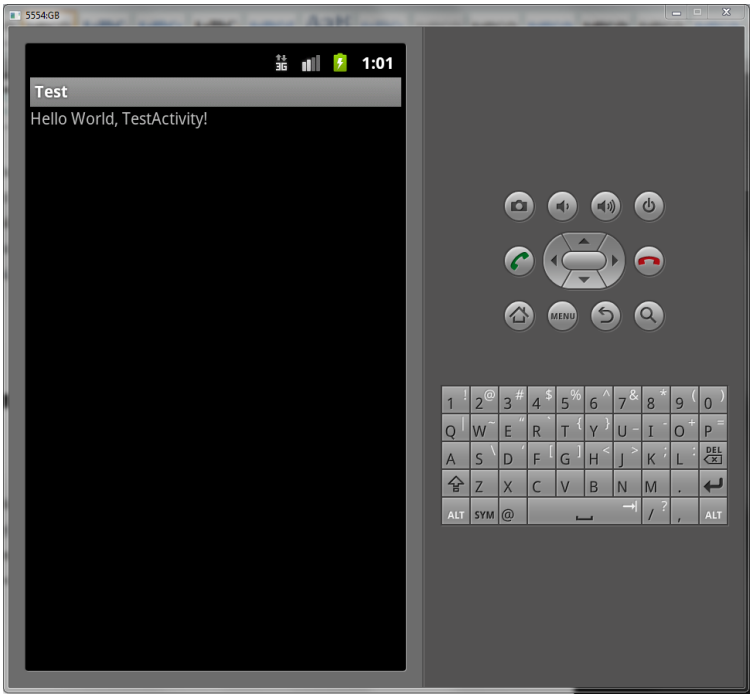
Obrázek 6: Průběh překladu Android projektu [7]

Po založení nového projektu je již možné aplikaci přeložit a spustit na fyzickém zařízení, nebo v emulátoru. Po přeložení se spustí emulátor Android zařízení. Jeho start je poměrně zdlouhavý a náročný na HW počítače. Po načtení emulátoru se ihned spustí projekt, který byl přeložen. V tomto případě nově založeného projektu se jedná o známou aplikaci, která na obrazovku vypíše: „Hello World“.

V levé části emulátoru se nachází vlastní obrazovka zařízení, vpravo se pak nachází klávesnice s rozložením qwertz a nad ní ovládací tlačítka, které se mohou nacházet na zařízeních s OS Android.

Při dalším překladu již není nutné znovu startovat emulátor, pokud tedy nebyl vypnut. V případě, že je emulátor stále aktivní, se aplikace automaticky přeinstaluje na aktuálně překládanou verzi a je spuštěna. Testování funkčnosti je tedy jednoduché a možné při jakýchkoliv úpravách aplikace.

Struktura zdrojových souborů se nijak neliší od klasické Javy, třídy a aktivity jsou řazeny do balíčků.



Obrázek 7: Spuštěná první aplikace na Virtuálním zařízení

2.6. XML návrhy

Zajímavostí jsou zde grafické návrhy založené na XML souborech, tedy oddělení grafické podoby od vlastní logiky aktivit. Je zde určitá podoba s webovými stránkami psanými pomocí HyperText Markup Language (HTML) a Cascading Style

Sheets (CSS). Samozřejmě je možné vytvořit design aplikace přímo ve zdrojovém kódu, avšak použití XML návrhu je značně jednodušší a rychlejší. Již u nově založeného projektu je XML soubor automaticky vygenerován pro spouštěcí aktivitu. Grafické prvky jsou zde stromově řazeny. V případě, že je nutné k prvkům přistupovat v samotném kódu, je jim možné nastavit unikátní id. Android nabízí několik základních komponent, například tlačítko, textový popisek, textová pole, komponenty pro zobrazení obrázků a videa.

Ve vývojovém prostředí je možné vytvářet XML dvěma způsoby. Prvním je přímý zápis kódu, stejně jako při vytváření vlastního kódu aplikace. Druhou možností je použití funkce samotného vývojového prostředí pro vytváření XML souboru. Zde je možné prvky vybírat z nabídky, vkládat je a rozmísťovat na obrazovku aktivity. XML kód je v tomto případě generován automaticky.

Hlavním obalujícím atributem by měl být některý druh rozmístění (Layout), jedná se o typ rozvržení obrazovky. K dispozici jsou například následující: *LinearLayout*, *RelativeLayout*, *TableLayout* a *ScrolView*.

Obsahem těchto layoutů jsou již samotné komponenty. Každé komponenty mají své vlastní parametry, které nalezneme v dokumentaci na internetu. Důležitými parametry jsou velikost a id, pokud ke komponentům potřebujeme přistupovat.

Zvláštností je právě hodnota parametru id, jeho podoba je následující: „@+id/jedinečný_*název_atributu*“. Symbol „@“ značí, že XML analyzátor by měl použít zbytek id, který se nachází za lomítkem. Symbol „+“ znamená, že se jedná o nový identifikační řetězec a měl by být zapsán do automaticky generovaného souboru **R.java**. V samotném kódu se však používá pouze samotný název atributu.

```
<?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:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="@string/hello" />

</LinearLayout>
```

Ukázka kódu 3: XML soubor s rozvržením aktivity

2.7. Soubor R.java

Jedná se o automaticky generovaný soubor, který mapuje všechny soubory konkrétního projektu a přiřazuje jim konstanty, jsou to reference na všechny externí soubory. Konstanty jsou tříděny do tříd podle složek, ve kterých jsou umístěny.

```
package test.hw;

public final class R {
    public static final class attr {
    }
    public static final class drawable {
        public static final int ic_launcher=0x7f020000;
    }
    public static final class layout {
        public static final int main=0x7f030000;
    }
    public static final class string {
        public static final int app_name=0x7f040001;
        public static final int hello=0x7f040000;
    }
}
```

Ukázka kódu 4: Soubor R.java

2.8. Nabídky menu

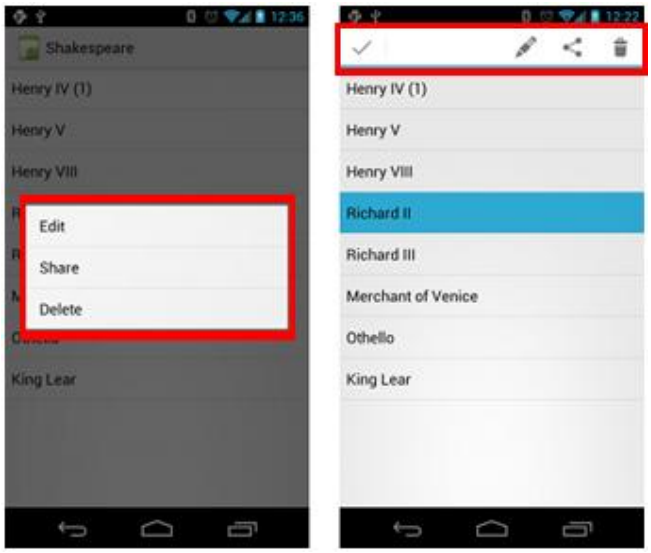
Dalším specifikem Android aplikací je menu voleb, které je vyvoláváno HW klávesou, která byla standardem těchto zařízení spolu s tlačítkem Domů a Zpět. S příchodem nejnovější verze Android 4.x je tento standard změněn a zařízení tak nemusí obsahovat žádné HW klávesy. Ty jsou totiž nově zobrazovány přímo na displeji a jedná se o klávesy Domů, Zpět a tlačítko pro vyvolání seznamu běžících aplikací. Tlačítko Menu se nyní přesunulo na různá místa přímo v aplikacích a má podobu tří teček.

