Střední průmyslová škola na Proseku
190 00 Praha 9, Novoborská 2

OBOR

18-20-M/01 INFORMAČNÍ TECHNOLOGIE

MATURITNÍ PROJEKT

TÉMA PRÁCE

Mobilní aplikace pro Android - Záruční listy

2015 Autor Malý David

Zadání praktické maturitní práce

PROHLÁŠENÍ

Prohlašuji, že jsem svou maturitní práci „Mobilní aplikace pro Android - Záruční listy“ vypracoval samostatně a použil jsem pouze podklady (literaturu, projekty, SW, atd.) uvedené v seznamu.

Nemám závažný důvod proti užití tohoto školního díla ve smyslu § 60 zákona č. 121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon).

V Praze dne 23.3.2015 ……

podpis

PODĚKOVÁNÍ

Tímto bych rád poděkoval všem, kteří mi pomohli s touto prací, a to díky praktickým radám, nebo připomínkám, které mi pomohly můj úkol udělat ještě lépe, než bych kdy zvládl sám.

Zejména chci poděkovat vedoucímu mé práce, panu Ing. Přemyslu Vaculíkovi za poskytnutí podpory a poskytnutí nového úhlu pohledu, který mi mnohokrát otevřel oči.

Panu Ing. Jiřímu Šilhánovi za mnoho cenných rad.

A také mé rodině za pomoc a trpělivost.

ANOTACE

Jméno autora	David Malý
Název maturitní práce	Mobilní aplikace pro Android - Záruční listy
Rozsah práce	90 stran
Školní rok vyhotovení	2015
Škola	Střední průmyslová škola na Proseku
Vedoucí práce	Ing. Přemysl Vaculík
Využití práce	Výukové účely v oboru Informační technologie v předmětu Programování.
Klíčová slova	Praktická maturitní dlouhodobá práce, Android, Záruční listy
Anotace	V textu je popsáno zpracování praktické dlouhodobé maturitní práce. Na téma vývoj aplikace sloužící pro evidenci záručních listů s operačním systémem Android.

ANOTATION

Autor	David Malý
Title of graduation work	Mobilní aplikace pro Android - Záruční listy
Extend	90 pages
Academic year	2015
School	Střední průmyslová škola na Proseku
Supervisor	Ing. Přemysl Vaculík
Application	Education purposes in the field of Information Technology in subject Programming.
Key words	Practise long-therm maturita exam, Android, Warranty lists
Annotation	Description of practise long-therm maturita exam. Topic of this document is development Android application for registration warranty lists on Android operation system.

Obsah

1. Cíl práce

Cílem práce je naprogramovat aplikaci, která bude pracovat s evidencí dat, konkrétně záručních listů a fotografií. Tato evidence má za úkol plnit dokumentační charakter a hlídat termín zákonem daného nároku na reklamaci zakoupeného zboží.

Součástí řešení pro uplatnění těchto nároků je fotodokumentace požadovaných dokumentů. Tyto dokumenty aplikace společně s dodatečnými informacemi umožňuje ukládat v mobilním zařízení k tomu určeném, na daném zařízení lze provádět zálohy a poskytuje i umožnění propojení záloh pomocí služby Google Drive.

Pro tento cíl je důležité zajistit funkčnost na většině zařízení s operačním systémem Android v předem určené verzi.

Dalším cílem bylo naučit se co nejvíce z vývoje pro mobilní operační systém Android a získat tak co nejvíce zkušeností z této oblasti. Hlavním záměrem tohoto cíle bylo získat znalosti hlavně z oblastí vývoje, které se týkají tvorby grafického rozhraní, nebo porozumění základnímu životnímu cyklu aplikace. Dokázat implementovat nativní součásti operačního systému Android jako NotificationManager – pro práci s notifikacemi, AlarmManager – pro probouzení zařízení v jasně daný čas anebo implementace GoogleDrive Api, které má za úkol implementovat nahrávání záloh na Internetové úložiště společnosti Google - GoogleDrive.

2. Úvod

Téma dlouhodobé maturitní práce se zabývá tvorbou mobilní aplikace pro operační systém Android – Záruční listy. Zajišťuje evidenci záručních listů koupeného zboží pro uplatnění zákonných nároků na reklamaci pro stát Česká republika.

Záměrem bylo vytvořit funkční prostředek, který ulehčí správu kopií záručních listů, nebo podobných dokumentů. Uživatel se tak nemusí starat o to, že by se mu mohly po delší době tyto dokumenty ztratit, nebo poničit. Nejlepší příklad pro znehodnocení záručních dokumentů jsou běžné účtenky, je možné je použít jako tento dokument, ale po delší době se stanou nepoužitelnými. Není možné použít je jako dokument prokazující datum koupě a identitu zboží.

Rozsah evidence jako je tomu v případě této aplikace, by bylo možné nasimulovat pomocí kombinace například kapesního fotoaparátu a dvou poznámkových bloků. Nicméně tento způsob by neumožňoval export do jiných zařízení, anebo uložení na vzdálené uložiště, bez vynaložení nutné dávky energie a dalších zařízení. Z tohoto důvodu byla zvolena „chytrá“ mobilní zařízení, která jsou v dnešní době nedílnou součástí běžného života občanů České republiky.

Volba mobilního zařízení byla pro účel této práce nejlepší možností. Nicméně taková zařízení se dělí, co se týče použitelnosti na několik typů. Toto dělení je stěžejní hlavně při výběru mobilního operačního systému, proto je jim věnována část tohoto dokumentu. Nejmasivnější zastoupení na českém trhu má operační systém Android, vlastněný společností Google. Jeho výběr byl tedy nejlepší možností. Tomuto operačnímu systému je věnována část dokumentu.

Mobilní aplikace pro Android - Záruční listy, jak již bylo zmíněno výše v této kapitole, eviduje kopie dokumentů, sloužící pro jednoznačnou identifikaci zakoupeného zboží a datumu koupě. Nabízí tedy silný nástroj při uplatňování záručního nároku, který byl umožněn prodejcem daného zboží.

Účel je splněn díky multifunkčnosti dnešních mobilních zařízení. Disponují velkým seznamem doplňků, které je možné dle libosti propojovat. Zařízení s mobilním fotoaparátem mohou sloužit pro archivaci fotozáznamů. Téměř všechna zařízení mají k dispozici vnitřní zapisovatelnou paměť a většina z nich umožňuje přístup k síti Internet. Ve spojení s operačním systémem Android se tak stávají ideálním kandidátem pro splnění cíle této práce.

Společnost Google, vlastnící operační systém Android umožnila vývojářům přístup k jejímu systému. Proto je možné aplikaci vyvíjet v domácích podmínkách, zdarma, s širokou podporou velké základny dalších vývojářů, a vše v jejím prostředí. Z hlediska nativní podpory se tedy nejedná o žádnou nadstavbu k existujícímu systému, aplikace přímo koexistuje se systémem a společně si umožňují využívání svých funkcí.

Disk Google (Google Drive) společnost Google upravila i pro používání ve spojení s vývojáři. Díky této službě a účet Google, který je vyžadován na extrémní většině mobilních zařízení s mobilním operačním systémem Android, je umožněn přístup k této službě. Zpřístupňuje tak funkci zálohování na vzdálený server i vývojářům, kteří nemají vlastní server.

Aplikace implementuje práci s fotoaparátem, vnitřním uložištěm, práci s připojením na síť Internet skrze standardní mobilní data a WiFi, práci s kompresí dat, notifikační služby, probouzení zařízení z režimu spánku v předem daný čas a nativní práci s popup (vyskakovacími) okny, animacemi.

Zahrnuje do své funkcionality i propojení s nativními grafickými funkcemi systému Android, jako jsou: používání standardního značkovacího jazyka XML pro grafické vykreslování, využívání existujícího kódu pro práci s rolovacími seznamy a meny, vytváření vlastních seznamů a men.

Pro komunikaci s uživatelem je použit systém získávání textu z přednastavených složek, který umožnil překlad celé aplikace do češtiny a angličtiny. Jako důkaz nativního zasazení do operačního systému Android je zde například přepínání mezi překlady, které se mění podle aktuálního jazyka celého systému, bez zásahu uživatele a jakékoliv práce programátora.

Umožňuje díky výše zmíněným funkcím evidenci dokumentů, nazvaných „záruční listy“. Tato evidence obsahuje informace o datu koupě, záruční době (v měsících), době na vrácení zboží bez udání důvodu, ceně zboží, výrobním čísle modelu a také poznámky ke každému záručnímu listu.

Fotografie jsou pořizovány přímo z aplikace pomocí nativní aplikace pro fotoaparát, nebo jakékoli jiné, která je propojená s operačním systémem. Jsou spojovány vždy s jedním „záručním listem“ a je možné jich k jednomu záručnímu listu pořídit i více. Ukládány jsou v plném rozlišení, ale jejich zobrazení je možné ve dvou variantách. První varianta se týká zobrazování miniatur, které se vytvářejí podle aktuální velikosti displeje zařízení a jsou proto na každém zařízení přiměřeně velká. Druhá varianta je jejich zobrazovaní v nativní aplikaci

pro procházení obsahu fotografií (ve většině případů aplikace zvaná Galerie), nebo pomocí aplikace k tomu určené a propojené s operačním systémem. Jsou ukládány do složky, kterou spravuje sama aplikace.

Informace o prodejci a pobočce, kde bylo zboží zakoupeno, jsou povinnou součástí každého dokumentu, který eviduje koupi. Není ale nutné každý obchod evidovat pokaždé znovu s každým „záručním listem“. Pro tyto účely je zde vytvořena sekce „Obchody“, kde je možné nalézt veškeré již evidované pobočky, které jsou spojeny se „záručními listy“. Eviduje se zejména: Název obchodu, město, poštovní číslo a webová adresa, tedy informace potřebné k bezpečnému nalezení požadovaného obchodu. U každého „záručního listu“ je možnost výběru z nabídky evidovaných obchodů, kde bylo zboží zakoupeno. Stejná funkcionalita je zaručena i co se týče práce s kategoriemi, do kterých je možné “záruční listy“ zařadit.

„Záruční listy“ jsou přehledně zobrazovány v nativně vytvořeném seznamu, kde se zobrazují i dny, zbývající do nejbližšího vypršení záruky/lhůty na vrácení zboží bez udání důvodu. Tento počet dnů je odlišen barevně podle naléhavosti: červená - 7, oranžová – 14, zelená - 21. Jednotlivé položky seznamu jsou řazeny nejdříve podle lhůty na vrácení zboží bez udání důvodu a poté podle doby, zbývající do vypršení záruční lhůty, v poslední řadě jsou již expirované „záruční listy“. Seznam je možno řadit podle jména, nebo ve výše popsaném řazení a to jak vzestupně, tak i sestupně.

Upozornění jsou nativně zakomponovaná do operačního systému Android, a proto bylo možné je bez větších potíží implementovat. Periody pro zasílání jsou: 1, 3, 7, 31, 122, 183 a 365 dnů před vypršením nejbližší lhůty vztahující se ke každému „záručnímu listu“. Kontroly, zda se neblíží jedno z těchto dat, jsou prováděny každý den ve 12:00 am., není nutná žádná akce uživatele, dokonce není potřeba, aby byl telefon v danou dobu používán, stačí, aby byl zapnutý, a operačním systém sám vyhodnotí nejlepší chvíli pro probuzení a vykonání kontroly. Aplikace tak zbytečně nezabírá žádné systémové zdroje a šetří tak baterii.

Aplikace rozšiřuje oproti zadání několik důležitých funkcí, které ji činí ještě sofistikovanějším nástrojem.

Systém záloh je implementován ve dvou formách, které jsou spolu vzájemně propojeny.

První z forem nabízí možnost uložení všech fotografií a celé databáze do jednoho komprimovaného archivu (.rar), s kterým může poté uživatel nakládat dle svého uvážení. Součástí této formy je i obnovení těchto souborů v případě smazání dat z aplikace. Tato

funkcionalita umožňuje uživateli všechna data, která uložil mít pod svou kontrolou, také zajišťuje přenositelnost těchto dat mezi různými zařízeními, které mají nainstalovanou tuto aplikaci.

Vytvořenou zálohu stačí umístit do umístění, které aplikace každému uživateli nabízí podle jeho nastavení operačního systému a na možnostech zařízení. Většinou se uložení liší podle toho, zda zařízení disponuje externím uložištěm (paměťová karta), nebo jestli nabízí pouze interní paměť.

Druhá forma zajišťuje komunikaci s účtem Google, díky kterému je možné přistoupit k uživatelskému disku Google, který mají všichni standardní uživatelé k dispozici. Je tak učiněno, aby bylo možné díky propojení s 1.formou, vytvořenou zálohu na tento disk uložit, zvolit si umístění zálohy na disku Google.

Další jazyk aplikace, kterým může komunikovat s uživatelem, je rozšíření oproti zadání. Protože lze předpokládat, že v České republice žijí lidé, kterým je jiný jazyk srozumitelnější nežli jazyk český, nachází se zde tedy plně funkční překlad do jazyka anglického.

Rozšíření zahrnující kategorie a seznamy obchodů, přispívají celkové přehlednosti a celkově zrychlují orientaci a běžnou práci s aplikací.

Barevná rozlišení expiračních lhůt, která jsou zobrazována hned v první obrazovce, kterou uživatel vidí po spuštění aplikace, pomáhají najít co nejrychleji vše, co uživatel hledal. S tím je spojen i grafický design aplikace, který navazuje na grafický model Holo Light. Na tento model zobrazování je zvyklá většina uživatelů operačního systému Android. Aplikace tento model rozšířila o několik barev, které usnadňují orientaci a jen vylepšují běžné uživatelské prostředí.

Tento dokument také obsahuje odbornou stať, ve které jsou zpracována témata z okolí mobilního operačního systému Android, jedná se zejména o témata: Mobilní operační systémy, Android, Vývoj pro Android, Design Aplikací pro Android.

- Rešerše na téma Mobilní operační systémy má za úkol seznámit se světem mobilních operačních systémů, jejich běžným výskytem při denním používání. Také se zmiňuje o historii těchto operačních systémů a ukazuje jejich vývoj až do dnešní doby, zabývá se především historií operačního systému Android.
- Rešerše na téma Android rozebírá téma mobilního systému Android hlavně v souvislosti se společností Google, která převzala hlavní korporátní část vývoje tohoto systému. V této rešerši jsou uvedeny základní principy fungování OS Android, Android Environmentu, základní dělení aplikací, životní cykly, obecné rozvrstvení aplikačního modelu a jeho obecný popis. V této rešerši je i povrchově popsán vývoj společnosti Google. Je zde uveden i aktuální graf rozdělení používání verzí OS Android a také jejich hlavní rozdíly.
- Rešerše Vývoj pro Android dále rozšiřuje předchozí rešerši o detailnějších popis Android Environmentu, životních cyklů a popisu modelu. Hlavním záměrem je popsat jednoduchost, se kterou může začínající vývojář proniknout do většiny funkcí, aniž by se zdržoval jejich vlastním programováním nebo úpravami, zabývá se tedy implementací existujících funkcí a jejich přizpůsobením ke konkrétním účelům vývojáře. K tomuto účelu je nutné věnovat kapitolu i oprávněním, které aplikace získávají při jejich instalaci.
- Na téma Design aplikací pro Android je napsána rešerše hlavně ve smyslu seznámení se základním smýšlením společnosti Google o grafickém designu, který je charakteristický svými pravidly. Zabývá se zkrácenou historií designu od prvních veřejných verzí OS Android, jejími hlavními tématy jsou ovšem design modely Holo a nejnovější Material.

Odborná stať seznamuje uživatele se základním používáním aplikace tak, aby se dosáhlo jejího používání v plném rozsahu a bez chyb. Pro tento účel je zařazena do dokumentu uživatelská dokumentace. Je prezentována formou série obrázků s doplňujícím textem tak, aby bylo naprosto jasné, jak aplikaci používat. Pro zvládnutí používání aplikace jsou vyžadovány pouze základní schopnosti používání OS Android a minimální znalost běžné ikonografie.

Odborná stať obsahuje i popis technického provedení aplikace formou technické dokumentace. Tato dokumentace má za úkol vysvětlit a přednést nejdůležitější funkce aplikace tak, aby byly snadno pochopitelné a bylo je možné po jejich porozumění bez větších problémů opravovat. Pro tento účel jsou v dokumentaci obsaženy výtažky důležitého kódu pro názornou ukázku a vysvětlení užitých postupů při tvorbě aplikace. Dané výtažky kódu jsou důležité i proto, že ukazují implementaci nativních součástí OS Android do aplikace.

Pro technickou dokumentaci je také důležitý popis minimálních požadavků mobilních zařízení, které jsou zde také rozepsány.

„I moudrý se stane hloupým, když na sobě přestane pracovat.“

- Orison Swett Marden

3. Stať

Odborná stať se zabývá odborným popisem provedení dlouhodobé maturitní práce a je určena pro začínajícího vývojáře programujícího aplikace na operační systém Android. Programovací jazyk JAVA je popsán v rámci odlišností od servletového provedení .

Základem pro pochopení OS Android a vývoje na tento operační systém jsou rešerše na témata:

- Mobilní operační systémy
- Android
- Vývoj pro Android
- Design aplikací pro Android

a technická a uživatelská dokumentace.

Rešerše jsou učeny pro pochopení a uvedení do správného kontextu pro vývojáře, kteří OS Android znají pouze díky běžným uživatelským zkušenostem při denním používání tohoto operačního systému.

Uživatelská dokumentace je určena pro seznámení s grafickým rozhraním aplikace a seznámení s jejím používáním s co nejvyšší efektivitou.

Technická dokumentace slouží především pro ukázku použitých postupů a komplexnosti aplikace, pro tento účel obsahuje UML diagram/y a příklady, které pomáhají pochopit základní funkce, jejich vzájemné provázání mezi sebou i na úrovni přímé komunikace s nativními funkcemi OS Android. Tato dokumentace obsahuje i popis minimálních požadavků na použité zařízení a minimální verzi OS Android.

3.1 Mobilní operační systémy

V době kdy počítače začínaly díky pokročilejším být menší a menší vznikla myšlenka, že by bylo praktické brát si svůj počítač s sebou a používat ho kdekoliv. Tento směr vedl ke vzniku prvních notebooků a malých pracovních stanic. Nicméně uživatelé nechtěli mít oddělené zařízení sloužící k práci od telekomunikačního zařízení. Chtěli vše v jednom, a protože v této době byly počítače stále velmi robustní a uživatel ani nepotřeboval všechny jejich funkce, vydala se tato myšlenka vstříc mobilním telefonům.

Myšlenka potřebovala ke své realizaci hlavně požadavky uživatelů, kterými se staly: výdrž baterie, volání, psaní SMS zpráv, připojení na mobilní sítě, paměť pro telefonní čísla a další. Na základě těchto požadavků se začaly vyvíjet první mobilní operační systémy.

