Razvoj android aplikacije

Kokot, Božidar

Sveučilište Jurja Dobrile u Puli

Fakultet ekonomije i turizma

«Dr. Mijo Mirković»

BOŽIDAR KOKOT

RAZVOJ ANDROID APLIKACIJE

Završni rad

Pula, 2015

Sveučilište Jurja Dobrile u Puli

Fakultet ekonomije i turizma

«Dr. Mijo Mirković»

BOŽIDAR KOKOT

RAZVOJ ANDROID APLIKACIJE

Završni rad

JMBAG: 0303038246, redoviti student

Studijski smjer: Informatika

Kolegij: Projektiranje informacijskih sustava

Mentor: Prof. dr. sc. Vanja Bevanda

Pula, rujan 2015.

IZJAVA O AKADEMSKOJ ČESTITOSTI

Ja, dolje potpisani _____, kandidat za prvostupnika _____ ovime izjavljujem da je ovaj Završni rad rezultat isključivo mojega vlastitog rada, da se temelji na mojim istraživanjima te da se oslanja na objavljenu literaturu kao što to pokazuju korištene bilješke i bibliografija. Izjavljujem da niti jedan dio Završnog rada nije napisan na nedozvoljen način, odnosno da je prepisan iz kojega necitiranog rada, te da ikoji dio rada krši bilo čija autorska prava. Izjavljujem, također, da nijedan dio rada nije iskorišten za koji drugi rad pri bilo kojoj drugoj visokoškolskoj, znanstvenoj ili radnoj ustanovi.

Student:

U Puli, 18. 09. 2015. _____

Sadržaj

Uvod

Mobilna tehnologija bilježi kontinuirani rast od svojih začetaka 1980-ih godina. U početku su mobiteli služili samo za slanje poruka tekstualnog sadržaja te za uspostavljanje poziva. Napretkom tehnologije i procesora; telefoni su proširili svoje funkcije pa su tako krajem 1990-ih, te ranih 2000-ih dobili mogućnost pohranjivanja multimedijalnog sadržaja. Pravi se pomak dogodio 2007. godine kada je kompanija Apple plasirala na tržište prvi pametni telefon iPhone. Ubrzo je svoju priliku vidio i Google te je plasirao operacijski sustav za pametne telefone Android. Danas je tržište pametnih telefona u stalnom rastu. Osnovna hipoteza rada je da mobilne aplikacije mogu značajno doprinijeti kvaliteti upravljanja pojedincu i organizacijama, te da je prilikom razvoja aplikacija potrebno poštivati osnovne koncepte planiranja i razvoja informacijskih sustava. U praktičnom dijelu prikazan je razvoj android aplikacije za pomoć upravljanju kućnim budžetom i preuzimanje informacija za kulinarske recepte. Rad je podijeljen na četiri dijela. U prvom dijelu razgrađen je koncept mobilnog poslovanja, u drugom je opisana arhitektura android sustava. Treći dio opisuje android komponente koje su potrebne za razvoj aplikacija dok četvrti prikazuje primjer razvoja android aplikacije.

1.Mobilno poslovanje

Mnoge kompanije, bile u začetku poslovanja ili već ustanovljene kompanije, nisu sigurne na koji način mogu ostvariti prihod putem svojih aplikacija. Tradicionalne su aplikacije imale nekoliko poslovnih modela. Prodavanje licenca i model pretplate bili su najpopularniji. Danas u mobilnom poslovanju postoje različiti poslovni modeli koji se mogu kombinirati i kojih se kompanije mogu pridržavati kako bi ostvarile profit. Ulaskom u 21. stoljeće raste utjecaj pokretnih (mobilnih) tehnologija te se koncept pokretnih tehnologija sve više uklapa u poslovanje organizacija pod terminom mobilno poslovanje ili poslovanje u pokretu. Australski znanstvenik Richi Nayak definira takvo poslovanje na sljedeći način: „Mobilno se poslovanje može definirati kao korištenje mobilnih tehnologija u razmjeni dobara, usluga,informacija i znanja.Mobilno poslovanje je izvršavanje transakcija obavljanih pomoću pokretne opreme, mobilnih mreža,bežičnih ili javnih.Mobilno poslovanje uključuje širok spektar poslovnih aktivnosti u okruženju poslovanja tvrtke s krajnjim korisnicima (B2C) i među tvrtkama (B2B).“ (Nayak 2010)

Sudeći po istraživanjima analitičke tvrtke Forrester (Lauren 2006) mobilne tehnologije moraju zadovoljiti određene uvjete kako bi se uklopile u koncept elektroničnog poslovanja:

- Prilagodba aplikacija elektroničnog poslovanja funkcionalnostima pokretnih uređaja odnosi se na funkcionalnosti i optimizaciju aplikacija na mobilne uređaje koji su znatno slabiji u usporedbi s tradicionalnim računalima.
- Stvaranje kulture pokretljivosti u poduzeću, koja predstavlja kulturu poslovanja putovanjem i širenja posla u različite krajeve.
- Ugradnja glasovnih aplikacija i usluga u portfelj elektroničkog poslovanja: kod standardne primjene elektroničkog poslovanja nije uobičajeno komuniciranje ljudskim glasom. Zahvaljujući mogućnostima pokretne tehnologije to se mijenja te pridonosi kvaliteti primjene elektroničkog poslovanja.
- Razvitak lokacijski zasnovanih i zavisnih aplikacija i usluga: iako se elektroničko poslovanje obavlja na internetu, usluge se uglavnom pružaju na konkretnim lokacijama te se korištenjem pokretnih tehnologija može kvalitetnije zadovoljiti lokacijski uvjetovana potražnja klijenta.
- Sigurnosni aspekt: kod električnog poslovanja smanjenje rizika u svrhu očuvanja sigurnosti podataka je izvor najvećih prijepora i u uvjetima uobičajenog električnog poslovanja, a tome pridonosi i komponenta pokretljivosti.

1. 1. Modeli mobilnog poslovanja

Modeli mobilnog poslovanja predstavljaju proces primjene pokretnih tehnologija u poslovne svrhe. U nastavku su opisani najkorišteniji modeli mobilnog poslovanja. Jedan od modela mobilnog poslovanja koji se na prvi pogled čini i najlogičnijim jest ***prodaja aplikacija***. Na taj se način mora izgraditi aplikacija koju veliki broj korisnika želi koristiti, ali za takav model mora se ulagati u marketing, naročito ako je kompanija nova i nema izgrađeno ime. Ovaj je pristup pogodan za organizacije koje su priznate u svom poslovanju.

Jedan od najkorištenijih modela danas je ***Freemium*** model (model besplatnih aplikacija). Freemium model obuhvaća mnogo različitih modela za ostvarivanje profita. Cilj mu je ponuditi aplikaciju besplatno kako bi se aplikacija mogla prenijeti što većem broju korisnika te jednom, kada je izgrađena zajednica korisnika, može se vršiti:

- Prodaja „PRO“ verzije aplikacije. U tom modelu korisnicima je na raspolaganju besplatna limitirana verzija. Nakon što su preuzeli aplikaciju, određeni postotak korisnika koji su preuzeli limitiranu verziju dodatno nadoplaćuju za naprednije funkcije aplikacije.
- Reklamiranje, odnosno prodaja oglasa unutar aplikacije. Takav pristup je jednostavniji, ali zahtijeva tisuće preuzimanja za ostvarenje profita.

Model kupovine unutar aplikacija – sličan je freemium modelu, ali za razliku od freemiuma gdje se samo jednom vrši plaćanje za prelazak na punu verziju aplikacije, ovaj model nudi kupovinu virtualnih dobara više od jednog puta. Ovakav model je pogodan za igrice kao što su „Candy Crush“, „FarmVille“ itd. koje ostvaruju profite putem freemium modela i modela kupovine raznih virtualnih dobara unutar same aplikacije.

Prodaja dodatnih paketa – ovakvim pristupom moguće je nuditi dodatni paket korisnicima koji imaju instaliranu aplikaciju, kako bi se proširila funkcionalnost aplikacije. Ovaj se pristup kombinira s reklamiranjem, odnosno oglašavanjem unutar aplikacija za određenu cijenu.

M-trgovina – model koji generira najviše prihoda u industriji mobilnih aplikacija. Ovo područje trenutno samo eksponencijalno raste te sve više korisnika prihvaća taj način kupovine. Za ovakav pristup važna su komponenta „backend procesi“. Mogu se prodavati proizvodi iz skladišta organizacije te se mogu koristiti i virtualna skladišta za prodavanje tuđih proizvoda.

Royalties (prodaja tantijema) – ovim modelom kompanije nude svoje aplikacije proizvođačima mobilnih uređaja kako bi ih instalirali na svoj uređaj prije prodaje. Pri svakoj

aktivaciji uređaja proizvođači isplaćuju dogovoreni iznos vlasnicima aplikacije. Ovakav model je dostupan samo poznatim tvrtkama te nije namijenjen kompanijama u začetku poslovanja.

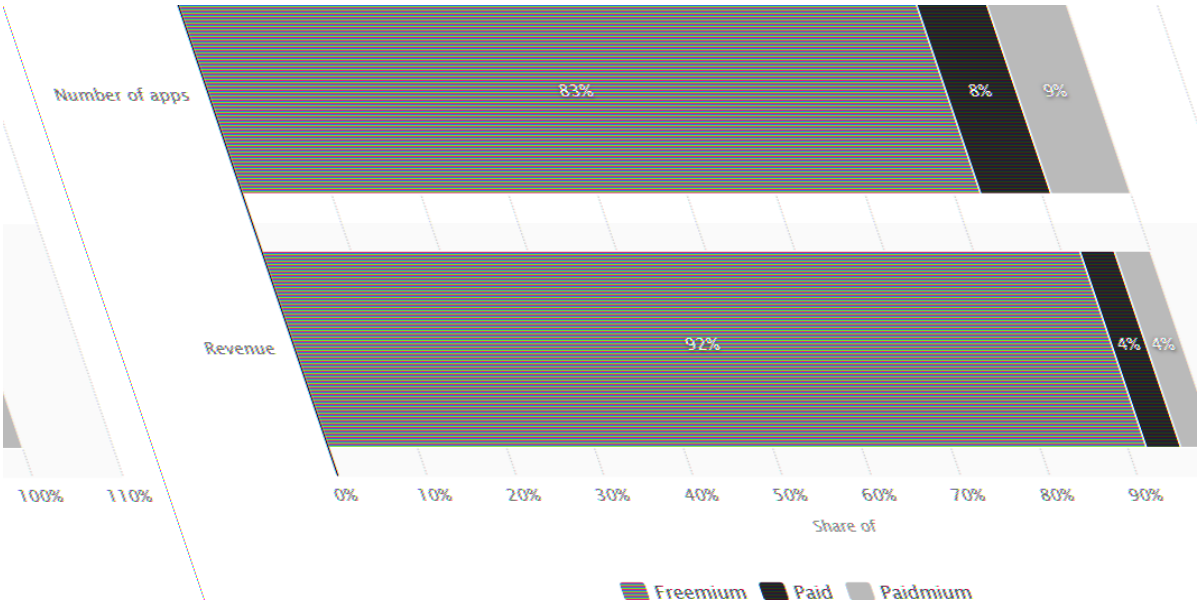
Izgradnja aplikacija za treću stranu – model koji je najsigurniji za programere jer je najmanje rizičan, budući da klijenti unaprijed plaćaju. Klijenti dostave dokumentaciju s osnovnim zahtjevima, na temelju kojih programeri razvijaju aplikaciju.