Menu voleb se specifikuje také pomocí XML souboru. Samotné menu je nutné přidat do aplikace přidáním metody *onCreateOptionsMenu()* a metody zpracovávající výběr některé položky z menu *onOptionsItemSelected()*. [8]

```xml
<?xml version="1.0" encoding="utf-8"?>
<menu xmlns:android="http://schemas.android.com/apk/res/android">
    <item android:id="@+id/signoff"
          android:title="Jiný profil" />

    <item android:id="@+id/changeValue"
          android:title="Nastavit kurz" />

    <item android:id="@+id/backup"
          android:title="Zálohovat na SD" />

    <item android:id="@+id/restore"
          android:title="Obnovit ze SD" />
</menu>
```

Ukázka kódu 5: XML soubor s rozvržením menu

Obdobně se vytváří kontextové menu, které se používá například u výčtů. Vyvolání menu tohoto typu je možné po podržení prstu na některém prvku výčtu.



a) Menu v Androidu 2.4.3 **b) Menu v Androidu 4.0**

Obrázek 8: Příklady kontextového menu aplikací [8]

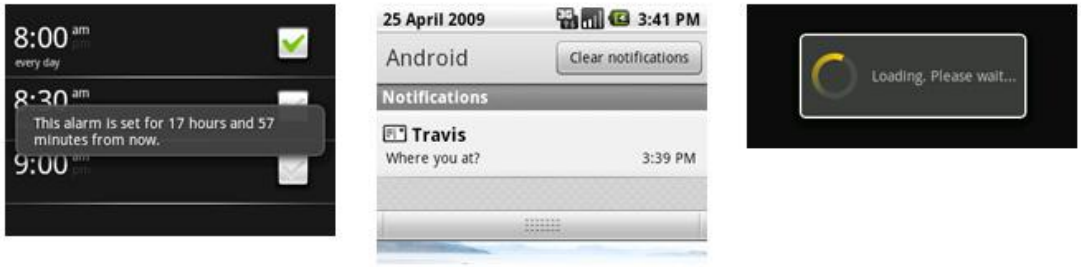
Podoba menu se liší v různých verzích OS (viz Obrázek 8). Ve starších verzích bylo možné zobrazit pouze kontextové menu (viz Obrázek 8a), nově je možnost zobrazit menu v horní liště pomocí grafických symbolů (viz Obrázek 8b).

2.9. Notifikace

Operační systém nebo konkrétní aktivita potřebuje občas dát vědět o nějaké nastalé situaci. K tomuto jsou zde notifikace, které toto umožňují. Některé notifikace vyžadují interaktivitu uživatele, například notifikace o buzení. [9]

Android nabízí následující tři druhy notifikací:

- Bubliny: Jedná se o upozornění, které je zobrazeno v bublině přes aktuálně zobrazenou aktivitu. Zobrazuje se například po aktivaci budíku, aby informovala uživatele, že je budík opravdu zapnutý (viz Obrázek 9a).
- Upozornění ve stavovém řádku: Toto upozornění je zobrazeno v horní stavové liště jako ikona a bývá doprovázeno textovým popisem, který je možné zobrazit při stažení stavové lišty. Při kliknutí na zprávu upozornění je možné spustit příslušnou aktivitu, například příchozí SMS jsou zde notifikovány a při kliku na text je zobrazena celá zpráva (viz Obrázek 9b).
- Dialogy: Jsou zobrazeny v okénku, které částečně překrývá aktuální aktivitu a zobrazují běžně informace o nějakém postupu, například načítání dat (viz Obrázek 9c). Zde se právě jedná o notifikaci, která může vyžadovat aktivitu uživatele, a to je možné přidáním tlačítek do dialogu.



a) Bublina **b) Upozornění ve stavovém řádku** **c) Dialog**

Obrázek 9: Příklady notifikací [9]

3 Praktická část

3.1. Návrh aplikace

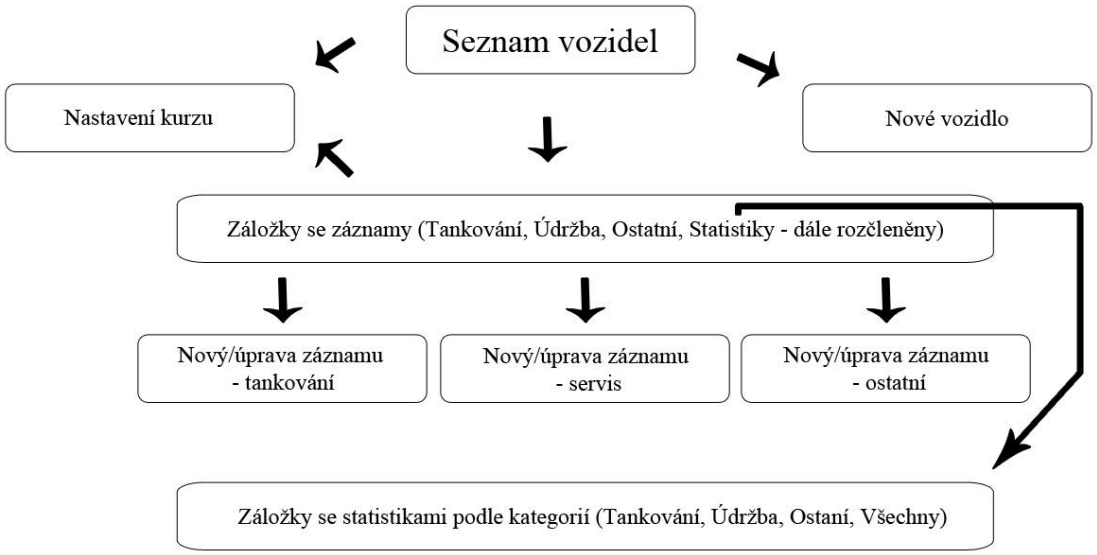
Důležitým rozhodnutím byl výběr platformy, pro kterou bude aplikace navržena a implementována. Od počátku byly volbou operační systémy pro mobilní telefony, a to z prostého důvodu. Mobilní telefon má víceméně každý stále u sebe, takže je možné přidávat záznamy ihned po zakoupení či natankování paliva. Odpadá tedy nutnost zapisovat si stav tachometru a další údaje, stejně tak uchovávání a hromadění účtenek za tankování. Další výhodou je stálý přehled o tankování, ale také o servisu vozidla. Není nutné nikde hledat, kdy bylo co provedeno na vozidle. Vše je uloženo v jedné aplikaci, kde je možné řadit záznamy podle stavu tachometru i podle data. Je tedy jednoduché zjistit, co je aktuálně potřeba provést na vozidle. Aplikace samozřejmě nenahradí účtenky nutné k reklamaci zboží nebo daňové doklady pro služební vozidla pro účetnictví.

Dalším přínosem je automatické počítání průměrné spotřeby a dalších užitečných a zajímavých statistických údajů. Počítání průměrné spotřeby nejvíce ocení majitelé starších vozidel, které ještě nejsou vybaveny palubním počítačem, který by průměrnou spotřebu počítal. Aplikace je také užitečná pro majitele služebních vozů, protože budou mít na jednom místě uchovány všechny záznamy o výdajích vozidla.

Vlastnosti a funkce aplikace vycházejí z rešerše již existujících aplikací a vlastních zkušeností s používáním těchto aplikací. Hlavním cílem je navrhnout komplexní aplikaci pro správu veškerých výdajů týkajících se automobilu v jedné aplikaci a výpočet statistik ze záznamů, které mohou být zajímavé pro uživatele. Důležitá je také možnost zálohy a obnovy uživatelských dat ze souboru (kopie databázového souboru), který bude možné přenášet mezi mobilními telefony, pro případ výměny telefonu či jeho poškození.

3.2. Vlastní implementace

Aplikace je složena z několika aktivit. Každá aktivita obstarává jednotlivé části aplikace. Design jednotlivých aktivit je vytvořen pomocí příslušných XML souborů a tlačítka mají své vlastní grafické ikonky vytvořené přímo pro tuto aplikaci. Barevné schéma aplikace bylo zvoleno do odstínů šedé, aby aplikace vypadala decentně a přehledně.