Psal se rok 1997 a společnost Ericsson vydala svůj první telefon GS88 Penelope a poprvé v historii jej označila za smartphone.

V roce 2000 se stává mobilní operační systém Symbian nejpoužívanějším mobilním operačním systémem, bohužel se ale vyvíjel jen pomalu a tak zaostal.

2002

- společnost Blackberry vydává svůj první operační systém
- Microsoft vydal svůj Pocket Pc

2007

- Apple uvádí na trh svůj smartphone iPhone
- Vznik Open Handset Alliance
- Oznámení Androidu

Na trhu se v tuto chvíli objevila velká spousta různých operačních systémů. Některé byly pro cílové zákazníky mnohem použitelnější než jiné např. první iPhone byl doslova revolucí. Společnost Apple se pokusila přiblížit k běžným uživatelům, necílila jako Blackberry nebo OHA na PDA a povedlo se ji to.

Směr vývoje mobilních operačních systémů se změnil a začal klást větší důraz na koncové zákazníky. Vyvinula se nutnost zpracovávat co nejpříjemnější grafická rozhraní a zpřístupnění co nejvíce funkcí uživateli.

Mobilní operační systémy spolu začaly soupeřit stejně jako jejich předchůdci na domácích počítačích. Snažily se zaujmout hlavně množstvím aplikací, grafickým zpracováním a v dnešní době i silnými marketingovými kampaněmi, soudními spory a vtipy na konkurenci. Nejdůležitějšími operačními systémy se z hlediska vývoje staly systémy od společností Blackberry, Apple, OHA.

Blackberry si vysloužila post nejbezpečnějšího operačního systému. Ten získala například ve své době neviděným šifrováním veškeré komunikace zařízení i jeho obsahu, neustálou synchronizací s vlastními servery, které měly za úkol nejen aktualizovat, ale i autentizovat uživatele na jednotlivých zařízeních. Vydala se ale cestou PDA a tak konkurence, která zvolila jiný směr, získala velký náskok. Blackberry OS zastaral a snaží se najít opět způsob, jak se vrátit mezi hlavní operační systémy.

Apple vydal svůj smartphone iPhone spolu s operačním systémem iOS. Vyvolal u cílových zákazníků všeobecné nadšení a nabídl možnosti dosud neviděné. Oproti komplikovaným PDA vsadil na jednoduchost a co nejpříjemnější komunikaci s uživatelem, stal se tak přijatelný pro velkou skupinu potencionálních uživatelů. Byl také první, kdo přemýšlel o marketingu spojeným s aplikacemi a spustil službu Appstore. Společnost Apple udala směr, kterým se vyvíjejí mobilní operační systémy dnes.

Open Handset Alliance je konsorcium společností zabývajících se trhem mobilních technologií. Jednou z mnoha činností je tvorba standardů pro mobilní technologie, kterými se poté výrobci mohou řídit. Další činností je i záštita vývoje mobilního operačního systému Android. Tento systém byl v počátcích velmi podobný Blackberry OS a za to byl často kritizován. V době, kdy společnost Apple vydala iPhone, Android nebyl skoro použitelný a i když ještě nebyl uveden do prodeje, už byl zastaralý. OHA nicméně ve vývoji pokračovala a mobilní operační systém Android se stal nejpoužívanějším.

Dnes jsou mobilní operační systémy v naprosté většině nedílnou součástí každého prodaného telefonu a jejich potenciál se neustále zvyšuje. Během několika let se použití těchto systémů, zejména Android, rozšířilo i na poli domácích spotřebičů, inteligentních domácností nebo vozidel. Způsob, kterým tyto systémy zaplavily trh, byl velmi rychlý, a těžko lze odhadovat, jak bude vývoj pokračovat dál. Nicméně mobilní operační systémy začínají nahrazovat i operační systémy na domácích počítačích. Tak by se dalo očekávat vytvoření jednotného unifikovaného prostředí, které smaže rozdíly mezi mobilními a desktopovými operačními systémy.

3.2 Android

Linux, jeden z nejstarších a dodnes používaných operačních systémů je vzdálený příbuzný Androidu. Operační systém Android je založený na stejném jádru, které se upravilo pro použití v mobilních technologiích. Zůstala stejná struktura, ale změn bylo tolik, že v dnešní době si tyto systémy jsou podobné jen velmi málo.

Požadavky, které tyto změny zapříčinily, byly například: nutnost snížení nároků na energetické zdroje mobilních zařízení, nižší nároky na hardware, mnohem menší kapacity pamětí oproti počítačům a mnohé další. Je také nutné brát v potaz, že doba, ve které tyto požadavky vznikly, není až tak dávno minulá.

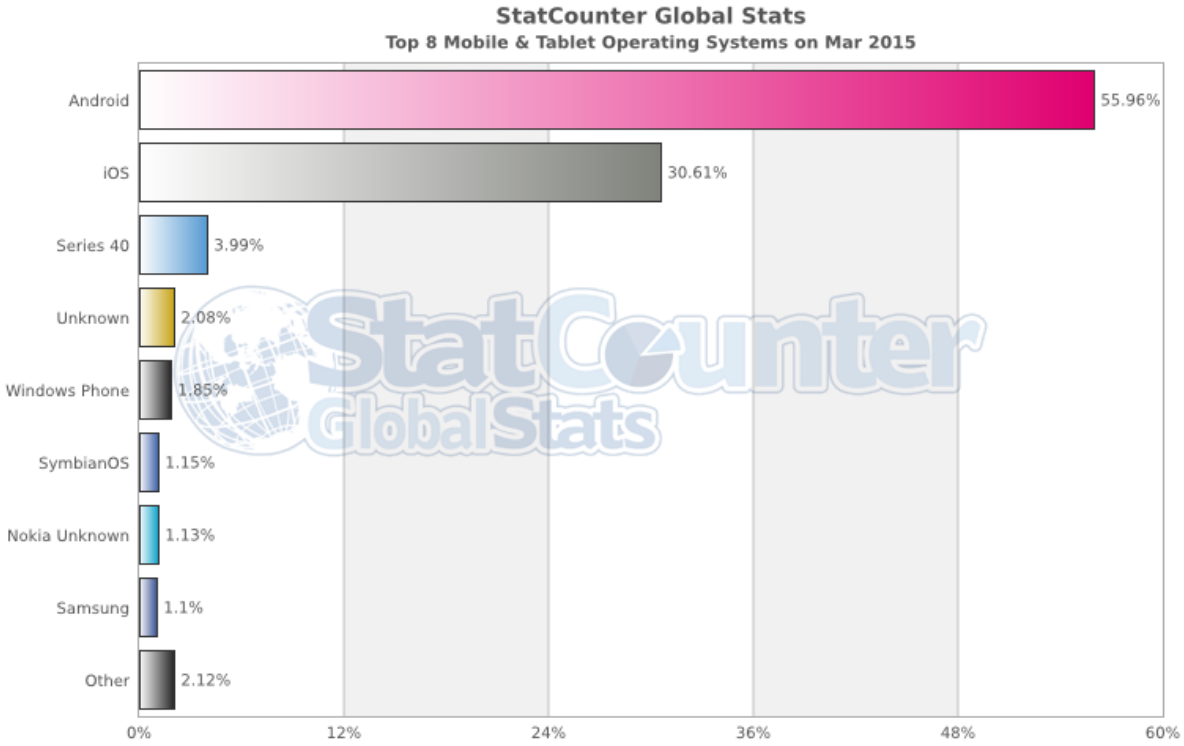
Roku 2003 vznikla společnost Android Inc., která byla založena v Kalifornii. Tuto společnost založili: Andy Rubinson, Riche Miner, Nick Sears a Chris White. Jejich operační systém byl původně vyvíjen pro potřeby tehdejších kamerových zařízení. Nabízel graficky přívětivé uživatelské rozhraní, podporu více architektur procesorů a disponoval základními aplikacemi pro záznam videa, zvuku, prohlížeč souborů apod.

Tuto společnost odkoupila roku 2005 korporace Google Inc. a vyvinula tak ucelenou platformu pro další vývoj, a tato společnost, tak získala mnoho patentů v oblasti mobilních technologií, čímž Google Inc. vstoupila na trh s mobilními technologiemi, v té době ovládnutý Blackerry, iOS a Windows Phone.

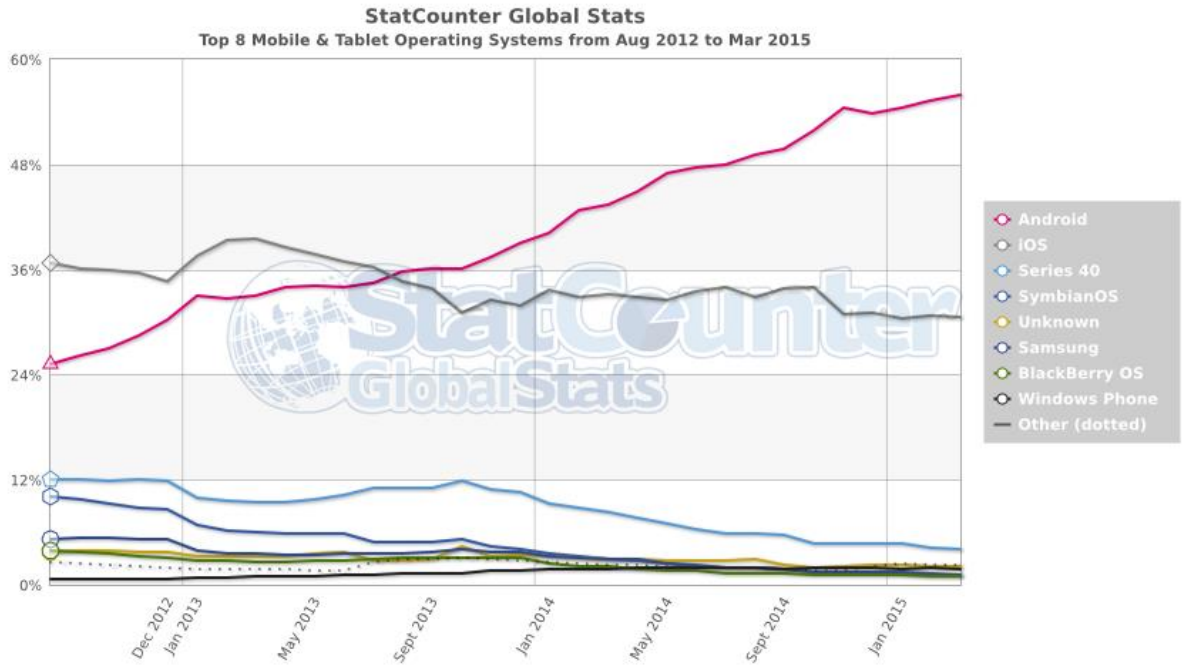
Pátého listopadu 2007 bylo vytvořeno konsorcium Open Handset Alliance společnostmi zejména: LG, HTC, Intel, NVIDIA, Qualcomm, Samsung a také Google Inc. Toto konsorcium si dalo za cíl vytvořit mobilní platformu, která má platit jako jednotný standard. Jeden z dalších cílů je tuto platformu progresivně vyvíjet jednotně s účelem snížení nákladů a jednodušší distribuce.

Ve stejný den Open Hadnset Alliance ohlásil svůj první produkt, Android, otevřenou mobilní platformu, kterou mohl vyvíjet kdokoliv a zdarma pod licencí Apache 2.0 (1) a GPLv2. Společnost Google také dala najevo, že má s touto platformou velké plány.

Dnes je Android zastoupen více než 55% oproti ostatním mobilním operačním systémům, což je vidět na následujících grafech (Obrázek 1 a Obrázek 2) a stal se tak nejvýznamnějším mobilním operačním systém dneška.



Obrázek 1 - graf využití mobilních operačních systémů - Březen 2015
Zdroj: (11)



Obrázek 2 - graf využití mobilních operačních systémů – Srpen 2012 - Březen 2015
Zdroj: (10)

3.2.1 Historie důležitých Android verzí

Sladkosti, koláče, koblihy jsou nedílnou součástí propagace Android. Společnost Google Inc. se rozhodla zajistit propagaci hlavně tím, že zajistí uživateli příjemný pocit po zaslechnutí jednoho z jejich produktů. Google+ - ten co nabízí webové služby Google a něco navíc (sociální síť), Picasa – uživatel si připadá jako malíř při práci s jeho obrázky, Google Play – nakupování je hra/zábava, KitKat – sušenka, Lollipop – lízátko, Gingerbread – perníček apod.

Tímto směrem se vydal i při zavádění jejich nového systému Android a kolem roku 2007 bylo vydáno několik beta verzí (Milestone, Beta). (2)

3.2.1.1 0.5, Milestone

První veřejnou verzí byla Milestone (mezník) 3, Android 0.5. Tato verze fungovala pouze na emulátorech a byla přijata jako „pouhý klon Blackberry“. Toto tvrzení bylo uvozeno i z faktu, že první telefon (který nebyl uvolněn na veřejný trh) velmi připomínal tehdejší směr telefonů Blackberry. Toto zařízení mělo jméno Sooner (příliš brzy) a vytvořila ho společnost HTC.

Roku 2007 byl nový operační systém Android považován za „zastaralý“, jelikož o několik měsíců po jeho představení společnost Apple ukázala svůj nový IPhone. Tak musela společnost Google s jejich nehotovým a neatraktivním počinem začít opět od začátku.

V této verzi nebylo možné nastavovat hlavní plochu ani widgety. Bylo zde pouze několik ikon ve spodní části, které mohly rotovat nebo se ukázat v jednoduchém menu. Byla zde ale dodnes používaná tři tlačítka (zpět, domů a menu). A zařízení bylo možné ovládat pomocí pěticestného d-padu nebo dotykově. Důležité je, že na chodu systému se přímo podílely aplikace. Aplikace samy pracovaly s notifikacemi, dnes tuto činnost obstarává systém.

Nicméně oproti iOS vynikal multitaskingem, sdílením zdrojů mezi různými aplikacemi a později vlastním přepínačem aplikací, který obstarával například chvílí, kdy uživatel pracoval s aplikací a někdo zavolal, systém sám zobrazil popup okno a umožnil komunikaci, aniž by musel vypnout předchozí aplikaci. Díky popup oknům, která mohla obsahovat i více informací jako adresář kontaktů, položil tento systém základ pro moderní XWindow UI, které nebylo to té doby na žádném mobilním operačním systému. Oproti konkurenci také obsahovala webový prohlížeč s aktuálním jádrem Webkit, který držel krok s počítačovou platformou MacOS X s prohlížečem Safari 2.

Synchronizace stejně jako u Blackberry byla umožněna díky vzdáleným serverům. Zajišťovala tak aktualizace operačního systému, cloud-to-device messaging.

3.2.1.2 0.9, Beta

Rok 2008 se stal důležitým datem pro Android, protože s jejich novou verzí Beta začínal připomínat dnešní verze operačního systému. Umožňoval práci s aplikacemi na hlavních obrazovkách, podporoval jich i více, další (pouze first-party aplikace). Přidal podporu pro „vytahování“ notifikačního panelu a připravil se pro používání i ve chvíli, kdy bude v telefonu nainstalovaných i více aplikací. Pro tuto fázi bylo možné přidávat na domácí obrazovky i složky a do nich ukládat aplikace.

Widgety byly omezeny hlavně tím, že byly pouze 3 – obrázek, hodiny a vyhledávání. Zajímavostí je, že do seznamu obrázků společnost Google integrovala i kategorii „koupené obrázky“ chtěla tak do budoucna prodávat obrázky, nikdy tento plán nezrealizovala.

Přinesla i obrovské množství nastavení jako zvukové profily, anebo možnost kopírovat a vkládat text pomocí dlouhého podržení nad ním . Stala se tak přízviskem „customize everything“(nastavit cokoliv). Také to byla první verze, která umožnila horizontální přetočení obrazovky. Tuto možnost využívaly i všechny aplikace a na rozdíl od dnešní verze opravdu šlo horizontálně přetočit cokoliv.

V době kdy Android byl připravován na více aplikací, 2.generace Iphone přinesla vlastní distribuci pomocí obchodu s aplikacemi pomocí Appstore.

3.2.1.3 1.0

V Říjnu 2008 byl představen Android ve verzi 1.0. Ukázán byl na prvním telefonu, který byl volně ke koupi na T-mobile G1 viz. Obrázek 3.

Obrázek 3 – T-mobile G1
Zdroj: (3)

Bylo to poprvé, co byl Android viděn na opravdovém hardware a nejednalo se pouze o emulaci.

Nebyl o mnoho rozdílnější oproti 0.9, ale jeho hlavním přínosem bylo integrování Google Apps, které obsahovaly základní aplikace jako kalendář, emailový klient, Gmail, aplikaci pro nastavení, Youtube, a hlavně Market – obchod s aplikacemi. Tím se otevřel novým možnostem a dohnal konkurenci. Díky Open Handset Alliance mohli vývojáři začít zdarma vyvíjet aplikace a později je i zveřejňovat pomocí distribuce Market. Myšlenka Marketu byla v tom, že oproti Appstore chtěl Google prodávat nejen aplikace, ale vše ostatní okolo Android (tuto myšlenku se mu podařilo naplnit hlavně během posledních dvou let v České republice).

Google také nelimitoval vývojáře, aby se drželi jeho grafického rozhraní jako to dělá Apple a možná i z tohoto důvodu a také díky možnostem zdarma začali vývojáři pracovat i na aplikacích pro Android. Velkým náskokem před konkurencí bylo zavedení oprávnění, které mohly aplikace vyžadovat (to aplikoval Apple až někdy kolem roku 2012).

Problém s uživatelským rozhraním byl hlavně v tom, že programátoři se Android nedohodli, jestli použijí tmavé barvy se světlým písmem, anebo světlé s černým (tuto možnost zrušila až verze Kitkat, rozhodlo se pro světlé pozadí s černým textem).

3.2.1.4 2.0, Éclair

Říjen 2009 - výrobce Motorola přinesl 2. generaci operačního systému Android. Ten byl představen na telefonu Motorola Droid a umožnil tak mnohem větší výkon, displej i RAM.

Kromě samotné nové verze je důležité zmínit, že byla uskutečněna masivní marketingová kampaň. Ta prezentovala Android jako mnohem lepší alternativu k iOS, která ho měla naprosto rozdrtit. Pomohlo tomu i grafické zpracování celého telefonu. I samotný systém byl zpracován ostře a vyzývavě.

Nexus One byl velký milník pro společnost Google, protože tímto krokem se chtěla společnost uvést na trh s mobilními telefony a prodávat je přímo zákazníkům. Tento telefon navrhla a vyráběla společnost HTC. Mělo se jednat o experimentální telefon, který není vázaný na místního poskytovatele sítě.

Měl myšlenku, že změní běžnou politiku, která panovala ve Spojených státech Amerických. Ta určovala, že si zákazník nejdříve vybral poskytovatele a poté telefon, chtěl to naopak. Bohužel Nexus One se moc neprodával, byl pro některé zákazníky příliš drahý (asi 529 dolarů, běžně cena byla asi 200 dolarů za smartphone).

3.2.1.5 2.1

Hlavním přínosem této verze byly animace, které dohnaly konkurenci. Pokračující marketingová kampaň sklízela úspěch u zákazníků a tak se Android stával stále populárnějším.

Další novinkou byla živá pozadí, celkové upravení vzhledu celého operačního systému.

S touto verzí také vstoupila společnost Google do boje se společností Apple, který skrze narážky a vtipy neskončil dodnes.

3.2.1.6 2.2, Froyo

Hlavní přednost tohoto updatu bylo zpřístupnění Flash pro silnější telefony a upravení základních aplikací jako například fotoaparát, nebo aplikace Google Googgles, která dokázala rozpoznávat obličeje na fotografiích.

Umožnil také přesouvat aplikace na SD kartu, což opět předstihlo konkurenci.

3.2.1.7 4.0 Ice Cream Sandwich

Říjen 2011 byl představen Android 4.0, který stavěl na experimentálním Androidu 3.0 Honey Cup. Přinesl naprosto nové grafické rozhraní postavené tak, aby vyhovovalo i větším displejům, zejména pak tabletům, ale i smartphonům.

Samsung Galaxy Nexus, byl první telefon s Androidem, který měl 720p rozlišení a LTE modem. Měl 4,65 palce velký displej, který byl podle mnoha uživatelů kritizován jako příliš velký (dnes se tak malý telefon shání pouze v nejnižší cenové kategorii). Byl také první telefon, který přinesl dvoujádrový procesor.

Jedna z novinek byla i absence kapacitních nebo jiných hardwarových tlačítek (vyjma tlačítek pro hlasitost). To umožnilo neplýtvat místem na telefonu a vytvářet tak mnohem tenčí rámečky okolo displeje. Také by se mohl telefon otáčet do všech směrů a tlačítka zobrazovaná na displeji by se vždy přizpůsobila. Tuto možnost, ale většinou umožňují pouze tablety. Novinky byly například zahrnuty v monitorování přenosu dat tak, aby uživatel měl přehled, kolik dat se přeneslo za určité období.

Výrazným posunem v tvorbě grafického rozhraní bylo uveřejnění oficiální Google design návodů na to, jak by vývojář měl pracovat s grafikou tak, aby byla co nejpodobnější vizi od Google. Uveřejnil, tak ukázku designu Holo, který mohl kdokoliv implementovat a získat tak stejná grafická rozhraní jako používaly Google služby.

Apple používal jinou strategii, neuveřejňoval návody a neukazoval vývojářům jak pracovat s designem. Jednoduše aplikace, které nevyhovovaly, neuveřejnil na Appstore. A tak mají vývojáři pro Android nejen volnou ruku při designovaní jejich aplikací, ale mohou také, pokud nechtějí vytvářet vlastní design sáhnout po návodu od společnosti Google.

Tímto návodem zároveň Android splnil všechny potřeby pro návrh uživatelsky přívětivého prostředí a jednalo se o plně funkční mobilní operační systém, který mohl být považován za opravdu konkurenceschopného. Říjen 2012 byl ve znaku spuštění Google play services, který definitivně rozdělil Google apps na několik balíků. Google Play Services – aplikace komunikující se servery Google (Sign-in, Remote Wipe, CTDM, Account setup), Play store – aplikace sloužící pro vyhledávání a práci se soubory (Drive, Play apps, Gmail, Maps, Chrome) a Android apps – běžné aplikace nutné pro status „telefon“ (nastavení, vytáčení hovorů, lock screen a základní ovladače a jádro).

3.2.1.8 4.4, Kitkat

Říjen 31 2013 byl na den Halloweenu vypuštěn Android Nestlé, který byl předzvěstí Kitkatu. Ten byl pomocí update nasazen během několika dalších dnů.

Hlavním změnou v novém Androidu bylo mnohem menší vytížení paměti. Obecně se jednalo o velké optimalizování celého operačního systému. S tím bylo spojeno i to, že se na některých zařízeních vše chovalo mnohem hůře a tak to i zůstane, pokud taková zařízení nedostanou další update.

Optimalizace se týkaly omezení zbytečných animací podle hardwaru zařízení, přidání režimu spotřeby apod.

Tímto updatem se společnost Google snažila zbavit zbylých Gingerbread zařízení, která představovala stále velký podíl na trhu. Společnost chtěla, aby se tato starší zařízení opět pomocí update dostala do povědomí a snížila se tak celková roztříštěnost operačního systému. Dalším krokem bylo ukončení podpory vývoje pro Gingerbread verzi.

3.2.2 5.0 Lollipop

25.6.2014 byl představen, ale až 12.11.2015 byla spuštěna aktualizace pro vybraná zařízení.

Velkou změnou prošel celý design aplikace, který je dnes nazývaná Material Design, který značně zjednodušil veškeré uživatelské rozraní.

Slibuje mnohem lepší práci s baterií, už proto, že nastupuje po Androidu 4.4.4W na zařízení jako jsou chytré hodiny nebo náramky.

Kromě toho definitivně přepnul z běhového prostředí Dalvik (měly ho všechny předchozí verze, jen v Kitkatu bylo možné přepínat) na prostředí ART, které by mělo zvýšit celkový výkon systému i aplikací a jejich odezvu. Společnost Google mluví až o čtyřnásobném zlepšení. Tato verze podporuje i 64bitové procesory a aplikace.

Nová zařízení budou také dodávána s automaticky zapnutým šifrování obsahu telefonu, což povede ke zvýšení bezpečnosti při odcizení za cenu lehkého snížení výkonu.

Nově se snaží získat post multimediálního zařízení. Podporuje až 8 zvukových kanálů včetně 5.1 a 7.1, podporuje USB Audio, podpora RAW formátu při pořizování fotografií.

Umožňuje nově i sdílené zařízení, pokud si uživatel zapomene telefon, jednoduše si půjčí jiný s Androidem ve verzi Lollipop a může po autentizaci přistupovat k veškerému obsahu v jeho zařízení.

3.3 Vývoj pro Android

Společnost Google je jedním z hlavních zastánců komunitního vývoje. Vede rozsáhlou webovou prezentaci, vydává manuály a aplikační prostředí, pořádá i konference a co je hlavní, zavázala se k dlouhodobému vývoji systému Android.

Díky všem těmto službám se komunita vývojářů velmi rozrůstá. Hlavní devízou operačního systému Android je fakt, že vývoj je zcela zdarma. Jediné, co může vývojář zaplatit, je uvedení jeho aplikace na Google Play (obchod nejen s aplikacemi pro Android) a prémiové funkce společnosti Google.

Není tedy žádná překážka, aby mohl kdokoli vyvíjet aplikace i student střední školy. Pro úspěšný start vývoje je důležité seznámit se se základní teorií, která bude z části popsaná v této práci.

Důležité je znát základy programovacího jazyka Java a objektového programování. Za předpokladu, že programátor má tyto znalosti, je nutné vysvětlit základní rozdíly mezi běžným Java API a Android API.

3.3.1 JAVA a VM

Operační systém Android je napsán v programovacím jazyku, který je velmi podobný Javě, ale má tolik rozdílů, že nepodléhá patentům ani licencím společnosti Oracle (vlastník a vývojář Java). Vedlo se na toto téma mnoho soudních sporů. Z tohoto důvodu Android API nepoužívá ani JVM (Java Virtual Machine) pro zpracování kódu. Místo toho používá vlastní virtualizované stroje (VM) jako Davlik, nebo ART. Nicméně pro kompilaci je vyžadován Java SDK.

Jeden z rozdílů je například v kompilaci, kde u běžných aplikací napsaných v programovacím jazyku Java je kód kompilovaný do bytového kódu a spuštěn. U Dalviku je kód zkompilovaný do proprietárního (uzavřeného) bytového kódu a spuštěn v přesně navrženém VM. Ten je navržený tak, aby používal co nejméně systémových zdrojů. Používá jednu 16 bitovou sadu instrukcí, místo 8 bitových + operandy (instrukce pro každou činnost s proměnnou).

Další velký rozdíl je, že Dalvik neumí linkovat knihovny, takže musí být vždy zkopírovány do interního úložiště aplikace a to zvětšuje jeho objem.

Rozdíl mezi Dalvik a ART VM je hlavně v času kompilace. Dalvik používá Just-In-Time (JIT) kompilaci a ART Ahead-of-Time (AOT). Podstata je v tom, že JIT kompilace spouští instrukce aplikace v době, kdy jsou přímo vyžadovány a AOT kompilace je spouští předem. Místo čekání na to, až se kód zpracuje, jsou již k dispozici uložené výsledky. Tyto výsledky jsou ukládány při instalaci aplikace, kdy se zpracovává většina (někdy i celý) kód. Je možné tedy očekávat úsporu baterie a zvýšení výkonu za cenu delší instalace.

Důležité je i přinesení nového správce paměti Rosalloc do ART, u Dalviku se jedná o velmi rozšířený Garbage collector. Princip Garbage collectoru je obecně řečeno takový, že spravuje velký blok paměti, který rozděluje podle potřeby. Rosalloc vytváří jeden paměťový blok pro každé vlákno aplikace a ten dále dělí. Větší bloky jsou určeny pro nejvyšší nároky aplikace a menší shlukují méně paměťově náročné potřeby aplikace.

Nutné je ale zmínit, že ART bylo v beta verzi na Androidu ve verzi Kitkat a je tedy stále velmi nové. Zlepšení oproti Dalviku je měrné, ale opravdový pozorovatelný výsledek bude k vidění až ve chvíli, kdy se objeví aplikace optimalizované i pro ART.

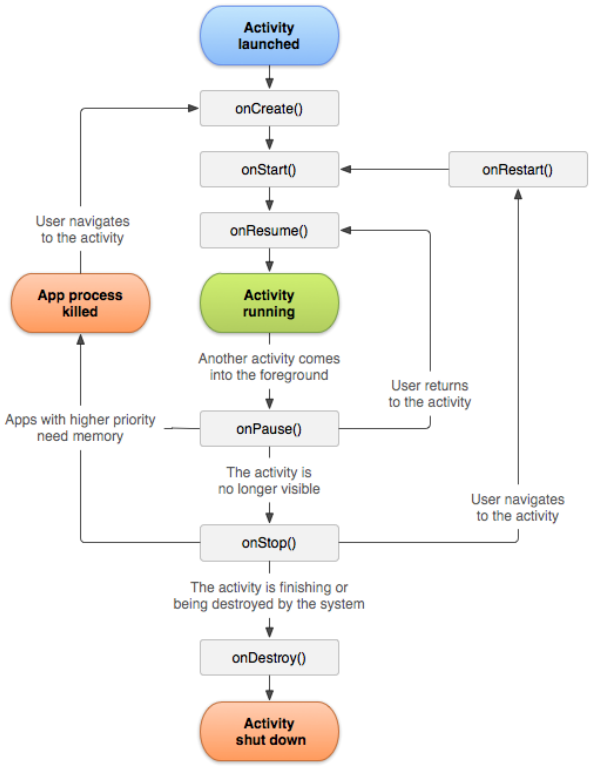
3.3.2 Programování pro Android

Pro začátek je naprostou nutností získat balíček pro vývoj aplikace (SDK), ten je na webových stránkách určených pro vývojáře. Tento produkt je zdarma a nevyžaduje při jeho používání žádnou autorizaci ani registraci. Dalším krokem je získat správný program pro tvorbu aplikací pro operační systém Android. Společnost Google doporučuje použít jejich nový produkt „Android Studio“, ale je možné i použít velmi rozšířený „Eclipse“ s nástroji pro vývoj na Android (ADT). Poté stačí spustit vybraný produkt pro programování (Eclipse/Android Studio) a začít pracovat. Není nutné stahovat další balíčky sloužící pro vývoj, protože poslední stáhnutá verze balíčku pro vývoj vždy obsahuje aktuální nástroje. Jako poslední krok je instalace balíčku Java SDK, také zdarma.

3.3.2.1 Activity

Základní teorie pro tvorbu grafického rozhraní se zabývá vším okolo tzv. Activity. Volně vysvětleno se jedná o jednu celou obrazovku (bez lišty upozornění a navigačních tlačítek), kterou vyplňuje aplikace. Stejně tak na Activity je cílené i programování celé funkčnosti.

Pro popis životního cyklu jedné Activity je zde názorný Obrázek 4



Obrázek 4 – Životní cyklus aplikace
Zdroj: (4)

Každá Activita je spouštěna hlavní metodou nazvanou *OnCreate()*, kde se deklarují všechny proměnné a základní struktura Activity. Pro vývojáře je to jedna z nejdůležitějších metod, protože skrze ní projdou všechny Activity na svém startu.

Metody jako *OnStart(), OnResume(), OnPause(), OnStop(), onDestroy(),* nebo *OnRestart()* jsou volány samotným systémem. Většinou se jedná o akce, které vynutí Activitě změnit svůj aktuální režim. Například ve chvíli, kdy Activita není viditelná je ve stavu Stop. Když systém potřebuje paměť Activitu vypne, ale předtím dá programátorovi možnost uložit poslední stav a to tím, že zavolá metodu *OnDestroy()*.

Programátor také musí vědět, že Activita se vytváří vždy ve chvíli, kdy je zobrazována na popředí. To se děje i ve chvíli, kdy aplikace změní svojí orientaci (pohled) z vertikální na horizontální a opačně. Pokud na to vývojář zapomene, může se stát, že ztratí všechna data, která nebyla uložena.

3.3.2.2 Procesy

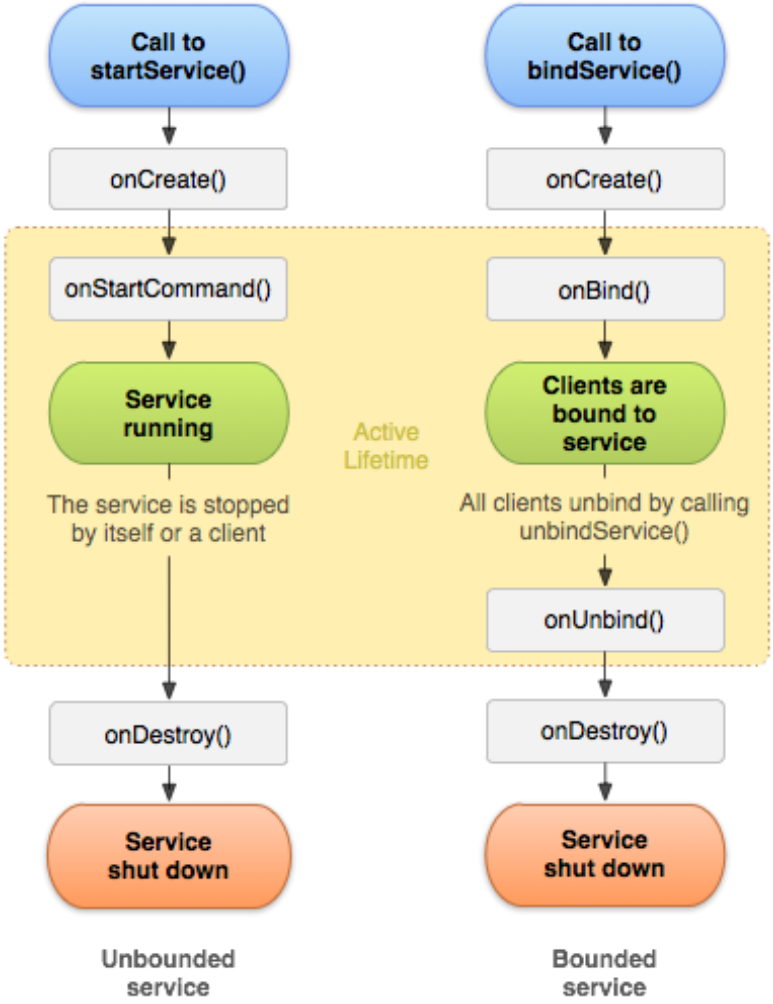
Aplikace na platformě Android, stejně jako ve všech moderních operačních systémech běží ve svém vlastním procesu. Je možné, ale vytvářet i sub-procesy. Jedná se o kód, který se nemusí zpracovávat ve stejné řadě jako hlavní činnost aplikace.

Nejjednodušší příklad je: stahování souborů z Internetu. Jedna část kódu zajišťuje samotné stahování, druhá kontroluje stav dat a třetí aktualizuje ukazatel stavu probíhajícího stahování.

V operačním systému Android si lze představit procesy jako Activity a sub-procesy jako Services. Pracují stejně jako ve většině Linuxových operačních systémech. Nedokáží přistupovat ani spravovat uživatelské rozhraní přímo. Jsou určeny pro práci na pozadí, což se hodí zejména ve chvíli, kdy je vyžadován chod aplikace, i když není na popředí.

Services se dělí na dva základní typy běžný Service a BinService. První typ běží tak dlouho, jak je vyžadováno. Druhý typ běží od chvíle, kdy je zapnut a vypne se ve chvíli, kdy přestane existovat zpětný odkaz na Activitu, které ho zavedla. Je tedy možné ho vypnout i ve chvíli, kdy nedopracoval. To zajišťuje mnohem větší bezpečnost pro stabilitu systému, protože Android API nemá nic jako globální prostor. Ve chvíli, kdy opustí aplikace svůj vymezený prostor, programátor, nemá kontrolu nad spuštěným procesem, pokud by ho nechtěl složitě hledat mezi ostatními.

Životní cyklus Services je podobný, ale je mnohem jednodušší než-li u Activit. Jedná se pouze o zavolání požadovaného typu, zavedení, zpracování, pokud je nutné tak odepnutí od Acitivy a zničení.



Obrázek 5 – Services životní cyklus
Zdroj: (5)

3.3.2.3 Oprávnění

Důležitou funkci, kterou přinesl Android oproti dřívějším verzím iOS, je zavedení oprávnění pro většinu komponentů, které smí aplikace používat. Jedná se důležité bezpečnostní stanovisko. Oprávnění se píší společně s hlavním popisem aplikace do speciálního manifest souboru. Na jeho základě se vypisuje například název aplikace a také se z něj čerpají data pro upozornění na potřebná oprávnění. Uživatel má tak možnost před každou instalací aplikace souhlasit nebo nesouhlasit s používáním těchto oprávnění.

Linuxu znalí programátoři jsou zvyklí, že oprávnění se dělí na čtení, zápis a spouštění. Atributy jsou pro ně například: skryté, jen sdílení, úplné řízení apod. Operační systém Android zavádí, pojem „oprávnění“ zcela jinak.

Pro tento systém je daný pojem spíše doklad o tom, jaké funkcionality a komponenty systému bude používat. Aplikace se musí například před instalací dotazovat, zda smí číst adresář kontaktů, jestli může vytáčet telefonní čísla, nebo posílat SMS.

Nebýt manifestu, který tyto informace shromažďuje, jediný způsob, jak zjistit, co aplikace dělá, by byla analýza kódu.

Bezpečností prvek pracuje i s možností, kdy aplikace například požaduje vytáčení telefonního čísla, ale o toto oprávnění nezažádala. Systém pracuje pokaždé na stejném principu, jakmile aplikace vykazuje činnost, která vyžaduje oprávnění (většinou se jedná o volání daných funkcí), tak systém zkontroluje na základě manifestu, jestli k tomu má oprávnění. Pokud ho nemá, aplikaci jednoduše nepovolí provést danou činnost. Na tuto možnost většina vývojářů nepomýšlí, a proto aplikace často skončí chybovou hláškou.

3.4 Design aplikací pro Android

Od verze HoneyCup se společnost Google rozhodla udávat systému Android a vývojářům jednotný styl pro jejich grafická zpracování. Dosáhla tak jednotného vzhledu, i když není povinností vývojáře, aby je využil.

Hlavním důvodem proč společnost Google podnikla tento krok, byla potřeba změnit celou vizáž operačního systému. Před Android verzí 3.0 nebylo UI (uživatelské rozhraní) považováno za hezké, ba spíše naopak. Díky nástupu různých návodů, jak tvořit společný styl aplikací s vizí společnosti, je možné vytvořit dobře vypadající aplikaci i bez nutnosti znalostí tvorby UI.

Implementace všech základních prvků designovaných prostředí do SDK (software developtment kit) otevřela vývojářům zcela nové možnosti práce. Od této chvíle mohou vývojáři naprosto volně využívat všech grafických možností, která nabízí operační systém Android.

Volnou ruku mají i ti, kteří chtějí experimentovat. Společnost Google nijak nelimituje možnosti a tak není žádný problém na základu existujícího návrhu vytvořit vlastní design. Není žádnou novinkou, že například firemní aplikace stojí na pevném podkladu dobře fungujícího UI, které obohatí o vlastní barvy a loga.

Grafický styl ovlivňuje vše, co uživatel vidí. Udává směr pro vizualizaci grafických prvků jako například: jednotlivá tlačítka, checkboxy, posuvníky, barevné rozložení celé obrazovky, překrývání, průhlednost apod. Důležité je ale i to, že grafický styl ovlivňuje i UI, které má za úkol předkládat uživateli vše ve srozumitelné formě. Jeho hlavním úkol je vnést do grafického zpracování řád a zamezit chaosu.

Nejjednodušší příklad je jednoduché vyskakovací okno, které po uživateli žádá odpověď „souhlasím/nesouhlasím“. Uživatel je zvyklý, že tlačítko „souhlasím“ je vždy na jedné straně a „nesouhlasím“ na druhé. Ve chvíli, kdy by se vývojář rozhodl je dát nad sebe nebo jen obrátil pozice těchto tlačítek, vnesl by tak chaos do celého systému. Hlavní záměr UI je tedy bezpochyby intuitivnost.

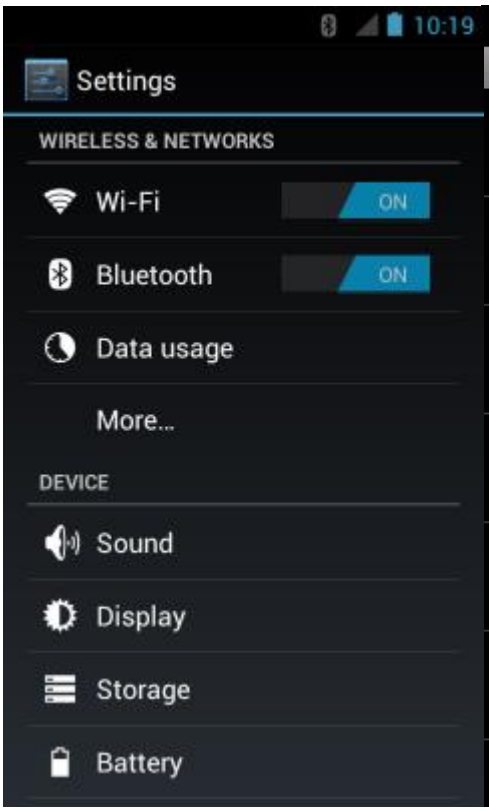
Možné je i jednotlivé styly kombinovat. V dnešní době jsou dostupné pouze dva styly a těmi jsou Holo(ICS) a Material Design.

3.4.1 Holo

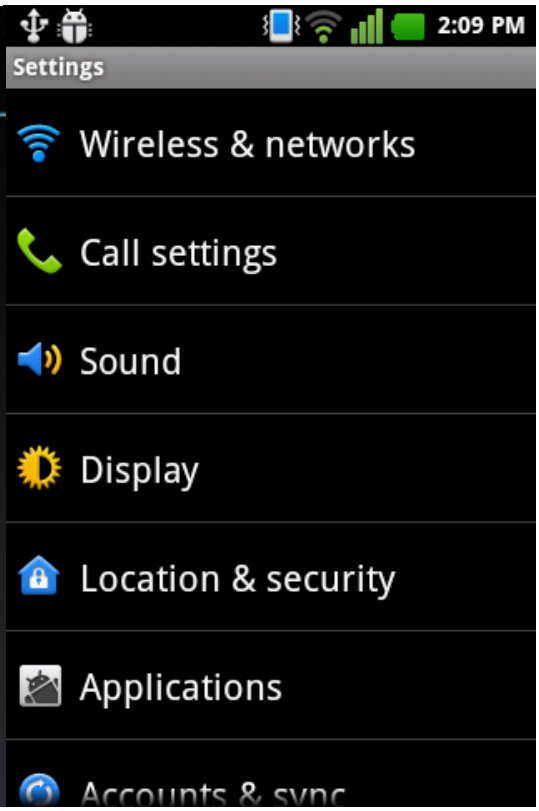
Oproti předchozímu grafickému stylu byl styl Holo revolucí v operačním systému Android. Celý systém se stal mnohem hezčí a přehlednější a nabízel možnosti, jako docílení požadovaného stavu na méně kroků.

Přinesl také výraznou změnu, která se týkala barevnosti celého operačního systému, kde kombinoval kontrast světlých a tmavých barev se světle modrou, která je spojovala.

Stejně jako v minulých verzích nebylo jasné, jestli zvolit tmavé nebo světlé pozadí a tak zde byla možnost obojího viz. Obrázek 6 a Obrázek 7.



Obrázek 6 – Android 4.1.2



Obrázek 7 – Android 2.3

Velkou změnou bylo i zavedení plochy, kde se nacházely nejdůležitější ovládací prvky pro pohyb a práci s aplikací. Zde měl uživatel na očích vše, co potřeboval pro práci.

Další novinkou byla přímá asociace obsahu s ovládacími prvky. Viděl-li uživatel obsah, který obsahoval seznam, bylo jasné, že by tento seznam mohl být rozbalen a dalo by se s ním pracovat. Stejně tak, ve chvíli kdy uživatel viděl jméno kontaktu, bylo jasné, že další interakce bude se seznamem kontaktů, pokud viděl telefonní číslo, šlo o adresář.

3.4.2 Material Design

Společnost Google specifikovala Material Design, jako nový jazyk, kterým je možné komunikovat s uživatelem. Tento design je přímo zakomponovaný do Android 5.0, ale je možné většinu jeho funkčnosti přenést i do starší verze.

Jeden ze záměrů a hlavní myšlenky Holo Designu bylo využití stejného stylu, jak pro aplikace běžící na telefonech, tak pro ty na tabletech. Ve výsledku aplikace vypadaly podobně, ale ne stejně. Material design tuto myšlenku posunul o krok dál.

Material Design vychází ze studií papíru a jeho skládání. Myšlenka tkví v tom, že uživateli je ukázán jakýsi stůl, na kterém leží několik papírů, a ty se překrývají. Nejvrchnější listina je uživateli nejblíže a proto se bude zabývat hlavně jí, ale může se podívat i na tu pod ní a libovolně je míchat jako karty, kde každá má svou výšku, šířku i délku a barvu, kterou pak připomínají nálepky nebo štítky.

Tento záměr měl přinést mnohem větší intuitivnost, jelikož uživatel nebyl nucen učit se s novým grafickým vzhledem, když už ho zná z běžného každodenního života.

Pracuje tedy se základními fyzikálními zákony o šíření světla a vzájemném ovlivňování materiálů. Není například možné, aby dva objekty zasahovaly do sebe. Ten svrchní vždy vrhá stín na objekt pod ním. Nicméně je zde i několik odchylek od praxe. Svrchní objekt, i když je mnohem větší než objekt pod ním, vrhá stín pouze na objekt pod ním, nikoliv na celý svůj prostor, jak by se dalo očekávat. Materiály také není možné skládat a překládat, je možné je pouze zvětšit, čí zmenšit.

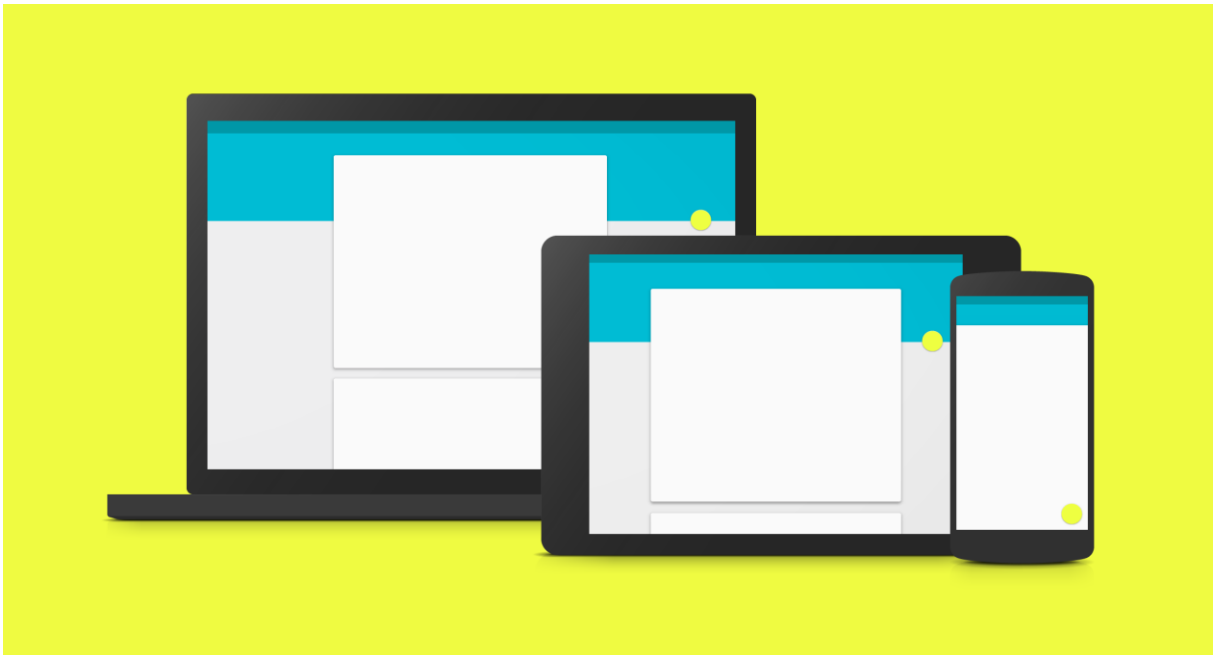
Jedná se tedy o skládání po vrstvách, kde každá ovlivňuje vrstvu nad sebou a pod sebou.

Práce s vrstvami a šířením světla dává tak pocit, že pracuje ve 3D prostředí, což ho opět přibližuje blíže k intuitivnímu ovládání.

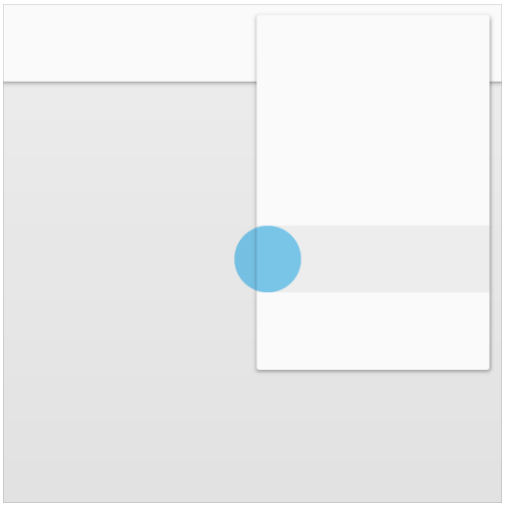
Podobnost vzhledu aplikací je dosažena jednoduchou logikou, kde místo aby se skládal celý styl jinak, je uživateli ukázán pouze menší nebo větší pracovní prostor. Pracuje se na principu přenášení věcí z menšího na větší pracovní prostor, či naopak. Pouze se udělají větší hromady nebo se vše rozprostře podle potřeby.

Animace jsou nedílnou součástí tohoto grafického stylu a opět navazují na celkovou logiku. Objekt, který je interaktován a vyvolává animaci, je vždy s danou animací přímo spojen. Není možné, aby se na jedné straně obrazovky stala interakce a na druhé se provedla animace. Animace musejí být také neměnné, nesmí se stát, že by se příchod animace provedl rychleji než odchod. To uživatele neleká, protože je zvyklý na stejnou rychlost změn a nemá pocit, že by mu něco uniklo.

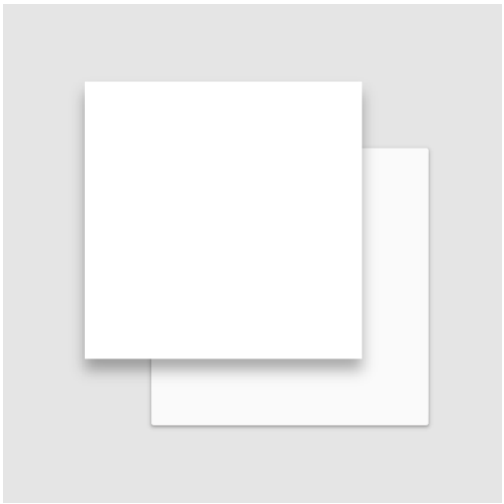
Celý Material Design je tedy cílen na maximální intuitivnost. Ukázka viz Obrázek 8, Obrázek 9 a Obrázek 10.



Obrázek 8 - Material Design – multiple screens
Zdroj: (6)



Obrázek 9 - Material Design – shadow
Zdroj: (16)



Obrázek 10 – Material Design – cards
Zdroj: (17)

4. Uživatelská dokumentace

Tato část dokumentu se zabývá uživatelským průvodcem aplikací, aby mohl kdokoli aplikaci používat tak, jak je koncipovaná.

Průvodce ukazuje instalaci, způsob chování aplikace při prvním spuštění a poté tvorbu a editaci záručního listu včetně pořízení fotografie.

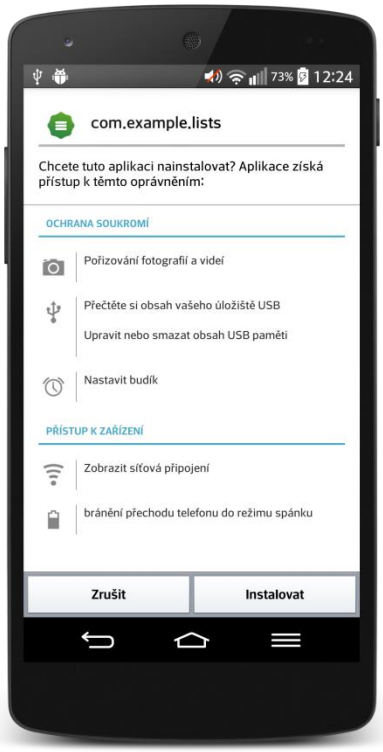
Ukazuje také tvorbu obchodů a popisuje postup exportování/importování dat a zálohování na uživatelský disk služby GoogleDrive.

4.1 Instalace aplikace

Pro úspěšné nainstalování aplikace je třeba splňovat minimální požadavky na zařízení. Je-li operační systém Android v nižší verzi než 4.4 (minimální povolená je 4.1.2), nebo jsou nainstalovány aplikace, kterou mohou ohrozit činnost této, je možné očekávat nepředpokládané chování.

Postup instalace:

1) Povolení instalace z neznámých zdrojů v nastavení operačního systému Android
2) Vložení souboru *lists.apk* do datového úložiště zařízení
3) Instalace aplikace pomocí nativního instalátoru aplikací
4) Spuštění aplikace



Obrázek 11 - Installation

4.2 První spuštění

Po prvním spuštění aplikace se uživateli ukáže obrazovka, viz Obrázek 12

Obrázek 12 – Empty lists

Jazyk textů, které jsou uživateli zobrazeny, se odvíjí od nastavení jazyka operačního systému Android. Pro obrázky je použit Český jazyk.

V pravém horním rohu se nacházejí základní navigační tlačítka pro tuto obrazovku:

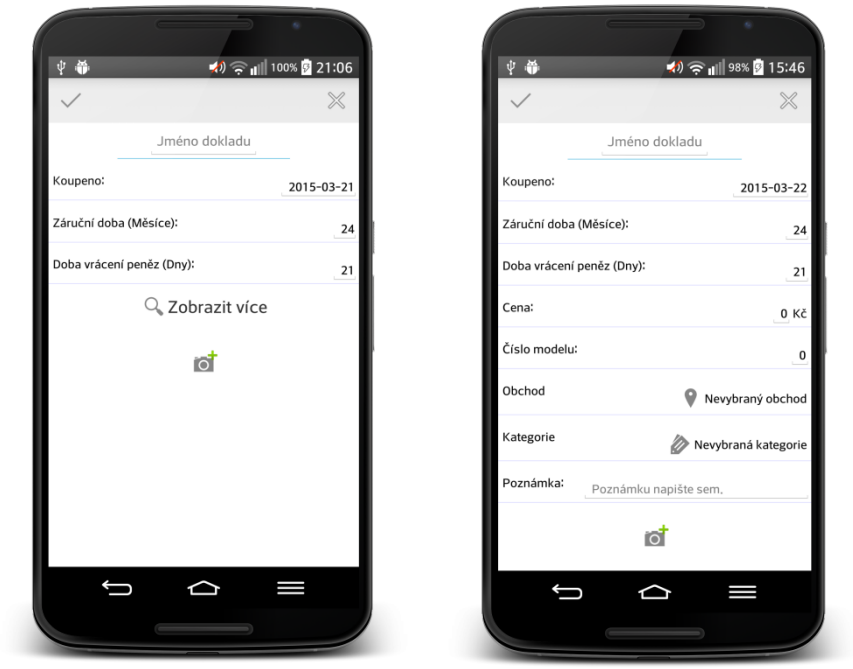
1) Založení nového záručního listu
2) Řazení seznamu záručních listů
3) Další možnosti aplikace

Více o navigačních prvcích hlavní obrazovky v kapitole Hlavní obrazovka se záručními listy.

Po stisknutí tlačítka „+“ se uživateli zobrazí další obrazovka.

4.3 Založení záručního listu

Druhá obrazovka ukazuje uživateli všechna pole, která může vyplnit. Po stisknutí ikony lupy, nebo nápisu zobrazit více se zobrazí další volitelné položky, jak je ukázáno na obrázku Obrázek 13.



Obrázek 13 – New list

Na obrazovce jsou v navigační liště (horní šedá zóna) dvě ikony:

- Levá je pro potvrzení založení záručního listu
- Pravá pro zrušení zakládání a návrat na první obrazovku

Obsahem této obrazovky jsou hlavně uživatelsky vyplnitelná pole jako například Jméno dokladu, počet měsíců záruční lhůty. Po zobrazení dalších polí zde jsou například pole cena, modelové číslo.

Za předpokladu, že jsou vytvořeny kategorie nebo obchody, se po kliknutí na položku pro obchod nebo kategorii zobrazí seznam obchodů/kategorií pro výběr.

Jediné pole, které je uživatel povinen vyplnit je název záručního listu.

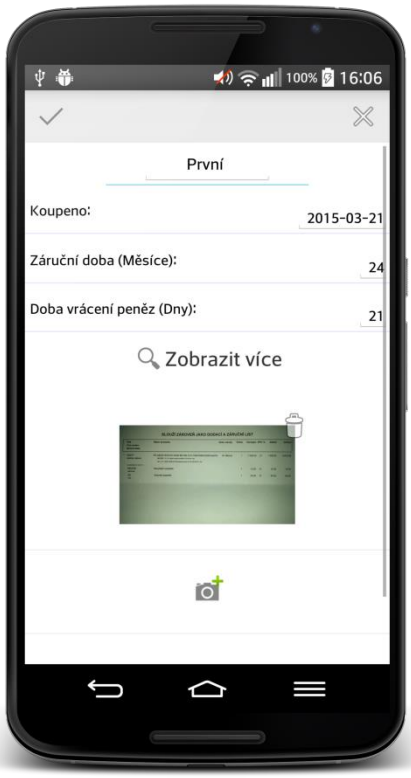
Po stisknutí tlačítka pro potvrzení se zobrazí obrazovka s uloženými informacemi o záručním listu.

Po stisknutí tlačítka fotoaparátu se zeleným plus se zobrazí obrazovka pro pořízení fotografie.

4.4 Editace záručního listu

Do obrazovky pro editace záručního listu je možné se dostat dvěma způsoby: Odchod je možný pomocí potvrzení nebo zamítnutí úprav.

1) Po vytvoření záručního listu a pořízení fotografie
2) Z obrazovky pro ukázání dat o záručním listu.



Obrázek 14 – Edit list

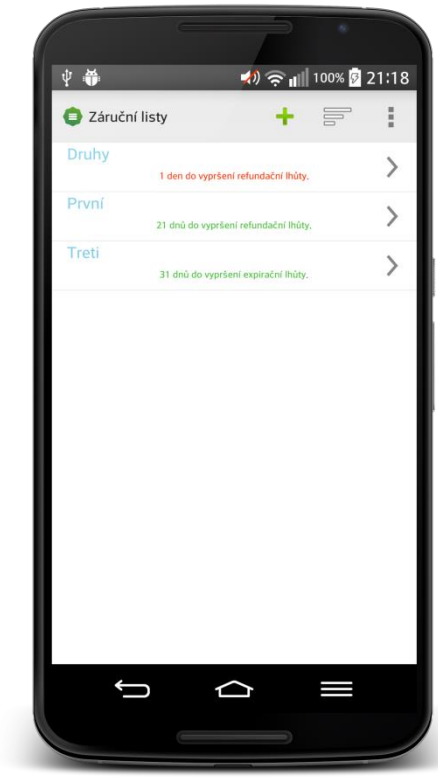
Tato obrazovka (Obrázek 14) vypadá a chová se podobně jako obrazovka pro vytvoření záručního listu. Jediný rozdíl je v uživatelsky editovatelném obsahu. Ten je zde zobrazen tak, jak byl zadán při poslední editaci/vytvoření daného záručního listu.

Na této obrazovce je také možné spravovat fotografie týkající se záručního listu.

- Tlačítko fotoaparátu s plus zajišťuje vyvolání aplikace pro pořízení fotografie, které se následovně přidá k záručnímu listu
- Tlačítko koš po stisknutí smaže fotografii, u které je zobrazeno (uživateli je zobrazeno potvrzovací okno akce)
- Po kliknutí na fotografii se spustí aplikace pro zobrazování obrázků (nativně Galerie)

4.5 Hlavní obrazovka se záručními listy

Po opětovném spuštění aplikace, nebo po vrácení se zpět na hlavní obrazovku, je uživateli zobrazen seznam vytvořených záručních listů.



Obrázek 15 - After

Na obrázku výše je jasně viditelné, že se barva textu mění podle doby, za kterou vyprší záruční lhůta nebo lhůta na vrácení peněz bez udání důvodu.

Po kliknutí na položku seznamu je uživateli zobrazena obrazovka s informacemi o daném dokladu.

Po kliknutí na tlačítko pro řazení má uživatel možnost zvolit řazení:

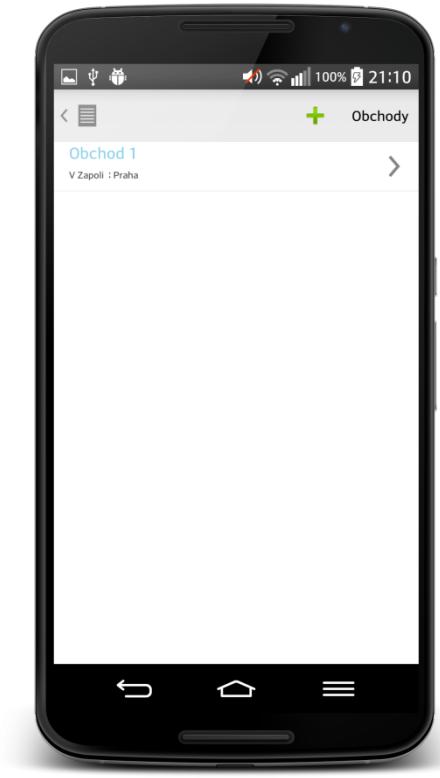
- Základní -> podle doby za kterou vyprší jedna z lhůt záručního listu
- Podle jména -> seznam se seřadí podle názvů záručních listů
- Obě možnosti je možné kombinovat se vzestupným nebo sestupným řazením

Menu:

- Zde jsou položky Záloha, Obchody, Kategorie a O aplikaci. Každá z nich spustí příslušnou obrazovku.

4.6 Obchody a kategorie

Po vybrání možnosti z menu na hlavní obrazovce je možné spustit obrazovku pro obchody nebo kategorie.



Obrázek 16 - Shop

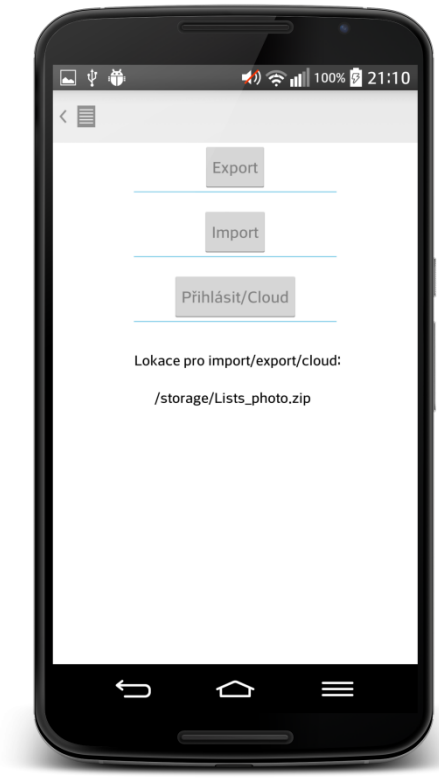
Obrazovky těchto možností se chovají stejně jako hlavní obrazovka se záručními listy. Je zde seznam a tlačítko pro přidání nové položky.

Seznam je řazený abecedně a u každé položky jsou zobrazeny dodatečné informace. U obchodů se jedná o název obchodu, ulici a město. Každou položku je možné stejně jako v seznamu na hlavní obrazovce rozkliknout a získat tak na další obrazovce podrobnější informace, nebo danou položku smazat, či editovat.

Tlačítko v levém horním rohu slouží pro návrat zpět do hlavní obrazovky se záručními listy.

4.7 Záloha

Po vybrání možnosti z menu na hlavní obrazovce je možné spustit obrazovku pro zálohování dat záručních listů.



Obrázek 17 - Backup

Na této obrazovce (Obrázek 17) jsou tři tlačítka:

1) Export – zkopíruje data a fotografie a založí archivační soubor
2) Import – z exportovaného souboru upět všechna data načte a obnoví stav aplikace a obnoví stav aplikace při exportu

3) Přihlásit/Cloud
    a. Pokud se uživatel ještě nepřihlásil ke svému účtu Google skrze aplikaci, je vybídnut k přihlášení
    b. Pokud je již přihlášen, objeví se dialogové okno, které umožní zadat lokaci, kam se má uložit exportovaný soubor na službu Disk Google. Odeslání a uložení souboru je poté v režii GooglePlayServices.

Navíc je na této obrazovce zmínka o lokaci, kam se exportuje soubor a do jaké složky má být daný soubor uložen včetně názvu pro importování dat a fotografií do aplikace.

5. Technická dokumentace

V této části dokumentu je popsán základní model tříd a jejich použití, obsahuje i návrh databáze, popis otevření a práce se zdrojovými soubory aplikace. Začleněny jsou dvě ukázky.

První ukázka je modelový příklad spuštění aplikace, inicializace databáze, vytvoření záznamu o jednom záručním listu a pořízení fotografií.

Druhá ukázka se týká kódu vybrané třídy a demonstruje způsob, jakým je členěn kód, práci s jinými třídami a implementaci grafického rozhraní.

Zdrojové soubory obsahující nezkompilovaný kód jsou roztříděny do balíčku podle filosofie programovacího jazyka JAVA. Tyto balíčky jsou implementovány jako samostatné celky. Pro vlastní zpracování instrukcí nepotřebují žádné jiné projektové soubory, kromě nativních knihoven operačního systému Android.

Každý balíček i aplikace nainstalovaná do operačního systému Android musí mít vlastní unikátní název, aby nedošlo ke konfliktu. Pro tento projekt je vybrán název: *lists* a pro vnitřní identifikaci projektových zdrojů se používá *com.example.*balíček. Název je vybrán s ohledem na studijní použití.

Celý projekt je psán v anglickém jazyce, stejně tak i popisy všech obrázků ukazující části maturitní práce.

5.1 Práce se zdrojovými soubory aplikace

Celý projekt je napsán v programátorském IDE Eclipse, které spadá pod licenci *Apache 2.0* (1) a zdrojové kódy Google Play Services Lib spadají pod licenci *CreativeCommons* (7).

Pro umožnění zpracování zdrojových souborů projektu jsou minimální požadavky:

- Exportovaný projekt viz příloha
- Eclipse verze: 3.7, 4.2, 4.3, 4.4
- Operační systém:
    - Windows
    - Mac
    - Linux/GTK
- Android Development Tools for Eclipse
- Android SDK:
    - API - 19 (Android v.4.4.2) - celé
    - Android SDK Tools Rev. 24.1.2
    - Android SDK Platform-tools Rev. 21
    - Android Support Repository Rev. 11
    - Android Support Library Rev. 21.0.3
    - Google Play services Rev. 22
- Do projektu je doporučeno importovat google-play-services_lib získatelnou ze SDK
    - Důležitá pro chod Cloudových služeb aplikace, musí být aktuální
- JAVA standard edition verze 8, update 11
    - JAVA SDK
- Je doporučeno se ujistit zda *Eclipse* obsahuje rozšíření *LogCat*, kde se zobrazující konzolové debuggovací zprávy.
- V Eclipse nastavit
    - knihovnu GooglePlayServices_lib
    - Build Version API 20(L)

- Možné problémy
    - Jiná veze Eclipse může ukazovat duplicitní XML záznamy
        - Kompilátor je při určitém nastavení ignoruje
        - Řešení – smazat
    - Chybějící knihovy
        - Řešení – opakovat znovu všechny body
    - Eclipse nemá target version
        - Nastavit na API 19

Pro pouhé čtení zdrojových souborů je potřeba běžný textový editor, jsou exportovány ve formátu *.java*.

5.2 Minimální požadavky a konfigurace

Pro instalaci aplikace na zařízení je nutné splňovat minimální požadavky:

- Operační systém:
  - Android s minimálním API 19 (Android v.4.4.2)
  - Pouze pro testování možné API 16 (Android v.4.1.2)
  - Operační systém Android bez nadstaveb od výrobce
- Nainstalované služby
  - Google Play Services
  - Google Drive
  - Aplikace pro pořizování fotografií umožňující spolupráci s jinými aplikacemi
  - Aplikace pro zobrazování fotografií umožňující spolupráci s jinými aplikacemi
- Volné místo
  - Minimálně 6 MB pouze pro aplikaci
- Operační paměť:
  - Minimálně 256 MB
- Procesor:
  - Procesor splňující standard ARMv7
- Nastavení:
  - Debuggovací mód
  - Povolení instalace z neznámých zdrojů

Doporučené zařízení:

- LG Nexus 5

5.3 Layout, AndroidManifest a překlad

Kostru pro dodatečná data vytváří značkovací jazyk XML. Pomocí tohoto jazyka jsou tvořeny nejdůležitější části aplikací, které nemají přímo činný charakter.

AndroidManifest – Soubor ve kterém se nachází informace o aplikaci, tento soubor je čten ještě před instalací samotné aplikace. Obsahuje například minimální verzi operačního systému Android, oprávnění, která aplikace vyžaduje, název samotné aplikace včetně všech registrovaných komponentů, se kterými bude operační systém pracovat. Komponenty, které jsou registrované, umožňují operačnímu systému vytvářet například historii *Activit*, směrování podnětů pro spuštění *AlarmManageru* aplikací, nebo *Broadcast* – globální změny (infromace o tom, že se změnil čas, připojila sluchátka apod.).

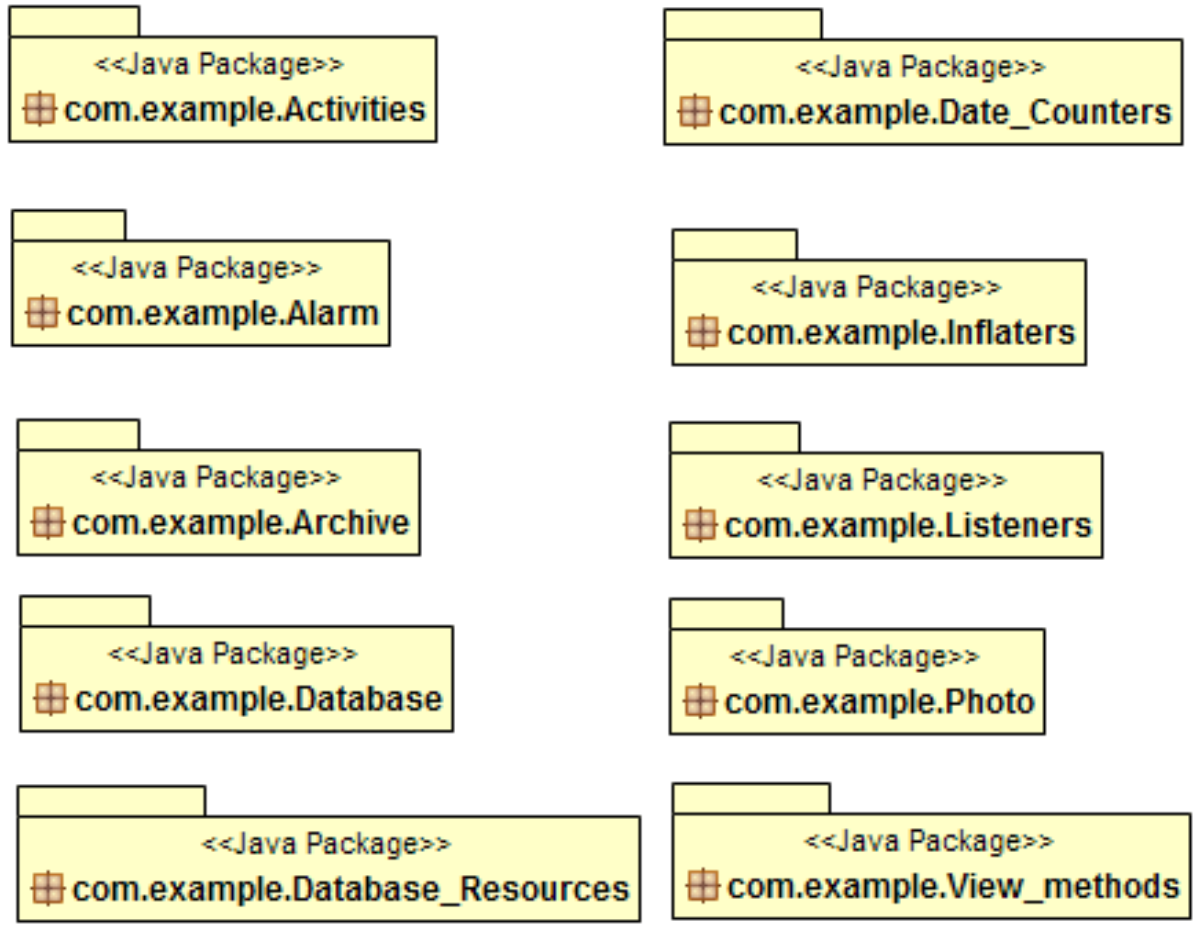
Layout – jedná se zápis grafického rozhraní ve formátu XML. Tento zápis by se dal přirovnat jako kombinace webového způsobu zápisu HTML a CSS. Pro *Layouty* stejně jako pro všechna doplňující data (zdroje) je vyhrazena vlastní složka. S těmito soubory se pracuje podle toho, v jaké složce se nachází, nikoliv podle jejich obsahu, formátu.

Jedna z přidaných funkcí nad rámec zadání je i překlad celé aplikace do českého a anglického jazyka. Díky vlastnosti pracování se soubory podle složek, je možné vytvářet asociace grafických prvku a jejich obsahu. Například je možné místo statického zadávání textu do textového pole uložit odkaz na jméno položky v jedné z těchto složek. Jména položek budou v každém jazyku stejná, ale mění se jejich obsah.

Operační systém Android, bez práce programátora aplikace, sám rozlišuje, jakou složku pro daný jazyk má používat. To umožňuje pracovat i s překladateli, kteří se nikdy nemusí dotknout kódu.

5.4 Java balíčky

Projekt je členěn pomocí balíčků do logických celků tak, aby bylo možné se jednoduše orientovat anebo pracovat na každém balíčku separátně od projektu. Obrázek 18 ukazuje názvy jednotlivých balíčků.



Obrázek 18 – Packages (Balíčky)

Popis jednotlivých balíčků:

Activities:

- Obsahuje veškeré třídy, které se v programování aplikací pro operační systém Android nazývají Activity. Jedna Activita je synonymem pro jednu zobrazenou obrazovku aplikace.
    - Třídy *NewList, EditList, InfoList* apod.

Alarm:

- Obsahuje třídy pracující s nativní knihovnou operačního systému Android – *AlarmManager*. Zajišťuje spouštění třídy *CheckAlarm* v daný čas pro kontrolu záručních listů.
    - Třídy *Alarm, CheckAlarm*

Archive:

- Zajišťuje exportování, importování, archivaci fotografií včetně celé databáze pro zálohování nebo pozdější uložení na Cloud.
    - Třídy *ExportImport, ZipUnzip*

Database:

- Implementuje vytvoření, upgradování databáze, její naplnění základními hodnotami a zprostředkovává komunikaci mezi aplikací a databází.
    - Třída *MySQLiteHelper*

Database_Resources:

- Balíček, který slouží jako nejnižší vrstva při komunikaci aplikace s databází. Jednotlivé třídy zastupují stejné tabulky v databázi, vytváří dotazy, zpracovávají I/O funkcionalitu. Třídy jsou rozděleny na zastupující třídu a třídu tvořící dotazy.
    - Třídy *WarrantyList, WarrantyLists_SQL, Shop, Shops_SQL*

Date_Counters:

- Obsahuje třídu sloužící pro vypočítání data konce refundací, záručních lhůt nebo zjištění data pro zobrazení notifikace pro jednotlivé záruční listy.
    - Třída *Date_Counters*

Inflaters:

- Balíček pro práci s grafickými prvky, které se dodatečně vkládají do již existujícího uživatelského rozhraní, jako jsou popup okna nebo dynamické seznamy.
    - *Třídy Adapter, PopupMethods*

Listeners:

- Obsluhuje veškeré uživatelské akce jako klikání na objekty, potvrzování vyskakovacích oken apod.
    - Třídy *ContenHandleMethods, NavigationMethods*

Photo:

- Balíček pro práci s fotografiemi. Zajišťuje pořizování, zpracování, uložení fotografií a jejich asociaci s daným záručním listem. Umožňuje i jejich zobrazení. Spolupracuje s aplikací pro pořizování fotografií a galerií.
    - Třídy *Adapter, Photo_camera*

View_methods:

- Tento balíček zpracovává a kontroluje veškeré uživatelské vstupy jako například zadávání dat přes klávesnici.
    - Třídy *EditMethods, GetCheckMethods*

5.4.1 Activities

Activity jsou třídy, které mají jako jediné přístup ke grafickému rozhraní aplikace. Tyto třídy rozšiřují nativní třídu operačního systému Android *Activity*. Jedna třída je synonymem pro jednu obrazovku, která je uživateli zobrazena.

Obrázek 19 níže ukazuje jednotlivé Activity a jejich metody.

Obrázek 19 - Activities

Popis jednotlivých Activit je rozdělen na logické celky:

Záruční listy:

NewList – Určená pro interakci s uživatelem a provádí ho jednoduchým vytvořením záručního listu.

- Implementuje: *NavigationListeners a DatabaseResources*

InfoList – Určená pro zobrazení záručního listu včetně fotografií. Implementuje možnost vrácení se zpět do seznamu záručních listů nebo pokračování na editaci, či zobrazení fotografií.

EditList – Pracuje podobně jak *NewList*, jen upravuje záruční listy místo zakládání nových.

ListsList – Zobrazuje seznam záručních listů. Implementuje *DatabaseResources, NavigationListeners*, řazení záručních listů, menu. Jedná se o hlavní obrazovku aplikace. Jejím popisem se zabývá ukázka č.2.

Obchody:

NewShop - Určená pro interakci s uživatelem a provádí ho jednoduchým vytvořením obchodu.

InfoShop – Určená pro zobrazení obchodu. Implementuje možnost vrácení se zpět do seznamu obchodů nebo pokračování na editaci.

EditShop– Pracuje podobně jak *NewShop* jen upravuje obchody místo zakládání nových.

ListsList – Zobrazuje seznam obchodů. Implementuje *DatabaseResources, NavigationListeners*.

Kategorie:

NewCategory - Určená pro interakci s uživatelem a provádí ho jednoduchým vytvořením kategorie.

InfoCategory – Určená pro zobrazení kategorie. Implementuje možnost vrácení se zpět do seznamu kategorií nebo pokračování na editaci.

EditCategory– Pracuje podobně jak *NewCategory* jen upravuje kategorie místo zakládání nových.

ListsCategories – Zobrazuje seznam kategorií. Implementuje *DatabaseResources, NavigationListeners*.

Cloud:

Cloud – Třída implementující přihlášení do účtu Google, ukládání souborů na GoogleDrive a pracuje s připojením do Google Cloud Computing. Více v kapitole DriveApi.

Inicializační Activita:

InitActivity – Třída navazující spojení s databází, zaručuje prvotní vytvoření databáze a zajišťuje její naplnění základními hodnotami, jako jsou například typy notifikací nebo záznamy určené pro obchody, které nepatří k žádnému záručnímu listu a naopak. Bez úspěšného provedení této inicializační procedury není možné aplikaci používat.

Archiv:

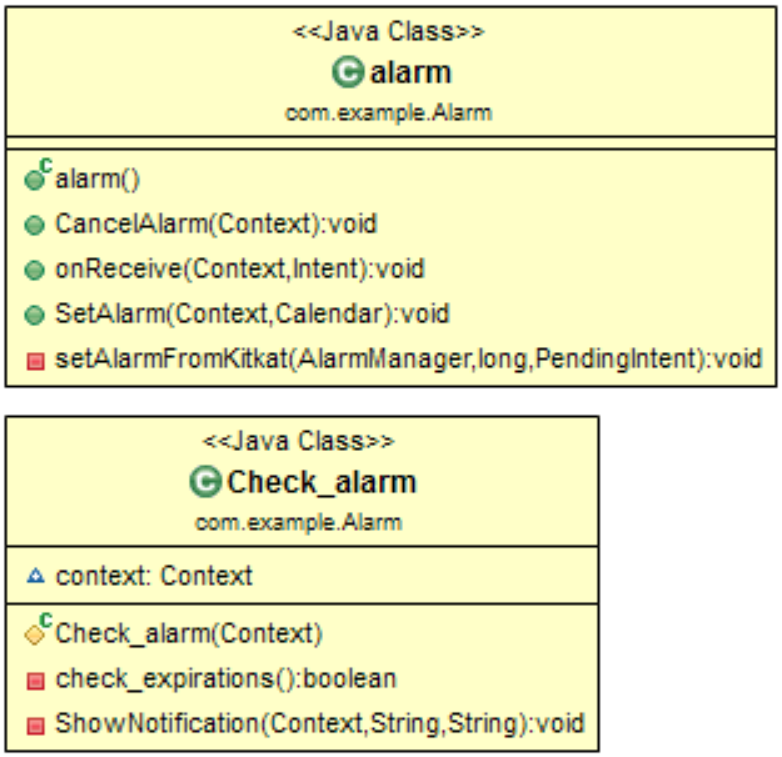
Archive – Slouží jako uživatelské rozhraní k provádění akcí, které jsou dále zpracovávány ve třídách *ZipUnzip*, *ExportImport*, nebo *Cloud*. Slouží pro archivaci databáze včetně fotografií, importování exportovaného souboru nebo jeho uložení na Cloud.

O aplikaci:

About: Activity, které pouze zobrazuje základní informace o aplikaci, jako je její verze nebo jméno tvůrce, či licence.

5.4.2 Alarm

Balíček *Alarm* obsluhuje práci s *AlarmManager* nativní knihovnou/službou operačního systému Android. Umožňuje tak spouštět danou sekvenci instrukcí, které se provedou v pokud možno nejbližší možnou chvíli od zadaného času. Kdy je umožní operační systém Android, záleží na tom, jestli je zařízení zrovna v režimu spánku, nebo je vypnuté.



Obrázek 20 - Alarm

Alarm – tato třída obsluhuje nastavování času pro spuštění dané sekvence instrukcí a také zachycuje toto spuštění ze strany operačního systému. Rozlišuje nastavování pro operační systémy Android před verzí Kitkat (4.4.x) a po ní. Umožňuje i nastavení časového spínače vynulovat.

Check_alarm – Tato třída je zpracovávána pouze na popud operačního systému, který umožnil její zpracování v daný čas. Je volána pouze metodou onReceive() ze třídy *Alarm.*

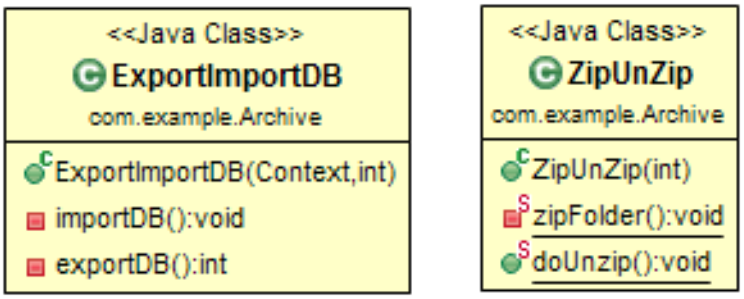
- Jejím účelem je kontrola dat pro vypršení záručních lhůt a refundací všech záručních listů, která ještě nevypršely. V případě nalezení shodného data s aktuálním datem spustí upozornění, které se objeví i s dodatečnými informacemi v navigační liště operačního systému Android.

5.4.3 Archive

Balíček *Archive* obsluhuje export, import, archivaci a obnovení databáze včetně fotografií.

Fotografie jsou uloženy ve vlastní složce na oddílu externí paměťové karty (vč. virtuální). Do této složky se vloží i kopie databáze a poté je celá složka archivována do souboru *.zip*.

Obnovení je opačný proces archivace, vytvoří se složka, do ní se nahrají fotografie vč databáze a ta se poté zkopíruje do složky určené pouze aplikaci pro její databáze.



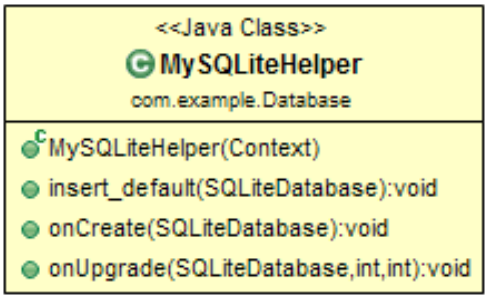
Obrázek 21 - Archive

ExportImportDB – Třída obsluhuje kopírování a vkládání souboru databáze.

ZipUnZip – Zpracovává vytvořenou složku pro archivaci nebo ji vytvoří. Obsluhuje archivaci této složky.

5.4.4 Database

Balíček, jehož účel je obsluha databáze z pohledu přístupu k ní.

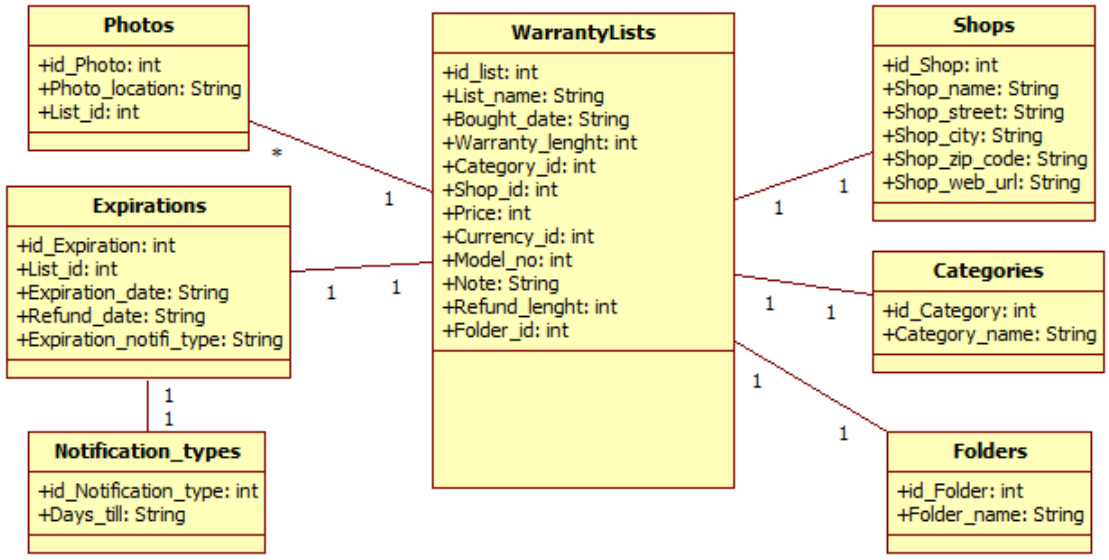


Obrázek 22 - Database

MySQLiteHelper – rozšiřuje API nativních knihoven pro práci s databází operačního systému Android. Zakládá databázi, zprostředkovává k ní přístup pomocí vytvoření nové instance na objekt *SQLite* a vkládá do databáze při prvním spuštění základní záznamy pro chod aplikace.

5.4.4.1 Návrh databáze

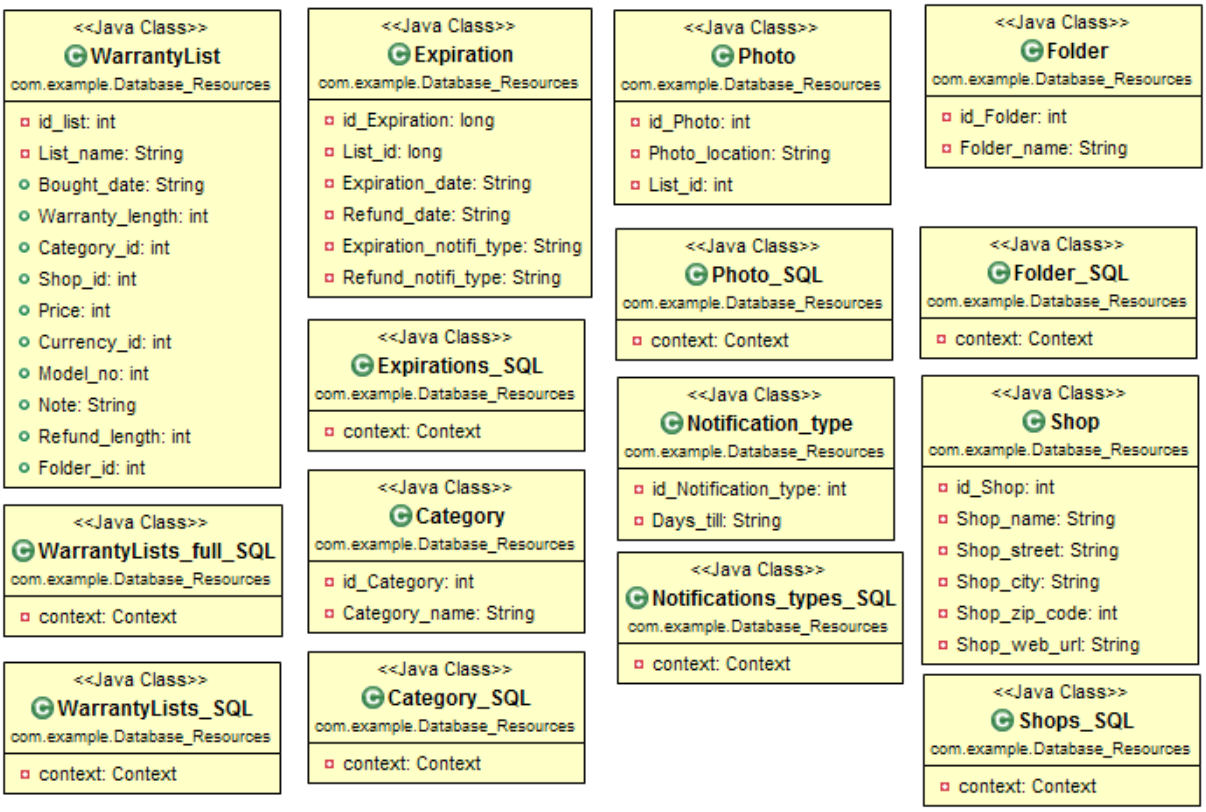
Použitá databáze je relační a přistupuje se k ní pomocí dotazovacího jazyka SQL nebo předem vytvořených funkcí z knihoven *SQLite* . Hlavní tabulka je WarrantyLists a na ní jsou ostatní napojené viz. Obrázek 23



Obrázek 23 – Database structure

5.4.5 Database_Resources

Balíček tříd sloužící jako spojová vrstva mezi relační databází a objektovém zpracovaní dat.



Obrázek 24 – Database_Resources

Třídy jsou tvořeny podle stejného schématu:

Třída zastupující tabulku v databázi:

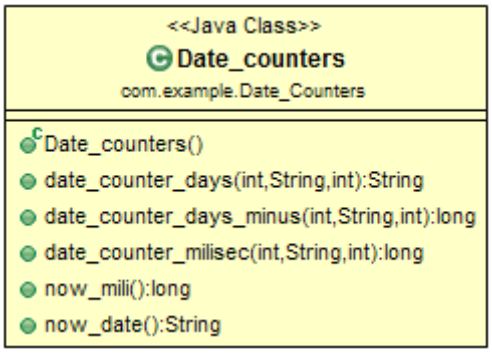
WarrantyList – tato třída je přímou kopií databázové tabulky WarrantyList a slouží jako objekt, ve kterém jsou přenášena data se strukturou této tabulky. Kromě základních atributů je rozšířen o *GET/SET* metody, které slouží jako I/O vrstva pro přístup k těmto datům.

Třída obsluhující SQL dotazy:

WarrantyList_SQL – třída implementující všechny potřebné dotazy na databázi např. *AddWarrantyList(WarrantyList), deleteWarrantyList(id).* Tyto metody mají jako vstup zastupující objekt tabulky napr. *WarrantyList* nebo identifikátor *id,* mohou vracet objekt zastupující tabulky např. *WarrantyList*, nebo pole těchto objektů, či *id* v případě nového záznamu.

5.4.6 Date_Counters

Balíček obsahující třídu pro zpracování dat o datu a čase.



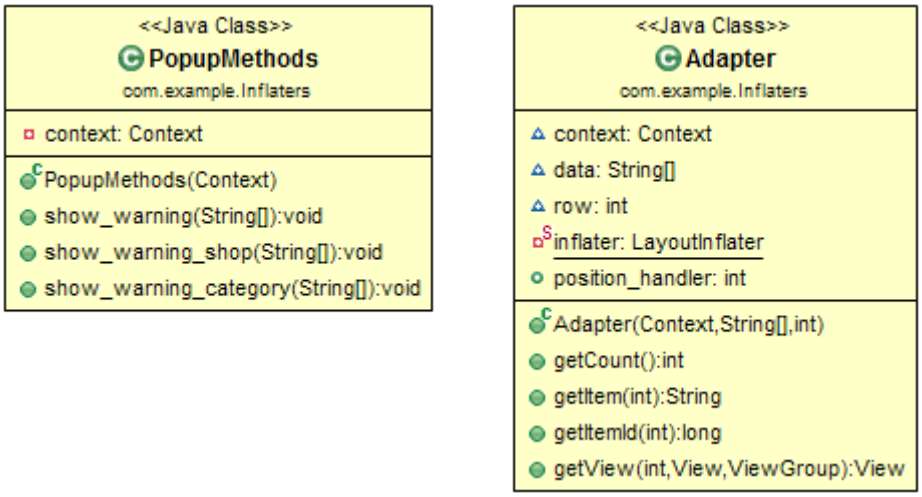
Obrázek 25 – Date_counters

Date_counters – obsahuje metody pro výpočet:

- Datumu na základě, počtu dnů a datu koupi -> kolik zbývá dnů
- Datumu, který vznikl odečtením data a počtu dnů -> pro notifikace
- Milisekund na základě, počtu dnů a datu koupi -> kolik zbývá milisekund
- Aktuální čas v milisekundách od roku 1969
- Aktuální datum formátované na *yyyy-MM-dd*

5.4.7 Inflaters

Balíček tříd, které pracují s grafickými prvky, které se vkládají do již existujícího uživatelského rozhraní.



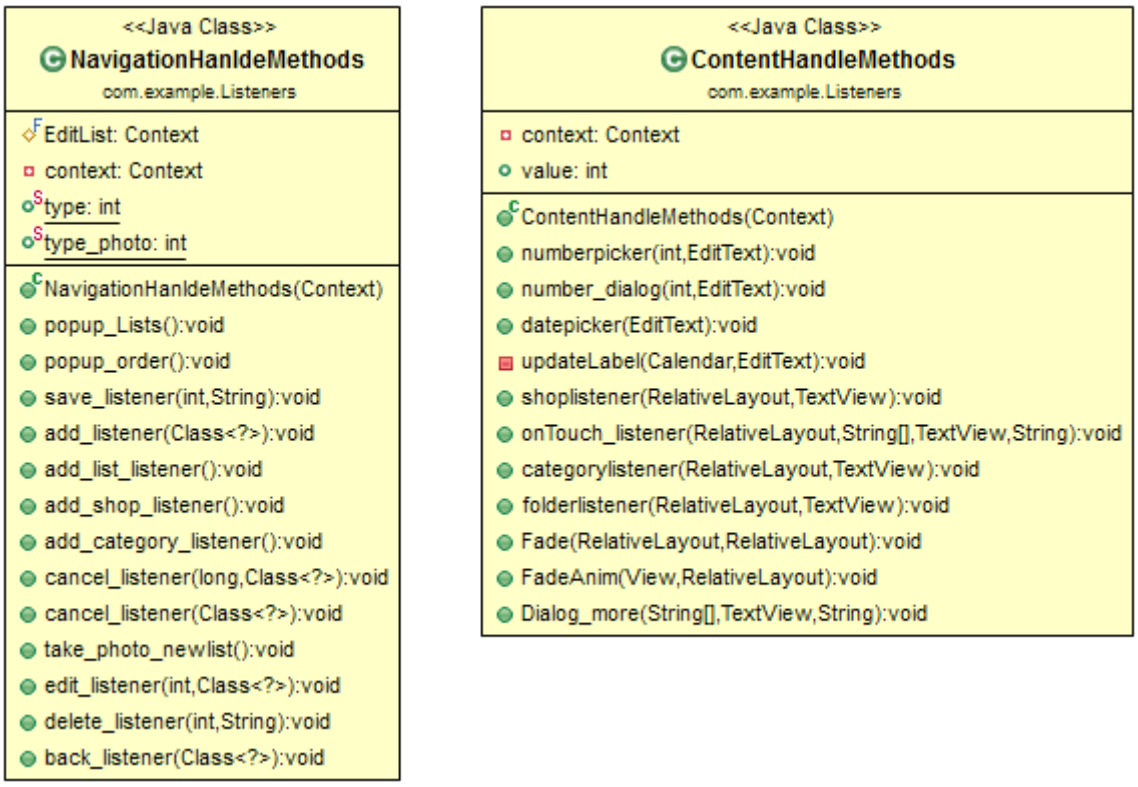
Obrázek 26 - Inflaters

PopupMethods – Třída obsahující práci s popup okny, která jsou ukazována například při špatném zadání uživatelského vstupu. Obsahuje metody pro zobrazování těchto oken pro jednotlivé *Activity* a zvláště pro kategorie a obchody.

Adapter – tato třída obsluhuje statický seznam, který je možné zpracovávat podle potřeby. Vytváří seznam na základě předem daných kritérií, ten poté může obohatit o dynamické části, jako jsou ikony, nebo barva textu. Tato třída také obsluhuje jednotlivé uživatelské vstup, jako například posouvání seznamu, či výběr jednotlivých položek. Tento seznam je vložen do již existujícího rozhraní, a proto je možné tento seznam tvořit dle potřeby za chodu aplikace, bez nutnosti vykreslení celé *Activity* znovu.

5.4.8 Listeners

Tento balíček obsahuje dvě třídy. Jedna je určená pro naslouchání veškerých uživatelských vstupů v oblasti navigační lišty aplikace, kde se nalézají například tlačítka zpět, nebo menu. Druhá slouží k naslouchání uživatelských vstupů v oblasti aplikace, kde se nalézá obsah.



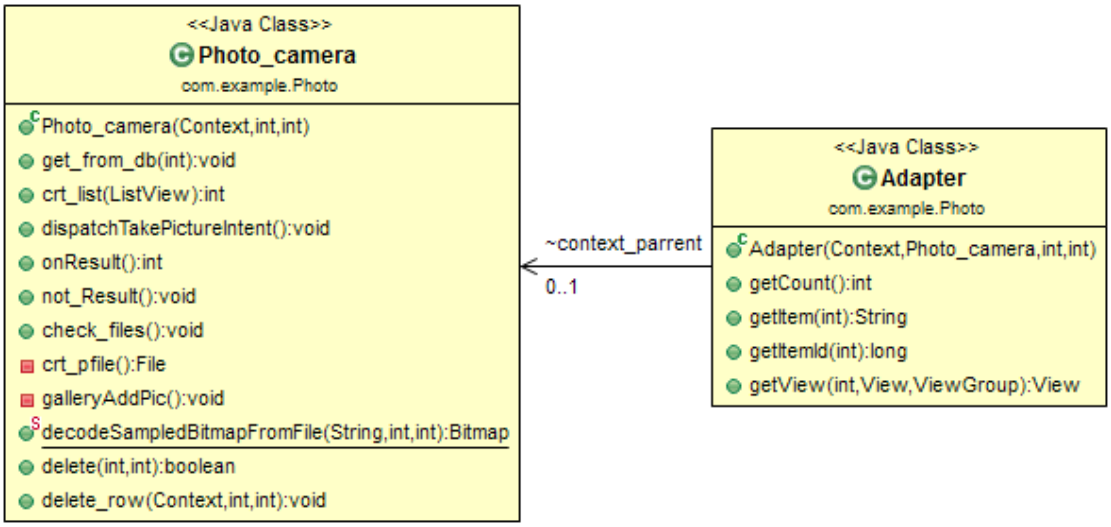
Obrázek 27 - Listeners

NavigationHandleMethods – Tato třída obsluhuje navigační lištu aplikace. Obsahuje všechny metody napříč všemi *Activitami*. Je součástí každé *Activity* a jedná se o základní navigační třídu aplikace.

ContentHandleMethods – Tato třída obsahuje všechny metody pro práci s obsahem uvnitř *Activit*. Je součástí zejména *New* a *Edit Activit*. Umožňuje zobrazování dalších parametrů záručních listů včetně animace nebo popup okna pro výběr data nebo počtu dnů.

5.4.9 Photo

Balíček pracující s fotografiemi. Umožňuje jejich pořizování, zobrazování v galerii, nebo v seznamu, ukládání do databáze, mazání fyzicky fotografií i záznamů z databáze, v případě nalezení záznamu o fotografii, která není k nalezení, umí tento záznam obejít.



Obrázek 28 - Photo

Photo_camera – Tato třída obsluhuje vše, co je s fotografiemi v této aplikaci, kromě archivace a zobrazení v seznamu, spojeno.

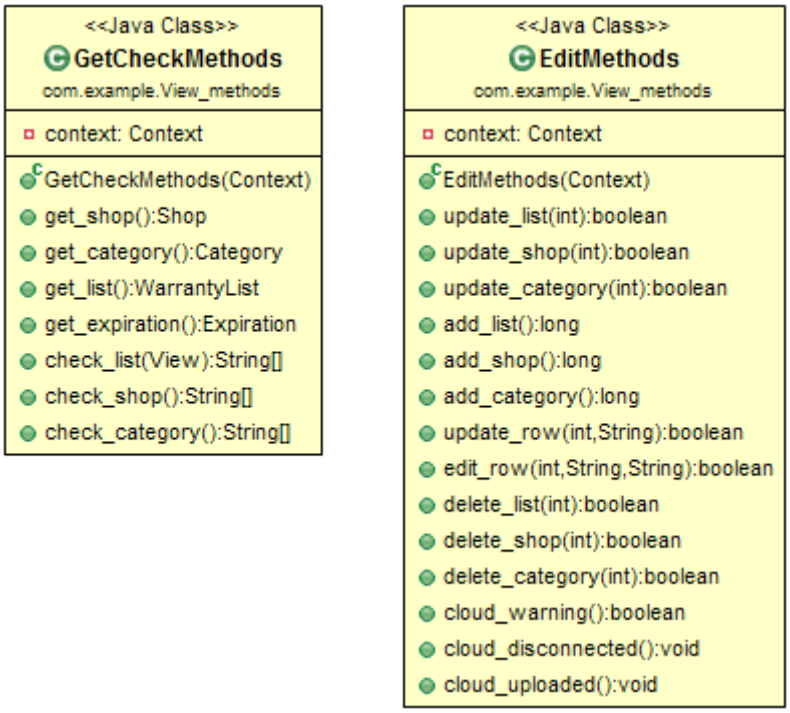
Umožňuje:

- Fotografie pořizovat pomocí uživatelem vybrané aplikace
- Ukládá fotografie na externí kartu, vytváří složku pro toto uložení
- Fotografie podle potřeby zmenšuje tak, aby miniatura odpovídala podle velikosti displeje. Na menších displejích jsou miniatury fotografií menší a na větších větší, ale vždy ve stejném poměru stran a velké vzhledem k okolnímu grafickému rozhraní.
- Tvoří dynamické seznamy na základě počtu fotografií, vkládá do nich možnost mazání, nebo zobrazování v galerii. Tyto možnosti umožňuje podle aktuální *Acitivity*
- Fotografie maže z fyzického umístění i záznamy o nich v databázi

Adapter – na základně seznamu získaného z třídy *Photo_camera* zobrazuje seznam fotografií. Je potomkem třídy *Photo_camera* a žádná jiná třída ji nevyužívá.

5.4.10 View_methods

Balíček tříd slouží ke zpracování uživatelských vstupů, jejich kontrole a následnému přístupu k záznamům v databázi.



Obrázek 29 – View_methods

GetCheckMethods – Tato třída dělí své metody na dva druhy:

- Metody získávající uživatelem zadaná data a ukládá je do objektů zastupující tabulku v databázi
- Metody kontrolující ona data

EditMethods – Tato třída na základě úspěšného uživatelského vstupu mění, přidává, nebo maže záznamy z databáze.

5.5 Drive API

Drive API je balíček knihoven obsažený v GooglePlayServices_lib volně získatelný z Android SDK. Celé tyto knihovny zprostředkovávají přístup ke službám společnosti Google.

Jedna z těchto služeb je GoogleDrive, která umožňuje na základě účtu u společnosti Google ukládat data na jejich Cloudové uložiště.

Pro využití této služby je nutné mít výše zmíněný účet Google a zaregistrovat svoji aplikaci v Google Developer Console. Aplikace popsaná v tomto dokumentu, je registrována jako testovací, a proto je možné, že bude dosažen maximální počet uživatelů a Cloudové služby nebudou fungovat, jak mají.

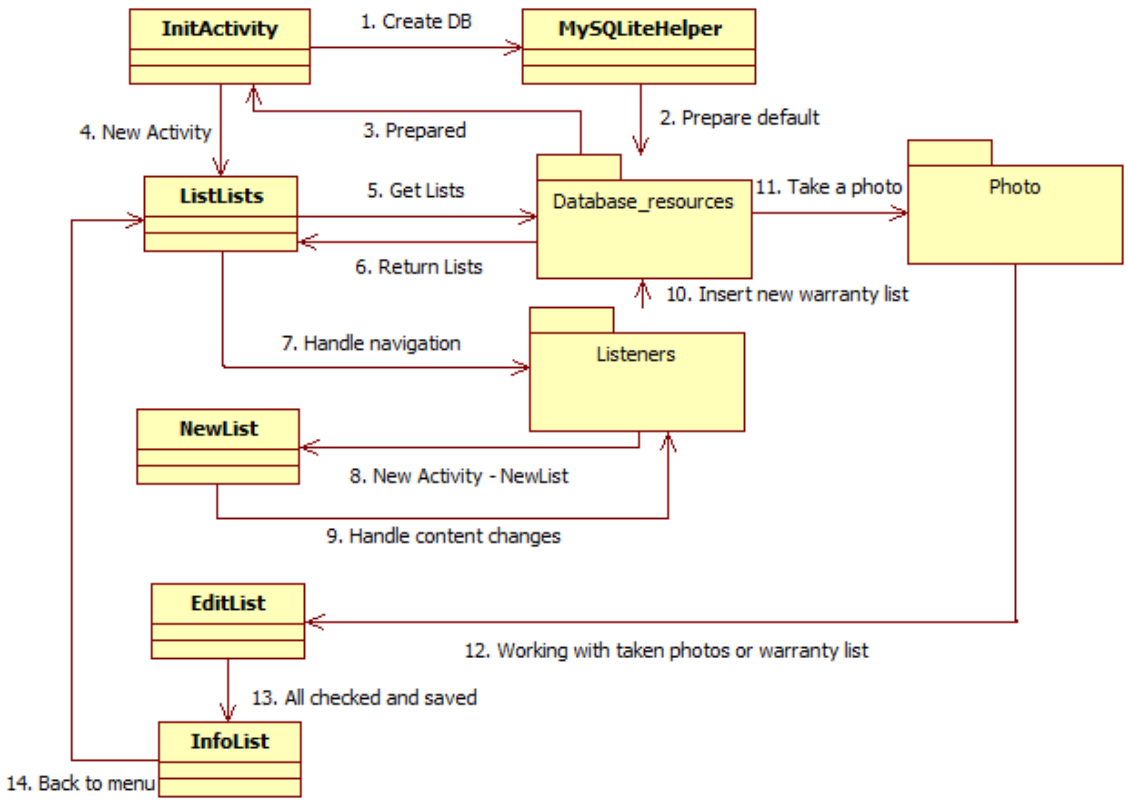
Implementace této knihovny tříd s sebou přinesla velké zvýšení celkového objemu dat aplikace, zejména při zachování všech překladů a doprovodných obrázků.

Do aplikace přinesla nové možnosti komunikace se sítí Internet a rozšířila tak, celkovou rozsáhlost.

5.6 Modelová situace No.1

Popis modelové situace, která ukazuje proces spuštění aplikace a vytvoření záručního listu včetně pořízení fotografie (Obrázek 30).

Pro přehlednost jsou vynechány některé třídy zejména *DateCounters*, zastupující třídy tabulek databáze, nebo třídy pro práci s uživatelem zadanými daty, místo nich jsou použity balíčky.



Obrázek 30 – Model situation

Popis jednotlivých kroků:

1. *InitActivity* inicializuje databázi pomocí třídy *MySQLiteHelper*
2. Naplňuje databázi základními záznamy, jako jsou notifikační typy nebo prázdné kategorie a obchody.
3. Po dokončení celé procedury pro přípravu databáze se opět činnost vrací do *Activity InitActivity,* kde se nastaví *Alarm* na každodenní kontrolu někdy kolem 12:00am. Spouští novou *Activitu ListLists*
4. *ListLists* vykreslí uživatelské rozhraní
5. Dotáže se databáze o data týkající se záručních listů
6. *Database_resources* provede dotaz na databáze, převede získaná data na list a ten vrátí do *ListLists*
7. *ListLists* zpracuje a vykreslí seznam, který získala. V modelové situaci se očekává, že uživatel chce založit nový záruční list, proto se zpracuje akce stisknutí tlačítka.
8. Tlačítko bylo rozpoznáno jako spouštěč nové *Activity NewList* a je tedy spuštěna
9. Uživatel zadá požadovaná data v pořádku a jsou odeslána k překontrolování.
10. Nová data se uloží do databáze
11. Předpokládá se, že uživatel stiskl tlačítko pro pořízení fotografie a tak je činnost aplikace přesměrována do třídy *Photo*, kde je fotografie pořízena, zpracována a uložena.
12. Třída *EditList* je zavolána, aby uživatel mohl pracovat s pořízenou fotografií/přidat další.
13. Po uložení je činnost přesměrována do *Activity InfoList,*kde uživatel vidí výpis informací o záručním listu včetně fotografií. Takto ji uvidí pokaždé, dokud nebude proedena úprava záručního listu/fotografií.
14. Předpokládá se, že uživatel stiskl navigační tlačítko telefonu, nebo tlačítko v navigační oblasti pro akci zpět. Činnost aplikace se vrátí zpět do *ListLists,* kde vidí v seznamu

5.7 Modelová situace No.2

Tato modelová situace ukazuje třídu *ListLists* včetně komentářů uvnitř kódu.

```
public class ListsList extends Activity
{
	public int end_butt = 0; // Proměnná pro počet stisků zpět
	public static int  sort_type = 0; // Proměnná pro typ řazení
	public static int asc_desc = 0 ; // Proměnná pro vzestupné/sestupné řazení
	static Context context; // Activity Context

	// Po vytvoření Acitivity se zavolá tato metoda
	// s parametry posledního stavu proměnný apod.
	@Override
	protected void onCreate(Bundle savedInstanceState)
	{
		super.onCreate(savedInstanceState); // Obnovení posledního uloženého stavu
		context = this; // Uložení Contextu
		setContentView(R.layout.lists_activity_new);// Vykreslení daného Layoutu
		navigate_listeners(); // Inicializace listenerů pro vstupy uživatele
		showall(); // Metoda pro získání a zobrazení záručních listů
	}

	public void navigate_listeners()
	{
		// vytvoření nové instance NavigationHanldeMethods
		NavigationHanldeMethods listeners = new NavigationHanldeMethods(this);
		listeners.add_listener(NewList.class); // Listener pro uživatelský stisk +
		listeners.popup_Lists(); // Menu
		listeners.popup_order(); // Řazení
	}
	// Použití pro libovolné HW tlačítko telefonu
	// Chytá tlačítko "zpět" a "více", při každém odmáčknutí tlačítka
	@Override
	public boolean onKeyUp(int keyCode, KeyEvent event) {
		if (keyCode == KeyEvent.KEYCODE_MENU)
		{
			final ImageButton button1 = (ImageButton) findViewById(R.id.Overflow);
			button1.performClick(); // Umělé zavolání stitku uživatele pro zobrazení rolovacího menu
			return true;
		}
		else
			if ( keyCode == KeyEvent.KEYCODE_BACK)
			{
				if (end_butt == 1) this.finish(); // Aplikace se vypne až druhy stisk uživatele "zpět" tlačítka
				else
				{
					end_butt = 1;
					// Toast je malá zpráva, která se zobrazí uživateli na krátkou chvíli
					Toast.makeText(this, "Press back button once more to end application.",
					Toast.LENGTH_LONG).show();
				}
			}
		return super.onKeyDown(keyCode, event);
	}
```

```
	// Procedura pro zobrazení seznamu záručních listů
	public static void showall()
	{
		// Nová instance třídy pro SQL dotazy na záruční listy do DB
		WarrantyLists_SQL WarrantyList = new WarrantyLists_SQL(context);
		// Do seznamu se naplní hodnoty z metody pro získání záručních listů, SQL dotaz je včetně řazení
		List<String> List = WarrantyList.getAllWarrantyLists_View(sort_type, asc_desc);
		if(List.size() != 0) // Ošetření prázdného seznamu
		{
			RelativeLayout cont = (RelativeLayout) ((Activity) context).findViewById(R.id.Content);
			cont.setVisibility(View.VISIBLE); // Místo pro zobrazení seznamu je v základu shované
			// Pro Adapter je nutné staticky definovaný Array ne List, vč velikosti
			String[] stockArr = new String[List.size()];
			stockArr = List.toArray(stockArr);
			// Identifikace místa pro zobrazení seznamu a pro práci Adaptéru
			final ListView listview = (ListView) ((Activity) context).findViewById(R.id.listview);
			// Na identifikované místo se vykreslí statický seznam z třídy Adapter
			listview.setAdapter(new Adapter(context, stockArr, R.layout.row_new));
		// Přidělení Listeneru na seznam, listener je static proto je nutné předávat ukazatel na context a ne pouze this
			listview.setOnItemClickListener(new OnItemClickListener()
			{
				@Override
				// Metoda, která se provede po kliku na jeden prvek
				public void onItemClick(AdapterView<?> arg0,View arg1,int pos,long arg3)
				{
					// Identifikace klinutého elementu, získání jeho dat
					String selected_item = listview.getItemAtPosition(pos).toString();
					String[] colums = selected_item.split(";"); // Ze získaných hodnot vytvoření Array
		// Inicializace objektu pro zasílání dat se schopností spustit jinou Activitu/Service apod. Pošle se do Activity InfoList
					Intent intent = new Intent(context, InfoList.class);
					int id = Integer.parseInt(colums[0]); // Získání id záručního listu ze získaných hodnot
					intent.putExtra("id", (int) id); // Vožení do senderu
					context.startActivity(intent); //Spuštění Activity InfoList a zaslání id
				}
			});
		}
		else
		{
// Pokud je List prázdný zobrazí se uživateli místo seznamu jiný element, upozorňující na to, že může začít přidáním nového záručního listu
			RelativeLayout cont = (RelativeLayout) ((Activity) context).findViewById(R.id.Content);
			cont.setVisibility(View.GONE);
			RelativeLayout start = (RelativeLayout) ((Activity) context).findViewById(R.id.start);
			start.setVisibility(View.VISIBLE);

		}
	}
}
```

6. Závěr

Dlouhodobá maturitní práce byla pro mě největší výzvou za celé čtyři roky studia střední průmyslové školy. Za tuto dobu jsem si zvykl na rozsah projektů v rámci 30 stránek, nebo maximálně 200 řádků kódu. Tento dokument má kolem 70 stránek a aplikace sama obsahuje něco málo pod 6 000 řádků kódu, což mne posunulo daleko za hranice mých dosavadních možností. Tato práce mě tak pohltila, až mě samotného překvapuje, že jsem ji celou zvládl sám a ještě měl čas přidávat funkcionalitu nad rámec zadání.

Naučil jsem se spoustu nového a aplikace sama poskytuje silnou základnu pro další pokračování ve vývoji. Začal jsem jako programátor znalý PHP, JS, C, C# a dalších programovacích/skriptovacích jazyků. JAVA mne míjel velkým obloukem do doby, než-jsem si jej vybral jako hlavní programovací jazyk pro tuto aplikaci a poznal ho tak mnohem blíže.

V tuto chvíli myslím, že nám dobré základy pro další vývoj různorodých aplikací na platformě Android. Vždyť jen aplikace implementovala tvorbu a práci s uživatelským rozhraním, práci s databází a její celkové vytvoření, komunikaci s ostatními aplikacemi a službami, používání účtů Google, využití GoogleDrive, zpracování souborů při jejich archivaci a využití nativních knihoven operačního systému Android jako *AlarmManager, NotificationManager*.

Seznámil jsem se s programovacím modelem Model-view-controller. Naučil jsem se pracovat a koexistovat s operačním systémem Android, využívat jeho historii aplikací, pracovat s *BroadcastReceivery*. Zjistil jsem, jak šetřit zdroje mobilního zařízení tak, abych nezruinoval jeho HW nebo SW zdroje. Pochopil jsem, jakým způsobem funguje životní cyklus aplikací. Sžil jsem se s nativními třídami operačního systému.

Kromě programování jsem pronikl do pozadí vývoje aplikací, poznal jsem, jaké je pracovat s Google Developer Console, která se používá při zveřejňování aplikací, nebo při používání služeb Google.

Získal jsem velkou zkušenost s celkovým vývojem aplikací, která se jistě v praxi neztratí, a mohu ji považovat za jednu z nejdůležitějších přínosů studia střední školy. Shrnu-li celé studium, myslím, že mne škola na tuto práci po celou dobu připravovala.

Tématem práce bylo vytvoření aplikace pro platformu Android, která umožňuje evidenci záručních listů včetně pořizování fotografií. Jejím dalším hlavním tématem bylo naučit žáka co nejvíce nad rámec běžné středoškolské výuky tak, aby mohl své nově nabyté zkušenosti předvést před maturitní komisí a prokázat tak své kvality jako studenta hodného titulu žáka s ukončeným středoškolským vzděláním.

V zadání nicméně nebylo, jakým způsobem tohoto cíle docílit. Mohl jsem například evidovat pouze jeden záruční list a jeden obchod včetně jedné fotografie s notifikací o tom, že někdy vyprší jeho záruční doba a to celé na jedné obrazovce.

Místo toho jsem vytvořil systém, který umožňuje evidovat velké množství záručních listů včetně mnoha fotografií u každého a stejně tak i obchodů včetně mnoha obrazovek pro komunikaci s uživatelem, popup okna. Jako vzor jsem si zvolil Holo Design a splnil jsem jeho návrh UX.

Přidal jsem i kategorie, řazení záručních listů, celou aplikaci jsem přeložil do dvou jazyků, které se mění podle toho, jakým jazykem komunikuje celý operační systém Android s uživatelem.

Implementoval jsem vlastní řešení archivace fotografií a celé databáze pro zaručení přenositelnosti dat aplikace. K tomu jsem přidal Cloudové řešení pomocí účtu Google, kam uživatel může rovnou archivovat všechna data. Tato archivovaná data může opět nahrát do aplikace na jiném zařízení.

Tento projekt a SPS Na Proseku mi daly velkou životní zkušenost, bez které bych se neobešel. Bez přípravy, kterou mi tato škola dala, bych nikdy nedokázal tento projekt s úspěchem dokončit.

7. Seznam použité literatury a zdroje

1. **Apache.** apache.org. *Apache.* [Online] Apache, 21. 3 2015. [Citace: 21. 3 2015.] https://www.apache.org/licenses/LICENSE-2.0.

2. **Google Inc.** www.android.com/history/. *www.android.com.* [Online] Google Inc., 25. 2 2015. [Citace: 3. 3 2015.] https://www.android.com/history/.

3. **gs.statcounter.com.** cdn.arstechnica.net/wp-content/uploads/2014/04/t-mobile_g1.jpg. *gs.statcounter.com.* [Online] gs.statcounter.com, 3. 3 2015. [Citace: 3. 3 2015.] http://cdn.arstechnica.net/wp-content/uploads/2014/04/t-mobile_g1.jpg.

4. **Google Inc.** developer.android.com/images. *developer.android.com.* [Online] Google Inc., 3. 3 2015. [Citace: 2015. 3 5.] http://developer.android.com/images/activity_lifecycle.png.

5. —. developer.android.com/images. *developer.android.com.* [Online] Google Inc., 3. 3 2015. [Citace: 3. 3 2015.] http://developer.android.com/images/service_lifecycle.png.

6. —. material-design.storage.googleapis.com/publish. *material-design.storage.googleapis.com.* [Online] Google Inc., 3. 3 2015. [Citace: 3. 3 2015.] http://material-design.storage.googleapis.com/publish/v_2/material_ext_publish/0Bx4BSt6jniD7TDlCYzRROE84YWM/materialdesign_introduction.png.

7. **creativecommons.org.** creativecommons.org/licenses. *creativecommons.org.* [Online] creativecommons.org, 22. 3 2015. [Citace: 22. 3 2015.] http://creativecommons.org/licenses/by/2.5/.

8. **Lacko, Ľuboslav.** *Vývoj aplikací pro Android.* Praha : COMPUTERS PRESS, 2015. str. 472. 9788025143476.

9. **arstechnica.com.** arstechnica.com/gadgets/2014/06/building-android-a-40000-word-history-of-googles-mobile-os/. *arstechnica.com/gadgets/.* [Online] arstechnica.com, 16. 6 2014. [Citace: 2015. 2 15.] arstechnica.com/gadgets/2014/06/building-android-a-40000-word-history-of-googles-mobile-os/.

10. **gs.statcounter.com.** gs.statcounter.com/#mobile+tablet-os-ww-monthly-201208-201503. *gs.statcounter.com.* [Online] gs.statcounter.com, 3. 3 2015. [Citace: 3. 3 2015.] http://gs.statcounter.com/#mobile+tablet-os-ww-monthly-201208-201503.

11. —. gs.statcounter.com/#mobile+tablet-os-ww-monthly-201503-201503-bar. *gs.statcounter.com.* [Online] gs.statcounter.com, 3. 3 2015. [Citace: 3. 3 2015.] http://gs.statcounter.com/#mobile+tablet-os-ww-monthly-201503-201503-bar.

12. **Google Inc.** developer.android.com/design/. *developer.android.com.* [Online] Google Inc., 3. 3 2015. [Citace: 3. 3 2015.] https://developer.android.com/design/index.html.

13. —. developer.android.com/design/style/iconography.html. *developer.android.com.* [Online] Google Inc., 3. 3 2015. [Citace: 3. 3 2015.] https://developer.android.com/design/style/iconography.html.

14. —. developer.android.com/guide/. *developer.android.com.* [Online] Google Inc., 3. 3 2015. [Citace: 3. 3 2015.] https://developer.android.com/guide/index.html.

15. —. www.google.com/design. *www.google.com.* [Online] Google Inc., 17. 11 2014. [Citace: 3. 3 2015.] http://www.google.com/design/spec/what-is-material/environment.html#.

16. —. material-design.storage.googleapis.com/publish. *material-design.storage.googleapis.com.* [Online] Google Inc., 3. 3 2015. [Citace: 3. 3 2015.] http://material-design.storage.googleapis.com/publish/v_2/material_ext_publish/0Bx4BSt6jniD7bDZac2JGV2RUNk0/whatismaterial_properties_physical3.png.

17. —. material-design.storage.googleapis.com/publish. *material-design.storage.googleapis.com.* [Online] Google Inc., 3. 3 2015. [Citace: 3. 3 2015.] http://material-design.storage.googleapis.com/publish/v_2/material_ext_publish/0Bx4BSt6jniD7aVhXV0EtZ29OSU0/whatismaterial_properties_physical5.png.

8. Seznam obrázků

9. Slovník pojmů

- Operační systém
    - Základní programové vybavení zařízení pro obsluhu hardware a komunikaci s uživatelem
- Hardware zkr. HW
    - Fyzické vybavení zařízení
- Software zkr. SW
    - Programové vybavení
- Servlet
    - Server, který je obsluhován programovacím jazykem JAVA
- Pupup okno
    - Vyskakovací okno se zprávou, jejímž účelem je dotázat se uživatele, aby potvrdil své rozhodnutí
- Toast
    - Vyskakovací okno se zprávou informačního charakteru
- Widget
    - Ovládací prvek umístitelný na uživatelskou plochu operačního systému
- Komprese dat
    - Zmenšování objemu dat s nutností pozdější dekomprese
- Dekomprese dat
    - Uvedení dat, která prošla kompresní procedurou zpět do stavu před kompresí
- Cloud-to-device messaging
    - Služba společnosti Google Inc. zajišťující vynucení komunikace zařízení se serverem
- Disk Google (GoogleDrive)
    - Služba společnosti Google Inc. pro uchovávání uživatelských dat na datových serverech

10. Přílohy

10.1 Příloha 1 - Třída EditMethods – update_list

```
private Context context;

// konstruktor, který vyžaduje Context activity, která ho spouští
public EditMethods(Context context)
{
    // uloží Context do proměnné přístupné pouze touto třídou
    this.context = context;
}

// Změna záručního listu
// Parametr id
// Získání hodnot z Activity, který je identifikovaná díky Contextu
public boolean update_list(int id)
{
    // nová instance GetCheckMethods
    final GetCheckMethods check_methods = new GetCheckMethods(context);
    // získání uživatelem zadaných hodnot a uložení do objektu Warran-
  tyList
    WarrantyList list_update = check_methods.get_list();
    // přidání k hodnotám id z parametru metody
    list_update.Setid_list(id);
    // získání dat o expiracích a uložení do objektu Expiration
    Expiration expiration_update = check_methods.get_expiration();
    // přidání id z parametru funkce
    expiration_update.SetList_id(id);
    // nová instance sql metod pro tabulku Warranty_Lists
    WarrantyLists_SQL sql = new WarrantyLists_SQL(context);
    // změna hodnot v databázi pro daný záruční list
    sql.updateWarrantyList(list_update);
    // nová instance sql metod pro tabulku Expirations
    Expirations_SQL exp_sql = new Expirations_SQL(context);
    // změna hodnot v databázi pro daný záruční list
    exp_sql.updateExpiration(expiration_update);
    Intent intent = new Intent(context, InfoList.class);
    intent.putExtra("id", id);
    // spuštění Acitivity InfoList s id upraveného záručního listu
    context.startActivity(intent);
    return true;
}
```

10.2 Příloha 2 - Třída EditMethods – add_list

```
    public long add_list()
    {
        // Nová instance GetCheckMethods
        final GetCheckMethods check_methods = new GetCheckMethods(context);
        // získání uživatelem zadaných hodnot a uložení do objektu
WarrantyList
        WarrantyList list_update = check_methods.get_list();
        // nová instance sql metod pro tabulku Warranty_Lists
        WarrantyLists_SQL sql = new WarrantyLists_SQL(context);
        // zapíše nový záruční list do databáze a vrátí jeho id
        long id = sql.addWarrantyList(list_add);
        // pridá k objektu expirace id nového záručního listu
        expiration_add.SetList_id(id);

        // nová instance sql metod pro tabluku Expirations
        Expirations_SQL exp_sql = new Expirations_SQL(context);
        // zapíše novou expiraci
        exp_sql.addExpiration(expiration_add);
        // vypíse do debug logu zprávu o novém záručním listu
        Log.d("Add was completed", list_add.toString());
        return id;
    }
```

10.3 Příloha 3 – Třída NavigationHandleMethods – práce s řazením

```
    public void popup_order()
    {
        // identifikace tlačítka
        final ImageButton button1 = (ImageButton) ((Activity)
context).findViewById(R.id.Sort);
        // piřazení eventu (po stisku) -> Listener k danému tlačítku
        button1.setOnClickListener(new OnClickListener()
        {

            @Override
            public void onClick(View v)
            {
                // nová instance objektu PopupMenu (nativní lib v Android)
s parametry kontextu Activity a tlačítka
                PopupMenu popup = new PopupMenu(context, button1);
                // inflate grafického meny menu_sort a získání přístupu k
němu
                popup.getMenuInflater().inflate(R.menu.menu_sort,
popup.getMenu());
                // piřazení eventu (po stisku) -> Listener ke každé položce
menu
                popup.setOnMenuItemClickListener(new
PopupMenu.OnMenuItemClickListener()
                {
                    // metoda po stisknutí jedná z položek menu
                    @Override
                    public boolean onMenuItemClick(MenuItem item)
                    {
                        Intent intent;
                        // Po identifikaci dané akce se změní hodnoty v
proměných a znovu se vypíše seznam záručních listů
                        switch (item.getItemId())
                        {
                            // výběr pro základní řazení
                            case R.id.Menu_default_sort:
                                ListsList.sort_type = 0;
                                ListsList.asc_desc = 0;
                                ListsList.showall();
                                break;
                            // výběr pro řazení dle jména
                            case R.id.Menu_by_name:
                                ListsList.sort_type = 1;
                                ListsList.showall();
                                break;
                            // vzestupně
                            case R.id.Menu_asc:
                                ListsList.asc_desc = 0;
                                ListsList.showall();
                                break;
                            // sestupně
                            case R.id.Menu_desc:
                                ListsList.asc_desc = 1;
                                ListsList.showall();
                                break;
                        }
                        return true;
                    }
                });
                popup.show();
            }
        })
```