Pretplata – model koji je prikladan za aplikacije koje nude medijski sadržaj. Za određeni mjesečni ili godišnji iznos korisnici mogu koristiti sve medije koje organizacija nudi. Ovakav poslovni model zahtijeva veliki budžet ako organizacije žele nuditi svoj sadržaj, dok nudeći tuđi sadržaj organizacije zarađuju mali profit.

TPI (Trošak po instalaciji) – model u kojem organizacije mogu generirati profit za svaku aplikaciju koja je instalirana preko njihove aplikacije. Ovakve aplikacije ostvaruju veći profit od aplikacija koje nude oglašavanje. Ako aplikacija postigne veliki uspjeh, zainteresirane kompanije su spremne platiti veliki iznos novca za oglašavanje svojih aplikacija unutar aplikacija u vlasništvu organizacija. Zainteresirane kompanije koje razvijaju svoje aplikacije mogu izdvojiti veći iznos novca za oglašavanje na aplikacijama budući da unaprijed mogu izračunati koliki prihod mogu očekivati od reklamiranja.

Mnoge kompanije, koje su započele svoje poslovanje prije razvijenog društvenog umrežavanja i razvijenih mobilnih tehnologija, moraju prilagoditi svoje poslovanje današnjim dinamičkim uvjetima. Takve organizacije mogu iskoristiti mobilno poslovanje za:

- Unaprjeđenje postojećeg posla – takav pristup zahtijeva pronalazak usluge koja se već nudi a koja bi se mogla unaprijediti. Primjerice, aplikacija Bringify.net nudi mogućnost naručivanja hrane, suradnjom s takvim vrstama aplikacija mnoge organizacije mogu proširiti opseg posla s već postojećom uslugom.
- Proširenje korisnika – prednost mobilnog poslovanja jest i proširenje utjecaja proizvoda na korisnike koji ne bi koristili proizvode određene organizacije. (Panian 2010)



Slika 1 **Kompozicija trgovine aplikacijama po poslovnim modelima, izvor: http://www.statista.com/statistics/293636/app-store-composition-business-models/, pristupljeno: 01. 09. 2015.**

Na *Slici 1* prikazana je kompozicija trgovine aplikacijama po poslovnim modelima. 83% aplikacija na vrhu rang liste koriste freemium model i ostvaruju 92% poslovnih prihoda.

Pri mobilnom poslovanju posebice su važan segment društvene mreže (Panian 2010). Organizacije mogu uživati u mnogim blagodatima korištenja socijalnih medija (primjerice mogu izgraditi odnos između klijenata i zaposlenika tvrtke) za stvaranje novih dimenzija pružanja usluga klijentima, te za praćenje konkurencije na tržištu.

Kako društvene mreže postaju sve više implementirane u moderno društvo i način na koji se komunikacija vrši, tako se i mnoge organizacije odlučuju na integriranje svoje marketinške strategije s društvenim mrežama. Ulazeći u takvu vrstu odnosa moraju se uzeti u obzir čimbenici koji su se pojavili sa suvremenom tehnologijom koja je izmijenila odnos tvrtke i njenih klijenata te donijela novu poslovnu filozofiju. Kod razumijevanja društvenih medija trebaju se uzeti u obzir njezine osnovne karakteristike:

- Otvorena konverzacija – kod tradicionalnih medija, kao što su televizija i novine, sva se komunikacija odvija isključivo jednosmjerno, od pošiljatelja do velikih masa koje primaju informaciju. Kod socijalnih medija uloga se promijenila, pa se tako sada komunikacija odvija dvosmjerno, te organizacije u relativno kratkom vremenskom roku dobivaju veliku količinu povratnih informacija koje ne bi mogle dobiti tradicionalnim načinima.
- Zasnovanost na zajednici – svakoj je organizaciji u cilju stvoriti što veću zajednicu koja će dijeliti zajedničke interese i koja će biti sklona suradnji, a društvene mreže su idealan alat za osnivanje takvih zajednica.
- Sudjelovanje – jednom kad je osnovana zajednica, njezini članovi su spremni dati vlastiti doprinos i dijeliti učinak.
- Povezanost – društvene mreže dopuštaju lagan i brz protok informacija velikom broju ljudi putem dijeljenja Internet poveznica.

Dolaskom društvenih mreža mijenjaju se i odnosi zaposlenika, pa tako i oni očekuju nesmetanu komunikaciju i pristup aplikacijama koje im pomažu da budu što efikasniji u svom poslu. Korelacija mobilne tehnologije i socijalnih medija posebno dobiva na značaju kada se uzme u obzir utjecaj socijalnih medija na tradicionalne djelatnosti. Kako tvrdi McKinsey & Company „Usmena predaja je primarni faktor kod donošenja odluka o nabavi u poduzećima u 20 do 50% slučajeva. Utjecaj usmene predaje je najveći kada potrošač kupuje nešto po prvi puta ili kada su proizvodi razmjerno skupi, kada to iziskuje više istraživanja, vremena, razmatranja više opcija i konzultiranja tuđih mišljenja. A najzanimljivije je to što usmena predaja sve više postaje javno dobro, a sve je manje posljedicom komunikacije u uskom krugu ljudi. Danas digitalna usmena predaja počinje funkcionirati prema načelu jedan prema više ili puno jer se recenzije proizvoda postavljaju kao blogovi, a mišljenja se razmjenjuju trenutnom razmjenom poruka u društvenim mrežama. Sve se češće pojavljuju Web mjesta čija je jedina svrha hvaliti ili kuditi brandove.“ (Bughi 2010).

2. Android operacijski sustav

Android je open-source operacijski sustav građen s Linux jezgrom. Android Inc. osnovali su Andy Rubin i Rick Miner 2003. godine kako bi razvijali programe za pametne uređaje. Nakon dvije godine otkupio ih je Google koji je predstavio Android operacijski sustav za pametne telefone 2007 godine. Android pokreće cijelu paletu uređaja od pametnih

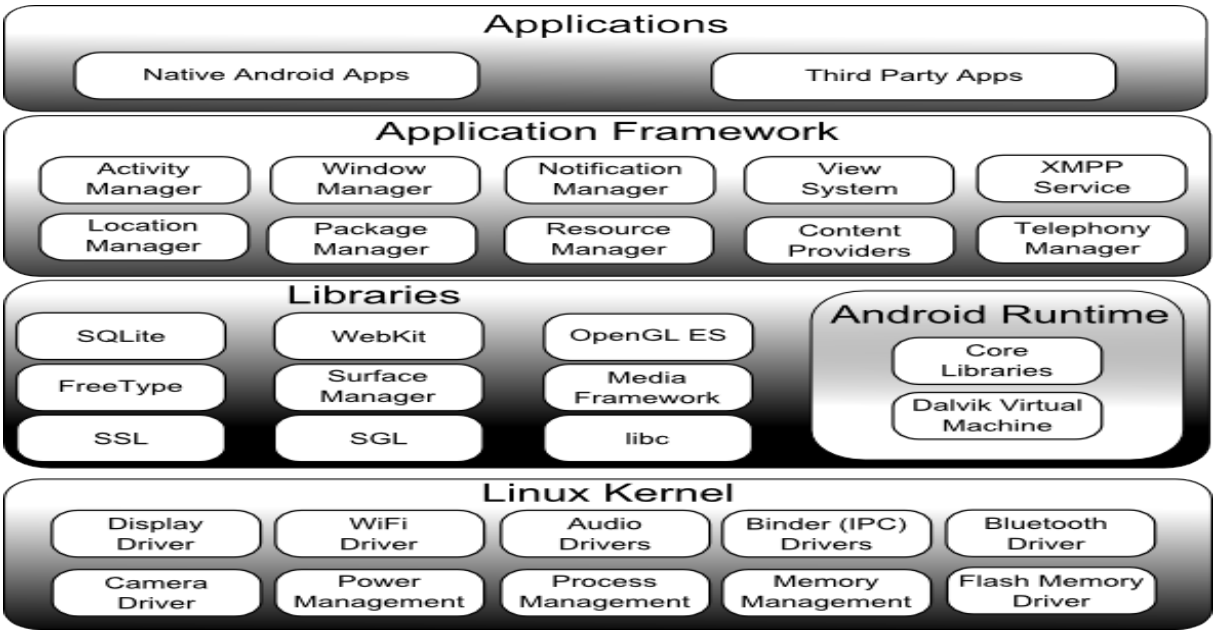
telefona do tableta, te po istraživanjima IDC (International Data Corporation) 82.8% iznosi udio Androida u tržištu pametnih telefona. (IDC 2015)

Android predstavlja glavnu alternativu iOS platformi za mobilne aplikacije, ali za razliku od iOS-a android projekti se mogu razvijati na različitim operacijskim sustavima od OS X, Windowsa do Linuxa. Android aplikacije se razvijaju u Java programskom jeziku, a to mnogim programerima koji dolaze primjerice iz C# pozadine predstavlja lakši prijelaz na novu tehnologiju.

2.1. Komponente arhitekture android sustava

Android ima arhitekturu u formi stoga (stack) kompresiranih aplikacija, operacijskog sustava, vremena izvršenja, middlewarea, usluga i biblioteka. Svaki sloj u stogu odgovara elementima unutar svakog sloja te su elementi usko povezani kako bi pružili optimalno aplikacijsko okruženje za mobilne uređaje.

Kao što je prikazano na *Slici* 2, na dnu android arhitekture nalazi se Linux jezgra. Linux jezgra pruža određeni nivo apstrakcije između hardwarea samog uređaja i gornjih slojeva stoga. Android jezgra bazirana je na Linux verziji 2.6, koja pruža multitasking, sistemske usluge kao što su memorija, procesi te pruža mrežni sloj zajedno s driverima za hardware, kao što su display uređaja, Wi-Fi te audio.