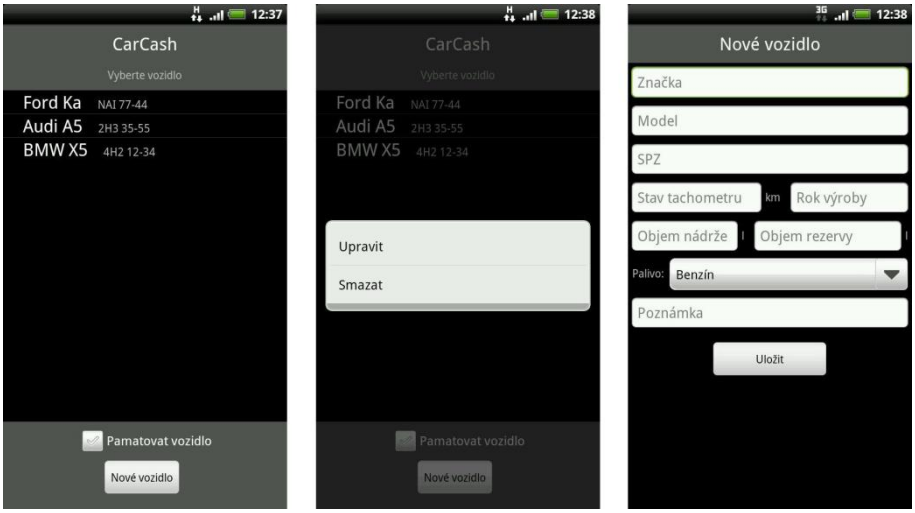
Obrázek 10: Schéma aktivit aplikace

Pro uchovávání vozidel a jejich záznamů je využita interní SQLite databáze. [10] Práce se záznamy je tedy velice jednoduchá pomocí Structured Query Language (SQL) příkazů vykonávaných pomocí metod, které nabízejí standardní knihovny pro Android. Tyto metody vracejí data v objektech třídy *Cursor* [11], což jsou objekty, kde se k datům přistupuje pomocí klíčů, které jsou shodné s klíči v databázi. Lze si je představit jako tabulku v databázi, kde jsou sloupce pojmenovány klíči. V aplikaci je také využito ukládání do binárního souboru a použití *SharedPreferences* (viz dále).

Binární soubory jsou použity pro ukládání vypočítaných statistik, aby je nebylo nutné při každém zobrazení vypočítávat, to by bylo náročné na výkon zařízení a také na jeho výdrž. *SharedPreferences* slouží pro ukládání jednoduchých datových typů a jsou využity pro ukládání identifikátoru aktuálně zvoleného vozidla, pro druh řazení záznamů ve výpisech, atd. Všechny tyto soubory jsou ukládány do paměti zařízení a jsou privátní. Má k nim tedy přístup pouze tato aplikace a takzvaný uživatel *root*. Jedná se o uživatele, který má přístup ke všem prostředkům zařízení, je to obdobné jako u Linuxu. Dnešní zařízení nemají povoleného tohoto uživatele, avšak je možné ho povolit a používat, tímto však dochází ve většině případů ke ztrátě záruky.

Na hlavní obrazovce navrhované aplikace (viz Obrázek 11a) je seznam uložených vozidel s možností jejich editace, mazání a přidávání nových vozidel (viz Obrázek 11b). Pro přidání vozidla (viz Obrázek 11c) je nutné zadat výrobce vozidla, model, státní poznávací značku (SPZ), aktuální stav tachometru, rok výroby, objem nádrže, objem rezervy, druh paliva a je možné vyplnit poznámku k vozidlu. Objem

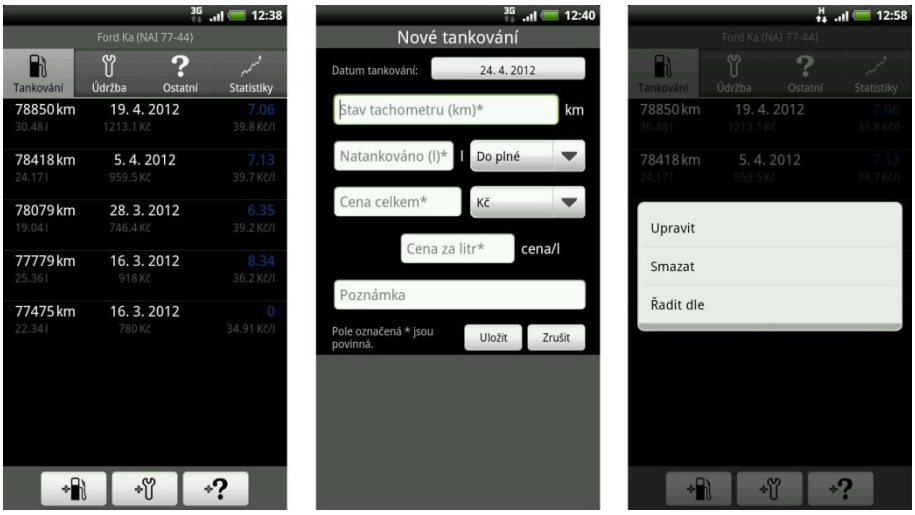
nádrže a rezervy jsou obzvláště důležité pro počítaní spotřeby. Je tedy dobré je vyplnit co nejpřesněji. Tyto veškeré údaje je možné zobrazit pouze při editaci, není nutné je nikde více zobrazovat, jsou spíše vhodné pro uživatele pro pořádek mezi vozidly. Při výběru vozidla je zobrazen v seznamu vozidel výrobce, model a SPZ, což pro jednoznačnou identifikaci stačí, jelikož SPZ je unikátní pro každé vozidlo.



a) Úvodní obrazovka **b) Menu záznamu** **c) Nové vozidlo**

Obrázek 11: Aplikace – Úvodní aktivity

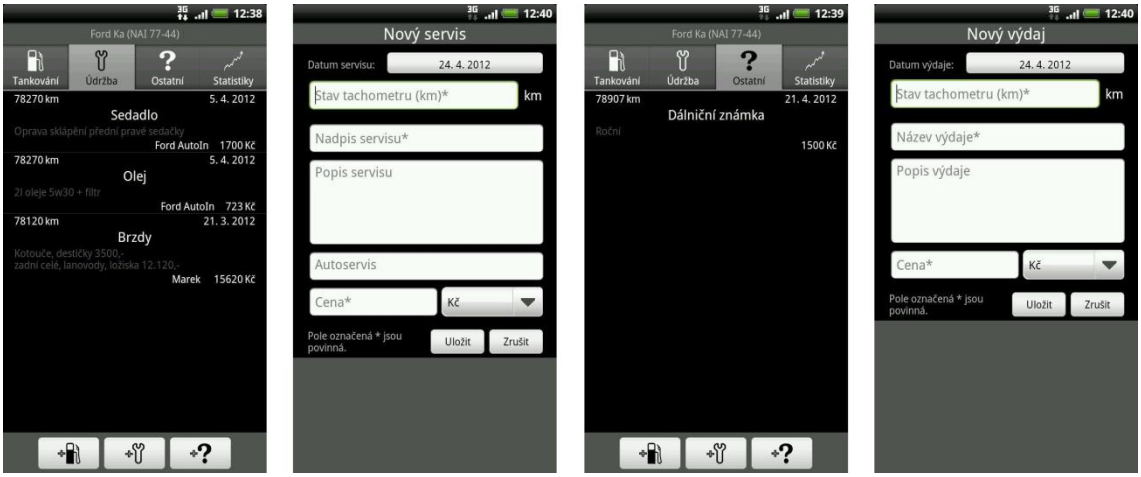
Následně po vytvoření a vybrání profilu vozidla se aplikace přesměruje na obrazovku vypisující záznamy pro vybrané vozidlo rozčleněny do sekcí **Tankování**, **Údržba** a **Ostatní** (viz Obrázek 12a). Tato obrazovka má však 4 záložky, poslední záložkou jsou **Statistiky**, které jsou také rozděleny na 4 záložky, první tři dle sekcí záznamů a poslední **Celkové statistiky** (viz Obrázek 14). Na záložkách se záznamy jsou vždy v dolní části aplikace tři tlačítka pro přidání záznamu do určité sekce. Přidání jakéhokoliv záznamu vyžaduje zadání některých povinných atributů a některých volitelných (pole označená „*“ jsou povinná). Aktivity pro přidání záznamů se také liší svoji podobou a obsahem, podle toho, do které sekce má být záznam přidán. Pro každý druh záznamu je tedy vytvořena vlastní aktivita s odlišnou podobou. Tyto aktivity jsou využity i pro editaci záznamů, pouze jsou zde vyplněna pole dle uložených dat v databázi.



a) Výpis záznamů **b) Nové tankování** **c) Menu záznamu**

Obrázek 12: Aplikace – sekce Tankování

Veškeré záznamy je samozřejmě možné editovat a mazat. Také je možné přidávat záznamy, které patří chronologicky mezi již zadané, to však vyžaduje úpravu některých stávajících záznamů. Záznamy je nutné upravovat pouze v sekci tankování, a to z důvodu, že se změní průměrná spotřeba, která je počítána pro každý záznam. V případě vložení, změny nebo smazání některého záznamu je následující záznam ovlivněn a je nutné ho aktualizovat, toto probíhá automaticky a uživatel nemusí nic potvrzovat.



