VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

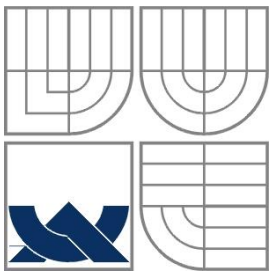
NÁKUPNÍ RÁDCE PRO ANDROID

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

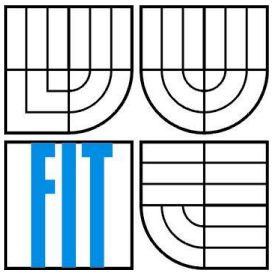
AUTOR PRÁCE
AUTHOR

VLADIMÍR ELIÁŠ

BRNO 2015



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ
FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

NÁKUPNÍ RÁDCE PRO ANDROID

SHOPPING ADVISOR FOR ANDROID

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE
AUTHOR
VLADIMÍR ELIÁŠ

VEDOUCÍ PRÁCE
SUPERVISOR
Ing. LUKÁŠ MARŠÍK

BRNO 2015

Abstrakt

Tato práce se zabývá studií, návrhem a implementací mobilní aplikace pro inteligentní telefony s operačním systémem Android na skenování čárových kódů a následnou komunikací s databází. Také popisuje postup při vytváření moderního grafického uživatelského rozhraní i s názornými ukázkami. Účelem aplikace je urychlit proces získávání informací o produktech nabízených v obchodech a zároveň pomoci s výběrem vhodného produktu na základě uživatelových vstupů a pokročilých algoritmů.

Abstract

This bachelor thesis deals with study, design and implementation of mobile application for smartphones with Android operating system. The application scans barcodes and then communicates with the database. This thesis also describes how to create a modern graphical user interface with illustrative examples. The main purpose of application is to speed up the process of obtaining information about the products offered in shops and at the same time help choosing the right product based on the user's input and advanced algorithms.

Klíčová slova

Android, čárové kódy, mobilní aplikace, databáze, nákupní rádce

Keywords

Android, bar codes, mobile application, database, shopping advisor

Citace

Vladimír Eliáš: Nákupní rádce pro Android, bakalářská práce, Brno, FIT VUT v Brně, 2015

Nákupní rádce pro Android

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pána Ing. Lukáše Maršíka.
Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

……
Vladimír Eliáš
20. května 2015

Poděkování

Rád bych poděkoval vedoucímu mé bakalářské práce panu Ing. Lukášovi Maršíkovi za množství užitečných rad a také Rudolfovi Halmimu, který mi dal cenné rady na začátcích implementace.

Obsah

1 Úvod

Podľa webovej stránky emarketer.com malo v roku 2014 smartfón[1] až 1.75 miliardy obyvateľov Zeme. Z tejto štatistiky vyplýva, že každý štvrtý obyvateľ na svete vlastní smartfón a až 70% z nich funguje na platforme Android. Na základe týchto trendov bolo rozhodnuté vyvinúť novú aplikáciu pre zariadenia so systémom Android. Cieľom bolo nájsť takú, ktorá na trhu ešte chýba a má potenciál na úspech u širokej verejnosti. Po dlhšom uvážení sme sa rozhodli pre aplikáciu napomáhajúcu pri každodennej činnosti – nákupoch.

V dnešnom svete plnom rôznych supermarketov so širokým sortimentom produktov, ktoré ľudí lákajú veľkými akciami, pestrými obalmi či rôznymi reklamami, je ťažké vybrať si práve taký produkt, po akom túžime. Predstavme si športovca, ktorý túži po výrobkoch len s vysokým obsahom bielkovín, človeka trpiaceho obezitou hľadajúceho potraviny s nízkym obsahom tukov, alergika vyhýbajúceho sa určitým alergénom, osobu nútenú dodržiavať špeciálne diéty či obyčajného zvedavca, ktorý by rád vedel, čo obsahuje produkt, ktorý kupuje. Týmto všetkým ľudom dokážeme pomôcť vďaka mobilnej aplikácií Nákupný radca.

Aplikácia pomocou fotoaparátu nasníma čiarový kód produktu a vďaka unikátnemu identifikačnému číslu nájde na serveri v databáze odpovedajúce informácie o snímanom produkte. Tieto informácie sa potom zobrazia používateľovi vo forme grafov, obrázkov a textu. Používateľ tak získa základné informácie o tovare akými sú napríklad zloženie, nutričné hodnoty, obsah alergénov, či zaradenie tovaru do kategórie priradením špecifického symbolu(tučný, s nízkym obsahom tuku, na rast svalovej hmoty). Okrem iného tu bude možnosť spustenia módu porovnávania produktov, kedy budeme môcť naskenovať viacero produktov naraz a aplikácia na základe osobných kritérií (hodnotenie produktu/obsah tukov, cukrov, bielkovín) nám sama vyberie ten najvhodnejší produkt.

Na začiatku tejto práce sa dočítame o základných faktoch operačného systému Android (kapitola 2), bližšie si predstavíme technológiu čiarových kódov a nazrieme na spôsoby komunikácie systému Android s databázou umiestnenou na vzdialenom serveri (kapitola 3). Neskôr sa povenujeme návrhu aplikácie (kapitola 4) a popíšeme si aj jej samotnú implementáciu (kapitola 5). Na záver sa dozvieme niečo o testovaní výslednej aplikácie (kapitola 6).

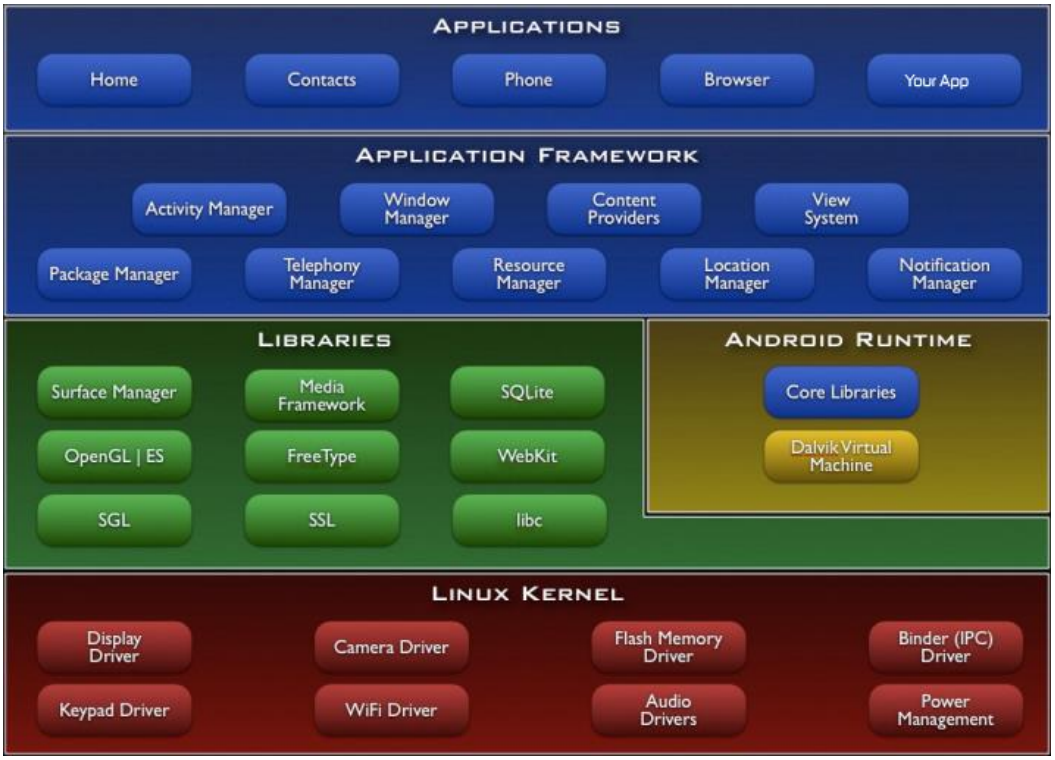
[1] Z angl. smartphone, pojem označujúci inteligentný mobilný telefón

2 Android

Spoločnosť Android Inc. bola založená v Kalifornií v Októbri 2003 Andy Rubinom, Richom Minerom, Nickom Searsom a Chrisom Whiteom. Cieľom bolo vytvoriť inteligentné mobilné zariadenie, ktoré by nahradilo doterajšie mobilné telefóny. V roku 2005 nastal zlomový okamih, keď technologický gigant Google odkúpil firmu Android. Do vývoja operačného systému zasiahla aj OHA (Open Handset Aliance), čo je konzorcium 84 firiem, medzi ktoré patria firmy napríklad ako Google, HTC, Sony, Dell, Intel, Motorola, Qualcomm, Texas Instruments, Samsung Electronics, LG Electronics či T-Mobile. Táto spoločnosť si dala za cieľ vytvoriť otvorené štandardy pre mobilné zariadenia, čo potvrdila predstavením svojho vlajkového produktu – operačný systém Android.[1]

2.1 Architektúra

Operačný systém Android je založený na jadre Linuxu. Celý systém je rozdelený do piatich vrstiev ako je možno vidieť na obrázku 2.1, pričom každá má svoju špecifickú úlohu.



Obrázok 2.1: Rozdelenie OS do piatich vrstiev [2]

Linuxové jadro

Najnižšiu vrstvu tvorí jadro operačného systému Linux 2.6 s rôznymi redukciami funkcií a ich prispôsobeniam na mobilné zariadenia. Vývojári bežných aplikácií s jadrom do priameho kontaktu

neprídu vďaka ADB (Android Debug Bridge), ktorý poskytuje prístup k príkazovému rozhraniu Linux Shell a jeho prostredníctvom tak môžeme zadávať jadru operačného systému príkazy. Táto vrstva zabezpečuje komunikáciu s hardwarom a obsahuje základné hardwarerové ovládače. Tie sa starajú napríklad o správu procesov, pamäte či periférnych zariadení ako fotoaparát, klávesnica alebo displej. Aplikácie a služby sú spúšťané v oddelených procesoch a dosť často potrebujú vzájomne komunikovať. Preto je na platforme Android implementovaný medziprocesorový ovládač komunikácie a volania metód s názvom Binder, ktorý umožňuje viacerým procesom zdieľať údaje. Súčasťou jadra je aj správa napájania, ktorá zabezpečuje, aby energeticky najnáročnejšie moduly (procesor a obrazovka), boli pri dlhšej nečinnosti vypínané. To, že Android je postavený na populárnom a osvedčenom jadre, z neho robí ľahko prenášateľný systém medzi veľkým množstvom zariadení.[3]

Natívne knižnice

Ďalšiu vrstvu tvorí základný balík knižníc napísaných v C/C++ kóde, ktoré poskytujú aplikáciám prístup k rôznym komponentom systému Android. Tvoria akúsi medzivrstvu medzi rôznymi komponentami vyšších vrstiev a linuxovým jadrom. Patria medzi ne napríklad:

- WebKit (slúži na renderovanie a zobrazovanie webových stránok)
- libc (odvodená BSD knižnica)
- SQLite (knižnica pre prístup k relačným databázam)
- SSL (knižnica zodpovedná za sieťovú bezpečnosť)
- OpenGL (knižnica na vykresľovanie 2D a 3D grafiky)

Android Runtime

Táto vrstva zahŕňa najmä kľúčový komponent zvaný Dalvik Virtual Machine (ďalej len DVM), čo je vlastne analógia Java Virtual Machine (ďalej len JVM) na klasických počítačoch prispôsobená špeciálne pre operačný systém Android. DVM bol vyvinutý Danom Bornsteinom ako reakcia na licenčné podmienky JVM a jej vysokú energetickú a pamäťovú náročnosť. Každá androidová aplikácia je samostatný proces využívajúci vlastnú inštanciu virtuálneho stroja Dalvik. Ten zabezpečuje beh spustiteľných súborov s príponou „dex“. Súbory dex vznikli kompiláciou z klasických súborov „class“ a „jar“. DVM umožňuje simultánny beh viacerých inštancií virtuálnych strojov poskytujúcich bezpečnosť, izoláciu a správu pamäte.

Aplikačný framework

Najdôležitejšia vrstva pre vývojárov. Poskytuje prístup k veľkému počtu služieb, ktoré môžu byť použité priamo v aplikáciách. Tieto služby môžu sprístupňovať dáta v iných aplikáciách, prvky používateľského rozhrania, upozorňovací stavový riadok, aplikácie bežiace na pozadí, hardvér používaného zariadenia a mnoho ďalších služieb a funkcií.

- Package Manager – tento modul je v podstate databázou, ktorá obsahuje zoznam všetkých aplikácií nainštalovaných na zariadení. Jeho vizuálnym obrazom je domovská obrazovka zariadenia, pričom každá ikona reprezentuje balíček aplikácie.
- Window Manager – spravuje všetky okná, ktoré tvoria aplikácie.
- Activity Manager – spravuje životný cyklus všetkých aplikácií.
- Content Providers – spravuje zdieľané dáta medzi aplikáciami.
- Telephony Manager – túto službu využívame ak chcem použiť hlasový hovor v našej aplikácii.
- Notification Manager – umožňuje všetkým aplikáciám zobrazovať vlastné upozornenia.

Aplikácie

Aplikácie tvoria najvyššiu vrstvu Android architektúry. Sem patria všetky predinštalované mobilné aplikácie ako SMS klient, webový prehliadač, správca kontaktov, kalendár, ale aj všetky dodatočne stiahnuté aplikácie.