Slika 2 **Arhitektura android sustava, izvor: http://www.techotopia.com/index.php/An_Overview_of_the_Android_Architecture#The_Android_Software_Stack, pristupljeno: 01. 09. 2015.**

Na *Slici 2* prikazana je arhitektura android operacijskog sustava posloženog po stogovima. Android koristi Linux jezgru, međutim sam Linux OS prvotno je namijenjen za upotrebu na kompjuterima u formi osobnih računala te servera. Android aplikacija izvodi se kao proces direktno na Linux jezgri, kroz svoju instancu na Dalvik virtual machine (VM). Izvođenje aplikacija na Dalvik VM-u pruža mnogo prednosti, primjerice aplikacije se ne mogu umiješati u rad operacijskog sustava ili drugih aplikacija niti mogu direktno pristupiti hardwareu. Google je razvio Dalvik VM koji je efikasniji od Java VM-a u pogledu korištenja memorije te su napravljene na način da dopuštaju izvođenje više instanci unutar resursa mobilnog uređaja. Kako bi pokrenuli Dalvik VM, aplikacijski kod mora biti transformiran iz standardne Java datoteke u Dalvikov.dex format, koji ostavlja 50% manji memorijski otisak od Java bytecodea. Dalvik je odnedavno zamijenjen ART sustavom. Razlika između Dalvik VM-a i ART-a je u kompiliranju. Dalvik VM je koristio JIT način kompiliranja koji je kod

prevodio pri izvođenju i korisnikovim napredovanjem kroz aplikaciju dodatni kod se prevodi u strojni jezik. ART sustav se kompilira pri instaliranju tako da sprječava zastoj programa.

Android programske biblioteke predstavljaju Java klase spremljene u pakete koje programeri mogu koristiti ako se za to pojavi potreba. Ključne android biblioteke dostupne programerima su:

- android.app – pruža uvid u aplikacijski model te je osnova za sve android aplikacije
- android.content – olakšava pristup komponentama
- android.database – koristi se za pristup podatcima te uključuje SQLite baze podataka
- android.graphics – pruža 2D grafičke API biblioteke koje uključuju boje, filtere itd.
- android.hardware – pruža pristup hardware komponentama, kao što su svjetlosni senzor i dr.
- android.opengl – Java sučelje za OpenGL (grafički 3D standard)
- android.os – pruža pristup standardnim operacijskim sistemskim uslugama uključujući poruke
- android.media – pruža programske klase koje omogućuju reproduciranje audio i video sadržaja
- android.net – pruža uvid u mrežni sloj
- android.text – koristi se za manipuliranje prikaza tekstualnog sadržaja
- android.util – pruža izvršenje zadataka kao što su pretvorba brojeva, te manipulacije datumom i vremenom
- android.view – fundamentalna biblioteka za upravljanje korisničkim sučeljem
- android.widget – komponente za korisničko sučelje kao što su gumbi, lista vrijednosti itd.
- android.webkit – namijenjen za pregledavanje web sadržaja

Application framework predstavlja programsko okruženje u kojemu se android aplikacije razvijaju i izvode. Koncept application frameworka je izgradnja aplikacija iz komponenta koje se mogu opet upotrijebiti. Ovakav koncept omogućava da se aplikacija sa svim svojim svojstvima može opet upotrijebiti od strane druge aplikacije. Application framework uključuje sljedeće ključne usluge:

- Activity manager – kontrolira sve aspekte aplikacijskog tijeka
- Content Providers – dozvoljava aplikacijama da dijele podatke s drugim aplikacijama
- Resource manager – pruža pristup aplikacijskim resursima kao što su komponente korisničkog sučelja

- Notification manager – dozvoljava aplikacijama da prikažu upozorenja korisnicima
- View System – služi za kreiranje korisničkog sučelja
- Package Manager – dozvoljava aplikacijama da pronađu informacije o drugim aplikacijama trenutno instaliranim na uređaju
- Telephone Manager – informacije o telefonskim uslugama na uređaju
- Location Manager – pruža aplikacijama mogućnost da ažuriraju lokaciju

Na vrhu arhitekture android sustava nalaze se aplikacije, koje se sastoje od nativnih (ugrađenih) aplikacija koje dolaze s uređajem (e-mail,web browser) te aplikacija instaliranih nakon kupovine uređaja. Prije početka razvoja aplikacije potrebno je napraviti prigodno okruženje za programiranje, a za svako razvijanje android aplikacija potrebno je:

- Text editor za pisanje koda
- Android framework za povezivanje napisanog koda
- Android alati za kompiliranje koda
- Emulator za testiranje aplikacije

3. Osnovna obilježja i komponente u razvoju

Android aplikacije se razvijaju u Java programskom jeziku, te se kompilirani Java kod zajedno s podatcima i resursima aplikacije sažima u paket podataka, koji se arhiviraju pod .apk sufiksom. Ovako arhivirana datoteka je distributor aplikacije za instaliranje na mobilni uređaj. Jedno od obilježja androida jest da jedna aplikacija može koristiti elemente druge aplikacije, uz uvjet da imaju dopuštenje. Kako bi to funkcioniralo, sustav mora biti u mogućnosti pokrenuti aplikacijski proces kad je bilo koji aplikacijski dio potreban, te da pozove Java objekt za potreban dio, stoga android aplikacije nemaju jedan ulaz za sve komponente u aplikaciji, nego koriste osnovne komponente koje sustav može pozvati kad su potrebne.

3.1. Eclipse i Android studio

Eclipse je integrirano programsko okruženje razvijeno u Java programskom jeziku. Sadrži osnovni radni prostor (workspace) te priključke (plug-in sistem) za prilagođavanje programskog okruženja. Eclipse ima svoj programski razvojni alat (SDK), koji sadrži posebne Java alate namijenjene za Java programere. Android Development Tools (ADT) je priključak za Eclipse, koji je namijenjen dostavljanju integriranog programskog okruženja za razvijanje android aplikacija. ADT proširuje mogućnosti Eclipsea te dopušta programerima izradu

novog android projekta, aplikacijskog korisničkog sučelja i spremnje aplikacije u .apk ekstenziju, kako bi je mogli dalje distribuirati. ADT je bio službeni IDE (integrirano programsko okruženje) za android, ali ga je zamijenio Android studio.

Android studio je integrirano programsko okruženje (IDE) za razvijanje android aplikacija. Najavljen je u 2013. godini na Google I/O konferenciji, a prva stabilna verzija je izašla u prosincu 2014. Android studio se bazira na JetBrains softveru, te je namijenjen isključivo razvijanju android aplikacija. Android studio je zamijenio Eclipse kao Googleovo primarno razvojno okruženje. Android studio ima zaseban „build system“ koji predstavlja skup alata koji se koriste za izgradnju, testiranje i pokretanje aplikacija. Ovaj „build system“ je zamijenio Ant system koji se koristio kod Eclipse ADT-a kao primarni „build system“.

AVD Manager unutar Android studia predstavlja alat za upravljanje virtualnim uređajima koji služe za pokretanje aplikacija. AVD Manager dozvoljava korisnicima da sami izgrade svoj virtualni uređaj, koji mogu prilagoditi svojim potrebama. Android studio to postiže instalirajući Intel x86 HAXM koji služi za stvaranje emulatora (virtualnog uređaja na našem uređaju) pomoću kojega se mogu isprobavati i testirati aplikacije. Android studio nudi i mogućnost praćenja potrošnje memorije, pomoću kojeg se može otkriti gdje je određeni objekt smješten za vrijeme izvođenja aplikacije. Takva saznanja pružaju mogućnost bolje prilagodbe kroz pravilno pozivanje metoda kako bi optimizirali aplikacije i potrošnju memorije.

3.2. Android manifest

Svaka android aplikacija mora imati Android manifest datoteku koja je napisana u XML. programskom jeziku. Android manifest sadrži ključne informacije o aplikaciji koje operacijski sustav mora imati prije pokretanja aplikacije. U svakoj manifest datoteci potrebno je dodijeliti ime Java paketu u koji će se spremati aplikacija i koji služi kao jedinstveni identifikator. Manifest datoteka također opisuje svaku komponentu aplikacije (aktivnosti, services, broadcast receivers i content providers). U manifest datoteci potrebno je dodijeliti akciju svakoj aktivnosti u aplikaciji, kako bi sustav znao koju komponentu treba pokrenuti. Ako je kod razvoja aplikacije potrebno koristiti vanjske procese, kao što su spajanje na internet ili bluetooth, to je također potrebno definirati u android manifestu.

Android manifest ima određena pravila kojih se potrebno pridržavati, primjerice jedina dva elementa koja su obavezna i koji se mogu samo jednom pojaviti su „<manifest>“ i „<application>“.Većina drugih elemenata se može pojaviti i više puta. Element u android manifest datoteci može sadržavati samo drugi element. Elementi koji su na istom rangu unutar