a) Výpis záznamů **b) Nový záznam** **c) Výpis záznamů** **d) Nový záznam**

Obrázek 13: Aplikace – sekce Údržba a Ostatní

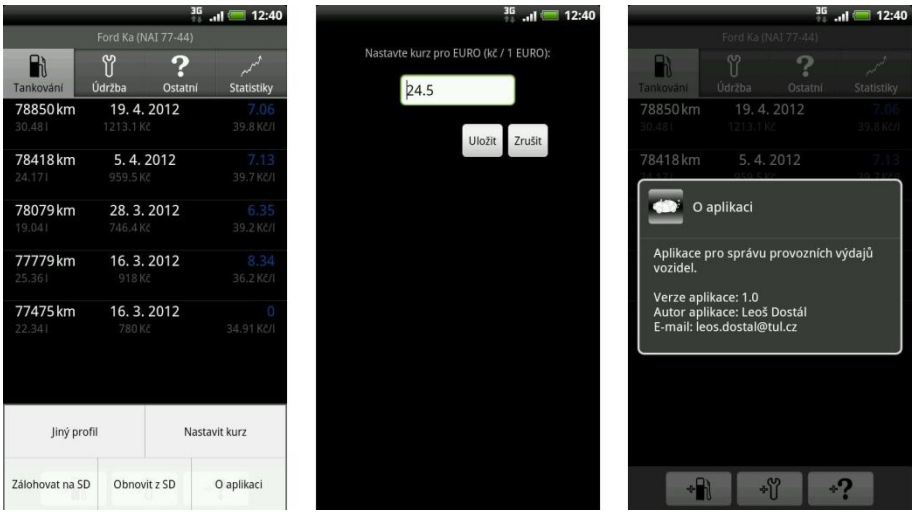
Aktivity pro přidání a editaci záznamů jsou ošetřeny tak, aby byla zadávána data, která odpovídají jak vlastnostem vozidla vyplněným v jeho profilu, tak ostatním záznamům v příslušné tabulce.

a) Statistiky Tankování **b) Statistiky údržby** **c) Ostatní statistiky** **d) Celkové statistiky**

Obrázek 14: Aplikace – sekce Statistiky

U výpisu záznamů je možné měnit jejich řazení v každé sekci podle příslušných atributů, které se tam vyskytují. V sekci **Tankování** lze řadit záznamy dle: data zadání, kilometrů, ceny a spotřeby. V sekci **Údržba** to jsou: datum zadání, kilometry, cena a názvy servisů. V sekci **Ostatní**: datum zadání, kilometry a cena.

V menu aplikace jsou následující možnosti: **Jiný profil**, **Nastavit kurz**, **Zálohovat na SD**, **Obnovit z SD** a **O aplikaci** (viz Obrázek 15a). Volba **Jiný profil** odhlásí aktuálně vybraný profil a přesměruje aplikaci na úvodní obrazovku s výběrem vozidel. Volba **Nastavit kurz** vyvolá novou aktivitu (viz Obrázek 15b), kde je pouze možné zadat kurz pro převod eura na koruny, podle kterého jsou přepočítávány záznamy zadané v eurech. Možnost **Zálohovat na SD** vytvoří kopii databáze na SD kartu, odkud si ji může uživatel zkopírovat například do počítače či jiného telefonu. Naopak volba **Obnovit z SD** vytvoří kopii souboru z SD karty do interní paměti zařízení na místo, kde je uložená původní databáze aplikace. Tato volba způsobí vypnutí aplikace, aby se zamezilo nechtěnému pádu, protože soubor databáze je neustále využíván aplikací a při jeho přepisu by mohlo dojít k pádu aplikace. Poslední možnost **O aplikaci** (viz Obrázek 15c) zobrazí pouze dialog s informacemi o verzi aplikace a autorovi.



a) Menu aplikace **b) Nastavení kurzu** **c) O aplikaci**

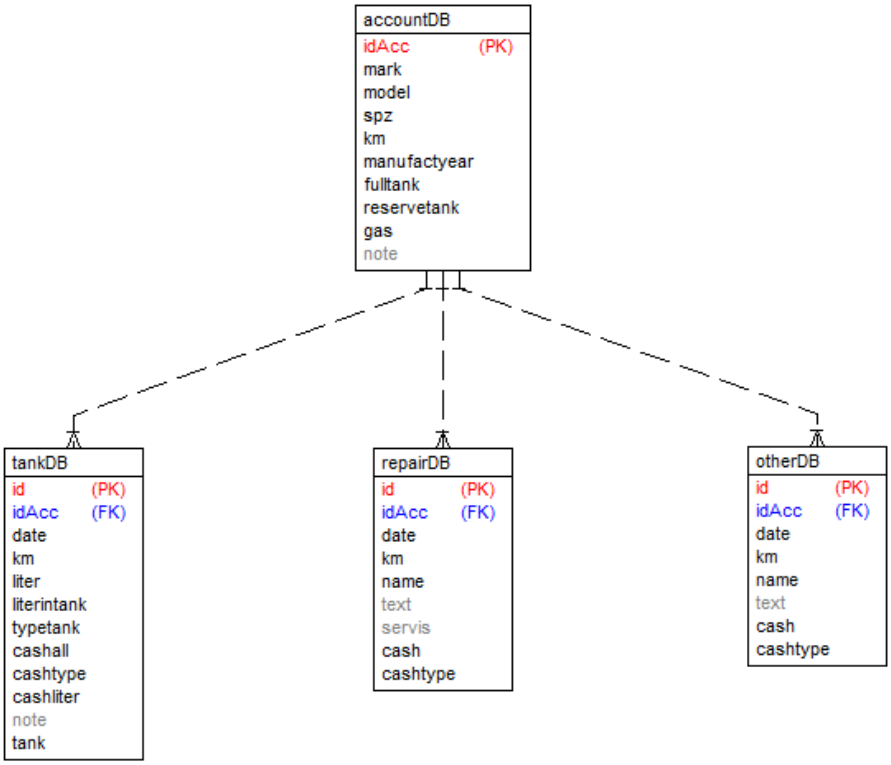
Obrázek 15: Aplikace – další aktivity

3.3. Zpracování záznamů

3.3.1. Databáze a tabulky

Pro ukládání záznamů a vozidel je použita SQLite databáze, která je již implementována v samotném Androidu. Jedná se o zjednodušenou SQL databázi, která nevyužívá k ukládání a uchovávání dat server, ale vlastní soubor. Tím je ulehčena možnost zálohování a obnovení uživatelských dat. Je možné jednoduše tento soubor zkopírovat respektive přepsat, není nutné vkládat jednotlivé obnovované záznamy.

Tabulky byly navrženy tak, aby nejlépe vyhovovaly aplikaci a byly uchovávány jen nejpotřebnější údaje, které mají nějaký vypovídající obsah pro uživatele. Struktura tabulek je pro lepší představu zobrazena na obrázku (viz Obrázek 16).