2.2 Verzie

Operačný systém Android si od svojho vzniku prešiel viacerými obmenami. Prvá verzia z roku 2009 – Android 1.5 Cupcake odštartovala v spoločnosti Google zvyk, pomenovávať všetky ďalšie verzie po sladkostiach. Nižšie sú chronologicky zoradené zvyšné verzie operačného systému Android:

ROK	VERZIA	NÁZOV
2009	1.6	Donut
2009	2.0-2.1	Eclair
2010	2.2	Froyo
2010	2.3	Gingerbread
2011	3.X	Honeycomb
2011	4.0	Ice Cream Sandwich
2012	4.1-4.3	Jelly Bean
2013	4.4	KitKat
2014	5.0-5.1	Lollipop

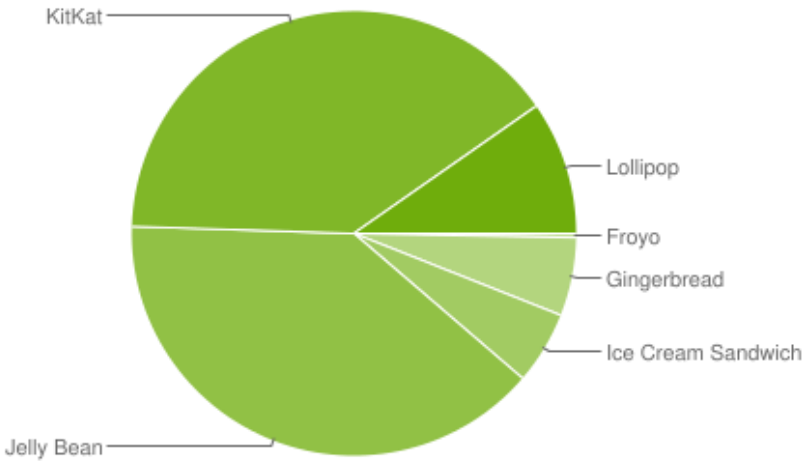
Tabuľka 2.1: Zoznam verzií Androidu s ich rokom vydania [4]

Každá nová verzia prináša do systému mnoho nových vylepšení a možností pre vývojárov. Tie ale nie sú spätne kompatibilné so staršími verziami, preto je potrebné neustále sledovať momentálny podiel distribúcií na trhu, aby sme pri návrhu a implementácií aplikácie mohli pokryť čo najväčšie množstvo

zariadení podporujúcich našu aplikáciu. Za týmto účelom spoločnosť Google pravidelne vydáva štatistiky používania jednotlivých verzií systému na zariadeniach. Vďaka nim sa môžu vývojári ľahšie rozhodnúť, ktoré verzie ich navrhovaná aplikácia ešte bude podporovať a ktoré už nie. Z nasledujúceho grafu s tabuľkou 2.3.2 vieme vyčítať, že prevažnú väčšinu zariadení tvoria zariadenia so systémom Jelly Bean a KitKat. Nový systém Lollipop zatiaľ v štatistikách zaostáva, no to je spôsobené aj tým, že ešte stále mnoho výrobcov ponúka svoje nové zariadenia so starším OS KitKat.

Verzia	Kódové označenie	API	Distribúcia
2.2	Froyo	8	0.3%
2.3.3 – 2.3.7	Gingerbread	10	5.7%
4.0.3 – 4.0.4	Ice Cream Sandwich	15	5.3%
4.1.x	Jelly Bean	16	15.6%
4.2.x		17	18.1%
4.3		18	5.5%
4.4	KitKat	19	39.8%
5.0	Lollipop	21	9.0%
5.1		22	0.7%

Tabuľka 2.2: Zastúpenie jednotlivých verzií Androidu k dátumu 4.Mája 2015 [5]



Obrázok 2.2: Grafické znázornenie jednotlivých Android verzií [5]

2.3 Životný cyklus aplikácie

Každá aplikácia sa skladá z aplikačných komponent nazývaných Aktivity. Každá Aktivita má svoje okno s používateľským rozhraním a špecifickou úlohou (poslanie e-mailovej správy, prezeranie mapy, vytáčanie čísla atď.). Všetky Aktivity sú navzájom prepojené a podľa situácie medzi sebou komunikujú. Typicky existuje jedna hlavná Aktivita nazývaná aj ako „MainActivity“, ktorá sa spúšťa automaticky pri prvom spustení aplikácie. Vždy, keď sa nová Aktivita spustí, stará sa pozastaví a čaká, kým ju používateľ nebude opäť potrebovať. Aktivita sa teda môže dostať do troch režimov:

1. Resumed/Running – Aktivita je viditeľná, je v popredí obrazovky a používateľ s ňou môže pracovať.
2. Paused – Aktivita je prekrytá čiastočne novou aktivitou, typicky nejakým dialógovým oknom. Znamená to, že pozastavená aktivita je stále viditeľná, ale dostáva sa do pozadia na úkor inej aktivity. Všetky dáta sa uchovávajú pre opätovné použitie. V kritických situáciách môže systém ukončiť pozastavenú aktivitu.
3. Stopped – Aktivita je kompletne prekrytá inou aktivitou. Napriek tomu sa všetky dáta stále uchovávajú pre neskoršie použitie. V tomto stave môže systém pri nedostatku pamäti hocikedy takúto aktivitu ukončiť.

To, čo sa vykoná vždy pri zmene režimu, definujú základné metódy, ktoré by mala obsahovať každá aktivita.

`onCreate()` – Metóda, ktorá sa vykoná pri prvom štarte novej aktivity. Tu sa vytvárajú nové pohľady, inicializujú premenné atď.

`onStart()` – Volaná po vykonaní metódy onCreate() alebo onStop(), kedy aktivita prechádza z režimu Stopped opäť do režimu Resumed.

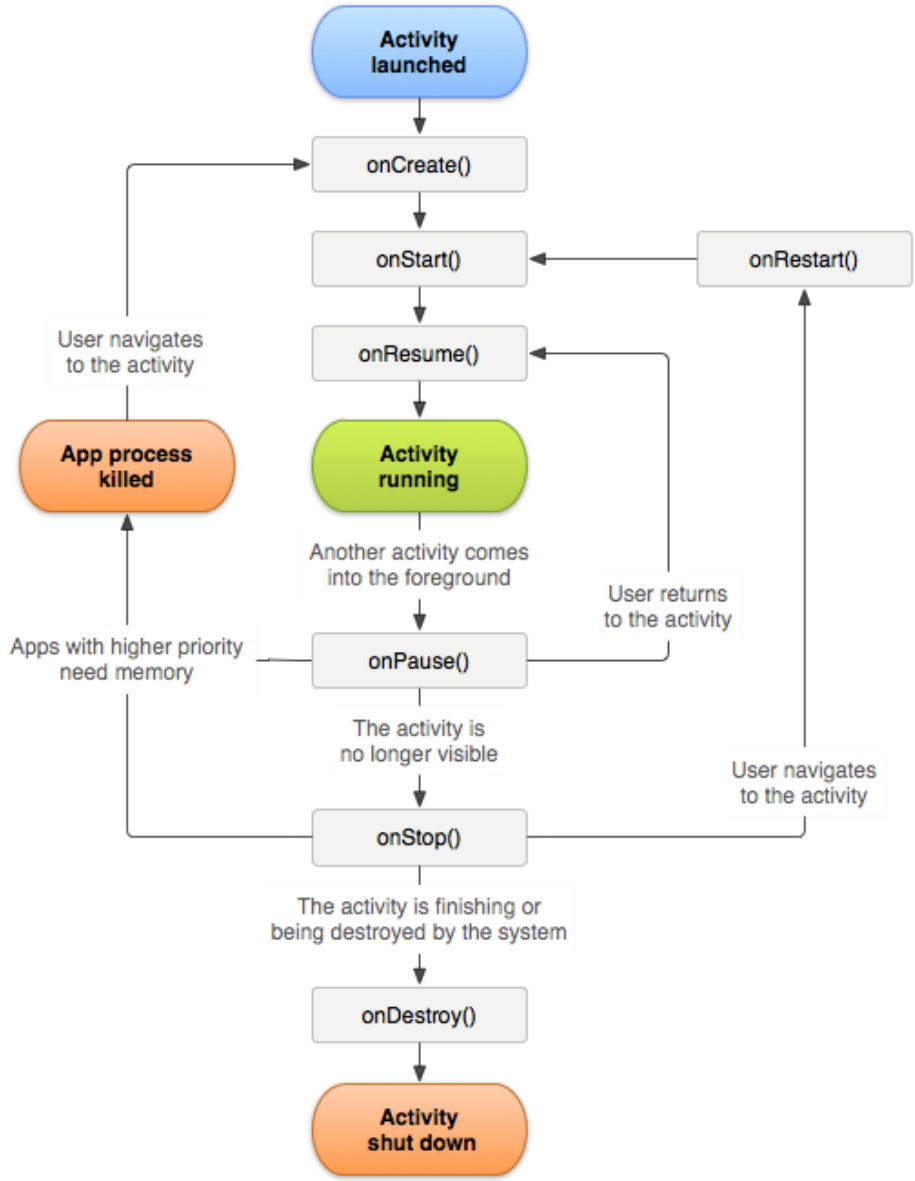
`onResume()` – Aktivita sa po vykonaní tejto metódy stáva aktívnou, čiže sa dostáva do popredia.

`onPause()` – Volá sa ak systém obnovuje inú aktivitu. Aktuálna aktivita sa dostáva do režimu Paused. V tejto metóde sa odporúča uložiť všetky potrebné dáta, ukončiť animácie a iné procesy, ktoré by mohli zaťažovať procesor. Operácie realizované v metóde musia byť časovo nenáročné.

`onStop()` – V prípade, že aktivita už nie je viac viditeľná, vykoná sa táto metóda.

`onDestroy()` – Po spustení tejto metódy daná aktivita zanikne.

Pre lepšiu predstavivosť si volanie týchto metód môžeme pozrieť na obrázku 2.3. Obrázok znázorňuje vytvorenie, spustenie a zrušenie aktivity. Taktiež tu môžeme vidieť situáciu, kedy aktivita musí byť operačným systémom ukončená pre nedostatok voľnej pamäti.



Obrázok 2.3: Grafické znázornenie životného cyklu aplikácie [6]

2.4 Vývojové prostredia

Vývojové prostredie (angl. IDE – Integrated developement enviroment) je dnes veľmi dôležitým pomocníkom pre programátorov pri vývoji každého projektu, preto bolo v našom prípade veľmi dôležité, zvoliť práve ten najvhodnejší. Na vývoj aplikácií pre systém Android existuje viacero prostredí, no najpoužívanejšími sú nasledovné tri:

IntelliJ IDEA

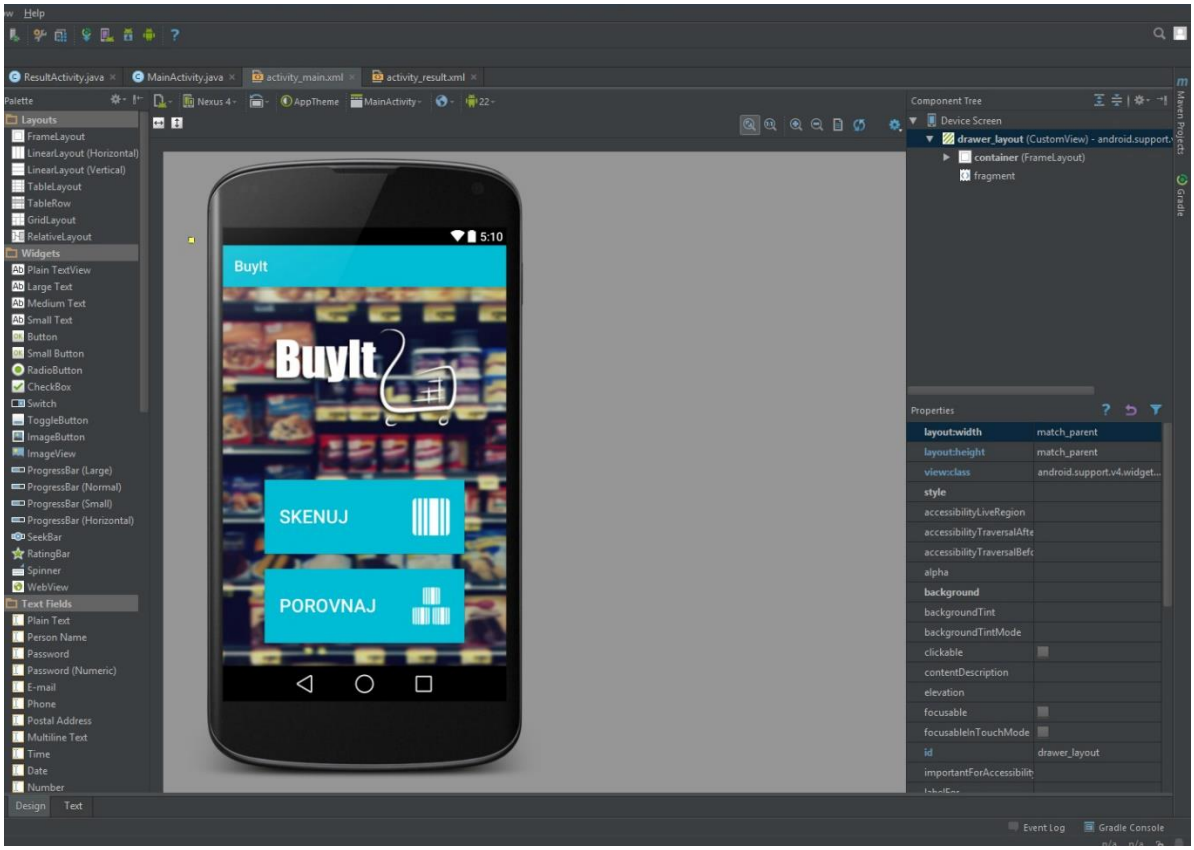
Vývojové prostredie vyvíjané firmou JetBrains s.r.o.. Program podporuje veľké množstvo programovacích jazykov ako napríklad Java, JavaScript, PHP, Python či Ruby. Samozrejmosťou je podpora tvorby aplikácií pre systém Android.

Eclipse

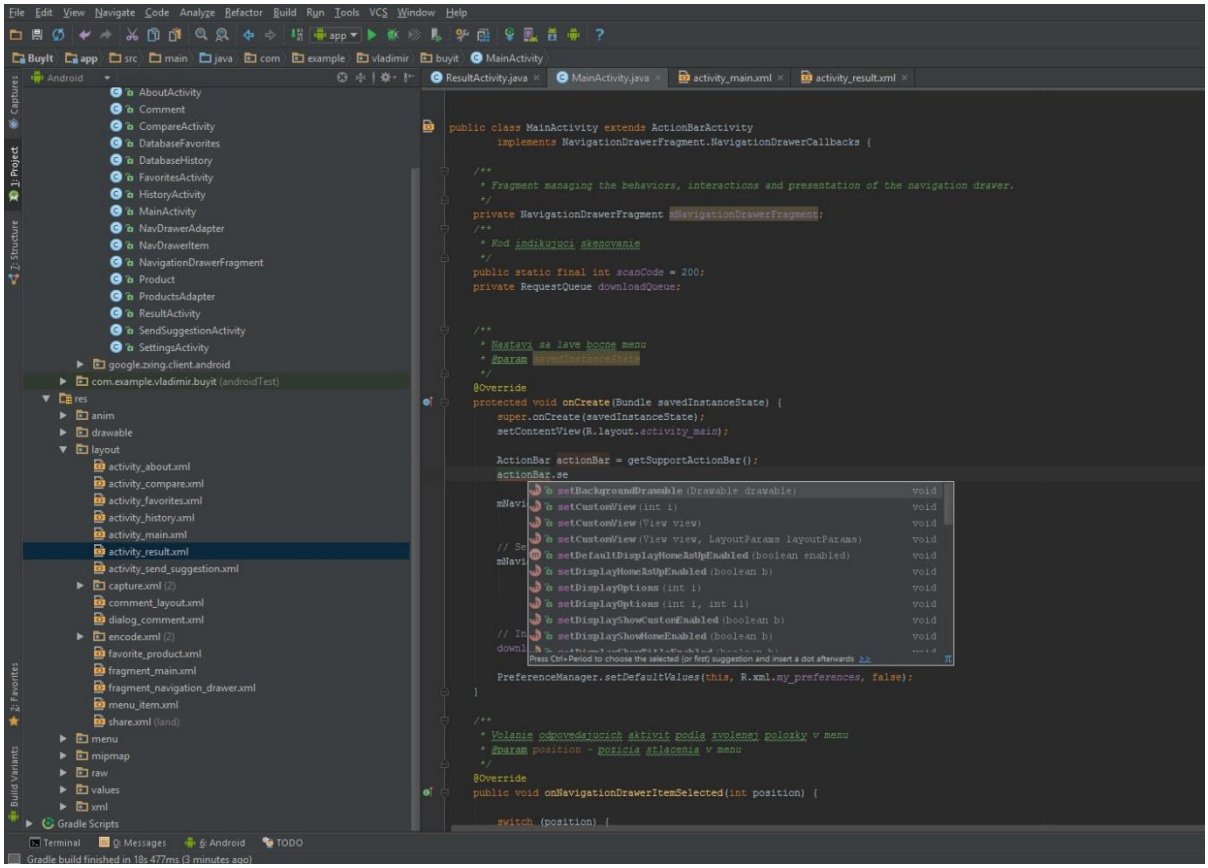
Stále veľmi populárne vývojové prostredie pre všetkých Android vývojárov. Samotný program neposkytuje podporu na vývoj Android aplikácií, no existuje oficiálny plugin, vydaný spoločnosťou Google – Android Development Tools (ADT), ktorý tento problém vyrieši. Po nainštalovaní pluginu sa stáva z programu Eclipse plnohodnotné vývojové prostredie pre systém Android.

Android Studio

Oficiálne vývojové prostredie vyvíjané spoločnosťou Google, založené na IntelliJ IDEA. Momentálne ešte stále nie je natoľko rozšírené, keďže prvá stabilná verzia tohto programu sa objavila až v decembri 2014. Program obsahuje okrem iného vstavaný emulátor operačného systému Android, intuitívneho dizajnéra na tvorbu grafického používateľského rozhrania či prepracovaný systém doplňovania kódu.



Obrázok 2.4: Ukážka dizajnéra v Android Studiu



Obrázok 2.5: Ukážka doplňovania kódu v Android Studiu

2.5 Material dizajn

Tento pojem, označujúci základné pravidlá a návyky pri tvorbe používateľského prostredia, prišiel spolu s najnovšou verziou systému Android 5.0 Lollipop. V podstate ide o nový vizuálny jazyk, ktorého cieľom je spojiť dobrý dizajn s novými inovatívnymi a technologickými možnosťami. Princíp spočíva v jednoduchosti, používaní základných geometrických útvarov a súbor plynulých preddefinovaných animácií. Slovom „Material“ bolo cielené označiť istú hmotu/objekt, ktorý sa nachádza v 3D priestore a teda má x, y a z-tkové súradnice, pričom x a y-nové súradnice sú variabilné ale z-tková má vždy hodnotu 1dp[2]. Ďalším dôležitým prvkom je použitie tieňov. Tiene sa môžu zväčšovať či zmenšovať na základe používateľovej interakcie s materiálom.

2.5.1 Ukážky základných prvkov

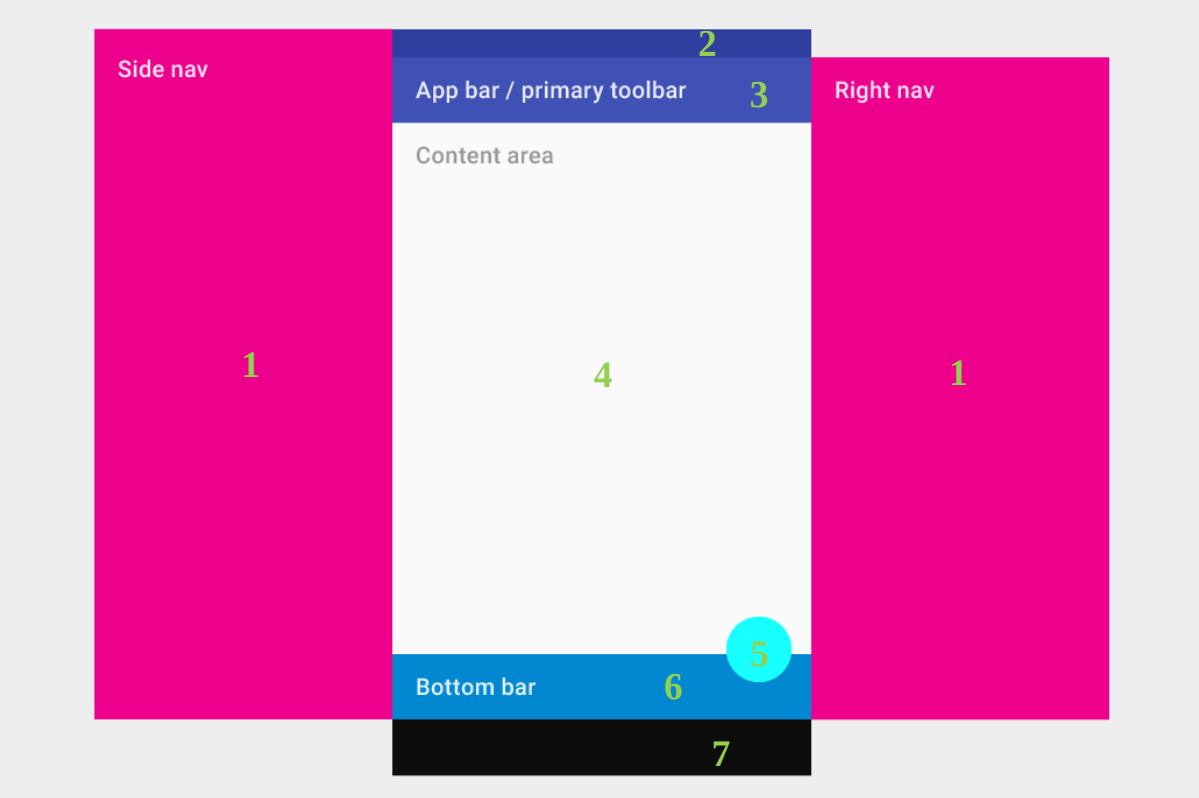
Oficiálny web pre Android vývojárov poskytuje základné ukážky rozloženia aplikácie, tlačidiel a množstvo ďalších užitočných rád ako navrhovať aplikácie v Material dizajne.

[2] Density independent pixel – virtuálna jednotka pixelu, ktorá nie je závislá na rozlíšení obrazovky

Rozloženie prvkov

Na obrázku 2.6 vidíme ukážku základného rozloženia prvkov typického pre Material dizajn.

Pravý a ľavý navigačný panel (na obrázku vyznačený číslom 1) je pri spustení aplikácie prekrytý hlavnou obrazovkou (v strede) a aktivuje sa buď gestom potiahnutia do strany alebo stlačením tlačidla. Formou animácie sa plynule vysúva zo strany, a vstupuje smerom do obrazovky, čím prekrýva hlavné okno aplikácie. Vzhľadom na to, že sa jedná len o demonštračnú ukážku, je to na obrázku znázornené naopak, teda bočné panely vyskakujú smerom von z hlavného okna.



Obrázok 2.6: Ukážka rozloženia prvkov pri Material dizajne

Hlavné okno je tvorené z viacerých prvkov, ktoré sme vložili do očíslovaného zoznamu. Takto očíslované prvky potom symbolizujú ich umiestnenie na obrázku.

1. **Side & right nav**

   Obsah ľavého navigačného panelu má mať charakter akéhosi menu celej aplikácie, zatiaľ čo pravý len doplňuje obsah momentálne otvoreného okna.

2. **Status bar**

   Táto lišta obsahuje systémové ikony a notifikácie. Jej výška je nemenná a to 24dp.

3. **App bar/action bar**

   Nadobúda spravidla vždy svetlejší farebný odtieň ako status bar. Slúži na umiestnenie rôznych, často používaných tlačidiel, ktoré zväčša vykonávajú funkcie ako: vyhľadávanie, obnovenie, zobrazenie pravého či ľavého navigačného panelu atď...

4. **Content area**

   Miesto vyhradené pre obsah aplikácie.

5. **Floating aciton button**

   Tlačidlo prezentujúce primárne určenie aplikácie. Používa sa iba na tvorbu pozitívnych akcií, napríklad vytvoriť, pridať, zdieľať, navigovať ale nikdy akcií deštruktívnych ako odstrániť, blokovať či iných nešpecifických akcií.

6. **Bottom bar**

   Môže obsahovať často používané akcie, či akcie, ktoré chceme dať používateľovi do pozornosti.

7. **Android navigation bar**

   Lišta obsahujúca ovládacie prvky zariadenia (nie je prítomná pri všetkých zariadeniach, keďže niektoré disponujú fyzickými tlačidlami pre ovládacie prvky).

Samozrejme, vývojári sa nemusia striktne držať tohto rozloženia, slúži skôr na inšpiráciu, prípadne na zjednotenie určitých prvkov dizajnu. Ten by mal byť prispôsobený hlavne účelu danej aplikácie a zahŕňať čo najmenšie množstvo vyššie popísaných prvkov, aby sa dosiahlo jednoduchosti, prehľadnosti a úspory miesta pre samotný obsah aplikácie.

Štýl písma

Na zobrazenie textu v aplikácií sa odporúča používať štýl Roboto pre všetky jazyky používajúce zápis: latinský, grécky a cyriliku. Pre ostatné jazyky bol vytvorený štýl Noto.

Roboto:

Roboto, ***Roboto Italic,*** **Roboto Bold**, ***Roboto Bold and Italic***

Roboto Condensed:

Roboto Condensed, *Roboto Condensed Italic*, **Roboto Condensed Bold**, ***Roboto Condensed Bold and Italic***

3 Technológie

Vďaka napredovaniu inteligentných telefónov a vývojových nástrojov sa otvárajú nové možnosti pre vývojárov mobilných aplikácií. Prostredníctvom dostupných technológií v najnovších smartfónoch ako sú fotoaparát, mikrofón, mobilné internetové pripojenie, Bluetooth či NFC je možné získať čoraz viac dát rýchlejšími a praktickejšími metódami. Pre naše účely si popíšeme dva externé zdroje dát, ktorých dáta získavame použitím fotoaparátu a mobilného internetového pripojenia.

3.1 Čiarové kódy

Čiarové kódy spadajú do oblasti tzv. „automatickej identifikácie“ tak isto ako magnetické kódy používané napríklad na kreditných kartách alebo strojovo čitateľné písmo OCR. Avšak čiarové kódy sú z tejto oblasti najpoužívanejšie. [8]

Výhody použitia čiarových kódov

V dnešnej dobe sa môžeme stretnúť so spôsobom identifikácie pomocou čiarových kódov skoro v každom odvetví priemyslu či služieb. Je tomu tak hlavne kvôli nasledovným vlastnostiam čiarových kódov:

- Presnosť – odhaduje sa, že pri ručnom zadávaní dát dochádza k chybe v priemere pri každom tristom zadaní zatiaľ čo pri použití čiarových kódov sa tento počet znižuje na jednu miliótinu
- Rýchlosť – snímanie kódov je minimálne trikrát rýchlejšie ako ručné zadávanie dát
- Flexibilita – vďaka moderným skenerom môžu byť čiarové kódy malé a použiteľné za každých podmienok
- Nízka cena – finančné nároky na zavedenie čiarových kódov sú zanedbateľné v porovnaní s ich prínosom v efektivite získavania dát vo väčších podnikoch

Princíp získavania dát z čiarových kódov

Každý kód sa skladá z tmavých čiar a zo svetlých medzier, ktoré sa čítajú pomocou snímačov vyžarujúcich väčšinou červené svetlo. Toto svetlo je pohlcované tmavými čiarami a odrážané svetlými medzerami. Snímač zisťuje rozdiely v reflexii a tie premieňa v elektrické signály zodpovedajúce šírke čiar a medzier. Tieto signály sú prevedené do čísel, resp. písmen, ktoré príslušný čiarový kód obsahuje. Znamená to teda, že každá číslica alebo písmeno je zaznamenávané v čiarovom kóde pomocou vopred presne definovaných šírok čiar a medzier.

Pri smartfónoch sa tento proces skenovania čiarových kódov líši. Najprv sa za použitia kamery v mobilnom telefóne vytvorí snímka čiarového kódu. Tá sa následne analyzuje tak, že sa fotografia oreže len na časť obsahujúcu čiarový kód, potom sa celý obrázok prevedie do čierno-bieleho obrázku z ktorého algoritmus získava postupnosť číslic nadobudnutých z bielych a čiernych čiar.

Dáta obsiahnuté v čiarovom kóde môžu zahŕňať čokoľvek: číslo výrobcu, číslo výrobku, miesto uloženia v sklade, číslo série alebo dokonca meno určitej osoby, ktorej je napríklad povolený vstup do inak uzavretého priestoru atď.

Na skenovanie čiarových kódov môžeme dnes využiť rôzne zariadenia fungujúce na odlišných princípoch. My si popíšeme asi najpoužívanejšie štyri typy skenerov:[9]

- Skenovacie pero – využíva zdroj svetla a fotodiódu, na skenovanie je nutné prejsť vrcholom pera plynulo po celom čiarovom kóde.
- Laserový skener – funguje na rovnakom princípe ako skenovacie pero až na to, že ako zdroj svetla používa laserový lúč. Ten je vďaka zrkadlu v zariadení nalomený tak, že pokryje celú plochu čiarového kódu.
- CCD skener – využíva stovky malých svetelných senzorov usporiadaných do radu, ktoré svietia na čiarový kód a merajú napätie okolitého svetla.
- 2D kamera – ako jediný dokáže snímať 2D kódy. Funguje na podobnom princípe ako CCD skener s tým rozdielom, že 2D kamera obsahuje stovky riadkov plných senzorov, ktoré sú usporiadané do dvojdimenzionálneho poľa.

3.1.1 Typy čiarových kódov

Existuje viacero typov čiarových kódov. Niektoré sú schopné zakódovať iba číslice, iné v sebe dokážu ukrývať aj písmená či špeciálne znaky. Každý typ kódu má svoj vlastný spôsob kódovania jednotlivých znakov do sústav čiar a medzier. Z dôvodu veľkej početnosti typov uvedieme len tie najznámejšie:[8]

EAN

Obrázok 3.1: Kód EAN

Najznámejšia forma čiarového kódu, s ktorou sa môžeme stretnúť aj pri označovaní produktov v obchodných reťazcoch. Tento kód je prideľovaný každému produktu medzinárodnou organizáciou GS1 so sídlom v Bruseli. Čiarový kód je kódovaný číslicami od 0 do 9 pričom každá číslica je kódovaná dvoma čiarami a dvoma medzerami. Typ EAN sa ďalej rozdeľuje na EAN-8 alebo EAN-13 (číslo za pomlčkou udáva počet číslic v kóde). Prvé dve alebo tri číslice identifikujú štát pôvodu (napr. Slovensko má 858), ďalšie číslice (väčšinou 4 až 6) určujú výrobcu

a zvyšné číslice okrem poslednej určujú konkrétny tovar. Posledná číslica slúži na overenie správnosti dekódovania.

Kód 128

Obrázok 3.2: Kód 128

Kód patriaci taktiež do systému EAN, odlišuje sa ale tým, že dokáže zakódovať okrem číslic aj znaky. Dokopy tak dokáže zakódovať až 102 znakov, pričom každý znak je reprezentovaný troma čiarami a troma medzerami. Týmto typom kódovania kódujeme informácie o danom výrobku ako je napríklad číslo dodávky, dátum výroby, dátum balenia, minimálna trvanlivosť, hmotnosť, dĺžka, objem, plocha atď.

Kód ITF

Obrázok 3.3: Kód ITF

Kód dovoľujúci vysokú hustotu zápisu (až 8 znakov na 1cm). Používa sa v rôznych odvetviach priemyslu. Dokáže kódovať číslice 0 až 9 pričom každá číslica je kódovaná buď piatimi linkami alebo piatimi medzerami. Jednotlivé znaky sa kódujú v pároch, z tohto dôvodu musí kód ITF vždy obsahovať párny počet znakov.

3.2 Databázy

Databázy slúžia na úschovu veľkého množstva dát typicky na serveri. Vďaka dopytovacím jazykom (napr. SQL) je možné tieto informácie triediť a získať tak len špecifické, používateľom vyžiadané informácie z databázy. Databázu si môžeme predstaviť ako množinu tabuliek s viacerými atribútmi, ktoré medzi sebou môžu nadobúdať určité vzťahy. Všetky dáta databázy sú perzistentné, tzn. dáta uchovávajúce sa aj po ukončení behu aplikačného programu či vypnutia počítača.

MySQL

Najznámejší a používateľmi najpoužívateľnejší voľne dostupný SQL relačný databázový server. Vytvorený bol vo Švédsku dvoma Švédmi a jedným Fínom: Davidom Axmarkom, Allanom Larssonom a Michaelom Wideniusom. Je kľúčovým prvkom stále viac populárnej sady nástrojov LAMP (Linux, Apache, MySQL, PHP/Perl/Python). Každá databáza v MySQL je tvorená z jednej alebo viacero tabuliek obsahujúcich riadky a stĺpce. V riadkoch je možné dohľadať jednotlivé záznamy, zatiaľ čo stĺpce udávajú ich dátový typ. Na komunikáciu s databázou sa používa dopytovací jazyk SQL. Medzi najznámejšie firmy používajúce MySQL patria napríklad Yahoo!, Google, Nokia, Youtube či Wikipedia.[10]

3.2.1 Posielanie dát

Počas komunikácie klienta s databázou sa zväčša prenáša veľké množstvo dát, ktoré je potrebné logicky členiť. Na to nám slúžia rôzne spôsoby zápisu dát, pre ktoré už existujú naprogramované knižnice spracovávajúce ich na objekty či iné programátorom použiteľné formáty. Vďaka tomu je prijímanie a následne spracovanie dát tak rýchle a efektívne.

XML

Skratka XML znamená Extensible Markup Language, čo napovedá o tom, že ide o značkovací jazyk, ktorý slúži na tvorbu dokumentov obsahujúcich štruktúrované informácie. Vychádza z jazyku SGML (Standard generalized markup language) taktiež ako jazyk HTML, ktorý je jazyku XML veľmi podobný. Oba používajú na popis súboru značkovacie symboly, pričom jazyk HTML sa špecializuje výhradne na popis stránok umiestnených na webe. Na druhej strane využitie jazyku XML je omnoho širšie, čo nám umožňuje zdieľať akékoľvek informácie v štruktúrovanej a konzistentnej podobe. Základnou stavebnou časťou XML dokumentu je element, ktorý je definovaný tzv. tagmi. Element sa skladá z počiatočného a ukončujúceho tagu. Všetky elementy nachádzajúce sa v dokumente sú ohraničené najvrchnejším tagom nazývaným koreňový element. Jazyk XML podporuje vnorené elementy, vďaka čomu je možné dosiahnuť hierarchickej štruktúry dokumentov. Názvy elementov popisujú ich obsah, zatiaľ čo ich štruktúra popisuje vzťahy medzi nimi. Na nasledujúcej ukážke je možné vidieť obsah jednoduchého XML dokumentu s vnorenými elementmi.[11]

```
<?xml version="1.0" encoding="UTF-8" ?>
<mliecne_vyrobky>
  <syr>
     <nazov>Syr_1</nazov>
     <druh>Kravský</druh>
     <vyrobene_v>Holandsko</vyrobene_v>
  </syr>
  <syr>
     <nazov>Syr_2</nazov>
     <druh>Ovčí</druh>
     <vyrobene_v>Slovensko</vyrobene_v>
  </syr>
</mliecne_vyrobky>
```

JSON

JavaScript Object Notation, skrátene JSON je ďalší, momentálne veľmi rozšírený spôsob zápisu dát do organizovanej podoby s dôrazom na jednoduchý prístup k dátam. Hoci JSON bol pôvodne odvodený od skriptovacieho jazyka JavaScript, je jazykovo nezávislým dátovým formátom. Vďaka svojej jednoduchosti je ľahko čitateľný aj človekom. Na začiatku sa celý obsah ohraničí zloženými zátvorkami, čím vlastne vznikne objekt. Medzi zátvorkami potom môžeme nájsť tieto základné dátové typy:[12]

- Číslo
- Reťazec
- Boolean
- Null

A dve dátové štruktúry:

- Objekt – neusporiadaná množina párov (názov/hodnota). Objekt začína ľavou „{“ a končí pravou „}“ zloženou zátvorkou. V tele objektu sa nachádza názov, za ktorým nasleduje dvojbodka a odpovedajúca hodnota. Viacero párov názvov a hodnôt sa oddeľuje pomocou čiarky.
- Pole – usporiadaná kolekcia hodnôt. Pole sa začína ľavou „[“ a končí pravou „]“ hranatou zátvorkou. Hodnoty sú oddelené čiarkou. Vyššie spomínaný dokument v XML si teraz prepíšeme do JSON-u:

```
{
  "mliecne_vyrobky": {
    "syr": [
      {
        "nazov": "Syr_1",
        "druh": "Kravský",
        "vyrobene_v": "Holandsko"
      },
      {
        "nazov": "Syr_2",
        "druh": "Ovčí",
        "vyrobene_v": "Slovensko"
      }
    ]
  }
}
```

Na základe porovnania oboch zápisov môžeme konštatovať, že JSON potrebuje na rovnaký zápis dát menej znakov oproti jazyku XML. Vďaka tejto vlastnosti sa preferuje čoraz viac tento zápis pri výmene dát, kde veľkosť odosielaného súboru zohráva kľúčovú rolu.

Architektúra REST

Celým názvom – Representational State Transfer je dnes často používaná architektúra, ktorá umožňuje pristupovať k dátam na určitom mieste pomocou štandradných metód HTTP. Spolu s metódou SOAP (Simple Object Access Protocol) patria k najpoužívanejším metódam na komunikáciu medzi serverom a klientom. REST bol vytvorený v roku 2000 pánom Royom Fieldingom, jedným zo spoluautorov HTTP protokolu. Komunikácia medzi klientom a serverom prebieha pomocou URL adresy, kde sa špecifikuje adresa servera a posielané parametre oddelené znakom „&“. REST je narozdiel od procedurálne orientovaného SOAP orientovaný dátovo, vďaka čomu je jednoduchší na použitie. Na nasledujúcom porovnaní je možné vidieť dve rovnaké požiadavky na server využitím metódy SOAP a REST:[14]

```
<?xml version="1.0"?>
<soap:Envelope
xmlns:soap="http://www.w3.org/2001/12/soap-envelope"
soap:encodingStyle="http://www.w3.org/2001/12/soap-encoding">
 <soap:body pb="http://www.acme.com/phonebook">
  <pb:GetUserDetails>
   <pb:UserID>12345</pb:UserID>
  </pb:GetUserDetails>
 </soap:Body>
</soap:Envelope>
```

Metóda SOAP používa na komunikáciu jazyk XML, kvôli čomu sa posiela nadbytočné množstvo dát. Pomocou metódy REST by tá istý požiadavka vyzerala takto:

```
http://www.acme.com/phonebook/UserDetails/12345
```

Okrem posielania menšieho množstva dát a jednoduchšej implementácie, táto architektúra prináša ďalšie výhody akými sú napríklad:

- Nezávislosť na platforme
- Nezávislosť na programovacom jazyku
- Nemá oficiálny štandard, vývojár si ho nadefinuje sám

4 Návrh

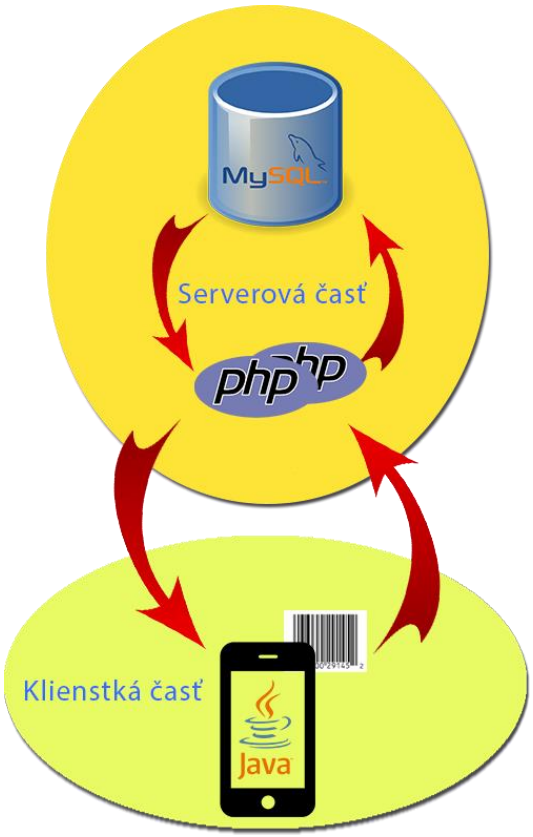
Dôležitou súčasťou vývoja každej aplikácie je jej správny návrh. Keďže naša aplikácia získava dáta zo vzdialenej databázy, bude potrebné si takúto databázu zaobstarať. Na internete existuje množstvo webhostingov, ktoré či už za poplatok alebo zdarma s rôznymi obmedzeniami, ponúkajú svoje serveri s možnosťou vytvorenia vlastnej MySQL databázy. Nami vybratý webhosting www.000webhost.com je ideálny pre vývojovú fázu projektu, keďže v bezplatnej verzií ponúka MySQL databázu, podporu PHP skriptov a 1500MB voľného priestoru.

4.1 Klient-server protokol

Komunikáciu servera s klientom si môžeme pozrieť na nasledujúcej grafike.

Klientská časť zahŕňa našu samotnú mobilnú aplikáciu. Ak vďaka nej naskenujeme čiarový kód produktu, získame tým unikátne čislo, pomocou ktorého vieme identifikovať každý produkt v našej vzdialenej databáze. Na to aby sme mohli komunikovať s touto databázou potrebujeme serverovú časť, ktorá sa skladá z PHP skriptov a MySQL databázy. Celá komunikácia prebieha nasledovne:

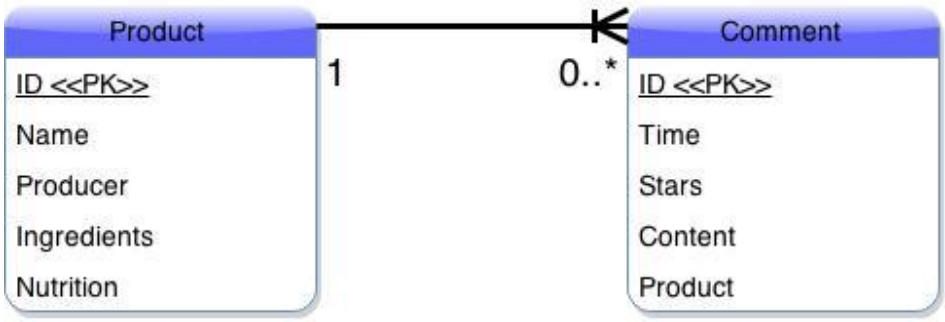
1. Používateľ si naskenuje produkt. Na základe jeho zvolenej funkcie sa odošlú dáta a vyhľadá sa odpovedajúci PHP skript na serveri.
2. Náš PHP skript tieto dáta spracuje, spojí sa s databázou a odošle jej požiadavku na konkrétne dáta.
3. MySQL databáza túto požiadavku vo forme SQL príkazu spracuje, vyhľadá požadované dáta v tabuľke a pošle ich späť našemu skriptu.
4. Skript tieto informácie prepíše do JSON formátu a odošle naspäť našej aplikácií.
5. Vďaka dátam uloženým v JSON formáte si ich aplikácia jednoducho transformuje do objektov, vďaka ktorým používateľovi bude schopná zobraziť žiadané dáta.



Obrázok 4.1: Grafické znázornenie komunikácie klienta so serverom

4.2 Návrh databázy

Na návrh databázy sme použili Diagram vzťahov entít (Entity relationship diagram alebo ERD). Ten obrazne popisuje jednotlivé tabuľky databázy a vzťahy medzi nimi.



Obrázok 4.2: Ukážka ER diagramu vzdialenej databázy

Naša databáza sa skladá z dvoch tabuliek a jednej čiary vyjadrujúcej vzťah medzi nimi. Z diagramu je vidieť, že jeden produkt môže obsahovať nula až N komentárov. Takisto tu vidíme v oboch tabuľkách kľúčové slovo <<PK>>[3], čo značí, že sa jedná o primárny kľúč tabuľky. Opíšme si teda detailne obe spomínané tabuľky.

Tabuľka Product obsahuje:

- ID – identifikačné číslo záznamu (identifier), získame ho z čiarového kódu
- Name – názov produktu
- Producer – výrobca produktu
- Ingredients – zloženie produktu
- Nutrition – nutričné hodnoty produktu

Tabuľka Comment obsahuje:

- ID – identifikačné číslo záznamu
- Time – čas, kedy bol záznam pridaný
- Stars – počet hviezdičiek v rozmedzí od jedna do päť
- Content – obsah komentára
- Product – identifikačné číslo produktu, ktorému tento komentár patrí

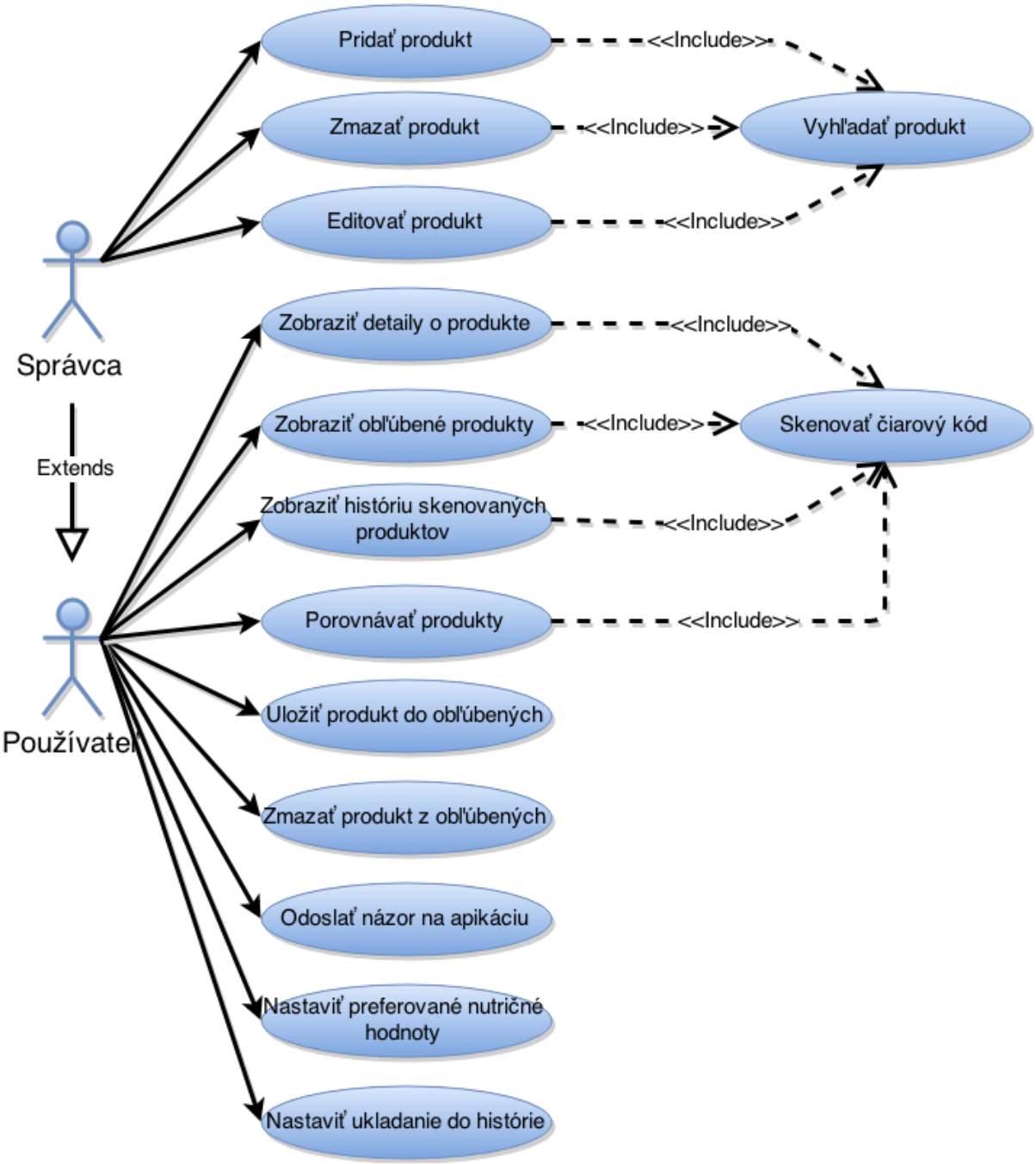
Vďaka správnemu návrhu sa nám bude databáza vytvárať rýchlo a jednoducho. Po vytvorení SQL kódu bude potrebné vytvoriť si MySQL server, na ktorom tento kód spustíme a tým tieto tabuľky vytvoríme. Okrem vzdialenej databázy budeme v aplikácií využívať aj lokálnu, uloženú v našom

[3] Primary key – v preklade znamená primárny kľúč.

mobilnom telefóne. Keďže tá nebude mať prístup k internetu, jej tabuľka „Product“ bude mať navyše riadok „stars“, čiže priebežný počet hviezdičiek aktualizujúci sa vždy po pripojení na internet.

4.3 Prípady použitia

Ďalším užitočným diagramom počas návrhu aplikácie je Diagram prípadu použitia (Use case diagram). Ten nám graficky zobrazuje úlohy dostupné pre každého aktéra používajúceho našu mobilnú aplikáciu.



Obrázok 4.3: Ukážka diagramu prípadov použitia

V diagrame vidíme dvoch aktérov – Správcu a Používateľa. Ako je možno vidieť na obrázku, správca dedí všetky prípady použitia od používateľa. Používateľ teda nemá možnosť pridania, zmazania či editovania produktu. Na vykonanie vybraných prípadov použitia bude potrebné najprv vykonať vyhľadanie produktu alebo skenovanie čiarového kódu ako to je znázornené na diagrame.

4.4 Logo a názov aplikácie

Keďže sa aplikácia bude využívať komerčne, musí mať atraktívny názov a jej logo. Preto sa pri tvorbe v oboch prípadoch snažilo čo najviac vystihnúť podstatu aplikácie či už graficky v podobe loga, alebo slovne v podobe názvu. Aplikácia slúži ako nákupný radca a preto dostala pomenovanie „BuyIt?“. Názov sa skladá z dvoch anglických slov a to: Buy (kúpiť) a it? (to?). Sú to slová, ktoré si používateľ kladie pri výbere produktu ešte pred použitím našej aplikácie. Na zodpovedanie tejto otázky preto použije našu aplikáciu. Spolu tak tieto slová tvoria krátke, ľahko zapamätateľné a výstižné meno pre našu aplikáciu.

Pri tvorbe loga sa použili farby dominujúce v celej aplikácií. Ako je vidieť na obrázku, logo tvorí kruh tyrkysovej farby, v ktorom je umiestnený názov aplikácie a nákupný košík imitujúci otáznik. Alternatívou pri tmavšom pozadí je použiť iba názov a košík bez obkolesujúceho kruhu.

Obrázok 4.4:
Logo aplikácie

5 Implementácia

Nespornou súčasťou vývojového procesu mobilnej aplikácie je určite jej implementácia. V tomto štádiu by sme už mali mať zozbierané, naštudované a vypracované všetky materiály. Na realizáciu celého projektu bolo potrebné použiť programovací jazyk Java, skriptovací PHP a značkovací XML. Celú implementáciu teda možno rozdeliť na tri nasledovné podkapitoly:

5.1 Grafické používateľské rozhranie

Skrátene GUI z ang. Graphic User Interface má veľký význam hlavne pre koncových používateľov, ktorý očakávajú intuitívny, jednoduchý a pekný dizajn. Z tohto dôvodu bolo rozhodnuté, že sa pri tvorbe GUI bude dodržiavať Material dizajn, ktorý bol popísaný v kapitole 3. Implementovať GUI je možné dvoma spôsobmi a to: použitím dizajnéra v Android Studiu alebo manuálnym písaním XML kódu. Pre naše potreby bola zvolená druhá metóda, keďže po získaní praxe písaním XML kódu je táto metóda oveľa efektívnejšia. Celkovo bolo vytvorených približne 20 súborov definujúcich vzhľad aplikácie, z ktorých väčšina slúži na popis dizajnu aktivity, prípadne popis len jej určitej časti. Tieto súbory môžeme nájsť v nasledujúcich zložkách:

Buyit\app\src\main\res\layout – popis dizajnov jednotlivých aktivít, prípadne prvkov

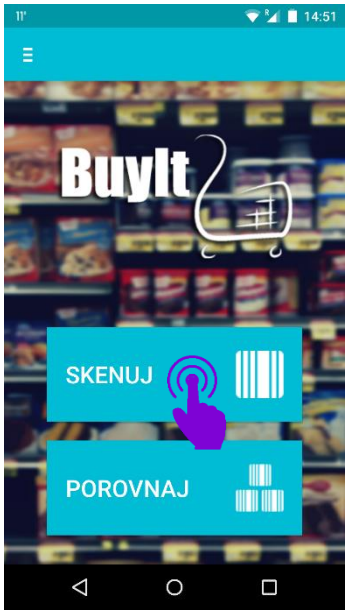
Buyit\app\src\main\res\drawable - animácie

Buyit\app\src\main\res\menu – popis horného menu každej aktivity

Buyit\app\src\main\res\xml – popis nastavení

5.1.1 Ukážky dizajnu aplikácie

V tejto sekcií si názorne ukážeme na snímkach obrazovky testovaného telefónu prechod medzi jednotlivými aktivitami aplikácie. Pre lepšiu predstavu nám bude fialový prst s krúžkom pri ukazováku symbolizovať miesto dotyku skutočného prsta.

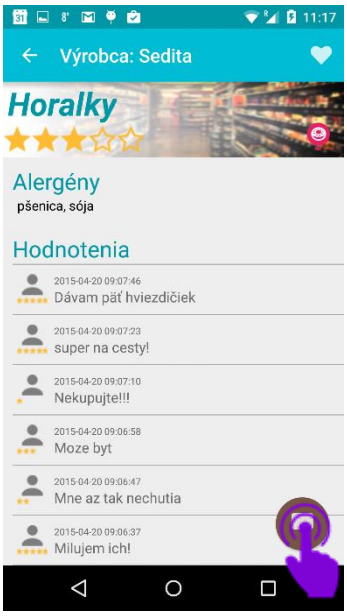


Obrázok 5.1:
Úvodná obrazovka

Obrázok 5.2:
Skenovanie produktu

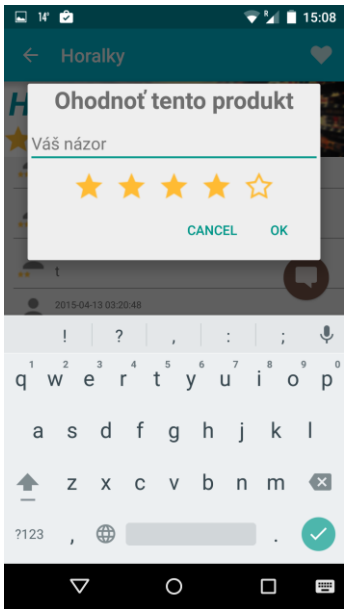
Obrázok 5.3:
Detail produktu č.1

Na obrázku 5.1 vidíme úvodnú obrazovku pozostávajúcu z troch tlačidiel: tlačidlo pre vysunutie menu v ľavom hornom rohu a dve dominantné tlačidlá v dolnej časti obrazovky určené na zahájenie skenovania čiarového kódu alebo porovnávanie produktov založené na používateľových nastaveniach. Po stlačení tlačidla „Skenuj“ sa zobrazí obrázok 5.2, na ktorom je vidieť ako používateľa naviguje aplikácia pre úspešné naskenovanie produktu či už textovou formou alebo aj graficky v podobe nestmaveného obdĺžnika obsahujúceho červenú skenovaciu čiaru. Akonáhle je čiarový kód produktu nasnímaný, spustí sa obrazovka, ktorá je zachytená na obrázku 5.3. Tu už má možnosť používateľ získať všetky informácie o produkte na jednom mieste. Ak si popíšeme obrazovku z hora nadol, tak ako na prvý narazíme na action bar (viď. podkapitola 2.6.1). Ten pozostáva z tlačidla pre návrat do úvodnej obrazovky, mena výrobcu produktu a tlačidla na uloženie produktu do obľúbených (ak je srdce vyplnené, znamená to, že tento produkt sa už nachádza v zozname obľúbených produktov, v opačnom prípade je toto srdce s priehľadným stredom). Pod action barom nájdeme názov produktu, jeho celkové hodnotenie vo forme hviezdičiek a kategóriu, do ktorej bol produkt priradený na základe jeho nutričných hodnôt. Táto lišta ostáva pevne pod action barom nehľadiac na to ako rolujeme so zvyškom obsahu.



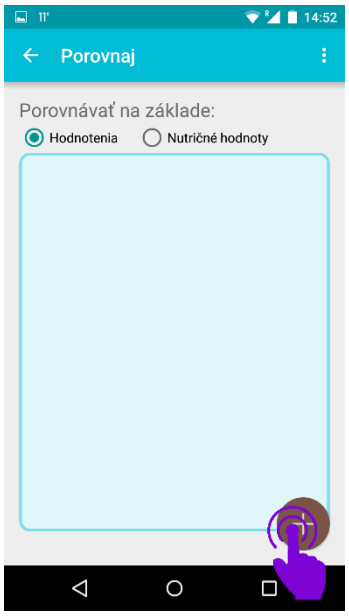
Obrázok 5.4:
Detail produktu č.2

Týchto kategórií môže byť viac alebo nemusí patriť do žiadnej. Obrazovka ešte obsahuje zloženie produktu, graf s nutričnými hodnotami, zoznam alergénov získaný zo zloženia produktu a hodnotenia používateľov. Posledné dve položky je možné vidieť po zrolovaní obrazovky na obrázku 5.4. Každé hodnotenie pozostáva z textu, počtu hviezdičiek a času jeho pridania, ktorý už ale pridáva samotná databáza. Bez vyplnenia hviezdičiek a textu aplikácia používateľovi nedovolí pridať nový komentár a upozorní ho na to vo forme krátkej správy umiestnenej v spodnej časti obrazovky. Po kliknutí na floating action button (viď. podkapitola 2.6.1) sa zobrazí dialógové okno, určené na vloženie nového komentáru (viď obrázok 5.5).



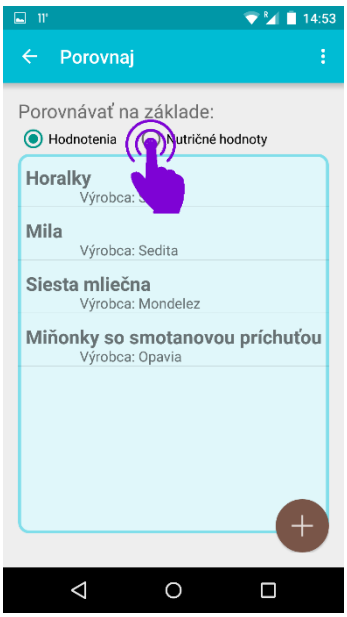
Obrázok 5.5: Pridávanie komentáru

Ak si na úvodnej obrazovke vyberieme druhé tlačidlo – „Porovnaj“, dostaneme sa do obrazovky, ktorú vidíme na obrázku 5.6.



Obrázok 5.6: Porovnávanie produktov

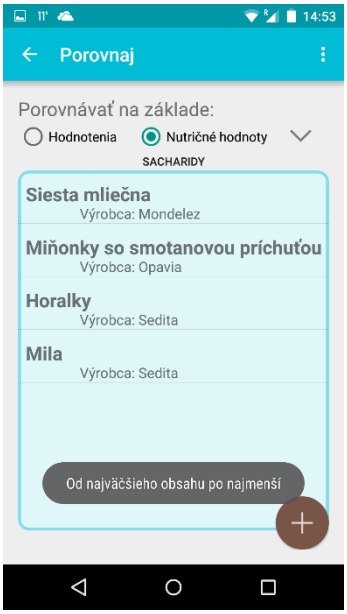
Obrázok 5.7: Skenovanie produktu



Obrázok 5.8: Výsledok porovnávania

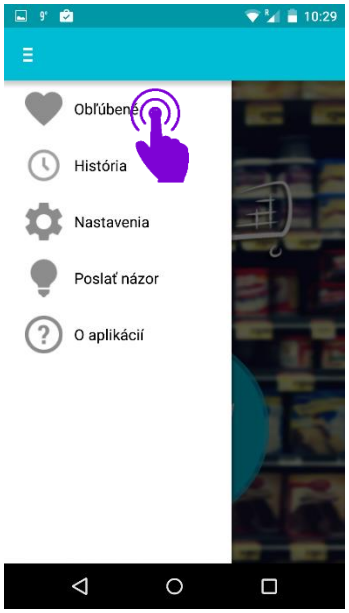
Dominantou tejto obrazovky je veľké slabomodré pole, ktoré slúži na zobrazovanie zoznamu porovnávaných produktov. Okrem toho tu vidíme opäť prvok material dizajnu a to konkrétne floating action button slúžiaci na pridávanie nových produktov do zoznamu. Po kliknutí na toto tlačidlo sa dostaneme na obrázok číslo 5.7, kde máme opäť možnosť pomocou skenera naskenovať čiarový kód

produktu a tým ho pridať do porovnávaného zoznamu. Ak tento proces zopakujeme pre všetky porovnávané produkty (v našom prípade 4-krát), naša obrazovka bude vyzerať ako na obrázku číslo 5.8. V tomto prípade sú naše všetky produkty zoradené na základe ich hodnotenia udeleného používateľmi od najlepšieho po najhoršie. Po kliknutí na hociktorý prvok zoznamu sa opäť dostaneme na detail produktu, ktorý je znázornený na obrázku 5.3. Ak si náš vytvorený zoznam chceme zoradiť podľa iných kritérií, môžeme si kliknúť na tlačidlo „Nutričné hodnoty“, tým získame aktualizovaný zoznam produktov znázornený na obrázku 5.9. Takisto si môžeme všimnúť, že nám tu pribudli automaticky dve tlačidlá. Prvé, vo forme šípky smerujúcej nadol, určuje radenie jednotlivých prvkov zoznamu (vzostupne alebo zostupne). Po kliknutí na toto tlačidlo nás aplikácia informuje v akom poradí sa momentálne všetky prvky nachádzajú a to oznamom v spodnej časti obrazovky a zmenou smeru šípky. Druhé tlačidlo nájdeme tesne nad zoznamom produktov, ktorého úlohou je informovať používateľa, na základe akej nutričnej hodnoty má zoradené produkty a zároveň po kliknutí slúži na zmenu preferovanej nutričnej hodnoty. Na záver si na tejto obrazovke môžeme ešte všimnúť v pravom hornom rohu tlačidlo v tvare troch bodiek. To nám po kliknutí ponúka možnosť vytvoriť nový zoznam porovnávaných produktov.

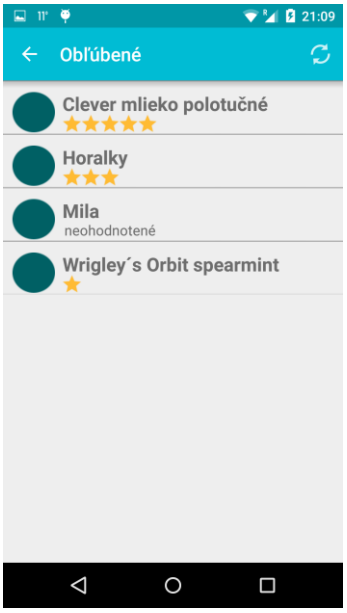


Obrázok 5.9: Zmena poradia zoznamu produktov

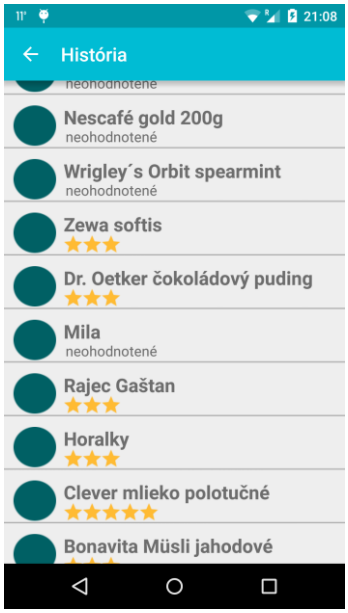
Okrem skenovania produktov aplikácia umožňuje zobrazenie histórie nasnímaných produktov, či prezeranie a spravovanie zoznamu obľúbených produktov. Všetky tieto produkty sú ukladané do lokálnej databázy v telefóne a preto sú dostupné aj bez pripojenia na internet. Pre prístup do týchto zoznamov je potrebné otvoriť bočné menu buď potiahnutím prsta z ľavého okraja aplikácie smerom do stredu, alebo kliknutím na tlačidlo menu na úvodnej obrazovke. Celý proces je znázornený na obrázkoch 5.10 až 5.12.



Obrázok 5.10:
Menu aplikácie



Obrázok 5.11:
Obľúbené produkty



Obrázok 5.12:
História produktov

Na prvom z trojice obrázkov vidíme bočné menu, z ktorého najprv vyberáme položku „Obľúbené“. Zobrazí sa nám zoznam obľúbených produktov, ktorý vidíme na obrázku 5.11. Keďže tento zoznam čerpá svoj obsah z lokálnej databázy uloženej v telefóne, nemá po svojom zobrazení aktualizované informácie z databázy aby bol dostupný aj bez internetového pripojenia. Ak používateľ získa internetové pripojenie, aktualizuje svoj zoznam jednoducho kliknutím na tlačidlo obnovenia v pravom hornom rohu. Po obnovení je používateľ informovaný o úspechu operácie. Používateľ si taktiež môže zobraziť detailné informácie o produkte (obrázok 5.3) po kliknutí na zvolený produkt. Na obrázku 5.12 ešte vidíme históriu skenovaní, ktorá funguje na podobnom princípe ako zoznam obľúbených. Štandardne aplikácia ukladá 10 naposledy naskenovaných produktov, tým pádom po naskenovaní 11-teho produktu je najstarší produkt automaticky vymazaný a nahradení novým. Toto chovanie sa dá zmeniť v nastaveniach, kde je možné nastaviť počet ukladaných produktov na: 0, 5, 10, 15, 20, 25. V bočnom menu má používateľ na výber ešte možnosť poslať svoj názor na aplikáciu na osobný email vývojára, prípadne si môže pozrieť informácie o aplikácií, kde nájde vysvetlenie ku každej kategórií, ktorá je priraďovaná produktom, aktuálnu verziu aplikácie a meno jej autora.

5.2 Komunikácia s databázou

Ako už bolo spomenuté vyššie, na komunikáciu s databázou nám slúžia skripty napísané v jazyku PHP. Naša aplikácia používa na základe vykonávanej operácie tri rôzne skripty:

Single_product.php

Ako už z názvu vyplýva, skript pracuje len s jedným produktom. Spúšťa ho aplikácia formou GET požiadavku odoslaného na server spolu s identifikačným číslom produktu. Nižšie je možné vidieť ukážku takejto URL požiadavky:

```
http://vlado.site11.com/single_product.php?id=8586011330746
```

Túto URL požiadavku si môžeme rozdeliť na tri časti:

- http://vlado.site11.com/ - názov domény, kde sa nachádza náš server
- single_product.php – názov skriptu, ktorý na našej doméne spúšťame
- ?id=8586011330746 – otáznik na začiatku definuje, kde začínajú parametre posielané skriptu. Za otáznikom vidíme len jeden parameter a to „id“ s hodnotou 8586011330746.

Ak už skript získa identifikačné číslo produktu, môže sa spojiť s databázou a jednoducho nájsť produkt v tabuľke s takýmto číslom. Táto operácia sa dá dosiahnuť kombináciou nasledovného PHP a SQL príkazu:

```
$product = mysqli_query($con,"SELECT * FROM Product where
id='$id'");
$comments = mysqli_query($con,"SELECT * FROM Comment where
'$id' = product");
```

V prvom riadku naplníme premennú `$product` produktom, ktorého id hodnota sa zhoduje s id hodnotou v tabuľke. V druhom riadku zase dostaneme do premennej `$comments` všetky komentáre, ktorých hodnota riadku „product“ sa zhoduje s hodnotou id skenovaného produktu. Tým pádom získame iba tie komentáre, ktoré prislúchajú nášmu produktu. Všetky tieto hodnoty sa potom vložia do asociatívneho poľa a prevedú do formátu JSON. Aby bolo možné rozlíšiť, kde v tomto reťazci končia informácie o produkte a kde začínajú o komentároch, vkladá sa medzi ne oddeľujúci znak „~“.

Multiple_product.php

Na rozdiel od vyššie spomínaného skriptu, tento prijíma na vstupe neobmedzené množstvo parametrov. Kvôli tomu bolo nutné zmeniť formu odosielania požiadavku aplikácie na server, keďže metóda GET nedokáže poslať naraz väčšie množstvo dát. Pre tento účel bola vybratá metóda POST, ktorá nám zaručí spoľahlivý prenos dát aj pri veľmi vysokom počte žiadaných produktov. Skript je potom až na drobné zmeny podobný prvému.

Upload_comment.php

Jediný z pomedzi skriptov, ktorý namiesto vyberania prvkov z tabuľky databázy ich naopak do nej vkladá. Slúži na uloženie nového komentára do databázy. Keďže skript vkladá komentáre vždy len po jednom, mohla byť opäť využitá metóda GET. Požiadavka, ktorú skript obdrží, obsahuje identifikačné číslo produktu, ktorému daný komentár patrí, počet hviezdičiek a obsah komentáru vo forme textového reťazca. Po vložení komentáru do databázy sa automaticky uloží aj čas jeho pridania.

5.3 Programovanie aplikácie

Programovanie logickej stavby aplikácie prebiehalo v jazyku Java. Všetky zdrojové súbory sú rozdelené do dvoch balíkov a to:

Balík google\zxing\client\android

Klasicky je súčasťou aplikácie Barcode Scanner od tvorcov ZXing Team, ktorých odkúpila spoločnosť Google a otvorila zdrojové kódy tejto aplikácie pre širokú verejnosť. Vďaka tomu bolo možné importovať časť tohto projektu do našej aplikácie a za pomoci malých zmien v kóde získať plnohodnotný skener čiarových kódov.

Balík example\vladimir\buyit

Obsahuje zvyšnú časť implementácie aplikácie. Okrem kódov pre jednotlivé aktivity tu je možné nájsť implementácie pre adaptéry, rôzne objekty a fragment.

- **Bočné vysúvacie menu**

Naposledy zmienený fragment – `NavigationDrawerFragment.java` slúži na naprogramovanie vyťahovacieho bočného menu. S tým súvisia ďalšie triedy ako `NavDrawerItem.java`, ktorá definuje jednu položku v menu zloženú z ikonky a jej popisu a `NavDrawerAdapter.java`, ktorá zase umožňuje zobraziť všetky tieto položky vo vertikálnom zozname.

- **Nastavenia**

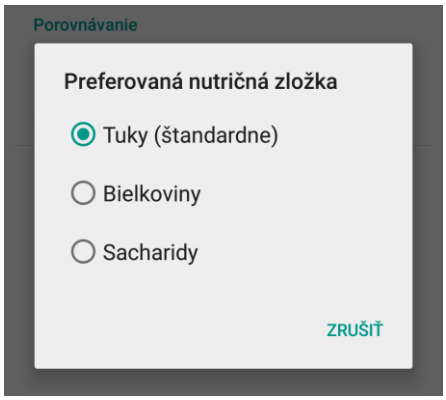
Keďže pri implementácií nastavení potrebujeme nejakým spôsobom ukladať používateľove zmeny natrvalo a takisto chceme zachovať určitú konzistenciu ich vzhľadu naprieč aplikáciami, spoločnosť Google vymyslela jednotný preddefinovaný postup pri implementácií nastavení, ktorého by sa mal držať každý vývojár aplikácií. Základom je aktivita, ktorú rozširuje trieda PreferenceActivity. Po spustení sa v metóde OnCreate(Bundle savedInstanceState) načíta vzhľad nastavení, ktorý sa ale od ostatných súborov definujúcich vzhľad aplikácie líši. Do neho sa vkladajú len špeciálne objekty podľa

množstva a druhu informácie, ktoré používateľ bude meniť. V našom prípade vyberáme z preddefinovaných možností a to konkrétne len z troch, preto sme použili objekt typu ListPreference. Nižšie môžeme vidieť časť kódu a jeho výsledok v aplikácií:

```
<ListPreference
    android:key="pref_preferred_nutrition"
    android:title="Preferovaná nutričná zložka"
    android:summary="Nutričná zložka, podľa ktorej bude prebiehať porovnávanie produktov"
    android:defaultValue="0"
    android:entries="@array/pref_nutrition_array"
    android:entryValues="@array/pref_nutrition_array_values">
</ListPreference>
```

Obrázok 5.13: Ukážka implementácie nastavení

A výsledok:



Obrázok 5.14: Ukážka GUI nastavení

- **Úvodná obrazovka**

Po spustení aplikácie sa nám ako prvý súbor spustí `MainActivity.java`. V ňom je zahrnutá implementácia úvodnej obrazovky. Tá pozostáva z inicializovania štandardných nastavení aplikácie, obsluhy výberu zo zoznamu v menu a akcií vykonaných po stlačení všetkých tlačidiel. V prípade, že používateľ stlačí tlačidlo „Skenuj“, zavolá sa aktivita z druhého balíka, ktorá má na starosti skenovanie čiarových kódov. Tá nám vráti identifikačné číslo produktu, z ktorého už opäť v našej aktivite tvoríme požiadavku typu GET, kontrolujeme dostupnosť pripojenia na internet a odosielame túto požiadavku na server. Úsek kódu, ktorý sa stará o prijatie a spracovanie správy zo servera je možné vidieť nižšie:

```
Response.Listener responseIntervalsListener = new Response.Listener() {
    @Override
    public void onResponse(Object response) {
        //hide progress
        findViewById(R.id.progress_main).setVisibility(View.GONE);
        try {
            if (response.toString().indexOf("!Not found!") != -1){
                Toast.makeText(MainActivity.this, R.string.
                    product_not_found, Toast.LENGTH_LONG).show();
            }
            else {
                JSONObject product = new JSONObject(response.toString().
                    substring(0, response.toString().lastIndexOf('~')));
                JSONArray comments = new JSONArray(response.toString().
                    substring(response.toString().lastIndexOf('~') + 1,
                    response.toString().indexOf("<!--") - 1));

                Product mProduct = new Product(product.getString("id"),
                    product.getString("name"), product.getString("producer"
                    ), product.getString("ingredients"), product.getString(
                    "nutrition"));
                ArrayList<Comment> allComments = new ArrayList<Comment>();

                for (int i = 0; i < comments.length(); i++) {
                    JSONObject comment = comments.getJSONObject(i);
                    Comment mComment = new Comment(comment.getString("id")
                        , comment.getString("time"), comment.getInt("stars"
                        ), comment.getString("content"));
                    mProduct.addComment(mComment);
                    allComments.add(mComment);
                }
                Intent resultIntent = new Intent(MainActivity.this,
                    ResultActivity.class);

                resultIntent.putExtra("ParentClassName", "MainActivity");
                resultIntent.putExtra("Comments", allComments);
                resultIntent.putExtra("Product", mProduct);

                startActivity(resultIntent);
            }
        }
        catch (JSONException e) {
            e.printStackTrace();
        }
    }
};
```

Obrázok 5.15: Ukážka implementácie prijímania správ od servera

Na začiatku vidíme schovanie točiaceho kolieska symbolizujúceho čakanie na odpoveď zo servera. Nasleduje overenie, či správa od servera neobsahuje reťazec „!Not found!“, čo by znamenalo, že skenovaný produkt sa na serveri nenašiel. V tom prípade dostane používateľ správu, ktorá ho o tom informuje, inak sa vytvorí objekt pre produkt a pole objektov pre komentáre typu JSON. Do nich sa vloží sformátovaná odpoveď servera v tvare JSON, ktorú ale napred treba upraviť, keďže produkt je oddelený od komentárov znakom „~“. Vzhľadom na to, že používame bezplatnú databázu, dostávame do každej odpovedi aj reklamu, ktorú je potrebné takisto odstrániť. Zo získaných JSON objektov si

vytvoríme objekty typu „Product“ a „Comment“, ktoré už potom budeme využívať v rámci celej aplikácie. Na konci si ešte vytvoríme nový intent[4] odkazujúci sa na aktivitu `ResultActivity.java`, vložíme doň získané informácie a spustíme ho. To ma za následok spustenie novej aktivity spomenutej vyššie, ktorá nám zobrazí detail produktu. Podobný kus kódu na prijímanie správ od servera sa vyskytuje v zdrojových súboroch aplikácie viackrát, no vždy s určitými obmenami.

- **Detail produktu**

Táto aktivita je zodpovedná za spracovanie a zobrazovanie všetkých informácií získaných z databázy. Ako jedinej sa nastavuje rodičovská aktivita dynamicky (aktivita, ku ktorej sa vrátime po stlačení tlačidla späť), vzhľadom na to, že túto aktivitu môžeme zavolať z rôznych obrazoviek. Niektoré získané dáta sa ešte pred zobrazením musia spracovať a to napríklad nutričné hodnoty produktov, ktoré sa najprv uložia do poľa a potom sa pomocou metódy `calculateData(float[] data)` a triedy `MyGraphview` transformujú na koláčový graf. Tu nastáva problém pri vykresľovaní grafu, pretože funkcia pracuje na vstupe s pixelmi, avšak ak chceme zachovať kompatibilitu pre všetky zariadenia s rôznymi uhlopriečkami, je potrebné si pred vykreslením získať hustotu bodov, ktoré má dané zariadenie na displeji a to vynásobiť počtom pixelov. Tým dostaneme našu želanú hodnotu nezávislú na veľkosti displeja. Taktiež sa musia pred každým vykreslením hviezdičiek, symbolizujúcich hodnotenie produktu, sčítať všetky počty hviezdičiek udelené v komentároch od používateľov a potom ich vydeliť ich počtom. Výpočet prebieha v metóde `calculateStars(List<Comment> items)` a samotné zobrazovanie výsledných hviezdičiek v `starsHandle(int number, int option)`. Táto metóda okrem zobrazovania spriemerovaných hviezdičiek zobrazuje aj hviezdičky v dialógovom okne (obrázok 5.5). Toto okno má na starosti metóda `commentOnClick(View v)`, ktorá sa aktivuje vždy po kliknutí na flying action button. Po korektnom vyplnení komentáru sa informácie odošlú na server pomocou metódy `downloadInfo(String content, int stars, final String product)`, ktorá sa svojím správaním veľmi nelíši od metódy používanej v aktivite MainAcitivy.java. Na zobrazovanie komentárov slúži metóda `refreshList(List<Comment> items)`. Tá vloží všetky komentáre do špeciálneho objektu, ktorý sa potom zobrazí na obrazovke.

- **Lokálne databázy**

Aby bolo možné mať dva rôzne zoznamy produktov (obľúbené produkty a história skenovaných produktov), bolo potrebné vytvoriť dve rôzne lokálne databázy. Prvá, `DatabaseFavorites.java`, spravuje naše obľúbené produkty. Vytvárajú sa v nej dve tabuľky

[4] Z angl. zámer – abstraktný objekt, určený na posielanie správ medzi aktivitami/aplikáciami

(Product a Comment), podobne ako to je v databáze na serveri. Trieda obsahuje metódy slúžiace na správu databázy, teda vytváranie, vyberanie a mazanie záznamov. `DatabaseHistory.java` je po implementačnej stránke podobná až na pár rozdielov. Na to aby sme mohli pracovať s takouto databázou, potrebujeme mať objekt reprezentujúci jeden záznam v databáze. Z tohto dôvodu boli vytvorené dve triedy: `Product.java` a `Comment.java`. Obe uchovávajú hodnoty, ktoré sú potom použité pri zobrazovaní informácií.

- ## Porovnávanie produktov

Na porovnávanie produktov slúži aktivita `CompareActivity.java`, ktorá sa spúšťa z úvodnej obrazovky po vyvolaní metódy `compareOnClick(View v)`. Po vytvorení si táto aktivita zistí nastavenia aplikácie a podľa nich nastaví meno tlačidlu nastavujúcemu nutričné hodnoty a tiež zobrazí zoznam produktov v prípade, že ho používateľ ešte nevymazal. Po stlačení tlačidla určeného na pridanie produktu do zoznamu sa opäť zavolá aktivita z druhého balíka. Jej odpoveď sa potom spracuje v metóde `onResponse(Object response)`, kde sa overuje navyše, či sa naskenovaný produkt už raz nenachádza v zozname produktov. V tom prípade používateľ obdrží krátku správu, inak sa produkt pridá do zoznamu pomocou metódy `sortProducts()`, ktorá ma okrem zobrazovania produktov za úlohu aj správne ich zoradiť podľa aktuálnych používateľových nastavení. To znamená, že vždy keď používateľ zmení kritéria na zobrazenie produktov, vykoná sa táto metóda.

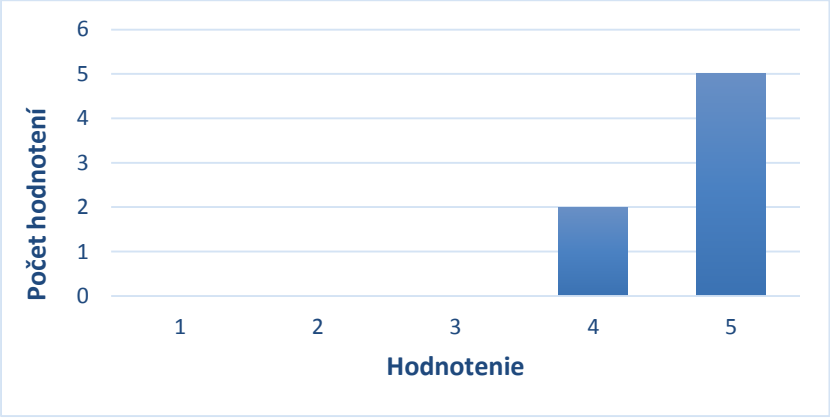
Zmeny kritérií zobrazovania produktov sa realizujú v dvoch metódach: `OrderingOnClick(View view)` a `onRadioButtonCompareClick(View view)`. Prvá zo spomínaných metód sa spustí vždy po kliknutí na tlačidlo v tvare šípky, ktoré zabezpečuje zmenu zoraďovania produktov (od najväčšieho po najmenšie a naopak). Druhá metóda sa aktivuje zase po stlačení na jedno zo skupiny tlačidiel, reprezentujúcich výber medzi porovnávaním na základe hodnotenia alebo nutričných hodnôt produktu. Po vybratí zoraďovania na základe hodnotenia je potrebné skryť prebytočné tlačidlá. O to sa stará metóda `nutritionButtonsAppearance(int rateBy)`.

6 Testovanie

Našim cieľom bolo vytvoriť aplikáciu, ktorá by používateľovi priniesla zážitok z jej používania, nepotreboval by žiaden návod, ako s ňou pracovať a rád by sa k nej vracal. Na to, aby toho bolo možné dosiahnuť, museli sme veľkú pozornosť venovať testovaniu. Spočiatku sa testovanie vykonávalo súbežne s implementáciou aplikácie, kedy sa skúšalo napríklad pracovať s extrémnymi vstupmi, na ktoré mala aplikácia v prípade chyby vždy reagovať krátkym a výstižným oznámením. Všetky tieto testy sa vykonávali na zariadení LG Nexus 5, na ktorom bola nainštalovaná posledná verzia Androidu 5.1. Po vyladení všetkých chýb na tomto zariadení bola aplikácia rozposielaná ďalším osobám s rôznymi zariadeniami (HTC Desire 5, Samsung Galaxy SII, Xiaomi Redmi Note 4G, OnePlus One a tablet Nexus 7). Všetky tieto zariadenia mali rôzne verzie Androidu, pričom najnižšia testovaná verzia bola 4.1.2, pri ktorej ako jedinej sa objavili menšie nedostatky, ktoré boli len vizuálneho charakteru a nijako nenarúšali chod aplikácie, ani neobmedzovali používateľa pri jej používaní. Žiadnej osobe, ktorá sa zúčastnila testovania nebolo vopred vysvetlené ako aplikácia funguje a načo slúži. Na testovanie im bola rozoslaná fotografia s troma čiarovými kódmi produktov, aby sa predišlo skenovaniu kódov, ktoré sa ešte nenachádzajú v databáze. Taktiež im boli rozoslané tri otázky, ktorých znenie a odpovede možno vidieť na nasledujúcich grafoch:

1. **Je aplikácia intuitívna?**

   (Otázkou bolo myslené či používateľ ľahko pochopí účel, ovládanie a rozmiestnenie prvkov aplikácie bez toho aby mu to vopred niekto ukázal. Hodnotilo sa číslami od 1 do 5, pričom 5 symbolizuje najlepšie hodnotenie)

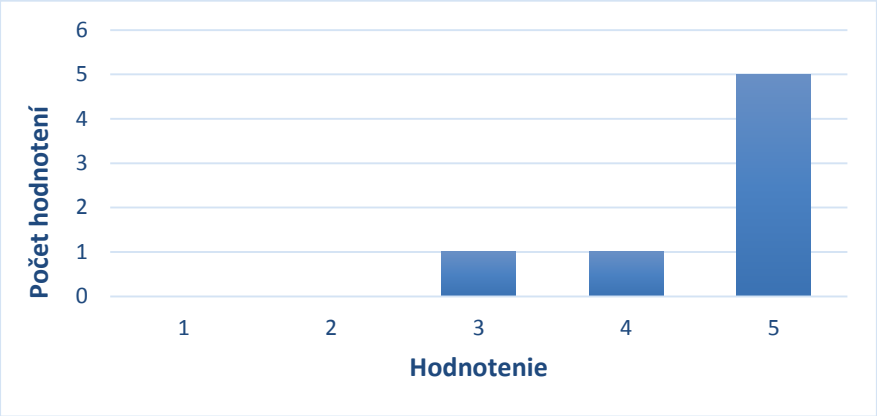


Graf 6.1: Výsledok prvej otázky

Aplikácia v teste uspela veľmi dobre, keď len 20% respondentov jej udelilo nižšie hodnotenie ako najvyššie možné a to s hodnotou 4.

2. Oslovil ťa dizajn aplikácie?

(Spôsob hodnotenia rovnaký ako pri otázke č. 1)



Graf 6.2: Výsledok druhej otázky

Hodnotenie dopadlo opäť veľmi dobre i keď horšie ako pri prvej otázke. Dôvodom bolo hodnotenie používateľa, ktorý mal zapnutú aplikáciu na verzií Androidu 4.1.2, tá totiž nezobrazuje isté detaily tak ako bolo zámerom. Časom sa ale plánuje upraviť zdrojový kód tak, aby vyzeral na všetkých zariadeniach úplne rovnako a tým sa nám snáď podarí eliminovať horšie hodnotenia za dizajn aplikácie.

3. Z desiatich návštev supermarketu, koľko krát by si použil/a aplikáciu na uľahčenie výberu produktu?

Graf 6.3: Výsledok tretej otázky

Z grafu je možné usúdiť, že najviac používateľov by aplikáciu využívalo v rozmedzí od 5 do 7krát z desiatich nákupov na pomoc pri výbere správneho produktu. Túto štatistiku možno taktiež pokladať za úspešnú, ale samozrejme bude nutné stále hľadať a pridávať nové funkcie do aplikácie aby sme dosiahli nabudúce ešte lepších výsledkov.

V nasledujúcom teste sme sledovali, ako dlho bude aplikácii trvať vyhľadanie produktu na serveri pomocou rôznych pripojení na internet. Pri každom spôsobe pripojenia sme merali čas 3krát a urobili priemer týchto časov.

Pripojenie	**Wi-Fi**	**4G**	**3G**	**2G**
Čas	1,51 s	1,44 s	2,06 s	2,13 s

Tabuľka 6.1: Výsledok testu rýchlosti komunikácie s databázou

Z uvedených výsledkov môžeme konštatovať, že spojenie aplikácie so serverom je pri každom druhu pripojenia časovo nenáročné, preto nebude problém ju používať aj v odľahlejších oblastiach s obmedzeným prístupom na internet. Príchodom nových technológií pripojenia a prechodom na rýchlejší, platený server sa tieto výsledky ešte môžu zlepšiť.

7 Záver

Práca vznikla za účelom uľahčiť a zmodernizovať proces vyberania produktov počas každodenného nakupovania. Vďaka tejto aplikácií sa budú môcť používatelia pri výbere produktov riadiť okrem ceny a lákavého obalu aj zložením a chuti produktu. To umožní používateľovi nielen zdravšie a chutnejšie sa stravovať, no môže to takisto viesť k podpore malých výrobcov, ktorý častokrát nemôžu konkurovať cenami či reklamnou kampaňou veľkým spoločnostiam, ktorých výrobky nemusia byť vždy tie najlepšie. Prostredníctvom špeciálne pridelených kategórií bude používateľ schopný vyhľadať správny produkt aj bez inak potrebných znalostí, ako by malo vyzerať jeho zloženie. Zobrazované grafy mu zase pomôžu lepšie si predstaviť, akú časť z obsahu produktu tvoria konkrétne nutričné hodnoty, čo je veľmi dôležité vedieť pri tvorbe vlastného jedálnička. Komentáre ľudí doplnené o hodnotenie s ich mierou spokojnosti s produktom takisto napomôžu používateľovi správne sa rozhodnúť pri jeho výbere. Vzhľadom na to, že v tejto oblasti doposiaľ neexistujú riešenia pre smartfóny, má táto aplikácia potenciál osloviť veľké skupiny ľudí.

Predtým ako ale bude zverejnená a spropagovaná širokej verejnosti, je potrebné ešte vyriešiť niektoré problémy. V prvom rade to je databáza, ktorú treba naplniť na počiatku aspoň najpredávanejšími českými a zahraničnými produktmi a dovoliť používateľom podieľať sa na rozširovaní databázy výmenou za nejakú odmenu. Taktiež aplikácia nezobrazuje všetky existujúce alergény úplne spoľahlivo, a preto je potrebné vymyslieť iný, dôveryhodnejší postup pri vyhľadávaní alergénov. Jedným z riešení, ktoré by mohlo oba tieto problémy aspoň čiastočne riešiť je vytvorenie webovej stránky, určenej práve výrobcom potravín, ktorý budú mať možnosť vložiť informácie o ich produktoch. To by nám mohlo zaručiť dostatočne veľkú databázu s relevantnými informáciami. Na všetky ostatné problémy je do aplikácie vložená možnosť odoslať návrh či pripomienku autorovi, vďaka ktorým bude možné aplikáciu ďalej inovovať.

V práci sme sa mali možnosť dočítať o technológií databáz, kde sme si aj ukázali akým spôsobom s nimi môžeme komunikovať a tiež sme si povedali základné fakty o čiarových kódoch (kapitola 3). Rovnako sme si popísali návrh aplikácie s vlastným protokolom používaným na komunikáciu klienta so serverom (kapitola 4). Nakoniec sme sa venovali implementácií aplikácie (kapitola 5) a vyhodnotili sme si výsledky jej testovania (kapitola 6).

Literatúra

[1] PC Life. [online]. [cit. 2014-12-13]. Dostupné z: http://www.pclife.cz/256-android-historie-a-funkce-aneb-chytry-mobilni-operacni-system/

[2] Android-App-Market. [online]. [cit. 2014-12-13]. Dostupné z: http://www.android-app-market.com/android-architecture.html

[3] LACKO, Ľuboslav. Vývoj aplikácie pre Android: Architektúra platformy Android. *PC Revue*. Október 2014. DOI: 1335-0226.

[4] Androidcentral. [online]. [cit. 2014-12-14]. Dostupné z: http://www.androidcentral.com/android-versions

[5] Android Developers: Dashboards. [online]. [cit. 2014-12-14]. Dostupné z: https://developer.android.com/about/dashboards/index.html

[6] Google: Activities. [online]. [cit. 2014-12-28]. Dostupné z: https://developer.android.com/guide/components/activities.html

[7] Google: Layout. [online]. [cit. 2014-12-19]. Dostupné z:http://www.google.com/design/spec/layout/structure.html#structure-ui-regions-guidance

[8] Kodys Slovensko: Trochu teórie o čiarovom kóde. [online]. [cit. 2014-12-15]. Dostupné z:http://www.kodys.sk/stranka/trochu-teorie-o-ciarovom-kode

[9] *Types of Barcode Readers* [online]. [cit. 2015-05-17]. Dostupné z: http://www.sumlung.com/en/about-barcode/230.html

[10] What is MySQL? [online]. [cit. 2015-05-12]. Dostupné z: https://dev.mysql.com/doc/refman/4.1/en/what-is-mysql.html

[11] O'Reilly - www.XML.com: A Technical Introduction to XML [online]. [cit. 2015-05-12]. Dostupné z: http://www.xml.com/pub/a/98/10/guide0.html?page=2

[12] *Introducing JSON* [online]. [cit. 2015-05-12]. Dostupné z: http://json.org/

[13] KALALI, Masoud a Bhakti MEHTA. *Developing RESTful Services with JAX-RS 2.0, WebSockets, and JSON*. Birmingham: Packt Publishing, 2013. ISBN 9781782178125.

[14] Learn REST: A Tutorial [online]. [cit. 2015-05-04]. Dostupné z: http://rest.elkstein.org/

[15] *How do barcodes and barcode scanners work? - Explain that Stuff* [online]. [cit. 2015-05-13]. Dostupné z: http://www.explainthatstuff.com/barcodescanners.html

[16] MURPHY, Mark L. *Android 2: průvodce programováním mobilních aplikací*. Vyd. 1. Brno: Computer Press, 2011, 375 s. : il. ; 23 cm. ISBN 978-80-251-3194-7

Zoznam príloh

Príloha 1

Plagát

Príloha 2

Obsah CD

- /src/ - zdrojové súbory
    - /app/ - obsahuje zdrojové súbory aplikácie vo forme projektu Android Studia
    - /server/ - obsahuje zdrojové súbory PHP skriptov, komunikujúcich na serveri s databázou
- /poster/ - plagát vo formáte JPG
- /video/ - video vo formáte MP4
- /doc/ - tento dokument vo formáte DOCX a PDF
- /app/ - aplikácia vo forme inštalačného súboru APK

Príloha 3

Inštalácia a testovanie aplikácie

Na nainštalovanie aplikácie je potrebné mať smartfón s operačným systémom Android s verziou API 14 a vyššou. Pre správny chod aplikácie sa odporúča testovať s verziou API aspoň 16, keďže to bola najstaršia testovaná verzia. V prvom rade je potrebné si nakopírovať súbor nachádzajúci sa na priloženom CD v zložke /app/ do smartfónu. Potom už len stačí otvoriť tento súbor v telefóne, ktorý sám vyzve používateľa k inštalácii aplikácie. Na testovanie aplikácie sú priložené taktiež čiarové kódy produktov nachádzajúcich sa v databáze.