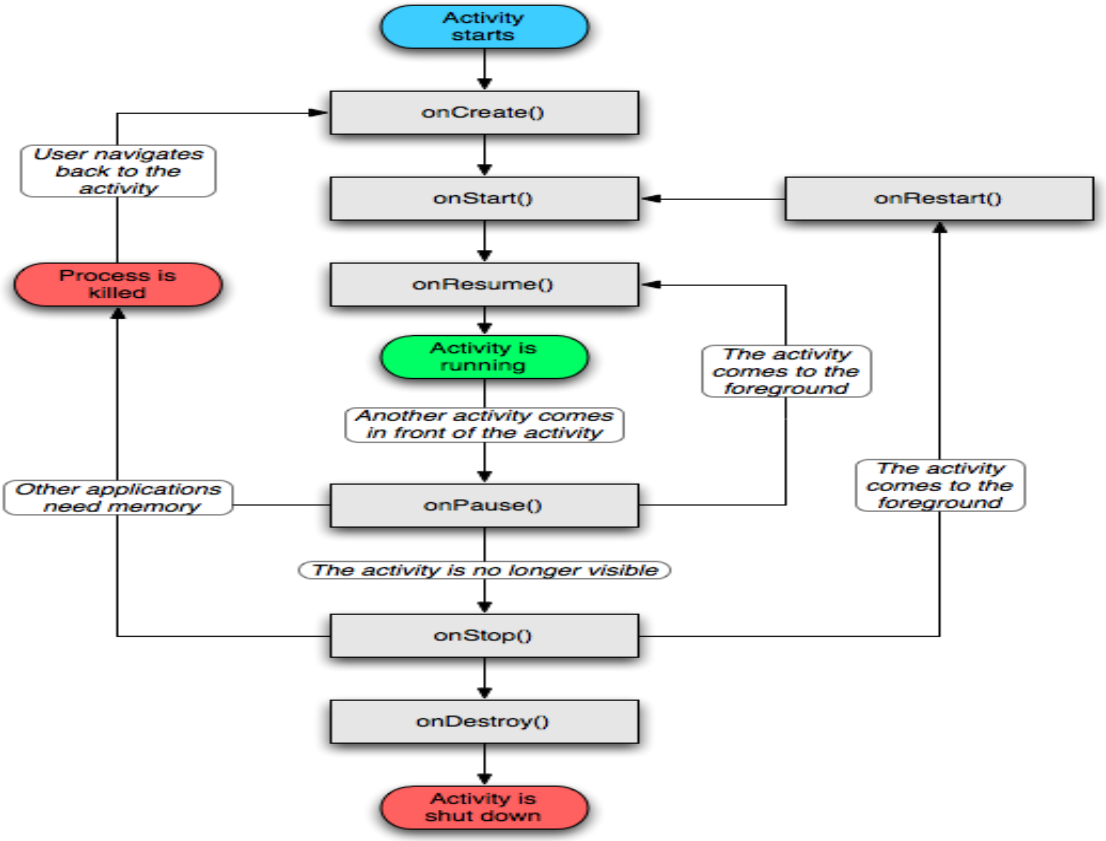
manifest datoteke uglavnom nisu složeni po nekom posebnom redu, pa tako primjerice elementi „<activity>“,“<provider>“,“<service>“ mogu biti poredani neovisno o redu. U android manifestu svi atributi nisu obavezni, ali određeni atributi moraju biti deklarirani kako bi element mogao funkcionirati. Izuzev atributa u <manifest> elementu svi drugi atributi moraju započeti s prefiksom „android:“. Neki atributi sadrže vrijednosti koje mogu biti prikazane korisnicima, primjerice ikonice aplikacije ili naziv aktivnosti. Vrijednosti ovakvih atributa moraju biti spremljene u određene izvorišne datoteke koje se prikazuju u formatu „@[package:]type:name“, gdje „package“ može biti izuzet, ako se nalazi u istom programskom paketu s aplikacijom, „type“ predstavlja tip resursa, primjerice string, “name“ predstavlja identifikator određenog resursa.

Ako aplikacija treba pristupiti određenim funkcionalnostima to se mora posebice naglasiti u android manifest-u pomoću „<uses-permission>“ elementa. Jednom kada je aplikacija instalirana na uređaju, korisnik mora odlučiti hoće li dozvoliti pristup određenoj funkcionalnosti, primjerice hoće li dozvoliti pristup za spajanje na internet.

Svaka aplikacija je spojena na opću knjižnicu android sustava, koja uključuje osnovne pakete za izgradnju aplikacija, međutim postoje i paketi koji su smješteni u svoju knjižnicu. Ako aplikacija zahtjeva korištenje takvih paketa potrebno je unutar android manifest datoteke posebno deklarirati element „<uses-library>“ s imenom paketa kojem želimo pristupiti.

3. 3. Komponente android aplikacije

Aktivnost predstavlja vizualno korisničko sučelje. Aplikacija se može sastojati od jedne ili više aktivnosti, a svaka android aktivnost mora biti definirana u android manifest datoteci. Unutar manifest datoteke aktivnost se definira, pridodaje joj se određena oznaka te akcija. Ako je akcija, unutar manifest datoteke, definirana kao „Launcher“ to znači da je to početna aktivnost te predstavlja ulaz u program; ako je definirana kao „Default“ mora biti pozvana da bi se pokrenula. Svaka android aktivnost također mora imati i svoju korenspondirajuću „layout“ datoteku, koja mora biti napisana u XML. programskom jeziku. Layout datoteka predstavlja korisničko sučelje aktivnosti, u njoj se mogu definirati widgeti i ostale komponente bitne za razvoj aplikacije. XML. datoteka sadrži određenu hijerarhiju u kojoj su komponente pohranjene, primjerice može se koristiti „Relative Layout“, kojem se mora odrediti širina i visina. Kako bi se layout XML datoteka povezala s Java kodom korenspondirajuće aktivnosti potrebno je unutar aktivnosti pozvati „setContentView“ metodu u koju se kao parametar postavlja XML datoteka.



Slika 3 **Životni ciklus android aktivnosti, izvor: http://www.skill-guru.com/blog/2011/01/13/android-activity-life-cycle/, datum pristupa: 01. 09. 2015.**

Na *Slici 3* prikazan je životni ciklus jedne android aktivnosti. Svaka aktivnost mora imati svoj tok i vijek trajanja, kao što je prikazano na gornjem dijagramu. Aktivnost se razlikuje od obične Java klase u tome što proširuje klasu „Activity“ te kao takva mora implementirati „onCreate()“ metodu. Ta metoda predstavlja stvaranje aktivnosti, unutar te metode definira se povezanost varijabli iz Java koda sa XML. datotekom. Aktivnost se može

naći u više stanja, kao što se može vidjeti na dijagramu, može biti u stanju izvođenja, može biti vidljiva korisniku ali nije u upotrebi te može biti u zaustavljenom stanju.

Service komponenta služi za korištenje komponenti vanjskih aplikacija unutar aplikacije koja se trenutno razvija. Service nema korisničko sučelje, nego se izvodi u pozadini. Primjerice, može pokretati pozadinsku muziku ili može dohvatiti podatke s interneta te dostaviti rezultat aktivnosti kojoj su ti rezultati potrebni. Kod servicea komponente se mogu vezati za service te na taj način upravljati internet transakcijama, reproducirati multimedijalni sadržaj ili komunicirati s content provider komponentom. Service se može pojaviti u dva oblika:

- Service – komponenta je započeta kada aplikacijska komponenta pozove service pomoću „startService()“ metode. Jednom kada je pozvana, komponenta se izvodi u pozadini čak i ako je komponenta koja je pozvala service uništena. Kada je operacija završena, service komponenta se sama zaustavlja.
- Bound – komponenta postaje vezana (bound) kada komponenta pozove metodu „bindService()“. Bound service nudi klijent – server sučelje koje dozvoljava komponentama da komuniciraju sa serviceom, šaljući određene zahtjeve, vraćajući rezultate te dozvoljavaju međuprocesnu komunikaciju. Bound service je aktivna samo dok je druga aplikacijska komponenta vezana za nju.

Service može funkcionirati na oba načina, ako se implementira nekoliko metoda, primjerice „onStartCommand()“ dozvoljava komponentama da se pokrenu a „onBind()“ metoda dozvoljava vezivanje. Neovisno o tome je li aplikacija pokrenuta ili vezana, svaka aplikacijska komponenta može pokrenuti service na isti način kako koristi i aktivnosti – pozivanjem intent komponente. Prije pozivanja intenta u manifest datoteci service se mora deklarirati kao privatna komponenta, kako bi se spriječilo uplitanje drugih aplikacija.

Broadcast receiver predstavlja android komponentu koja je zaslužna za uspješno identificiranje promjene u uređaju, primjerice primanja sms poruka, statusa baterije ili statusa Wi-Fi-a (spajanja na Internet). Android OS, kao i druge aplikacije, koriste poruke s Broadcast Receivera za primanje obavijesti o eventualnim promjenama na uređaju koje se onda mogu programirati.

Dvije su osnovne klase broadcast poruka koje se mogu primiti:

- Normalni broadcast – poruke su u potpunosti asinkronizirane, svi primatelji broadcast poruka se izvode u neodređenom redoslijedu, često i u isto vrijeme. Ovakav način je

efikasniji, ali primatelji poruka ne mogu vidjeti rezultate ili prekinuti API koji je uključen u slanje.

- Naručeni broadcast – poruke se dostavljaju jednom po jednom primatelju. Kada primatelju poruke istekne vrijeme, poruka se može poslati sljedećem primatelju ili se može u potpunosti otkazati, kako sljedeći primatelj ne bi primio poruku. Redoslijed primatelja poruke se može kontrolirati kroz atribut „android:priority“ unutar manifest datoteke.

Broadcast poruke se šalju pomoću Intent komponente, no bitno je napomenuti da je kod broadcast poruka intent komponenta znatno drugačija od uobičajenog intenta, koji se koristi za pozivanje aktivnosti jer ne postoji način da broadcast komponenta vidi intent kao kod „startActivity()“ metode koja poziva aktivnosti. Pokretanje aktivnosti je intent operacija koja mijenja stanje s kojim korisnik trenutno komunicira, dok je intent operacija kod broadcast-a pozadinska operacija.

Content provider upravlja pristupu podatcima pružajući svoje vlastito korisničko sučelje za upravljanje podatcima. Glavni zadatak content providera jest da se koristi od strane drugih aplikacija. Content provider dostavlja podatke drugim aplikacijama u obliku tablice. Redak u toj tablici predstavlja instancu tipa podatka, a svaki stupac predstavlja određeni set podataka prikupljen za određenu instancu. Android sustav uključuje content provider koji upravlja podatcima različitih formata, kao što su audio i video formati, te slike. Za primanje podataka iz providera, u manifest datoteci se mora posebno navesti element „<uses-permission>“ uz atribut koji se naziva jednako kako je definirano u provideru.

Komponenta zaslužna za pozivanje drugih aktivnosti ili komponenata zove se ***Intent***. Intent komponenta ima tri osnovna scenarija upotrebe:

- Pozivanje druge aktivnosti – Intent se može koristiti za pozivanje druge aktivnosti koristeći metodu „startActivity()“, a kao parametar se koristi intent varijabla koja je prethodno definirana. Pri definiranju intent varijable, potrebno je specificirati kontekst iz kojega pozivamo intent te aktivnost koja se poziva. U intent varijablu moguće je spremiti varijable koje su korištene u toj aktivnosti koristeći „putExtra()“ metodu.
- Pozivanje servicea – Intent se može koristiti i za pozivanje određenih service komponenti, kao npr. preuzimanje datoteke slanjem intent varijable u startService() metodu.

- Dostavljanje broadcast poruke – moguće je dostavljanje broadcast poruke drugoj aplikaciji koristeći intent tako da se proslijedi intent varijabla kao parametar u „sentBroadcast()“ metodi.

Intent filter predstavlja unos iz android manifest datoteke koji određuje tip Intenta koji određena komponenta želi primiti. Postoje dvije osnovne vrste intent komponente:

- Eksplicitni – eksplicitne vrste pozivaju komponentu po imenu, uglavnom se koristi kod pozivanja komponenti u vlastitoj aplikaciji, kada je poznato ime klase na koju se poziva.
- Implicitni – implicitna vrsta ne poziva određenu komponentu nego poziva neku određenu akciju koju će izvršiti. Takav pristup dozvoljava komponentama iz drugih aplikacija da ga koriste, kada se izradi implicitni intent, sustav pronalazi komponentu tako što uspoređuje intent s intent filterom, deklariranim u android manifest datoteci, u slučaju da intent odgovara intent filteru, sistem pokreće određenu komponentu te ju prosljeđuje u Intent.