Obrázek 16: Databázové tabulky aplikace

- tabulka **accountDB**: V této tabulce jsou uchovávány nejdůležitější údaje o vozidlu, které je nutné znát pro potřeby této aplikace. **IdAcc** je zde pouze pro jednoznačnou identifikaci vozidla. Atributy **mark** a **model** jsou zde pro uložení automobilky a modelu vozidla. Slouží pouze uživateli, aby nemusel vozidla rozeznávat pouze podle SPZ. Atribut **km** (stav tachometru) je důležitý pro kontrolu při zadávání nových záznamů, je kontrolováno, zda je stav tachometru zadávaného záznamu větší než tento zadaný při vytváření profilu vozidla, tato kontrola je vhodná například v případě překlepu a zadání nesprávného stavu tachometru. Podobně je to s atributem **manufactyear** (rok výroby), bylo by nemožné zadávat výdaje z data staršího než je automobil vyrobený. Atributy **fulltank** a **reservetank** (objem nádrže a rezervy) jsou opět nutné pro výpočet průměrné spotřeby, bez těchto údajů by ji nebylo možné spočítat. **Gas** (druh paliva) a **note** (poznámka) je čistě jen informativní pro uživatele, tyto údaje nemají žádný vliv na záznamy.
- tabulka **tankDB**: Slouží pro uchování záznamů o tankování pro všechny vytvořené vozidla. **Id** je použito pouze pro jednoznačnou identifikaci záznamu. **IdAcc** je použito pro identifikaci záznamu k příslušnému vozidlu. Podle tohoto

identifikátoru jsou selektovány záznamy při zobrazování. **Date** (datum) a **km** (stav tachometru) slouží informativně pro uživatele a v aplikaci hlavně pro kontrolu záznamu. Kontroluje se, zda je možné takový záznam chronologicky zařadit mezi stávající. Například není možné, aby byl stav tachometru 100 km dne 5. 5. 2012, když byl stav tachometru 150 km dne 1. 1. 2012. Dále je použit stav tachometru pro výpočet průměrné spotřeby, respektive rozdíl stavů tachometrů dvou za sebou jdoucích záznamů. **Liter** (natankovaný objem) a **literintank** (zůstatek v nádrži) jsou použity pro výpočet tankování. Druh výpočtu určuje atribut **typetank** (typ tankování). **Cashall** (cena celkem) je nutná pro výpočet průměrné spotřeby a **cashliter** (cena za litr) je automaticky vypočítávána při zadávání záznamu a slouží pouze informativně pro uživatele a pro kontrolu s údaji na účtence. **Cashtype** (měna) slouží pro uchovávání informace v jaké měně je záznam uložen. Do atributu **note** (poznámka) je možné zapsat další aspekty, které mohly ovlivnit spotřebu, například použití zimních pneumatik nebo střešního nosiče. Do atributu **tank** (průměrná spotřeba) je ukládána vypočítaná hodnota z předchozích atributů.

- tabulka **repairDB**: Slouží pro uchování záznamů o údržbě, také pro všechna vozidla. **Id** je také použitou pouze pro jednoznačnou identifikaci záznamu. Stejně tak **idAcc** je použito pro identifikaci záznamů příslušného vozidla. **Date** (datum) a **km** (stav tachometru) je zde opět pro kontrolu a pro uživatele, aby měl přehled, kdy byla údržba vykonána. **Name** (název) je zde jednoduše pro krátké pojmenování údržby a **text** (popis) slouží pro popsání údržby. **Servis** (autoservis) slouží pro uchování jména autoservisu, kde byla údržba provedena. Toto je vhodné například, pokud je potřeba něco reklamovat, ihned je jasné, kde byla údržba vykonána. **Cash** (cena) a **cashtype** (měna) slouží pro uchování ceny a měny údržby.
- tabulka **otherDB**: Slouží pro uchování ostatních záznamů. Tabulka je téměř totožná s tabulkou pro údržbu, pouze zde není atribut **servis**.

Pro práci s databází je v aplikaci obsažena třída, která se stará o všechny manipulace s databází. Při startu aplikace zajišťuje tato třída kontrolu, zda již databáze existuje. Pokud ano, tak vytvoří spojení s databází a je možné její použití pomocí metod v této třídě. V opačném případě je vytvořena nová databáze s příslušnými tabulkami podle předchozího schématu. Toto nastává pouze při první instalaci aplikace, nebo

v případě vymazání uživatelských dat aplikace, což je možné ve správě aplikací v OS. V případě reinstalace aplikace nebo aktualizace na novější verzi jsou data zachována a databázi není nutné znovu vytvářet.

```
private static final String DATABASE_ACCOUNT_CREATE =
        "create table accountDb (_id integer primary key autoincrement,"
      + "mark text not null, model text not null, spz text not null,"
      + "km integer not null, manufactyear integer not null, fulltank"
      + "integer not null, reservetank integer not null,"
      + "gas text not null, note text);";
```

Ukázka kódu 6: Textový řetězec s SQL příkazem jedné tabulky

Textový řetězec pro vytvoření tabulky v databázi není nijak složitý. Jedná se o normální SQL příkaz (viz Ukázka kódu 6), který je vykonán pomocí metody *execSQL*, které předáme pouze řetězec s SQL příkazem. Tato metoda je implementována v OS.

Přidání záznamu probíhá podobně. Do metody jsou předány všechny parametry, které odpovídají atributům v tabulce. Ty jsou následně zařazeny do struktury, kde každému klíči z databáze je přiřazena jeho hodnota. Následně jsou data pomocí metody *insert* vložena do tabulky. Metodě *insert* je předán řetězec se jménem tabulky a struktura dat (viz Ukázka kódu 7). Tato metoda vrací identifikátor nově vloženého řádku.

```
public long createAccount(String mark, String model, String spz, int km,
  int year, int ftank, int rtank, String gas, String note) {
        ContentValues initialValues = new ContentValues();
        initialValues.put(KEY_MARK, mark);
        initialValues.put(KEY_MODEL, model);
        initialValues.put(KEY_SPZ, spz);
        initialValues.put(KEY_KM, km);
        initialValues.put(KEY_MANUFACTYEAR, year);
        initialValues.put(KEY_FULLTANK, ftank);
        initialValues.put(KEY_RESERVETANK, rtank);
        initialValues.put(KEY_GAS, gas);
        initialValues.put(KEY_NOTE, note);
        return aDb.insert(DATABASE_TABLE, null, initialValues);
}
```

Ukázka kódu 7: Metoda pro přidání vozidla do databáze

Editace záznamu probíhá stejným způsobem, pouze je navíc metodě předán identifikátor záznamu, který má být upraven, a místo metody *insert* je použita metoda *update*, které jsou předány následující parametry: řetězec se jménem tabulky, struktura

s daty pro úpravu a podmínka, která určuje konkrétní záznam (viz Ukázka kódu 8). Tato metoda vrací počet ovlivněných řádků.

```
aDb.update(DATABASE_TABLE, updateValues, KEY_ROWID + "=" + rowId, null)
```

Ukázka kódu 8: Příkaz pro úpravu záznamu v databázi

Pro mazání záznamu je nutné znát pouze jeho identifikátor (viz Ukázka kódu 9). Následně je nutné zavolat metodu *delete*, které jsou v ukázce níže předány následující parametry: řetězec s názvem tabulky a podmínka určující jeden konkrétní záznam. Metoda vrací počet smazaných řádků.

```
public boolean deleteAccount(long rowId) {
    return aDb.delete(DATABASE_TABLE, KEY_ROWID + "=" + rowId, null);
}
```

Ukázka kódu 9: Metoda pro smazání záznamu z databáze

Pro výpis záznamů je opět nutné znát identifikátor vozidla, pro které se mají záznamy vypsat. Metodě se předává identifikátor vozidla a dle něho je vrácen *Curosor* se záznamy. *Cursor* je jakási struktura podobná databázi. Každému klíči z databáze je přiřazena hodnota. Pro výběr z databáze je použita metoda *query*. Ukázka kódu 10 ukazuje metodu pro získání všech záznamů o tankování, které jsou řazeny podle stavu tachometru sestupně. Metodě *query* je předán řetězec obsahující jméno tabulky, dále pole řetězců s klíči, které chceme získat z databáze, následně podmínka, která vybere pouze záznamy, které mají předané id a nakonec atribut, podle kterého budou data řazena.

```
public Cursor fetchAllTankTagKm(long id) {
   Cursor tCursor = aDb.query(true, DATABASE_TANK_TABLE, new String[]
     {KEY_ROWIDTANK, KEY_DATETANK, KEY_KMTANK, KEY_LITERTANK,
     KEY_CASHALL, KEY_CASHTYPE, KEY_CASHLITER, KEY_NOTETANK,
     KEY_TANK, KEY_LITERINTANK, KEY_TYPETANK},
     KEY_ROWIDACCAUNT + "=" + id, null, null, null,
     KEY_KMTANK + " DESC", null);
   return tCursor;
}
```