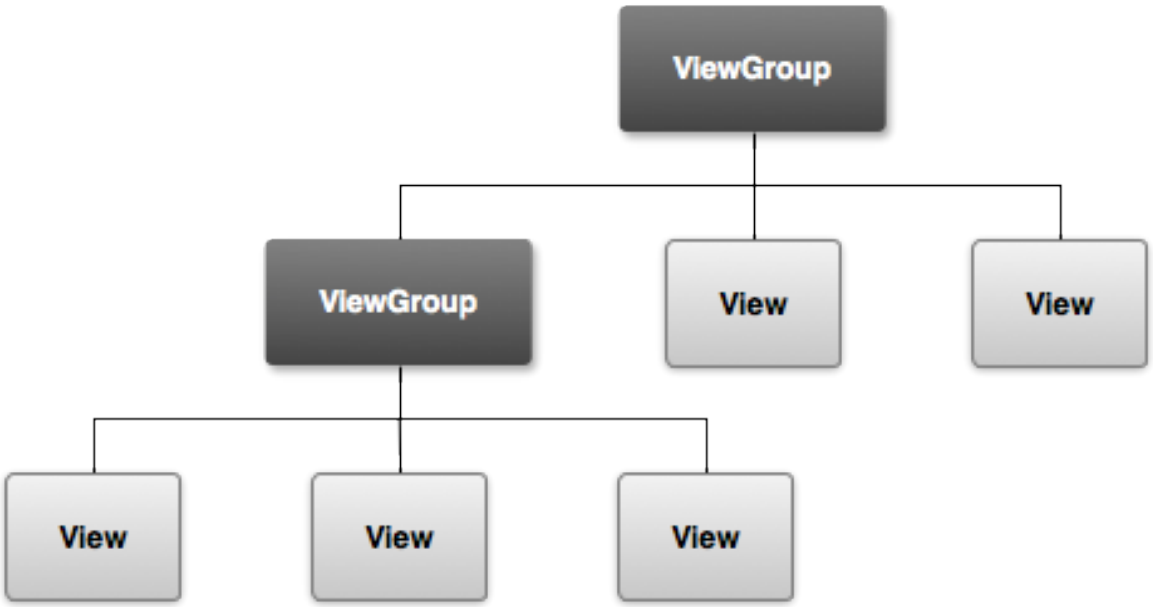
3. 4. Korisničko sučelje

Kao što je napomenuto, svaka aktivnost mora imati svoje korisničko sučelje koje se izgrađuje u XML kodu. Svaka komponenta u aplikacijama je podijeljena u hijerarhiji kroz View i ViewGroup; svaki ViewGroup je nevidljivi spremnik unutar kojeg se organiziraju i smještaju komponente Viewa. View može biti gumb ili bilo koja druga komponenta koja se prikazuje na korisničkom sučelju. Hijerarhijsko stablo može biti i jednostavno i složeno, ovisno kako su posloženi elementi. Vizualna struktura korisničkog sučelja definira se kroz „Layout“ komponentu koja se može deklarirati na dva načina:

- Deklariranje u XML kodu – pisanje XML koda koji odgovara pripadajućoj View klasi
- Instanciranje layout elemenata za vrijeme izvođenja – aplikacija automatski stvara View i View Group objekte

Prednost pisanja korisničkog sučelja u XML kodu je pružanje mogućnosti bolje podjele prostora unutar samog korisničkog sučelja. Takav opis korisničkog sučelja je odvojen od koda same aplikacije, što znači da se može mijenjati i prilagođavati izgled bez mijenjanja koda aplikacije i kompiliranja. Primjerice, može se napraviti XML layout koji odgovara različitim rezolucijama ekrana i različitim jezicima. Također, pisanje layout-a u XML kodu omogućava lakše vizualiziranje strukture korisničkog sučelja, što omogućava lakše pronalaženje grešaka u kodu. XML kod deklariranja elemenata korisničkog sučelja prati strukturu i nazive određenih

klasa i metoda, gdje ime elementa odgovara imenu klase i ime atributa odgovara imenu metode.



Slika 4 **Hijearhija korisničkog sučelja, izvor: https://developer.android.com/intl/zh-cn/guide/topics/ui/index.html, pristupljeno: 01. 09. 2015.**

Na *Slici 4* prikazana je hijerarhija korisničkog sučelja. Svaki ViewGroup služi kao određeni spremnik View komponenata koje mogu biti polja za unos podataka ili druge komponente koje se slažu na korisničko sučelje. Hijerarhijsko stablo može biti kompleksno ili jednostavno, ovisno o situaciji.

3. 5. Vanjski alati

PHP (PHP, HyperText Processor) je skriptni jezik koji se izvodi na strani servera, specijaliziran prvenstveno za razvijanje dinamičkih web aplikacija. Program koji se napiše u PHP-u ne zahtijeva prevođenje, nego se interpretira pri svakom izvršavanju. PHP interpretator može raditi po PHP CGI principu, odnosno tako što će interpretator postojati kao vanjska aplikacija koja se poziva da izvrši određenu skriptu svaki put kada bude pozvana od strane

korisnika, te može služiti i kao modul web-servisa. Danas je u najvećoj upotrebi kao modul web-servisa jer pruža znatno veću brzinu izvršavanja budući da je interpretator uvijek učitan u memoriju te se ne mora pozivati vanjska aplikacija. Scenarij izvođenja PHP skripte je sljedeći:

- Klijent zahtjeva PHP stranicu sa servera
- Server prosljeđuje zahtjev servisu za web
- Web-server prepoznaje da se zahtjeva PHP datoteka
- Web-server izvršava program pomoću PHP modula
- Tekst programa se šalje klijentu kao rezultat zahtjeva
- Klijent prepoznaje vrstu rezultata

PHP, nalik na većinu skriptnih jezika, ne sadrži početnu metodu nego jednostavno sadrži skup naredbi koje se izvršavaju jedna za drugom. Posljednja naredba predstavlja i kraj PHP programa.

SQLite je ugrađen sistem za upravljanje bazama podataka, sadržan u relativno maloj programskoj biblioteci. Za razliku od klijent–server sustava za upravljanje bazama podataka, SQLite nije samostalan proces s kojim aplikacija komunicira. Umjesto toga, SQLite biblioteka je sastavni dio aplikacije. Aplikacija koristi SQLite funkcionalnosti kroz pozivanja u samom kodu. U SQLite-u kompletna baza podataka je spremljena na uređaju. Danas je SQLite ugrađen u sve veći broj programa, tako primjerice Android operacijski sustav u sebi sadrži SQLite biblioteke. Zahvaljujući svom dizajnu koji ne uključuje server, SQLite aplikacije zahtijevaju manje konfiguracije nego klijent-server baze. Upravo zbog toga je SQLite i zaradio nadimak „zero-conf“ (bez-konfiguracije) jer ne zahtjeva pristup koji se bazira na lozinkama, umjesto toga kontrola pristupa se rješava u samoj datoteci baze podataka.

Loša strana dizajna bez servera jest to da može doći do situacije gdje se nekoliko procesa pokuša služiti pisanjem u datoteku baze podataka. SQLite se zahvaljujući svojom „server-less“ (dizajnu bez servera) dizajnu mora oslanjati na „file-system“ zaključavanje. Takav pristup ima manje znanja o drugim procesima koji pristupaju bazi podataka u isto vrijeme. Upravo zbog toga SQLite nije pogodan za operacije s intenzivnim zapisivanjem u bazu podataka. No, za jednostavne upite, SQLite pruža bolje performanse jer ne treba slati svoje podatke drugom procesu.

JSON (JavaScript Object Notation) je format otvorenog standarda koji se koristi za prijenos objekata složenih u atribut–vrijednost paru. To je primarni format koji se koristi za

sinkroniziranu komunikaciju između servera i preglednika. Iako je originalno nastao iz JavaScript jezika, JSON je format koji je neovisan te se može koristiti u više programskih jezika. JSON podržava samo nekoliko osnovnih tipova podataka:

- Broj – JSON ne dopušta takozvanu Nan (non-numbers) vrijednost niti pravi razliku između integer tipa podatka ili floating tipa
- String – predstavlja niz tekstualnih znakova
- Boolean – može se pojaviti u dvije vrijednosti, true (istina) ili false (neistina)
- Array – array ili lista predstavlja složenu listu podataka od kojih svaki član u listi može biti različitog tipa. U JSON-u array koristi uglate zagrade u kojima su elementi razdvojeni znakom zareza.
- Object – predstavlja skup podataka složenih kao ključ/vrijednost, gdje ključ mora biti tipa string. Budući da su objekti namijenjeni predstavljanju asocirane liste podataka, svaki ključ u objektu bi trebao biti jedinstven.
- Null – prazna vrijednost.

U nastavku završnoga rada razrađen je primjer izrade android aplikacije.

4. Primjer razvoja aplikacije

Na početku razvoja aplikacije okruženje za razvoj bio je Eclipse, jer je on imao službeni Google ADT plugin (dodatak) za razvijanje android aplikacija. No, u prosincu 2014. godine izašao je Android studio, koji je zamijenio Eclipse kao službeno razvojno okruženje za android aplikacije, pa se razvoj aplikacije nastavio na Android studio platformi. Android studio je bolji za razvoj android aplikacija jer mu je to primarni fokus, dok je kod Eclipsea platforma za razvijanje aplikacija plugin sistem (dodatak). Još jedna bitna razlika jest da Eclipse koristi plugin (dodatak) za dodavanje vanjskih paketa, dok Android studio koristi Mave dependencies koji sinkronizira pakete u projekt.

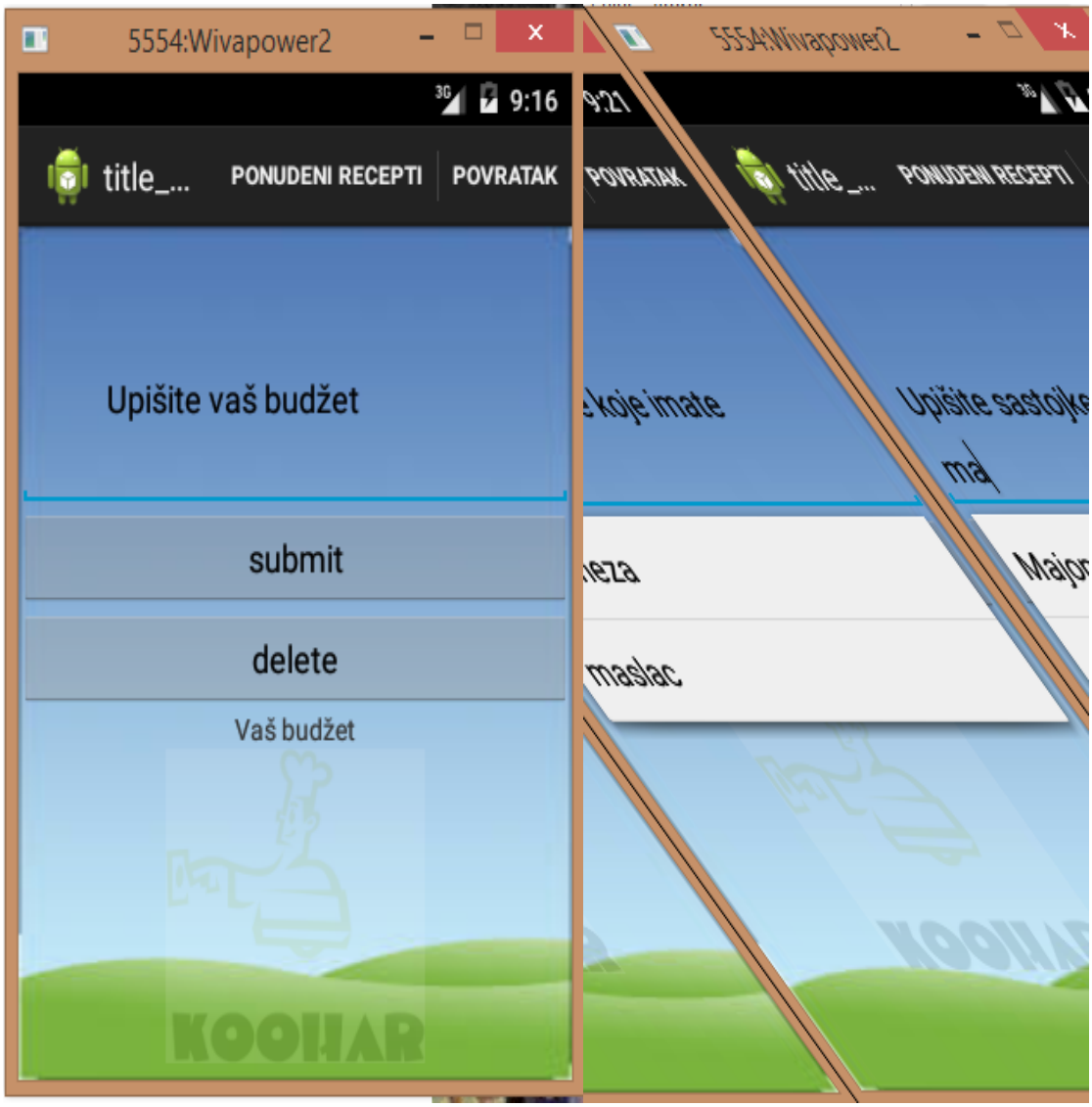
4. 1. Opis aplikacije

Razvoj android aplikacije naziva „Koohar“ je glavni dio praktičnog rada. Aplikacija služi za upravljanje kućnim budžetom te je prvenstveno namijenjena studentskoj populaciji. Aplikacija se sastoji od dva dijela – pregledavanja recepata za tjedan dana i pregledavanja recepata za jedan dan. Odabirom recepta za jedan dan korisnik upisuje sastojke koje ima na raspolaganju te mu sustav predlaže recepte koji sadrže unesene sastojke. Tjedni dio aplikacije je zadužen za upravljanje budžetom. Korisnik može unijeti iznos budžeta kojim raspolaže, te mu sustav predlaže recepte koji ulaze u upisani cjenovni rang.

Pri komercijalnoj upotrebi aplikacije najpogodniji poslovni model mobilnog poslovanja je freemium model (model besplatnih aplikacija) u kombinaciji s reklamiranjem unutar aplikacije. Aplikacija bi bila besplatna na tržištu, ali bi pri pokretanju prikazivala reklame. Ukoliko bi korisnik htio ukloniti reklame iz aplikacije, morao bi nadoplatiti određeni iznos. Također, moguća suradnja bi se mogla ostvariti i s lokalnim prodajnim centrima koji bi mogli proširiti svoju bazu korisnika nudeći svoje proizvode unutar te iste aplikacije.

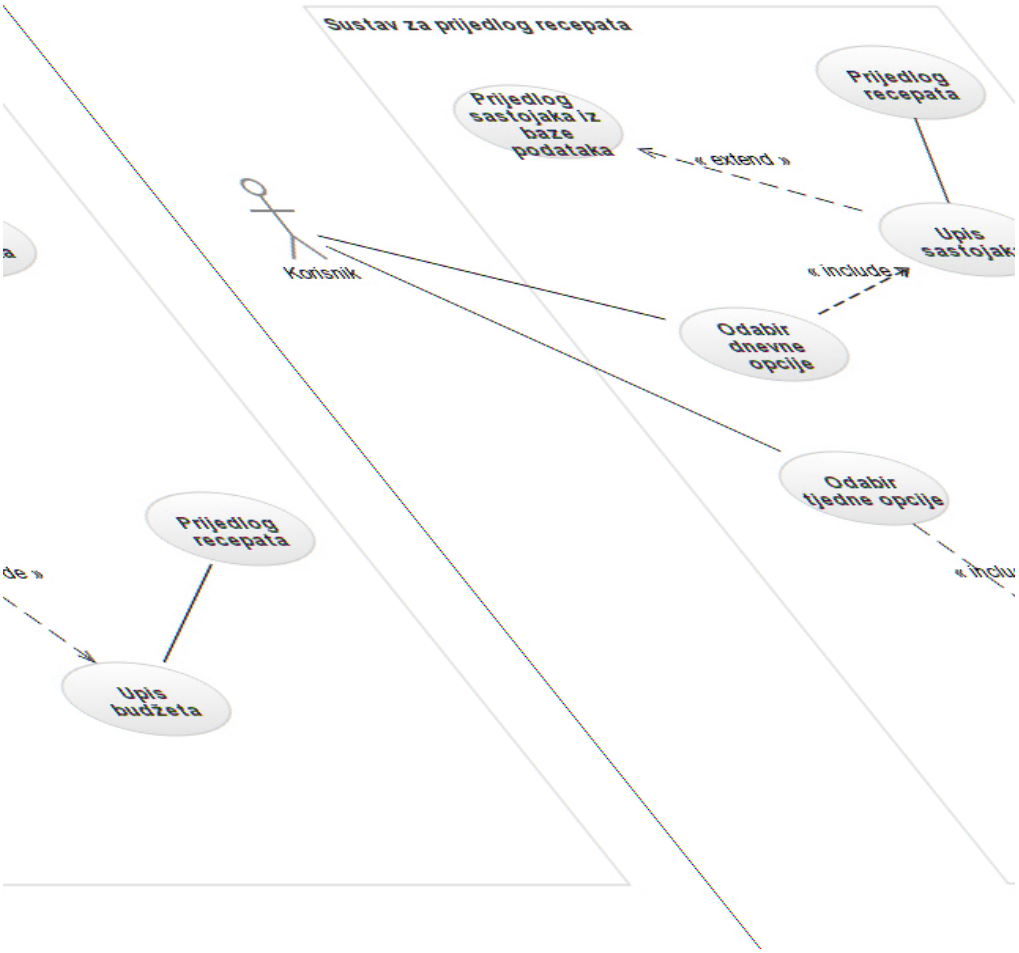
U svakom projektu uobičajeno je napraviti listu dobrih i loših strana projekta, tzv. SWOT analizu. Prednosti i nedostatci aplikacije bili bi:

- Prednosti (strength) – nedostatak sličnih alternativa na tržištu, sličan koncept ima coolinarka, web aplikacija za recepte koja sadrži dio za pretraživanje recepata po upisanim sastojcima
- Slabosti (weakness) – eventualni nedostatci ovakve aplikacije mogu biti ograničavanje na samo hrvatsko tržište. Također, nedostatak je i neizgrađena zajednica korisnika što znatno povećava ulaganja u marketing dok bi istovremeno druge aplikacije već izgrađenih imena, fokusirane na sličan koncept mogle preuzeti primat na tržištu.
- Mogućnosti (opportunities) – studentska populacija koja je ciljana skupina za korištenje ove aplikacije je ujedno i najveći konzument aplikacija i njima bi aplikacija za upravljanje budžetom uvelike olakšala studentski život. Suradnja bi se mogla ostvariti i s lokalnim prodajnim centrima koji bi mogli proširiti svoju bazu korisnika nudeći svoje proizvode unutar te iste aplikacije.
- Prijetnje (Threats) – otegotna okolnost za razvijanje aplikacije jest konkurencija s već izgrađenom korisničkom populacijom koja bi lako mogla implementirati ovakva programska rješenja u vlastitu aplikaciju.



Slika 5 **Tjedni i dnevni dio aplikacije, izvor: izradio autor**

Na *Slici 5* prikazano je korisničko sučelje tjednog i dnevnog dijela aplikacije. Na tjednom dijelu aplikacije uočljivo je polje za unos visine budžeta kojim korisnik raspolaže. Također, mogu se primijetiti i gumbi „submit“ i „delete“. Pritiskom na gumb „submit“ upisana vrijednost se sprema u varijablu te se ispisuje na ekranu, pritiskom na „delete“ moguće je izbrisati staru cijenu te unijeti novu.