Ukázka kódu 10: Metoda pro výběr záznamu z databáze

3.3.2. Zpracování dat v objektech třídy *Cursor*

Jsou vhodné pro uchovávání a zpracovávání dat, které vracejí metody z databáze. Reprezentují data pomocí jejich klíčů z databáze. Pro získání dat do jednoduchých proměnných z objektu třídy *Cursor* je nutné *cursor* projít cyklem, kde kontrolujeme, zda je již konec *cursoru*. V každém kroku cyklu je přístupný vždy jeden řádek z databáze a pomocí klíčů můžeme data získat do jednotlivých proměnných, důležité je zachovat datové typy. Po přečtení dat je *cursor* posunut na další řádek. Po přečtení celého *cursoru* je nutné ho uzavřít (viz Ukázka kódu 11).

```
Cursor o = tDbHelper.fetchAllTankTagKm(aRowId);
o.moveToFirst();
int pocetTank = o.getCount();
int [] kmsT = null;
float [] tanksT = null;
if (pocetTank != 0){
       kmsT = new int [pocetTank];
       tanksT = new float [pocetTank];
       int i = 0;
       while (o.isAfterLast() == false) {
             kmsT[i] = o.getInt(o.getColumnIndexOrThrow(Db.KEY_KMTANK));
             tanksT[i] = o.getFloat(o.getColumnIndexOrThrow(Db.KEY_TANK));
             i++;
             o.moveToNext();
       }
o.close();
}
```

Ukázka kódu 11: Zpravování objektu třídy Cursor

3.3.3. Kontroly zadávaných údajů

Samozřejmostí jsou kontroly prázdných políček ve všech aktivitách. Kontroly probíhají jen u povinných políček. Kontroluje se pouze délka zadaného řetězce, aby u polí, která nesmějí být nulová v databázi, byl nějaký řetězec či číslo. Na všechny problémy je uživatel upozorněn zobrazením upozorňující bubliny, která sama zmizí. Při vytváření nového vozidla je kontrolováno vyplnění všech údajů kromě poznámky. Při přidávání záznamů jsou kontrolovány údaje označené hvězdičkou v příslušných aktivitách.

Nejdůležitější kontroly probíhají při přidávání jednotlivých záznamů. Nejdůležitější kontrolou je chronologie záznamů (viz Ukázka kódu 12), ta je kontrolována ve všech sekcích. Tato kontrola se týká zadaného stavu tachometru a data, mezi nimiž je vazba. Jak bylo psáno výše.

```
public boolean getFit(int [] k, String [] d, String date, String km)
                                        throws ParseException{
   SimpleDateFormat cF = new SimpleDateFormat("yyyy-MM-dd");
   int hd = -1;
   int ld = -1;
   long datee = (cF.parse(date)).getTime();
   long dd;
   for (int i = 0; i < k.length; i++){
      dd = (cF.parse(d[i])).getTime();
      if(dd < datee){
         ld = i;
         break;
      }
   }
   for (int i = k.length-1; i >= 0; i--){
      dd = (cF.parse(d[i])).getTime();
      if(dd > datee){
         hd = i;
         break;
      }
   }
   int kmm = Integer.valueOf(km);
   boolean ano = true;
   if(ld >= 0){
      if(k[ld] > kmm){
         ano = false;
      }
   }
   if(hd >= 0){
      if(k[hd] < kmm){
         ano = false;
      }
   }
   return ano;
}
```

Ukázka kódu 12: Metoda kontrolující chronologii záznamů

Metoda přebírá pole se stavy tachometrů již uložených záznamů, toto pole je seřazeno sestupně dle stavů. Dále přebírá pole s daty záznamů, která jsou řazena tak, aby odpovídala stavům tachometru. Dále je zde kontrolovaný stav tachometru a příslušné datum.

V prvních dvou cyklech je nalezen index následujícího a předchozího záznamu podle data. Následně pokud je nalezen předchozí záznam, je jeho stav tachometru porovnán s vkládaným a pokud není vkládaný stav větší, metoda vrací *false*. Stejně tak to je v případě následujícího záznamu. Pokud je stav tachometru následujícího záznamu nalezen, je jeho stav tachometru porovnán s vkládaným a pokud je vkládaný větší, metoda vrací *false*.

Další kontrolou ve všech sekcích je porovnání data přidávaného záznamu s rokem výroby vozidla. Dále pak stav tachometru u přidávaného záznamu vůči stavu tachometru zadaném v profilu vozidla.

V případě přidávání záznamu, kde je zvolena měna v eurech, je kontrolováno, zda je nastaven kurz pro převod, pokud není je nutné ho nejprve nastavit, aby proběhl převod na koruny.

Poslední společnou kontrolou pro všechny sekce je kontrola budoucího data (viz Ukázka kódu 13). V tomto případě je porovnáváno aktuální datum s datem záznamu a pokud zadané datum je budoucí, je na to při ukládání záznamu uživatel upozorněn vyskakujícím dialogem.

```
SimpleDateFormat cF = new SimpleDateFormat("yyyy-MM-dd");
long datumSave = (cF.parse(fce.datum(date))).getTime();
long datumToday = (cF.parse(dd)).getTime();
if(datumToday < datumSave){
    Toast.makeText(newTankActivity.this,
       "Pozor: Zadané datum je budoucí.", Toast.LENGTH_SHORT).show();
}
```

Ukázka kódu 13: Kontrola budoucího data

Další kontrolou v sekci tankování je kontrola, zda se vejde zadané natankované množství do nádrže. To probíhá jednoduchým porovnáním hodnot objemu nádrže a zadaného množství v případě tankování do plné, v případě tankování k rezervě je k zadanému množství přičten objem rezervy a výsledek porovnán s objemem nádrže.

3.3.4. Výpočet spotřeby

Výpočet spotřeby je možný různými způsoby, které se odvíjejí od způsobu tankování. Implementovaná aplikace umožňuje dva způsoby tankování, a to následující: natankováno do plné a dojeto do rezervy. Druh tankování je pro každý záznam možné měnit. Při každém přidání záznamu či jeho editaci je vždy nutné najít předchozí záznam, protože je nutné zjistit jakým způsobem bylo v předchozím tankování tankováno a případně některé hodnoty z předchozího záznamu mohou být použity k výpočtu. Také je nutné nalézt následující záznam pokud existuje. Toto je nutné z důvodu změny ujetých kilometrů a tím i průměrné spotřeby, ta musí být také přepočítána.

- První způsob tankování je stále do plné nádrže, tento styl tankování dává nejpřesnější a nejjednodušeji vypočitatelnou průměrnou spotřebu. V tomto případě je průměrná spotřeba počítána z natankovaného množství paliva, protože pokud je vždy tankováno do plné, tak obsah natankovaný odpovídá obsahu, který byl spálen mezi tankováními. Pro výpočet spotřeby je tedy nutné zjistit z databáze pouze stav tachometru předchozího záznamu. Odečtením stavu tachometru předchozího záznamu od stavu tachometru přidávaného záznamu je získán počet ujetých kilometrů. Spotřeba je spočítána jednoduše vydělením natankovaného množství paliva počtem kilometrů a vynásobením výsledku stem.
- Druhým způsobem tankování je dojetí vždy do rezervy. Přesnější by mohlo být zadání zůstatku paliva v nádrži než pevně počítat s rezervou, ale drtivá většina vozidel má pouze ručičkový ukazatel stavu nádrže. Není tedy možné určit přesný zbytek paliva v nádrži, proto je nejlepší tankovat vždy do plné. V tomto případě je stejně jako v předchozím vypočítán počet ujetých kilometrů rozdílem stavů tachometrů přidávaného záznamu a předchozího. Avšak množství paliva pro výpočet průměrné spotřeby je množství natankované v předchozím záznamu. Pokud bylo v předchozím záznamu dojeto do rezervy, tudíž v nádrži byl po natankování objem rezervy a natankovaný objem. Když tedy následně bylo dojeto opět do rezervy, bylo spáleno pouze natankované množství odpovídající předchozímu záznamu. Výsledná průměrná spotřeba je již spočítána stejně jako v předchozím případě, tedy objem paliva vydělený počtem ujetých kilometrů a následně vynásobený stem.
- Třetím případem je aktuální tankování do plné nádrže a předchozí tankování, kdy bylo dojeto do rezervy. Počet ujetých kilometrů je počítán opět stejně rozdílem stavů tachometrů. Stejně tak je počítána i spotřeba, pouze je jinak vypočítáván objem paliva. Ten je vypočítán následujícím vzorcem: předchozí stav nádrže - (plná nádrž - natankované množství). Předchozí stav nádrže je v tomto případě objem rezervy plus natankovaný objem v předchozím záznamu. Odečtením aktuálně natankovaného objemu od plné nádrže je zjištěn stav nádrže před tankováním. Odečtením stavu nádrže před tankováním od stavu nádrže po předchozím tankování je získán objem paliva, který byl mezi záznamy spálen.

- Posledním případem je aktuální tankování, kdy bylo dojeto do rezervy a předchozí tankování do plné nádrže. Počet ujetých kilometrů a výsledná průměrná spotřeba je opět počítána stejně. Objem spáleného paliva je rozdíl objemu plné nádrže a objemu rezervy. Protože když bylo natankováno do plné, byla tedy analogicky plná nádrž a následně bylo dojeto do rezervy, tudíž rozdíl těchto objemů je výsledkem.

V každém z případů je nejprve hledán předchozí a následující záznam, pokud existují. Předchozí záznam je vyhledán a jsou z něho použity údaje pro výpočet průměrné spotřeby. V případě nalezení následujícího záznamu je jeho průměrná spotřeba přepočítána pomocí údajů z aktuálně vkládaného záznamu, protože ten se stal předchozím pro upravovaný záznam. Výpočet již probíhá stejně, opět je hledán jeden ze čtyř vztahů mezi záznamy a podle toho je přepočítána průměrná spotřeba.

3.3.5. Výpočty statistik

Aplikace provádí výpočty statistik, které vycházejí z uložených záznamů. Statistiky jsou rozděleny do sekcí stejných jako záznamy, navíc je zde sekce s celkovými statistikami. Statistiky jsou uloženy v souboru v paměti telefonu, a to z důvodu úspory výpočtů. Je zbytečné, aby byly statistiky stále dokola vypočítávány, aniž by nastaly nějaké změny. Statistiky jsou tedy vždy načteny ze souboru, každý účet(automobil) má vlastní soubor a také každá tabulka má vlastní soubor. Každý učet má tedy 4 soubory se statistikami. Statistiky jsou přepočítány pouze v případě přidání či editace některého záznamu. Přepočítávána je vždy pouze sekce, ve které došlo ke změně a samozřejmě celkové statistiky, protože ty jsou ovlivněné všemi sekcemi.

- sekce **Tankování**: Osahuje následující statistické údaje: celková průměrná spotřeba, průměrná cena za tankování, průměrná cena za litr, celkový počet ujetých kilometrů, celkový počet natankovaných litrů, celková cena všech tankování, počet tankování, minimální cena za litr, maximální cena za litr a průměrná cena za ujetý kilometr. Většina údajů jsou pouze průměry ze zadaných hodnot, nebo celkové součty hodnot a minimální a maximální prvky. Zajímavou hodnotou je zde průměrná cena za ujetý kilometr. Ta je počítána z celkového počtu ujetých kilometrů a celkové ceny záznamů. Tato hodnota je vhodná například pro výpočet ceny nějaké plánované cesty.

- sekce **Údržba**: Obsahuje následující údaje: průměrná cena oprav, celková cena oprav, počet oprav a průměrná cena za kilometr. Poslední údaj je opět počítán z celkové ceny oprav a ujetých kilometrů.
- sekce **Ostatní**: Obsahuje stejné údaje jako sekce údržba.
- sekce **Celkové statistiky**: Obsahuje celkovou cenu všech záznamů a celkovou průměrnou cenu za kilometr, jedná se pouze o součet průměrných cen za kilometr ze všech předchozích sekcí. Dále jsou zde pouze informativní údaje: možný počet kilometrů, které je možné ujet na plnou nádrž, možný počet kilometrů, které je možné ujet na poslední natankované množství a stav tachometru, kdy by mělo být opět natankováno. První dva údaje jsou počítány pomocí celkové průměrné spotřeby a množství paliva. Pokud tedy bude reálná průměrná spotřeba blízká celkové průměrné spotřebě, bude možné ujet vypočítané množství kilometrů. Poslední údaj je jednoduchý součet posledního stavu tachometru a počtu kilometrů, které je možné ujet na natankované množství. Tyto údaje jsou například vhodné pro plánování trasy, uživatel má alespoň nějakou představu, kolik kilometrů je s vozidlem schopen ujet.

Závěr

Bakalářská práce splnila všechny body zadání. V prvním bodě byl teoreticky rozebrán OS Android a vývojový balíček pro vývoj aplikací pro tento OS. Práce se zabývala použitím tohoto OS na reálných zařízeních, jimiž jsou převážně mobilní telefony, z toho vyplývá také popisovaná architektura OS. Android je uživatelsky velice přívětivý a přizpůsobitelný každému uživateli podle jeho potřeb.

Vývoj aplikací pro OS Android je do značné míry usnadněn dostupností vývojových nástrojů a pro jeho začátek je potřeba pouze Android SDK a vývojové prostředí Eclipse. V případě použití těchto nástrojů a vytvoření nového projektu vzniká ihned nová aplikace, kterou je po přeložení aplikace *Hello World.* Jedinou překážkou může být struktura a životní cyklus aplikace, ta je však v této práci také rozebrána. Aplikace je také možné ihned spustit a testovat ve virtuálním zařízení, které emuluje mobilní zařízení s OS Android ve zvolené verzi. Uživatel tedy nepotřebuje reálné zařízení s tímto systémem.

Poslední dva body zadání byly také splněny. Návrh aplikace vycházel z Bakalářského projektu, kde byla provedena rešerše již existujících programů s touto tématikou. Podařilo se tedy navrhnout a implementovat zamýšlené vlastnosti a vyvarovat se nedostatkům testovaných aplikací. Aplikace má spoustu užitečných funkcí, především obnovu a zálohu uživatelských dat a správu výdajů více vozidel, což v rešerši zmíněné programy ve větší míře nepodporovaly. Dále aplikace uchovává výdaje v sekcích, které evokují druh výdaje. Aplikace byla rozšířena o výpočty statistických údajů a o možnost přidávat záznamy s cenou v eurech, která je následně podle uživatelsky nastavitelného kurzu přepočítána na koruny.

Aplikaci lze do budoucna rozšířit o statistiky, které by byly zpracovány graficky, pro lepší představu. Dále je možno rozšířit aplikaci o uživatelské prostředí do více jazyků, a s tím přidání funkce na změnu hlavní měny aplikace. Určitým přínosem by také byla možnost exportu záznamů do Excelu, kde by mohl uživatel prohlížet záznamy i na počítači, případně navrhnout a implementovat desktopovou aplikaci, která by umožnila synchronizaci se stávající mobilní aplikací.

Seznam použité literatury

[1] DOSTÁL, Leoš. *Sledování nákladů automobilu*. Liberec, 2011. Bakalářský projekt. Technická univerzita v Liberci. Vedoucí práce Ing. Přemysl Svoboda.

[2] Android (operační systém). In: *Wikipedia*: *the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 2001- [cit. 2012-05-08]. Dostupné z: <http://cs.wikipedia.org/wiki/Android_(operační_systém)>

[3] What is Android?. *Android Developers* [online]. 2012 [cit. 2012-05-08]. Dostupné z: <http://developer.android.com/guide/basics/what-is-android.html>

[4] MURPHY, Mark. *Android 2*: *Průvodce programováním mobilních aplikací*. Brno: Computer Press, a.s., 2011. ISBN 978-80-251-3194-7.

[5] Activities. *Android Developers* [online]. 2012 [cit. 2012-05-08]. Dostupné z: <http://developer.android.com/guide/topics/fundamentals/activities.html>

[6] Installing the SDK. *Android Developers* [online]. 2012 [cit. 2012-05-08]. Dostupné z: <http://developer.android.com/sdk/installing.html>

[7] Building and Running. *Android Developers* [online]. 2012 [cit. 2012-05-08]. Dostupné z: <http://developer.android.com/guide/developing/building/index.html>

[8] Menus. *Android Developers* [online]. 2012 [cit. 2012-05-08]. Dostupné z: <http://developer.android.com/guide/topics/ui/menus.html>

[9] Notifications. *Android Developers* [online]. 2012 [cit. 2012-05-08]. Dostupné z: <http://developer.android.com/guide/topics/ui/notifiers/index.html>

[10] Data Storage. *Android Developers* [online]. 2012 [cit. 2012-05-08]. Dostupné z: <http://developer.android.com/guide/topics/data/data-storage.html>

[11] Cursor. *Android Developers* [online]. 2012 [cit. 2012-05-08]. Dostupné z: <http://developer.android.com/reference/android/database/Cursor.html>

Seznam příloh

Příloha A – Návod k obsluze

Příloha B – CD-ROM

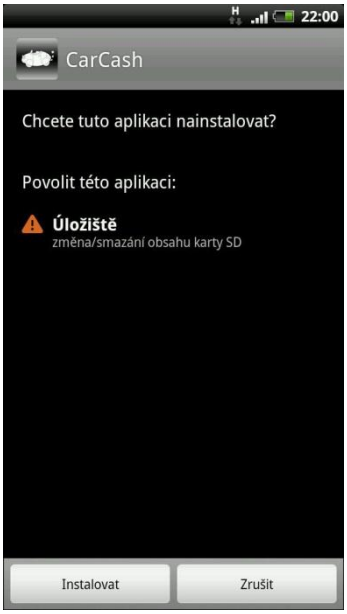
Příloha A

Návod k obsluze

Návod k obsluze

1. Instalace

Aplikaci je nutné nahrát na SD kartu zařízení, například pomocí USB kabelu. Následně je nutné mít nainstalovaného nějakého správce souborů, aby bylo možné aplikaci na SD kartě najít a spustit tak její instalaci. Po spuštění instalace budete vyzváni k potvrzení oprávnění, které aplikace požaduje, v tomto případě pouze změna/mazání obsahu karty SD a to z důvodu zálohy a obnovy uživatelských dat. Po potvrzení je aplikace nainstalována a připravena k použití.

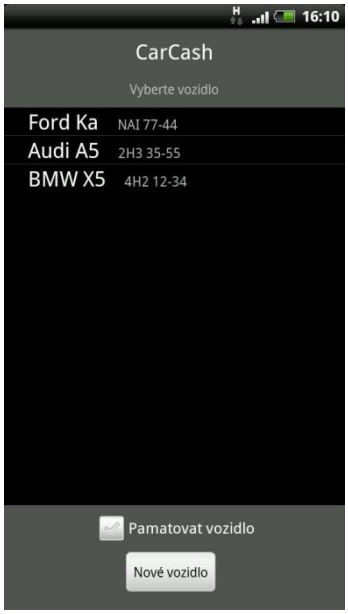


Instalace programu

2. Výběr a vytvoření vozidla

Při prvním spuštění aplikace je nutné vytvořit profil vozidla, pro které chce uživatel uchovávat výdaje. Pro vytvoření nového vozidla zvolte toto tlačítko ve spodní části obrazovky. Nyní vyplňte všechny údaje o vozidle (všechny údaje kromě poznámky jsou povinné a je s nimi pracováno dále při zadávání výdajů, vyplňte je tedy podle skutečnosti). Vytvoření profilu vozidla potvrdíte tlačítkem Uložit. Po uložení budete přesměrováni na výpis profilů, které jste vytvořili. Vytvořené profily vozidel je možné mazat a editovat. Toto je možné zvolením a podržením vybraného profilu. Zvolením jednoho profilu budete přesměrováni na výdaje tohoto vozidla. Před zvolením profilu je také možné zaškrtnout volbu zapamatování profilu, která znamená, že při

příštím spuštění aplikace budete rovnou přesměrování na výpis výdajů zvoleného profilu.



Výpis vozidel

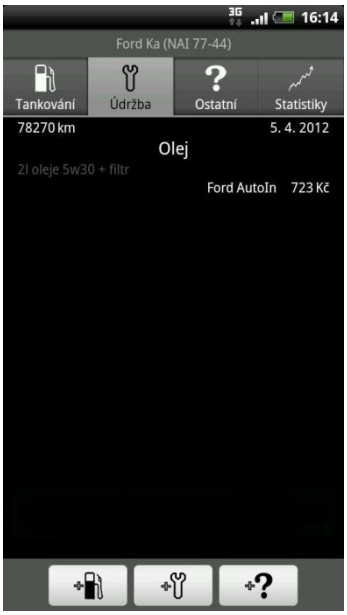
Vytvoření nového vozidla

3. Práce s výdaji

Obrazovka s výpisy výdajů je rozdělena na kategorie podle druhu výdaje a těmi jsou: Tankování, Údržba a Ostatní, poslední záložkou jsou Statistiky, které jsou také rozčleněny do záložek podle kategorií.

Ve všech kategoriích s výpisem výdajů jsou v dolní části obrazovky přístupná tři tlačítka pro přidávání výdajů podle kategorií.

Výpis záznamů tankování



Výpis záznamů údržby

Výpis statistik

4. Nový výdaj

Podle kategorie výdaje zvolte tlačítko pro jeho přidání, je jedno na které záložce se nacházíte, výdaj bude správně zařazen podle stisknutého tlačítka. Vždy budete přesměrováni na příslušnou obrazovku pro zadání atributů, které se liší podle druhu výdaje, povinné atributy jsou označeny hvězdičkou, uložení záznamu potvrdíte tlačítkem **Uložit**, naopak ukončení provedete tlačítkem **Zrušit**.

Přidání záznamu o tankování

Přidání záznamu o údržbě

Přidání záznamu do sekce ostatní

5. Úprava a mazání záznamů

Při zvolení vybraného záznamu je možné tento záznam smazat nebo editovat. Editace probíhá stejně jako zadání nového výdaje, pouze jsou zde předem vyplněny původní hodnoty a je možné je změnit.

Kontextové menu

6. Řazení záznamů

Toto je možné změnit opět zvolením některého záznamu a v menu zvolit **Řadit dle**, kde je možné následně vybrat, dle čeho mají být záznamy řazeny. Řazení je možné zvolit pro každou kategorii zvlášť.

7. Menu aplikace

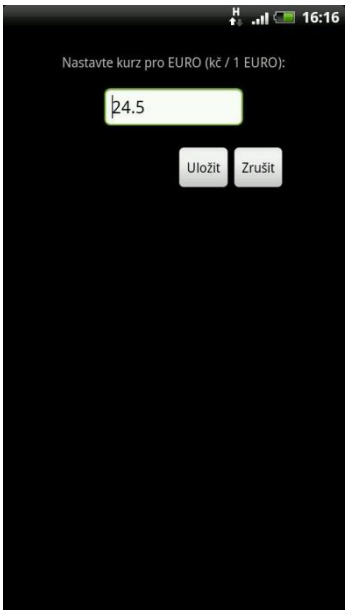
Menu je možné vyvolat HW klávesou Menu na přístroji. Menu je dostupné všude v aplikaci, kromě aktivit pro přidání nového záznamu. V menu lze nastavit kurz pro přepočet z eur na koruny. Toto je vhodné, když je přidáván záznam, který byl placen v eurech. Při přidání je zadána cena v eurech a je automaticky podle nastaveného kurzu přepočítána na koruny, kurz je však nejprve nutné nastavit. Dále je v menu možnost změnit profil, budete tedy přesměrováni na úvodní obrazovku s výběrem vozidel. V případě, že bylo zaškrtnuto pamatování profilu, je toto také zrušeno.

Dále jsou v menu možnosti zálohy a obnovy uživatelské databáze. Záloha je provedena na SD kartu do souboru CarCashDb. Obnova je prováděna ze souboru

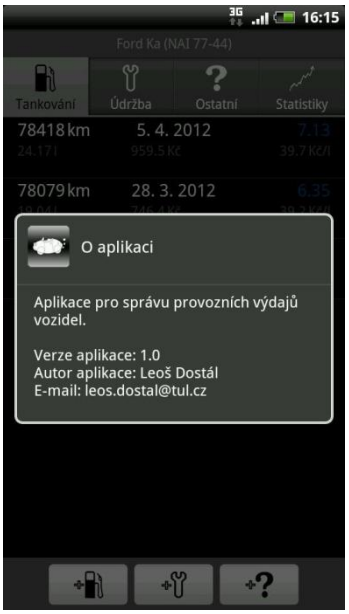
CarCashDb, který je nutné nejprve na SD kartu nakopírovat. Po obnově je aplikace automaticky vypnutá a je nutné ji znovu manuálně spustit.

Poslední možností v menu je **O aplikaci**, kde naleznete informace o verzi a autorovi aplikace.

Options menu aplikace



Nastavení kurzu



O aplikaci

Příloha B

CD-ROM

Obsah CD-ROM

Na přiloženém disku je kompletní projekt obsahující zdrojové kódy a instalační soubor CarCash.apk. Tímto souborem je možné aplikaci nainstalovat do mobilního telefonu s OS Android. Další součástí je bakalářská práce v elektronické podobě.