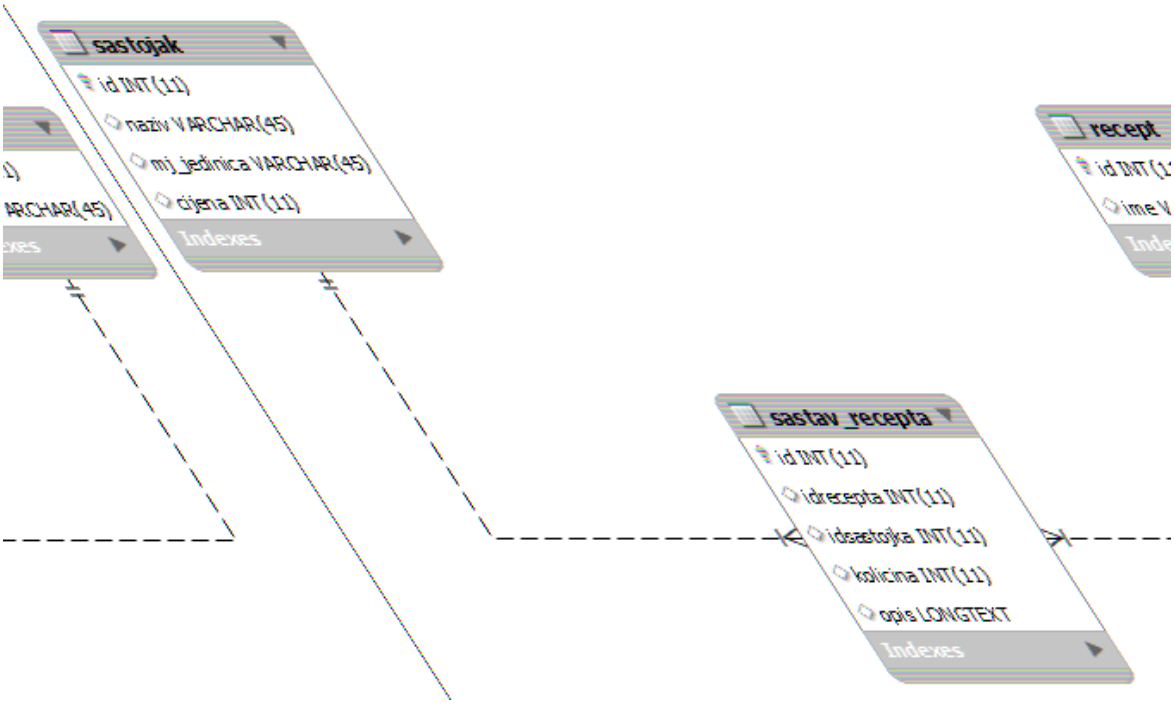


Slika 6 **Use-case diagram aplikacije, izvor: izradio autor**

Na *Slici 6* prikazan je dijagram slučaja upotrebe. Vanjski akter (korisnik) otvaranjem aplikacije odabire dnevnu opciju ili tjednu opciju. Odabirom dnevne opcije uključeno je upisivanje sastojaka; opcionalno se proširuje na prijedlog sastojaka iz baze podataka i korisnik dobiva listu predloženih recepata. Odabirom tjedne opcije uključeno je upisivanje budžeta na temelju kojeg se prikazuje popis predloženih recepata.

4. 2. Baza podataka

Baza podataka za ovaj projekt se zove „recepti.db“. Razvijana je u MySql-u, na besplatnom webhost računu.



***Slika 7* Notacija baze podataka, izvor: izradio autor**

Na *Slici 7* prikazana je notacija baze podataka koja se koristi u aplikaciji. U bazi podataka pohranjeni su recepti, a sama baza podataka se sastoji od tri tablice:

- Recept – tablica sadrži samo jedinstveni identifikator (id) te svoj naziv (ime).

- Sastojak – u tablici su pohranjeni sastojci neophodni za pripremu recepta; tablica sadrži jedinstveni identifikator (id), naziv sastojka (naziv), mjernu jedinicu (mj_jedinica) te sadrži cijenu određenog sastojka (cijena).
- Sastav_recepta – tablica zadužena za pripajanje sastojaka pripadajućem receptu; tablica sadrži jedinstveni identifikator (id), strani ključ (idrecepta) koji se referencira na tablicu recept, strani ključ (idsastojka) koji se referencira na tablicu sastojak, te sadrži količinu koja odgovara određenom sastojku za pripadajući recept, te opis odgovarajućeg recepta.

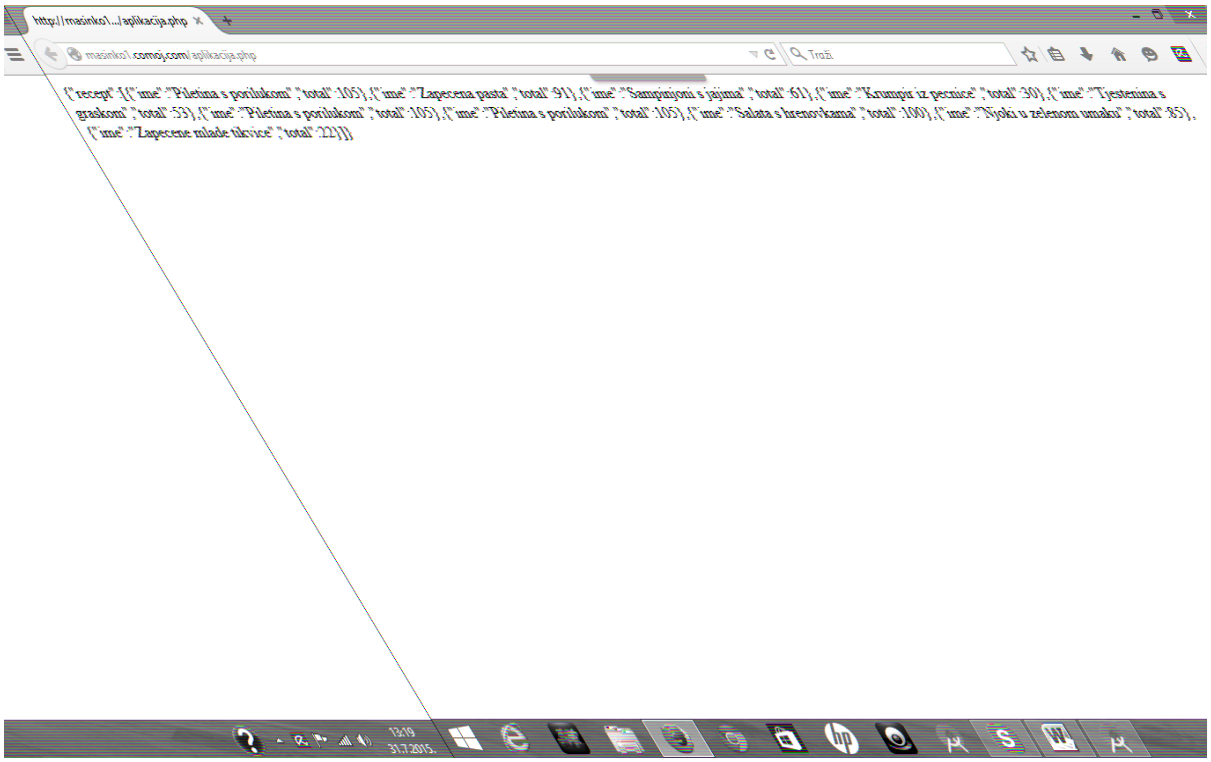
4. 3. Način preuzimanja podataka

Za preuzimanje podataka s baze korištena su dva pristupa, ovisno o smjeru aplikacije. Primjerice, za tjedni dio aplikacije korištena je PHP skripta, a za dnevni dio korištena je SQLite baza.

Php je skriptni jezik koji prvenstveno služi za razvijanje web aplikacija. U ovom radu je korišten tako što je napisana skripta koja izvršava sljedeći upit nad bazom podataka:
SELECT ime, SUM (sastojak.cijena) AS total FROM sastojak,sastav_recepta, recept WHERE recept.id = $rand AND sastav_recepta.idrecepta = $rand AND sastav_recepta.idsastojka = sastojak.id;

Ovaj upit povlači naziv recepta i broj sastojaka koji su povezani s tim receptom gdje $rand varijabla predstavlja php varijablu koja koristi rand funkciju za prikazivanje naizmjeničnih brojeva u rangu broja recepata te na kraju ta varijabla predstavlja redak (id) iz tablice recepti.

Nakon izvršenog upita, rezultat služi kao parametar u php funkciji „json_encode()“, json encode funkcija služi za pretvaranje php sadržaja u Javascript sadržaj (JavaScript Object Notation), odnosno varijable koje će se moći koristiti u Java programskom jeziku. Pravilno formatiran rezultat ovakve php skripte može se pogledati u Internet pregledniku.



***Slika 8* Prikaz preuzetih podataka, izvor: izradio autor**

Na *Slici 8* prikazan je izgled JSON formatiranog sadržaja koji se prikazuje na Internet pregledniku.

Ovako formatiran sadržaj koristi se u Java kodu tako da se pozove url adresa gdje se skripta nalazi: „HttpPostpost=newHttpPost ("http://masinko1.comoj.com/aplikacija.php");“. Nakon pozivanja skripte sadržaj se sprema u varijable pozivajući „getJSONArray“ funkciju koja kao parametar uzima polje iz url skripte, kao što je na *Slici 8* polje „recepti“. Na *Slici 8.* je vidljiva situacija gdje se isti naziv recepta pojavi više puta, kao što je primjerice „Piletina s porilukom“. Ovaj problem je riješen tako da je napravljena „for“ petlja koja prolazi kroz polje te ako jedan element u polju odgovara drugom elementu u polju poziva se funkcija koja briše ponavljajući element, tako su postignuti jedinstveni nazivi recepata.

SQLite baza podataka jest baza podataka napravljena za korištenje u razvijanju android aplikacija. Prednost ovakve baze kod razvijanja aplikacija jest to što se može koristiti

direktno u aplikaciji te ima sposobnost korištenja varijabli iz aplikacije direktno u SQL upitima. Nedostatak SQLitea jest taj što ima ograničenje veličine baze, odnosno ona je pogodna samo za male baze podataka. U aplikaciji Koohar SQLite baza je napravljena tako što je preuzeta baza podataka koja se nalazi na web-host računu te, koristeći priključak za internet preglednike „SQLiteBrowser“, baza je pretvorena u sqlite format, pogodan za korištenje u java programskom jeziku.

4. 4. Dio aplikacije za dnevne recepte

Kao što je naglašeno u opisu aplikacije, kod dijela aplikacije za dnevne recepte korisnik mora upisati sastojke od kojih želi pripremiti obrok. Kod upisivanja sastojaka korištena je komponenta „AutoCompleteTextView“ koja služi za prikazivanje prijedloga sastojaka na temelju slova koja su upisana. Prijedlozi su povučeni iz SQLite tablice „sastojak“ tako što su odabrani svi zapisi iz retka „naziv“ te su spremljeni u listu podataka (ArrayList). Nakon spremanja upita iskorištena je lista kao izvor podataka za prijedloge sastojaka. Nakon upisivanja sastojaka, pritiskom na gumb „Ponuđeni recepti“ korisnik se preusmjerava na aktivnost koja ispisuje listu s receptima koji odgovaraju upisanim sastojcima. Lista se ispunjava nazivima jela, također pomoću upita nad SQLite bazom. Upit prikazuje nazive recepata gdje redak „naziv“ iz tablice „sastojak“ odgovara nazivu upisanog sastojka iz prethodne aktivnosti; „id“ iz tablice sastojak odgovara stranom ključu „idsastojka“ iz tablice sastav_recepta; a „id“ iz tablice recept odgovara stranom ključu „idrecepta“ iz tablice sastav_recepta.

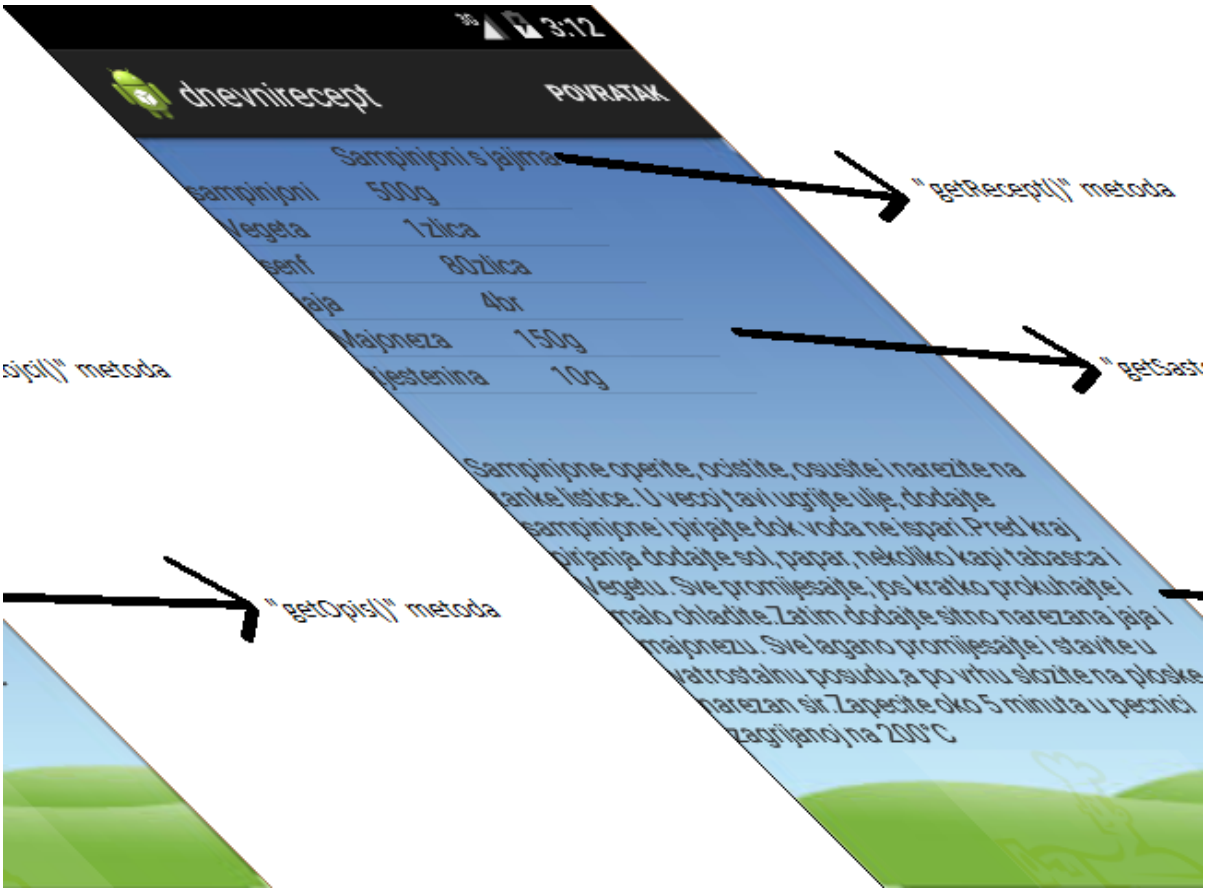
Slika 9 **Ponuđeni recepti za upisane sastojke, izvor: izradio autor**

Na *Slici 9* prikazana je lista podataka popunjena s ponuđenim receptima. Nakon što je odabran jedan recept iz liste, odabirom opcije „Pročitaj recept“ korisnik se preusmjerava na aktivnost koja prikazuje detalje recepta. Na toj aktivnosti korištene su tri metode iz posebne Java klase za manipuliranje podatcima iz SQLite baze. Te tri metode predstavljaju tri upita nad bazom:

- Prva metoda naziva se „getRecept()“ koja vraća tip podatka „String“ u obliku naziva recepta. Rezultat upita iz metode „getRecept()“ prikazuje naziv recepta koji odgovara odabranom nazivu iz prethodne liste.
- Druga metoda naziva se „getSastojci()“. Ova metoda vraća podatke spremljene u polje (ArrayList). U to polje sprema se naziv sastojka te količina sastojka

potrebna za odabrani recept. U toj metodi upit predstavlja sve retke iz tablice „sastojak“, te nazive iz tablice „recept“ uz uvjet da naziv recepta odgovara prethodno odabranom nazivu, te da polje „id“ iz tablice sastojak odgovara stranom ključu „idsastojka“ iz tablice sastav_recepta. Također polje „id“ iz tablice recept mora odgovarati stranom ključu „idrecepta“ iz tablice sastav_recepta.

- Treća metoda naziva se „getOpis()“. Ova metoda također vraća podatke spremljene u polje. Upit u ovoj metodi prikazuje redak „opis“ iz tablice „sastav_recepta“ uz uvjet jednak kao i u metodi „getSastojci()“.



Slika 10 **Prikaz recepta s odgovarajućim metodama, izvor: izradio autor**

Na *Slici 10* prikazana je posljednja stranica aplikacije koja sadrži ime prethodno odabranog recepta, pripadajuće sastojke odabranog recepta te opis odabranog recepta.

4. 5. Dio aplikacije za tjedne recepte

Kod dijela aplikacije za tjedne recepte korisnik mora upisati cjenovni budžet kojim raspolaže za tjedan dana, kao što je prikazano na *Slici 5*. Nakon što upiše budžet, odabirom na gumb „Ponuđeni recepti“ korisnik se preusmjerava na aktivnost gdje se u obliku liste prikazuju ponuđeni recepti unutar navedenog cjenovnog ranga. Upisana cijena sprema se u

varijablu koja predstavlja budžet. Zatim se u posebnoj Java klasi izvršava „do-while“ petlja, koja uzima naziv recepta i sumu određenog recepta iz JSON polja sve dok je upisana cijena veća od zbroja cijene sastojka recepta. Tako formatirano polje koristi se na sličan način kao i na dijelu aplikacije za dnevne recepte, a isto se polje koristi i za popunjavanje liste odabira naziva recepta.

Nakon odabira naziva recepta, pritiskom na gumb „Pročitaj recept“ korisnik se preusmjerava na aktivnost koja prikazuje detalje odabranog recepta. Detalji recepta se preuzimaju na sličan način kao i kod dnevnog dijela recepta. Također postoje tri metode („getReceptTjedni()“, „geSastojciTjedni()“,“getOpisTjedni()“) u posebnoj Java klasi za upravljanje SQLite bazom. Te su tri metode jednake kao i kod razvijanja dnevnog dijela aplikacije s jedinom razlikom da u uvjetu koriste prethodno odabrano ime iz tjednog dijela aplikacije.

Zaključak

Mobilno poslovanje je područje koji bilježi kontinuirani rast upravo zahvaljujući svojoj komponenti pokretljivosti. Veliki broj organizacija odlučuje se na mobilno poslovanje koje im olakšava svakodnevno organiziranje poslovanja. Mobilno poslovanje pridonosi unaprjeđenju komunikacije, kako između zaposlenih u organizaciji, tako i s klijentima.

Android operacijski sustav predstavlja dominantni operacijski sustav za pametne telefone.Android OS nije optimiziran kao što je njegova konkurencija iOS. Primjerice, neki pametni telefoni, koji imaju veću procesorsku snagu od iPhonea, pokazuju slabije performanse zato što se Android OS mora pokretati na velikom broju različitih modela mobilnih uređaja, dok se iOS fokusira samo na jedan model. No upravo zahvaljujući svojoj prilagodbi na različite modele mobilnih uređaja Android je najčešći operacijski sustav.

Pokazan je razvoj android aplikacije za upravljanje kućanskim budžetom. Aplikacija je prvenstveno namijenjena studentskoj populaciji. Aplikaciji nedostaje mogućnost korisničkog upisivanja novih recepata. Aplikacija pruža još puno mogućnosti za napredak. Potrebno je aplikaciju prevesti i na druge jezike, budući da je trenutno samo na hrvatskom jeziku. Također, otvoren je prostor za suradnju s lokalnim prodajnim centrima, tako bi aplikacija koristila sastojke koji su trenutno dostupni i po trenutnoj cijeni u određenom centru. Razvoj aplikacije nudi mnoge pogodnosti, potiče kreativnost i određene individualne slobode kod rješavanja problema. Razvoj je specifičan i dinamičan te ovisi o vještinama programera. Programska zajednica na internetu je vrlo rasprostranjena i široka budući da su android programeri vrlo aktivni na stručnim forumima kao što su „stackoverflow“ itd. Budući da se Android kontinuirano poboljšava i optimizira, može se očekivati da će ostati najučestaliji operacijski sustav na pametnim telefonima te da će rasti potražnja za android aplikacijama.

Literatura

Knjige:

[1] Neill Smyth – Android 4 App Development Essentials ,Payload media, 2014

[2] Matt L. Murphy – Busy Coders Guide To Android Development ,CommonsWare, 2015

[3] Željko Panian- Elektroničko poslovanje druge generacije, Ekonomski fakultet Zagreb, 2013.

[4] Nayak Richi- Wireless Technologies to Enable Electronic Business.Queensland University of Technology,Brisbane,2010

Web izvori:

[1] Android Developers – https://developer.android.com/intl/zh-cn/index.html. [Pristupljeno:20.08.2015].

[2]Skill Guru- http://www.skill-guru.com/blog/2011/01/13/android-activity-life-cycle/ [Pristupljeno:23.08.2015].

[3] Hughes Lauren – The total impact of deploying mobility services: Location,Voices,Guest Acces and Advanced security – http://www.cisco.com/web/strategy/docs/trec/The-Total-Economic-Impact.pdf

[Pristupljeno:01.09.2015]

[4]Technotopia.com-

http://www.techotopia.com/index.php/An_Overview_of_the_Android_Architecture#The_Android_Software_Stack. [Pristupljeno:20.08.2015].

[5]International data corporation -http://www.idc.com/prodserv/smartphone-os-market-share.jsp

[Pristupljeno:25.08.2015].

Popis slika: