Optimizacija izrade nativne android aplikacije za praćenje sportskog treninga

Strgar, Tomislav

SVEUČILIŠTE U ZAGREBU
GRAFIČKI FAKULTET

TOMISLAV STRGAR

OPTIMIZACIJA IZRADE NATIVNE ANDROID APLIKACIJE ZA PRAĆENJE SPORTSKOG TRENINGA

DIPLOMSKI RAD

ZAGREB, 2014.

SVEUČILIŠTE U ZAGREBU
GRAFIČKI FAKULTET

SMJER I MODUL:
Tehničko-tehnološki, multimedij

OPTIMIZACIJA IZRADE NATIVNE ANDROID APLIKACIJE ZA PRAĆENJE SPORTSKOG TRENINGA

DIPLOMSKI RAD

Mentor:
prof. dr. sc. Klaudio Pap

Student:
Tomislav Strgar

ZAGREB, 2014.

Rješenje o odobrenju teme diplomskog rada

SAŽETAK

Tema ovog diplomskog rada je izrada nativne Android aplikacije za odabir programa treninga, praćenje napretka i prehrane za treniranje u teretani napisane pomoću Java programskog jezika te preko automatiziranog web servisa za izradu aplikacija ShoutEm. Aplikacija će nuditi nekoliko vrsta treninga ovisno o željama korisnika. Potrebno je integrirati formule pomoću kojih će se obračunavati kilaže i intenzitet treninga te potreban dnevni unos kalorija baziran na kilaži, visini i dobi korisnika. Korisnici će moći bilježiti svoj napredak na dnevnoj bazi te pisati komentare za svoje potrebe. Korisniku će također ovisno o izboru programa treninga biti prikazane preporučene, "zabranjene" namirnice te suplementi. Aplikacija će zahtijevati unos podignutih kilaža od strane korisnika te automatski korigirati intezitet idućeg treninga ovisno je li korisnik uspješno ili neuspješno završio prethodni trening. Ovdje će se dokazati jednostavnost korištenja XML formata za izradu vizualnog sučelja naspram izrade grafičkog sučelja samo pomoću Jave. Aplikacija će također informirati korisnika o najvažnijim stavkama treninga kako bi rezultati bili što bolji u kraćem roku.

Ključne riječi: Java, XML, Android, ShoutEM, izrada

ABSTRACT

The topic of this thesis is the development of native Android application for sports training, progress tracking and planning sports nutrition for gym training which will be written in Java programming language and also made in ShoutEm web based service for application development. Application will offer few types of training based on users preferences. It is necessary to implement formulas which will calculate lifted weights, training intensity and daily calorie intake based on weight, height and age of the user. User will be able to track his own progress on a daily basis. Application will also mention recommended and „forbidden“ foods as well as supplements for achieving desired goals. Application will be able to track lifted weights for each training session. Simplicity of XML based layout in contrast to pure Java layout will be proven. Application will also inform the user about how and why this type of training works to provide the best results in shortest amount of time possible.

Key words: Java, XML, Android, ShoutEm, development

SADRŽAJ

1. Uvod

Kako su se mobilni telefoni napretkom tehnologije pretvorili u pametne telefone (eng. smartphone) te sada čine džepna računala koja svakodnevno imamo uz sebe i mogu nam asistirati u mnogo aspekata života, odlučio sam se okušati u izradi Android aplikacije. Kod odabira teme za aplikaciju i ovaj rad, prva ideja je bila aplikacija za praćenje sportskog treninga. Imajući u vidu da gotovo svatko barem u jednom periodu života krene u teretanu, to vrijeme zaslužuje biti kvalitetno iskorišteno. Sve češće ljudi zalaze u teretane sa svojim smartphoneovima na kojima imaju aplikacije sa nepotpunim informacijama. Znanja i ideje se također šire internetom i usmenom predajom, no većina njih je pogrešna te se veliki broj ljudi brzo obeshrabri jer ne vide rezultate. Cilj je okupiti informacije o kvalitetnoj prehrani i kvalitetnom treningu koje će biti od koristi i početnicima i naprednima. S obzirom da je provedena anketa pokazala da bi svima koji su u njoj sudjelovali takva aplikacija bila korisna, ideja se zadržala. Teorijski dio rada sadržavat će informacije o Java programiranju, XML-u te Android platformi zajedno sa njihovim manama naspram drugih programskih rješenja kako bi se pojasnio proces i količina rada koji su potrebni za izradu jedne aplikacije. Eksperimentalni dio prikazat će sâm proces izrade nativne aplikacije i pokušaj izrade identične aplikacije u ShoutEm servisu. Aplikacija je rađena s ciljem da bude od koristi i meni i svima koji žele imati kvalitetne informacije i dnevnik treninga na jednom mjestu koje je stalno pri ruci.

2. Teorijski dio

2.1. Java

2.1.1. Povijest Java-e

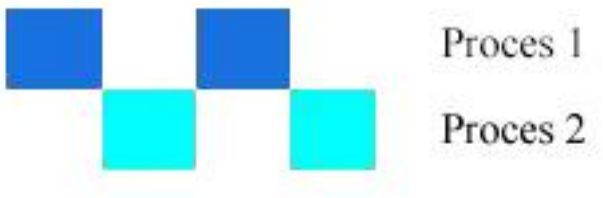
Java platforma i programski jezik su počeli kao interni projekt u Sun Microsystems u prosincu 1990.g., pružajući alternativu C i C++ programskim jezicima. Inženjer Patrick Naughton je bio prilično nezadovoljan stanjem Sun C i C++ sučelja za programiranje aplikacija (end. API – application programing interface) i alata. Ponuđena mu je prilika da radi na novoj tehnologiji i time je započeo Stealth Project koji je ubrzo preimenovan u Green Project kada su se James Gosling i Mike Sheridan pridružili Naughtonu. Zajedno s ostalim inženjerima, počeli su raditi u malom uredu na Sand Hill Road, Menlo Park u Kaliforniji. Pokušavali su razviti novu tehnologiju za programiranje nove generacije pametnih uređaja, što je za Sun trebala biti nova velika prilika da zavlada tržištem. Tim je prvotno razmatrao korištenje C++, ali taj je prijedlog odbijen iz nekoliko razloga. Prvo, s obzirom da su razvijali ugrađeni sustav sa ograničenim resursima, shvatili su da C++ treba previše memorije i da njegova kompliciranost dovodi do pogrešaka tijekom programiranja. Drugo, nedostatak mogućnosti da C++ ukloni nepotrebne informacije je značio da programeri moraju sami optimizirati memoriju sistema što je zahtjevna zadaća u kojoj se lako događaju pogreške. Kao treće, tim je također bio zabrinut zbog nedostatka prijenosnih objekata za sigurnost, distribuiranog programiranja i grananja zadataka (eng. threading). U konačnici, htjeli su platformu koja bi se lako mogla koristiti na svim vrstama uređaja. Bill Joy je zamislio novi jezik kombinirajući C i Mesa. U članku pod nazivom Next, predložio je Sun-u da njihovi inženjeri proizvedu objektno-orijentiran okoliš temeljen na C++. U početku je Gosling pokušao izmijeniti i proširiti C++, ali ta ideja je ubrzo napuštena u korist stvaranja novog jezika koji je nazvao Oak. Do ljeta 1992.g., već su bili u mogućnosti pokazati dijelove nove platforme uključujući Green OS, Oak jezik, programske knjižnice i hardver. Njihov prvi pokušaj demonstracije 3. rujna 1992. bio je usmjeren na izgradnju osobnog digitalnog pomoćnika (PDA) kojeg su nazvali Star7. Imao je grafičko sučelje i pametnog asistenta. U studenom iste godine, Green Project je postao podružnica tvrtke Sun Microsystems, a tim je preselilo u Palo Alto u Kaliforniji.

Bili su zainteresirani za stvaranje visoko interaktivnih uređaja, a kad je Time Warner izdao zahtjev za izradu primatelja kabelske televizije (eng. set-top box) promijenili su svoj cilj i odgovorili s prijedlogom za set-top box platformu. Međutim, kabelska industrija je osjetila da njihova platforma daje preveliku kontrolu korisniku i izgubili su utrku. Tako su se vratili pod okrilje Sun-a gdje se nalaze i danas. [1]

2.1.2. Način rada Java-e

Java je računalni programski jezik koji je simultan (istodobna obrada više akcija). Bazira se na klasama te je objektno-orijentiran.

Simultano računanje može se opisati kao više procesa izvođenih u isto vrijeme. U simultanom računanju, životni vijek više procesa se preklapa, ali izvedba se ne mora dogoditi u istom trenutku. Na primjer, istodobni procesi se mogu izvršiti na jednoj jezgri preko „vremenskih kriški“: jedan proces se odvija u određenom vremenskom periodu, a ako ne završi tijekom tog vremenskog odsječka, on se zaustavlja, drugi proces počinje ili se nastavlja, a onda kasnije izvorni proces bude nastavljen (slika 1). Na taj način višestruki procesi su djelomično izvršili svoj zadatak, ali samo je jedan aktivan proces koji se izvršava u tom trenutku. Simultano računanje može se izvršiti paralelno, primjerice dodjeljujući svaki proces zasebnoj procesorskoj jezgri (današnji višejezgreni procesori) ili distribuiranjem računanja preko mreže. To je poznato kao zadatak paralelizma, a ovaj tip paralelnog računanja je podvrsta istodobnog računanja.



Slika 1. Paralelni procesi izvođeni simultano

Klasno-orijentirano programiranje je stil objektno-orijentiranog programiranja (OOP) u kojem se nasljedstvo postiže definiranjem klase objekata umjesto samih objekata. U ovom modelu, objekti su cjeline koje kombiniraju stanja (tj. podatke), ponašanje (tj. postupke ili metode) i identitet (jedinstveno postojanje među svim drugim

predmetima). Struktura i ponašanje objekta su definirani pomoću klase koja je definicija ili nacrt svih objekata određenog tipa. Objekt mora biti izričito izrađen na temelju klase, a objekt stvoren na taj način se smatra ogledom te klase. Objekt je sličan strukturi s dodatkom metode pokazivača, kontrole pristupa te locira instance[1] klase (stvarne objekte te klase) u klasnoj hijerarhiji (bitno za *runtime* nasljedne značajke).

Objektno-orijentirano programiranje (OOP) je programerska paradigma[2] koja predstavlja koncept "objekata" koji imaju polja podataka (atributa koji opisuju objekt) i uz to vezane procedure poznatije kao metode. Objekti, koji su obično prikaz klase, koriste se za interakciju s drugim objektima za dizajn aplikacija i računalnih programa. C++, C, Delphi, Java, JavaScript, Perl, Python, Ruby i PHP su primjeri objektno-orijentiranih programskih jezika. [2]

Sintaksa Java-e u velikoj mjeri proizlazi iz C++. Za razliku od njega, koji kombinira sintaksu za strukturirano, generički i objektno-orijentirano programiranje, Java je izgrađena gotovo isključivo kao objektno-orijentirani programski jezik. Sav kod je napisan u klasama i sve je objekt, s izuzetkom primitivnih tipova podataka (npr. brojeva, Boolean vrijednosti[3] te znakova) koji nisu klase iz izvedbenih razloga. Za razliku od C++, Java ne podržava preopterećenje operatora ili višestruka nasljedstva za klase. To pojednostavljuje jezik i pomaže u sprečavanju potencijalnih grešaka [3]. Java koristi slične metode za komentiranje kao i C++. Postoje tri različita stila komentara: jedan red komentara označen je sa dvije kose crte (/ /), komentar kroz višestruke redove se otvara sa „/ *“ i zatvara sa „* /“. Posljednji je Javadoc stil komentara koji započinje sa „/ **“, a zatvara se sa „* /“ . Javadoc stil komentiranja omogućuje korisniku da pokrene Javadoc zbog sastavljanja dokumentacije za taj program. Primjer jednostavnog dijela koda sa komentarima iz Sudoku aplikacije prikazan je na slici ispod (programski isječak 1).

[1] Instanca – realizacija objekta u objektno-orijentiranom programiranju

[2] Paradigma – osnovna značajka programiranja, način izgradnje strukture i elemenata programa

[3] Boolean – vrsta podataka koji imaju samo 2 vrijednosti, obično „true“ i „false“

```
/* AUTO-GENERATED FILE. DO NOT MODIFY.
*
* This class was automatically generated by the
* aapt tool from the resource data it found. It
* should not be modified by hand.
*/
package org.example.sudoku;
public final class R {
        public static final class attr {
        }
        public static final class drawable {
              public static final int icon=0x7f020000;
        }
        public static final class layout {
              public static final int main=0x7f030000;
        }
        public static final class string {
              public static final int app_name=0x7f040001;
             public static final int hello=0x7f040000;
       }
}
```

Programski isječak 1. Primjer Java koda sa komentarima

2.1.3. Prednosti i mane Java programskog jezika

Java programski jezik je posebno dizajniran kako bi bio što je moguće manje ovisan o specifičnom uređaju na kojem se izvodi. Svrha je da se neka aplikacija programira samo jednom, ali se može pokrenuti na bilo kojem uređaju (eng. WORA - write once, run anywhere), što znači da kod koji se izvodi na jednoj platformi ne treba preurediti za rad na drugoj. Java je, od 2014.g., jedan od najpopularnijih programskih jezika u upotrebi, osobito za klijent-poslužitelj web aplikacije s prijavljenih 9 milijuna programera. Izvorno je razvijena od strane Jamesa Goslinga u Sun Microsystems (koji

se spojio u Oracle Corporation), a objavljen je 1995.g. kao osnovna komponenta Sun Microsystems Java platforme. Jezik preuzima većinu sintaksi iz programa C i C++. Originalna i referentna implementacija Java kompajlera, virtualne mašine i popis klasa su razvijeni u Sun-u 1991.g. i pušteni u promet 1995.g. Od svibnja 2007., u skladu sa specifikacijama iz Java Community Process, Sun je prenio većinu svojih Java tehnologija pod GNU opće javne licence. Drugi su također razvili alternativne implementacije Sun tehnologija, kao što su GNU kompajler za Javu, GNU standardne knjižnica (eng. library) i IcedTea-Web (browser plugin za aplikacije).

Glavna karakteristika Jave je prenosivost, što znači da programi napisani u Java programskom jeziku moraju raditi slično na bilo kojem uređaju/platformi. To se postiže kompajliranjem Java koda u posredničku verziju koda zvanu *Java bytecode* umjesto direktno u kod specifičan za taj uređaj. Zbog velikih sličnosti kodova, svaki uređaj uspješno interpretira Java kod. Krajnji korisnici koriste Java okruženje (eng. JRE – Java Runtime Environment) za samostalne Java aplikacije ili internet pretraživače za manje dijelove Java koda koji su obično dio nekih većih programa. Mana ovakvog pristupa je brzina izvođenja aplikacija te veća potražnja za memorijom uređaja od programa pisanih specifično za neki uređaj (npr. programi pisani u C++ imaju reputaciju veće brzine izvođenja od Java aplikacija). Ovaj problem je djelomično ublažen uvođenjem kompilacije koda u realnom vremenu (JIT kompilacija) koja predstavlja poboljšanje odlika Java programskog jezika u svrhu bolje interpretacije koda što podrazumijeva i veću brzinu. [4]

Java koristi automatski modul za upravljanje memorijom u životnom ciklusu objekta. Programer određuje kada se objekti stvaraju, a Java Runtime je odgovoran za oporavak memorije jednom kada objekti više nisu u uporabi. Kad više nema referenci na objekt, nedostupna memorija se oslobađa automatski. Nešto slično memorijskom curenju može se dogoditi ako kod ima referencu na objekt koji više nije potreban. Ako se pozivaju metode za nepostojeće objekte, javlja se iznimka. Glavna ideja Java automatskih modela za upravljanje memorijom je da programeri mogu biti pošteđeni tereta da ručno obavljaju upravljanjem memorijom. U nekim jezicima, memorija za stvaranje objekata

se raspoređuje na stog ili izričito dodjeljuje i oduzima sa heap-a[4]. U drugom slučaju odgovornost upravljanja memorijom snosi programer. Ako program ne ukine objekt, dolazi do nepotrebne potrošnje memorije. Ako program pokuša pristupiti ili osloboditi memoriju koja je već oslobođena, rezultat je nedefiniran i teško ga je predvidjeti, ali program će vjerojatno postati nestabilan i srušiti se. To se može djelomično otkloniti korištenjem pametnih pokazivača, ali to utječe na dodatnu kompliciranost. Ako nema dovoljno slobodne memorije za stvaranje novog objekta, to može uzrokovati trenutačno stopiranje programa. Java ne podržava C/C++ stil pokazivača gdje se adrese objekta i nedodijeljeni brojevi mogu koristiti naizmjenično. To osigurava sigurnost tipa podataka. Varijable Java primitivnih tipova podataka nisu objekti. Njihove vrijednosti se pohranjuju ili izravno u polja (za objekte) ili na stog (za metode), a ne na heap-u kao što to obično vrijedi za objekte. Zbog toga se Java ne smatra isključivo objektno-orijentiranim programskim jezikom. To je bila svjesna odluka dizajnera Java-e kako bi se poboljšale performanse [5].

2.1.4. Java Class Library

Java Class Library (JCL) je skup dinamičkih knjižnica naredbi koje sadrže određene skupove podataka te ih se jednostavnim naredbama poziva u kod, a koje Java aplikacije mogu koristiti u vrijeme izvođenja. Zbog toga što Java platforma ne ovisi o određenom operacijskom sustavu, aplikacija se ne može osloniti na nativne knjižnice koje se nalaze na tim platformama. Umjesto toga, Java platforma pruža sveobuhvatan skup standardnih knjižnica klasa koji sadrži funkcije zajedničke modernim operacijskim sustavima. JCL ima tri uloge unutar Java platforme: poput drugih standardnih knjižnica koda, one pružaju programeru dobro poznati niz korisnih sadržaja, poput kontejnerskih klasa i procesuiranje standardnih izraza. Knjižnica pruža apstraktno sučelje za zadaće koje bi inače bile jako ovisne o hardveru i operacijskom sustavu, kao što su pristup mreži i pristup datotekama. Neke temeljne platforme ne podržavaju sve značajke koje Java aplikacija očekuje. U tim slučajevima, implementacija knjižnica može ili oponašati potrebne osobine ili osigurati dosljedan način provjere prisutnosti određenih

[4] Heap memory – dio računalne memorije kojoj programer mora manualno pristupiti te nakon korištenja naredbama isprazniti

karakteristika. Jedan od primjera takvih knjižnica (eng. library) je hashmap[5]. Hashmap stvara tablicu raznih povezanih vrijednosti što olakšava dohvaćanje i korištenje podataka iz memorije kako bi brzina ostala zadovoljavajuća čak i u slučaju velikih aplikacija. Ispod je prikazan primjer korištenja *hashmap* knjižnice (programski isječak 2).

```
public class HashMap
  {
  public static void main( String[] args )
        {
        HashMap map = new HashMap();
        map.add( "automobil", "crveni" );
        map.add( "kamion", "plavi" );
        map.add( "motor", "zeleni" );
         map.add( "bicikl", "crni" );
```

Programski isječak 2. Primjer Java knjižnice

2.2. XML

2.2.1. Povijest XML-a

Svestranost standardiziranog jezika za dinamičan prikaz informacija (eng. SGML – standard generalized markup language) je bila primijećena od strane ranih digitalnih medijskih izdavača u kasnim 1980-im godinama prije uspona interneta. Do sredine 1990-ih neki korisnici SGML-a su stekli iskustvo sa tada novom tehnologijom -

[5] Hashmap – struktura podataka koja se koristi za implementaciju polja podataka, može vezati vrijednosti za odgovarajuće ključeve

World Wide Web-om. Vjerovali su da SGML nudi rješenja za neke od problema na koje će WWW naići u budućnosti kako bude rastao. Dan Connolly je dodao SGML na listu W3C[6] aktivnosti kada se pridružio osoblju 1995.g. Radovi su počeli sredinom 1996-e kada je inženjer Sun Microsystems-a, Jon Bosak, razvio plan izrade te okupio suradnike. Bio je dobro povezan u uskoj grupi ljudi koji su imali iskustva i sa SGML-om i WWW-om. XML je razvila grupa od 11 članova koje je podupirala interesna skupina od 150 ljudi. Zapis o odlukama dizajna je sastavio Michael Sperberg-McQueen 4. prosinca 1997. Važan doprinos dao je James Clark čiji su doprinosi prazan element (eng. empty element) i ime XML.

Grupa se nikad nije srela licem u lice, sav posao obavljen je korištenjem e-maila i telekonferencija. Najveće odluke o dizajnu napravljene su u kratkim intenzivnim periodima rada između kolovoza i studenog 1996. Rad se nastavio kroz 1997. te je XML 1.0 postao član W3C konzorcija u 10. veljače 1998. [6]

2.2.2. Način rada XML-a

XML (eng. Extensible Markup Language) je jezik koji definira skup pravila za kodiranje dokumenata u formatu koji je čitljiv i ljudima i strojno. To je definirano u XML 1.0 specifikaciji koju je proizvela W3C te nekoliko drugih srodnih specifikacija od kojih su svi otvorenog koda. Ciljevi dizajna XML-a su naglasiti jednostavnost, općenitosti i iskoristivost preko Interneta. To je tekstualni format podataka uz snažnu potporu preko unicode-a[7] za različite ljudske jezike. Iako se dizajn XML-a fokusira na dokumente, on se naširoko koristi za zastupanje proizvoljnih struktura podataka, npr. na web-uslugama. Mnoga sučelja za programiranje aplikacija (API) su razvijena kako bi pomogla programerima u obradi XML podataka, a nekoliko sustava shema postoji za pomoć u definiranju XML baziranih jezika. [7]

XML dokumenti ponašaju se kao drvo (eng. tree). Odnosi grana (eng. nodes) granaju se od osnovnih (eng. root) prema manjima. Grananje redom, od glavnog prema najmanjem

[6] W3C – međunarodna organizacija zadužena za bavljenje standardima interneta
[7] Unicode - standard za razmjenu podataka usmjeren na prikaz slova na način neovisan o jeziku, računalnom programu ili računalnoj platformi

dijelu, idu: *root, element, text, attribute, namespace, comment* [8]. Od 2009., stotine formata dokumenata su razvijeni pomoću XML sintakse, uključujući RSS, Atom, SOAP, i XHTML. Formati bazirani na XML-u su postali zadani za mnoge alate za uredsku produktivnost, uključujući Appleov iWork Microsoft Office (Office Open XML), OpenOffice.org i LibreOffice (OpenDocument). XML se također koristi kao osnovni jezik za komunikacijske protokole, kao što je XMPP. Primjene za Microsoft .NET Framework koriste XML datoteke za konfiguraciju. Apple ima provedbu registra na temelju XML. XML je došao u opću upotrebu za razmjenu podataka preko interneta. IETF RFC 7303 pruža pravila za izgradnju vrsta internet medija za upotrebu prilikom slanja XML. Također definira vrste medijskih program / XML i tekstualne / XML datoteka što govori samo to da su podaci u XML-u, a ništa o semantici. Korištenje tekst/XML je kritizirano kao potencijalni izvor problema za kodiranje i sugerirano je da bi mogao biti zastarjeo. RFC 7023 također preporučuje da se XML baziranim jezicima daju vrste medija koje završavaju na +XML. Npr. slika/SVG+XML za SVG. Daljnje smjernice za korištenje XML-a u umreženom kontekstu može se naći u RFC 3470, također poznatom kao IETF BCP 70. - dokument koji obuhvaća mnoge aspekte projektiranja i uvođenja XML baziranih jezika.

S obzirom da je Android platforma otvorenog koda, to omogućuje pristup svim aspektima mobilnog uređaja bilo da se radi o niskim razinama grafike pa do hardvera kao što je kamera na telefonu. Sa tako puno opcija pri korištenju Androida, XML ostvaruje bitnu ulogu u izradi aplikacija. XML stvara mnoge mogućnosti pri izradi aplikacije. On se obično koristi kao format podataka na internetu. Većina podataka se na internetu ionako nalazi u obliku XML-a, kao i slanje podataka web servisima. Izrada dijela aplikacije u XML-u je jako bitna ako aplikacija treba postići uspjeh. Srećom, postoji mnogo mogućnosti za rad u XML-u na Android platformi. Jedna od najvećih prednosti Androida je što koristi Java programski jezik. Android SDK[8] ne nudi sve što nudi Java, ali daje podršku veoma važnom dijelu Jave. Java platforma podržava više

[8] SDK – skup razvojnih alata koji omogućujerazvoj aplikacija za neku platformu

različitih načina rada sa XML-om, a većina XML programskih sučelja vezanih uz Javu su podržani od strane Androida.

2.2.3. Izrada grafičkog sučelja - XML vs Java

Grafički izgled aplikacije može se postići na 2 načina:

- Objavljivanjem elemenata korisničkog sučelja (eng. UI - user interface) u XML-u. Android nudi jednostavan XML rječnik koji odgovara prikazu klasa i podklasa, kao što su one za widgete[9] i grafičko sučelje
- Stvaranje rasporeda elemenata za vrijeme izvođenja aplikacije čime aplikacija može stvoriti objekte i programski manipulirati njihovim svojstvima

Android pruža fleksibilnost korištenja bilo koje ili obje metode u kombinaciji za deklariranje i upravljanje grafičkim sučeljem aplikacije. Čest je postupak deklariranja početnog izgleda u XML-u zajedno sa elementima i njihovim svojstvima što se može vidjeti na primjeru (programski isječak 3).

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
              android:layout_width="match_parent"
              android:layout_height="match_parent"
              android:orientation="vertical" >
    <TextView android:id="@+id/text"
              android:layout_width="wrap_content"
              android:layout_height="wrap_content"
              android:text="Hello, I am a TextView" />
    <Button android:id="@+id/button"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="Hello, I am a Button" />
</LinearLayout>
```

Programski isječak 3. Korisničko sučelje kodirano u XML-u

Nakon toga se pomoću Jave može dodati kod koji modificira stanje objekata.

[9] Widget – mala, jednostavna aplikacija koja je napravljena za korištenje na više platformi

Prednost deklariranja grafičkog sučelja u XML-u je što omogućuje bolje odvajanje grafičkih elemenata od koda koji kontrolira njihovo ponašanje. S obzirom da su u ovom slučaju grafički elementi odvojeni od Java koda, može ih se po potrebi mijenjati i prilagođavati bez mijenjanja koda aplikacije. Na ovaj način se također mogu napraviti sučelja za različite položaje zaslona, različite rezolucije te različite jezike. Povrh toga, deklariranje sučelja u XML-u olakšava vizualiziranje njegove strukture čime se lakše otklanjaju problemi. Zbog toga je aplikacija za praćenje treninga napravljena kombinacijom XML-a i Java koda.

2.3. Android

2.3.1. Što je android?

Android je operativni sustav (OS) za mobilne uređaje na temelju Linux jezgre, koji se trenutno razvija od strane Googlea. Sa korisničkim sučeljem koje se temelji na izravnoj manipulaciji, Android je dizajniran prvenstveno za touchscreen mobilne uređaje kao što su smartphone-ovi i tablet računala sa specijaliziranim korisničkim sučeljima za televiziju (Android TV), automobile (Android Auto) i ručne satove (Android Wear). U početku ga je razvijao Android Inc. kojega je Google financijski potpomogao, a kasnije i kupio u 2005. Android je pušten u promet 2007. zajedno s osnivanjem Open Handset Alliance – konzorcija hardverskih, softverskih i telekomunikacijskih tvrtki posvećenih unapređivanju otvorenih standarda za mobilne uređaje. Android je popularan među tehnološkim tvrtkama koje zahtijevaju gotov, jeftin i prilagodljiv operativni sustav za uređaje visoke tehnologije [9]. Androidova otvorena priroda je potaknula i ohrabrila veliku zajednicu programera i entuzijasta na korištenje otvorenog koda kao temelja za projekte kojima se dodaju nove mogućnosti za napredne korisnike ili unijeti Android na uređaje koji su službeno koristili neke druge operacijske sustave. Uspjeh ovog operativnog sustava ga je učinio metom za parničenje patenta kao dio tzv. ratova pametnih mobilnih telefona (eng. smartphone) između najvećih tehnoloških tvrtki. Unatoč tome što je u prvom redu namijenjen za upravljanje dodirom [10], također se koristi za igraće konzole, digitalne kamere i ostalu elektroniku. Od 2011. Android ima najveću instaliranu bazu od bilo kojeg mobilnog OS-a, a od 2013.

uređaji bazirani na Andoid sustavu se prodaju više nego Windows, iOS i Mac OS uređaji zajedno (slika 2) [11].

Period	Android	iOS	Windows Phone	BlackBerry OS	Others
Q2 2014	84.7%	11.7%	2.5%	0.5%	0.7%
Q2 2013	79.6%	13.0%	3.4%	2.8%	1.2%
Q2 2012	69.3%	16.6%	3.1%	4.9%	6.1%
Q2 2011	36.1%	18.3%	1.2%	13.6%	30.8%

Slika 2. Udio Androida na tržištu

2.3.2. Povijest Android verzija

Android 1.0 lansiran je sa HTC Dream / T-Mobile G1 uređajima u 2008. Primarna nova značajka Androida 1.0 je bila alatna traka sa obavijestima kojoj korisnici mogu pristupiti spuštanjem izbornika za pregled svih obavijesti. Google Sync opcija je također uvedena u ovoj verziji koja omogućuje sinkroniziranje kalendara Google računa i informacije o kontaktima iz telefona korisnika. Višestruki format početnog zaslona je također novitet od verzije 1.0 te se zadržao u svakoj sljedećoj verziji. Dodani su i widgeti koji pružaju korisnicima jednostavan pristup informacijama sa samo jednim dodirom prsta.

Android 1.1 pokrenut je u 2009. To je drugo izdanje Android OS-a koje je prvenstveno namijenjeno za poboljšanja Android 1.0. Glavna svrha objavljivanja je bila popraviti velik broj bugova[10] koji su bili prisutni u Androidu 1.0, s nekoliko dodanih manjih mogućnosti i prilagodbi. Osim tih ažuriranja, većina ostalih funkcija je ostala ista.

Android 1.5, kodnog imena Cupcake, je na tržište pušten u travnju 2009. Od ove verzije operacijskog sustava je pokrenuto nazivanje svake sljedeće verzije Androida po slatkim konfekcijama. U ovom izdanju uvedena je kamera uz sposobnost za upload[11] video sadržaja na Google Picasa servis kao i druge video web stranice poput YouTube. Osim

[10] Bug – greška u programu
[11] Upload – slanje podataka sa lokalnog sustava na udaljeni sustav

toga, ova verzija Android operativnog sustava je prva sadržavala funkciju kopiranja i lijepljenja teksta u internet pregledniku.

Android 1.6 (Donut), lansiran u rujnu 2009., je verzija Android OS-a koja je uvela potporu za veće rezolucije zaslona kako bi se zadržao korak sa napretkom tehnologije mobilnih telefona. Dizajn operativnog sustava je pojednostavljen i sadržavao je ažuriranu Google funkciju pretraživanja. To je omogućilo korisnicima da lako pretražuju aplikacije, kontakte i internetske podatke. Tu je također dodana aplikacija za razvoj vlastitih gesti zvana Gesture Builder.

Android 2.0/2.1 (Eclair, listopad 2009.) je predstavio poboljšanu funkcionalnost tipkanja na Motorola Droidu sa auto-ispravljanjem greški. To je postignuto uvođenjem značajnih poboljšanja u funkcionalnosti tipkovnice. Podrška za HTML5 također je prvi put predstavljena u 2.1, uz dodatne funkcije fotoaparata, novo sučelje za internet preglednik i Bluetooth mogućnosti.

Android 2.2 (Froyo, svibanj 2010.) označio je početak Nexus smartphoneova, počevši sa HTC Nexus One. Dodatne mogućnosti su veća funkcionalnost aparata. Ostale mogućnosti uključuju tethering[12] putem USB-a, novu mobilnu Hotspot funkcionalnost, podršku za Adobe Flash kao i mogućnost korištenja Android Marketa koji je kasnije preimenovan u Google Play te automatsko ažuriranje aplikacija. Korisničko sučelje je također nadograđeno kako bi se povećale performanse uređaja.

Android 2.3 (Gingerbread; prosinac 2010.) je operativni sustav pokrenut na drugom Nexus smartphoneu koji je proizveden od strane Samsunga, Samsung Nexus S koji je u trenutku izlaska na tržište nudio nekoliko, danas osnovnih, novouvedenih značajki kao što je podrška za NFC[13], poboljšanja u korisničkom sučelju za poboljšanje performansi uređaja, sposobnost internetskih poziva i poboljšanu tipkovnicu za brži unos podataka tipkanjem. Verzija 2.3 je također bila predviđena za veću rezoluciju zaslona kao i podršku za prednju kameru.

[12] Tethering – povezivanje jednog uređaja sa drugim

[13] NFC (near field communication) – set standarda za smartphoneove za brzi prijenos podataka na maloj udaljenosti (ne više od par cm)

Android 3.0 (Honeycomb; veljača 2011.) je prvi operativni sustav dizajniran za tablet računala. Bio je to prvi operativni sustav koji je uveo tipke na zaslonu za funkcionalnost dodira. Aplikacije su također redizajnirane za bolji pregled na većem zaslonu, zajedno s prvim internet preglednikom s mogućnošću višestrukih kartica. Višezadaćnost (eng. multitasking) je također poboljšana kako više ne bi bilo potrebno prebacivati između aplikacija. Umjesto toga, korisnici jednostavno mogu trenutno otvorene aplikacije prikazati na zaslonu sa samo jednim dodirom na gumb. U posljednjoj nadogradnji verzije 3.0, dodana je podrška za Google Wallet kao i podrška za višejezgrene procesore što je povećalo performanse i učinkovitost uređaja.

Android 4.0 (Ice Cream Sandwich; listopad 2011.) objavljen je sa značajnim poboljšanjima u sučelju. Najvažnije poboljšanje je uklanjanje hardverskih tipki koje su zamijenjene virtualnim tipkama na zaslonu te je time nastavljeno ono što je Android 3.0 započeo. Android aplikacija za glazbu je također zamijenjen novom Google Music aplikacijom. Ostale nove značajke uvedene u ovoj verziji Androida uključuju upravljanje aplikacijama koje se izvode u pozadini, odbacivanje obavijesti pokretom prsta, Android Beam koji je sigurna platforma za dijeljenje sadržaja, novi Roboto font, WiFi Direct i sposobnost preuređenja datoteka.

Android 4.1, 4.2, 4.3 (Jelly Bean) su pokrenute redom u lipnju 2012., studenom 2012. te srpnju 2013. Ova verzija Android operativnog sustava ponudila je potpuno novo sučelje što je davalo glatke i brže performanse. Google Voice pretraga također je poboljšana u ovoj verziji s poboljšanim mogućnostima pristupačnosti i podrškom za unos Brailleovog pisma. Android Beam funkcija za dijeljenje sadržaja doživjela je nadogradnju koja je uključivala opciju za lakši prijenos videa i fotografija. Osim toga, Google Now je uveden zajedno s djelotvornim poboljšanjima za notifikacije u Notification Centre. Jelly Bean je također ponudio nova poboljšanja u sigurnosti kao što su Smart App ažuriranja i enkripcija za aplikacije.

Android 4.4 (Kit Kat; listopad 2013.) je donio jednu od najvažnijih promjena na Android. Kit Kat je uravnotežio performanse projektom nazvanim Project Svelte. Svelte je Googleov pokušaj smanjenja zahtjeva sustava umjesto da ih poveća. Obično kada korisnici idu iz starije verzije OS na noviju, zahtjevi sustava obično se povećavaju. Suprotno tome, najnovija verzija Androida radi brže, glađe, učinkovitije i ima bolji

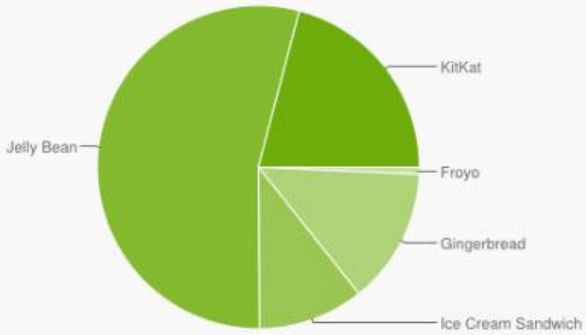
odaziv na uređajima koji imaju svega pola GB RAM-a. Također, nova verzija 4.4 predstavlja i uvođenje novih API[14]-ja u nastojanju da se pomogne programerima u izradi odličnih aplikacija koje će se pokretati glatko i imati bolji odaziv na većini uređaja, pogotovo na starijim uređajima s nižim performansama. Tu je i nova ugrađena postavka nazvana Process Stats. U proteklim verzijama Androida, korisnici su mogli vidjeti točno koliko memorije zauzima svaka aplikacija na uređaju. Process Stats je aplikacija koja funkcionira na sličan način, omogućujući korisnicima da vide koliko RAM-a pojedinačni programi koriste, što korisnicima daje preciznu sliku hoće li ili ne trebati više RAM-a. Nadogradnja na novi dizajn je u potpunosti posvećena izgledu operativnog sustava. Google je promijenio niz elemenata na koje su Android korisnici navikli od osnutka Androida 4.0. [12]

Izgled je znatno pojednostavljen – uvedena je novitet transparentnosti tipki duž dna zaslona te trake za obavijesti pri vrhu zaslona. Također, nove ikone za vrijeme i bateriju kao i svi indikatori pri vrhu zaslona su sada bijele boje. Malo je promijenjen i font cijelog sustava. Sve različite aplikacije ažuriraju sa novim čistim dizajnom, a odaziv na dodir tipki na dnu je mnogo suptilniji. Sveukupno, novi dizajn se pokazao mnogo jednostavnijim kao i poboljšanjem u fluidnosti cijelog OS-a. Opcija za pozive je značajno poboljšana u Android 4.4, kako u pogledu funkcionalnosti tako i dizajna. Dizajn je drugačiji jer organizira najčešće kontaktirane kontakte pri vrhu zaslona za maksimalnu prikladnost. Funkcija za pozivanje sada također djeluje kao prikladan okvir za pretraživanje. Ako korisnik želi tražiti lokalne objekte kao što su kafići, korisnici sada mogu jednostavno upisati "kava" i to će naći sve kafiće u blizini korisnika iz Google Maps zajedno s brojem telefona i adresom. Još jedno manje poboljšanje nalazi se kod primanja poziva. Ako korisnik prima telefonski poziv sa poslovnog broja tvrtke koji Google ima u svom ogromnom imeniku, to će automatski povući slike koje su relevantne za taj posao i prikazati sliku pozivatelju čak ako se tvrtka ne nalazi u popisu kontakata. Funkcija Host Card Emulation omogućuje bilo kojem uređaju sa sustavom 4.4 da oponaša transakcije temeljene na NFC-u tako da korisnici mogu čitati i pisati NFC bez obzira ima li uređaj NFC čip ili ne. Mnogo novih uređaja koji su opremljeni sa Androidom 4.4. dolaze opremljeni NFC čipom, ali kada je riječ o uređajima nižih

[14] API - skup određenih pravila i specifikacija koje programeri slijede tako da se mogu služiti uslugama ili resursima operacijskog sustava

specifikacija koji nemaju NFC čipove, oni će biti u mogućnosti iskoristiti ovu novu funkciju. Na početnom zaslonu za pokretanje, korisnici sada mogu brzo i lako pristupiti Google Now (dostupan samo na Nexus 5). Također, na Nexus 5 korisnici mogu koristiti glasovne naredbe, kao što su "Ok, Google", a to automatski otvara Google pretragu. Tu su i nove kartice i optimizacije u Google Now koje povezuju mnoge Googleove aplikacije zajedno. Na uređajima se danas mogu pronaći razne verzije, ovisno o starosti uređaja (slika 3) čiji broj na tržištu danas premašuje milijardu [13].

Version	Codename	API	Distribution
2.2	Froyo	8	0.7%
2.3.3 - 2.3.7	Gingerbread	10	13.6%
4.0.3 - 4.0.4	Ice Cream Sandwich	15	10.6%
4.1.x	Jelly Bean	16	26.5%
4.2.x		17	19.8%
4.3		18	7.9%
4.4	KitKat	19	20.9%



Slika 3. Udio Android verzija na smartphone uređajima

2.3.3. Način rada Androida

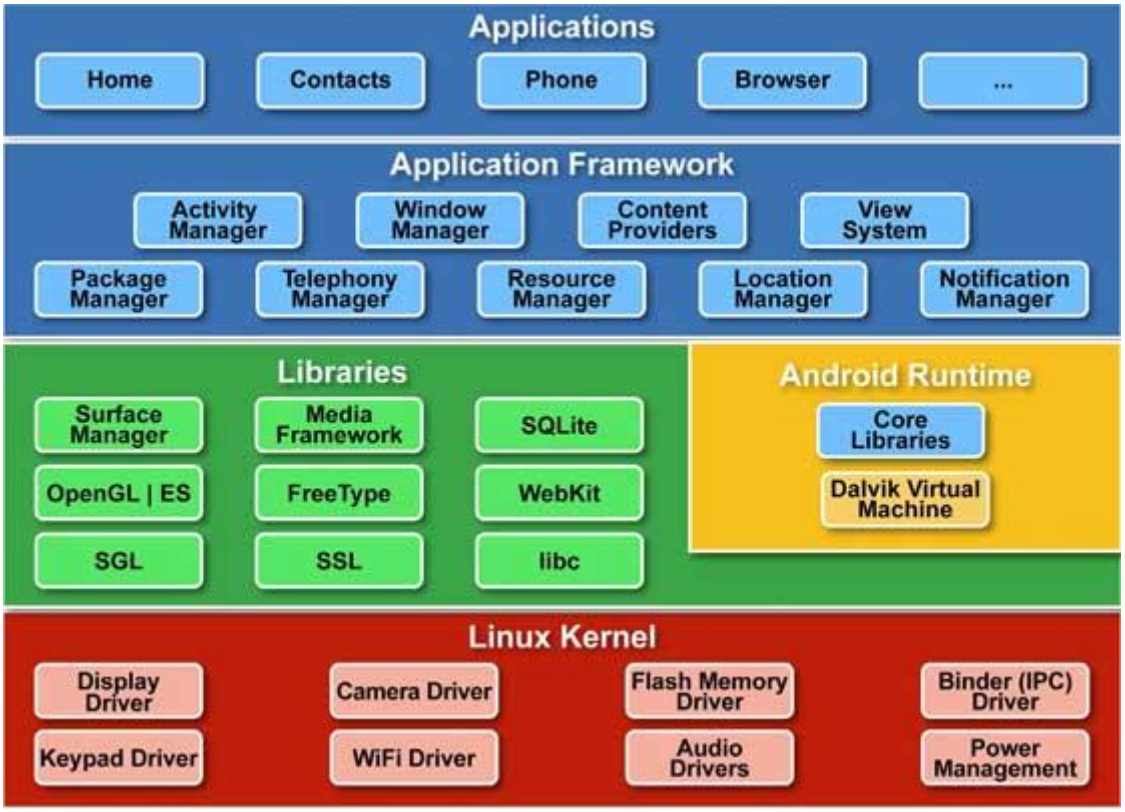
Android OS koristi dodir i kretnje za upravljanje koje otprilike odgovaraju stvarnim radnjama poput povlačenja prstima, dodirivanja, štipanja i obrnutog štipanja za manipulaciju predmetima na zaslonu i virtualnu tipkovnicu. Odgovor na unos korisnika je dizajniran da bude istovremen i pruža fluidno sučelje osjetljivo na dodir te često koristi vibraciju uređaja kako bi pružio taktilne povratne informacije korisniku. Interni hardver kao što su akcelerometrar**,** žiroskop i senzori koriste neke aplikacije kako bi odgovorile na dodatne akcije korisnika, primjerice podešavanje zaslona od uspravnog do horizontalnog, ovisno o tome kako je uređaj držan. Također omogućuje korisniku upravljanje vozilom u igrama okretanjem uređaja, simulirajući kontrolu nad volanom.

Android uređaji se bootaju[15] na početni zaslon, primarni navigacijski i informacijski centar na uređaju što je slično načinu na koji funkcioniraju stolna računala. Android početni zaslon se obično sastoji od ikona aplikacija i widgeta[16]. Ikone pokreću pripadajuću aplikaciju dok widgeti uživo prikazuju trenutna stanje te auto-ažuriranje sadržaja poput vremenske prognoze, korisnikov e-mail sandučić ili novosti izravno na početnom zaslonu. Početni zaslon može biti sastavljen od nekoliko stranica između kojih korisnik može birati naprijed-natrag pokretom prsta. Android početni zaslon je jako prilagodljiv omogućujući korisniku da prilagodi izgled i dojam uređaja vlastitom ukusu. Mnoge aplikacije dostupne na Google Play radikalno mijenjaju početni zaslon pa čak i oponašaju izgled drugih operativnih sustava kao što su Windows Phone. Većina proizvođača te neki mobilni *provideri* prilagođavaju izgled i dojam svojih Android uređaja kako bi se razlikovali od svojih konkurenata. Pri vrhu ekrana prisutan je status bar, koji prikazuje informacije o uređaju i njegovoj povezanosti. Status bar se može proširiti i bolje otkriti zaslon sa obavijestima gdje aplikacije prikazuju važne informacije ili ažuriranja (nedavno primljeni e-mail ili SMS poruka na način da se odmah ne ometa ili ne stvori neugodnost korisniku). Notifikacije o svježe primljenim porukama i slično su prisutne dok se ne pročitaju (dodirivanjem koje otvara odgovarajuću aplikaciju) ili odbace tako da korisnik makne zaslon. Počevši od verzije Androida 4.1, proširene obavijesti mogu prikazati više detalja ili dodatne funkcionalnosti, npr. glazbeni player može prikazati kontrole za reprodukciju, a obavijest o propuštenom pozivu pruža tipke za pozivanje natrag ili slanje SMS poruke pozivatelju. Android pruža mogućnost za pokretanje aplikacija koje mijenjaju zadani launcher[17], a time i izgled i vidljivo ponašanje Android uređaja. Ove promjene izgleda iz temelja mijenjaju osnovne značajke korisničkog sučelja.

S obzirom da su Android uređaji obično na baterije, Android je dizajniran za ekonomično upravljanje memorijom (RAM-a) kako bi se potrošnja energije svela na minimum za razliku od desktop operativnih sustava za koje se uglavnom pretpostavlja da su povezani s neograničenim izvorom električne energije. Kad Android aplikacija više nije u uporabi, sustav će je automatski suspendirati u memoriji – iako je aplikacija

[15] Boot – pokretanje računalnog sustava
[16] Widget – mala, jednostavna softverska aplikacija
[17] Launcher – program koji olakšava pokretanje drugih programa

tehnički još uvijek aktivna, suspendirane aplikacije ne troše resurse sustava (primjerice, bateriju ili procesorsku snagu) i besposleno čeka u pozadini dok nije ponovno potrebna. To ima dvostruku korist povećanja opće prijemljivosti Android uređaja jer aplikacije ne moraju biti zatvorene i ponovno od nule pokrenute svaki put, ali i obvezu da aplikacije u pozadini ne troše energiju nepotrebno. Android upravlja aplikacijama pohranjenima u memoriji automatski: kada je memorija niska, sustav će početi gasiti aplikacije i procese koji su bili neaktivni kroz neko vrijeme, ali obrnutim redoslijedom budući da su zadnje korištene (prvo najstarije). Ovaj proces je dizajniran da bude nevidljiv za korisnika tako da korisnici ne moraju upravljati memorijom ili gašenjem samih aplikacija. Svi dijelovi Android arhitekture mogu se prikazati jednostavnom shemom (slika 4). Međutim, konfuzija oko Android upravljanja memorijom je rezultirala pojavom aplikacija koje gase druge aplikacije koje su postale popularne na Google Play Store. Ove aplikacije „ubojice“ uglavnom čine više štete nego koristi.



Slika 4. Android arhitektura

2.3.4. Izrada aplikacija i njihovo tržište

Softverska aplikacija je skup jednog ili više programa namijenjenih za obavljanje poslova za određenu aplikaciju. Aplikacija ne može raditi sama, nego ovisi o softverskom sustavu kako bi se izvršila. Na primjer: MS Word, MS Excel, Tally softver, sustav upravljanja softverskim knjižnicama, sustav naplate, itd. Izraz se koristi kako bi se razlikovao takav softver od drugih vrsta računalnih programa - softverskog sustava koji upravlja i integrira računalne sposobnosti, ali ne obavlja izravno zadatke koji pomažu korisniku. Softverski sustav služi aplikacijama, koje pak služe korisniku. Primjeri uključuju računovodstveni softver, poslovni softver, grafički softver, medijske aplikacije te uredski paket. Mnogi aplikacijski programi se bave uglavnom dokumentima. Aplikacije mogu doći u paketu zajedno s uređajem i njegovim softverskim sustavom ili mogu biti objavljene zasebno te mogu biti kodirane kao sveučilišni projekti. Aplikacija primjenjuje moć određene računalne platforme ili softverskog sustava u određene svrhe. Neke aplikacije su dostupne u verzijama za nekoliko različitih platformi; druge imaju uže zahtjeve pa se stoga nazivaju nativnim aplikacijama, primjerice zemljopisna aplikacija za Windows, Android aplikacije za obrazovanje ili Linux igre. Ponekad se na tržištu pojavi nova i popularna aplikacija koja radi samo na jednoj platformi čime se povećava želja za tom platformom. To se zove „killer app“.

Dobro je poznato da razvijanje aplikacije za smartphone uređaje uključuje razradu planiranja i nekoliko procesa koji zajedno čine skladnu cjelinu. Sve počinje s idejom za aplikaciju, a zatim se ideja podvrgava pomnom planiranju, dizajniranju, razvoju aplikacije, testiranju i konačno, implementaciji aplikacije na planirane mobilne uređaje. Međutim, prije nego što aplikacija prođe kroz navedeni proces razvoja, postoji jedna vrlo bitna stavka o kojoj treba odlučiti. Razvijatelj (eng. developer) mora odlučiti točan način na koji želi stvarati i razvijati svoju aplikaciju, želi li razvijati aplikaciju kao nativnu ili kao web aplikaciju. Bitno je znati kako se jedne razlikuju od drugih te koje su im opće poznate prednosti i mane. Nativne aplikacije su aplikacije razvijene isključivo za jedan određeni mobilni uređaj i instaliraju se izravno na sami uređaj. Korisnici nativnih aplikacija obično ih preuzimaju putem raznih trgovina za aplikacije

na internetu ili na tržištu aplikacije kao što su Apple App Store, Google Play Store itd. Primjer nativne aplikacije je Camera+ za Appleove iOS uređaje. Web aplikacije su, s druge strane, u osnovi aplikacije aktivne na internetu, te su dostupne putem web preglednika na mobilnom uređaju. One ne moraju biti skinute na korisnikov mobilni uređaj kako bi im se pristupilo. Safari je dobar primjer mobilne web aplikacije.

Razvoj za Android je proces kojim se stvaraju nove aplikacije za Android OS. One se obično razvijaju u Java programskom jeziku koristeći Android SDK, ali postoje i drugi dostupni alati. Knjižnice podataka pisanih u C i C++ se mogu kompajlirati u ARM, MIPS ili x86 nativne kodove te instalirati koristeći SDK. Nativne klase se mogu pozvati iz Java koda te su standardni dio Android Java klasa. Čitave aplikacije se mogu kompajlirati i instalirati koristeći tradicionalne alate za razvoj. Međutim, prema Android dokumentaciji, NDK se ne bi trebao koristiti u svrhe razvoja aplikacija samo zato što se programer bolje snalazi u C/C++ jer korištenje NDK povećava kompleksnost te većina aplikacija od toga nema koristi.

Grafička knjižnica koju koristi Android za kontrolu pristupa uređaju naziva se Skia Graphics Library (SGL) koja se nalazi pod licencom otvorenog koda. Skia ima mogućnosti razvoja i za UNIX i za Win32 te time omogućuje razvoj aplikacija za više platformi te je također grafički modul koji pokreće Google Chrome.

Za razliku od razvoja aplikacija baziranih na Eclipse IDE, NDK se bazira na alatima koji koriste komandne linije te zahtijeva manualno pozivanje naredbi za izgradnju, smještanje i uklanjanje greški iz aplikacija.

Sa točke gledišta korisnika mobilnog uređaja, neke nativne i web aplikacije izgledaju i rade na isti način s vrlo malo razlike između njih. Izbor između ove dvije vrste aplikacija mora biti učinjen samo kad treba odlučiti hoće li se razvijati aplikacija s fokusom na korisnika ili na aplikaciju. Neke tvrtke razvijaju nativne i web aplikacije kako bi proširile doseg svojih aplikacija, a također pružaju dobro cjelokupno korisničko iskustvo. [14]

Proces razvoja ove dvije vrste aplikacija je ono što razlikuje jedne od drugih. Svaka mobilna platforma za koju se razvija nativna aplikacija propisuje svoj jedinstveni razvojni proces. U slučaju web aplikacija koje se izvode na web pregledniku mobilnog

uređaja, problem koji se javlja je taj da svaki od tih mobilnih uređaja ima jedinstvene značajke, ali zbog toga također dolazi sa svojim jedinstvenim problemima. Svaka mobilna platforma koristi drugi nativni programski jezik. Dok iOS koristi Objective-C, Android koristi Javu, Windows Mobile koristi C++ itd. Web aplikacije s druge strane, koriste jezike kao što su JavaScript, HTML 5, CSS3 ili druge izradbene okvire web aplikacija po željama programera. Svaka mobilna platforma nudi programeru svoj standardizirani SDK, razvojne alate i druge elemente korisničkog sučelja koje on može koristiti za razvoj vlastite nativne aplikacije s relativnom lakoćom. U slučaju web aplikacije, ipak ne postoji takva standardizacija i programer nema pristup SDK ili alatima bilo koje vrste. Usprkos tome, postoji nekoliko alata i razvojnih okvira koji su dostupni, a pomoću kojih on može implementirati aplikacije na više mobilnih platformi i web preglednike.

Nativne aplikacije su potpuno kompatibilne s hardverom uređaja i nativnim obilježjima kao što su brzinomjer, žirometar, kamere itd. Iako nativne aplikacije rade kao samostalni entiteti, problem je što korisnik mora stalno skidati nadogradnje [15].

Za razliku od njih, web aplikacije mogu pristupiti samo ograničenoj količini nativnih funkcija prijenosnih uređaja [16]. To su zapravo internetske stranice koje se ponašaju kao aplikacije. Web aplikacija se sama nadograđuje bez potrebe za korisničkom intervencijom. No, njima se nužno može pristupiti samo preko web preglednika [17].

Google play je Google-ova online trgovina, te ona danas broji više od milijun i dvjesto tisuća aplikacija. Čak je 80% aplikacija besplatno, dok se samo 20% aplikacija naplaćuje. Korisnici na Google Play mogu kupovati aplikacije, glazbu, filmove, igre i elektroničke knjige. Prilikom prodaje aplikacije Google korisniku isplaćuje 70% cijene aplikacije, a ostatkom se financiraju pružatelji usluga za online plaćanje. Za objavljivanje aplikacija na Market-u korisnici moraju posjedovati Google korisnički račun te kod prve prijave na Android tržište potrebno je uplatiti iznos od 25 američkih

dolara na ime računa. Prije objavljivanja same aplikacije programer mora proći kroz niz uputa i zadataka koje se nalaze na stranici za Google programere.

U srpnju 2013. Google Play Store je imao objavljenih više od 1 milijun Android aplikacija te više od 50 milijardi skinutih aplikacija. Istraživanje programera provedeno u razdoblju od travnja do svibnja 2013. je otkrilo da 71% mobilnih programera razvijaju za Android. Na konferenciji Google I/O 2014., tvrtka je otkrila da trenutno postoji više od jedne milijarde aktivnih mjesečnih Android korisnika [18] (aktivnih u trajanju od najmanje 30 dana), što je povećanje od 538 milijuna u lipnju 2013. Androidov izvorni kod je objavljen od strane Googlea pod besplatnim licencama, iako većina Android uređaja u konačnici na tržište izlazi sa kombinacijom otvorenog koda (eng. open source) i vlasničkog softvera [19].

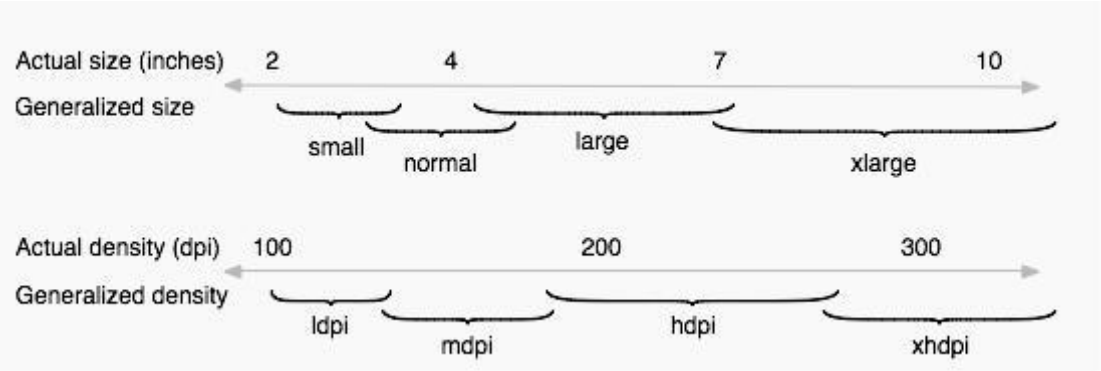
Zarada na nativnim aplikacijama može biti prilično teška jer pojedini proizvođači mobilnih uređaja mogu postaviti ograničenja na integrirane usluge kod određenih mobilnih oglasnih platformi i mreža. Suprotno tome, web aplikacije omogućuju ostvarivanje zarade putem reklama te punjenjem članarine. Važna razlika je u načinu isplate. Kod nativnih aplikacija odabrana trgovina aplikacijama vodi brigu o prihodima i troškovima, dok za web aplikacije *developer* sam treba postaviti vlastiti sustav plaćanja.

Nativne aplikacije su također skuplje za razvoj. Međutim, one su brže i učinkovitije jer rade u tandemu s mobilnim uređajem za koji su razvijene. Također, one su uvjerenje u kvalitetu s obzirom da im korisnici mogu pristupiti samo preko specijaliziranih trgovina na internetu. Web aplikacije mogu rezultirati višim troškovima održavanja na više mobilnih platformi. Također ne postoji specifično regulatorno tijelo za kontrolu standarda kvalitete tih aplikacija.

S obzirom da je Android platforma napravljena za rad na raznovrsnim uređajima (satovi, pametne naočale, smartphone, tableti..) koji imaju različite veličine i rezolucije zaslona, potrebno je aplikaciju prilagoditi kako bi izgledala dobro na svakom od tih uređaja. Android sustav pomoću svojeg koda rasterećuje uređaj te podešava aplikaciju rezoluciji uređaja na kojem se nalazi. [20]

Iako Android na sebe preuzima odgovornost skaliranja kako bi aplikacija radila na različitim zaslonima, *developer* bi trebao na sebe preuzeti odgovornost za optimizaciju. Time se poboljšava iskustvo korisnika te imaju dojam da je aplikacija rađena upravo za njihov uređaj pa sam na taj način i ja radio grafičko sučelje aplikacije za praćenje sportskog treninga.

Kako bi se pojednostavio način dizajna sučelja za više uređaja, Android razlikuje niz stvarnih veličina zaslona i gustoće u nekoliko generaliziranih veličina (slika 5).



Slika 5. Prikaz Androidovog grupiranja raznih rezolucija u veće skupine

Kako bi se osiguralo da je prikazani sadržaj fleksibilan i prilagodljiv, potrebno je koristiti *wrap_content* i *match_parent* naredbe. *Wrap_content* namješta visinu i širinu elementa na najmanju potrebnu kako bi element odgovarao spremniku, dok *match_parent* postavlja visinu i širinu elementa kako bi odgovarao većem elementu u kojem se nalazi. [21]

3. Eksperimentalni dio

3.1. Anketa

Tijekom odlučivanja o sadržaju aplikacije za praćenje treninga vodio sam se svojim potrebama te željama i razmišljanjima vlastite okoline koja predstavlja raznolik i reprezentativan uzorak. Kako bih doznao što više o njihovim potrebama i trenutnom znanju, odlučio sam izraditi kratku anketu (slika 6) sa najbitnijim pitanjima vezanima uz kvalitetan trening. Anketa provedena na uzorku od 10 osoba pokazala je da gotovo svi imaju potrebu za jednom sveobuhvatnom aplikacijom koja bi im bila pomoć tijekom i nakon treninga. Rezultati ankete bili su slični mojim pretpostavkama te sam odlučio napraviti aplikaciju koja je od velike pomoći početnicima te može dobro doći čak i naprednim sportašima.

Smatrate li da vam je potreban kvalitetan dnevnik za praćenje napetka kroz treninge?

		Response Percent	Response Total
Da		80.00 %	8
Ne		20.00 %	2
		Total Respondents	10
		(skipped this question)	0

Smatrate li da bi vam koristila aplikacija koja sadrži kompletan pregled svih potrebnih elemenata za uspješan trening?

		Response Percent	Response Total
Da		100.00 %	10
Ne		0.00 %	0
		Total Respondents	10
		(skipped this question)	0

Ocijenite svoje znanje o prehrani za postizanje željenih rezultata

		Response Percent	Response Total
1		20.00 %	2
2		40.00 %	4
3		10.00 %	1
4		30.00 %	3
5		0.00 %	0
		Total Respondents	10

Ocijenite svoje znanje o kvalitetnom treningu u teretani

		Response Percent	Response Total
1		10.00 %	1
2		10.00 %	1
3		20.00 %	2
4		30.00 %	3
5		30.00 %	3
		Total Respondents	10

Slika 6. Rezultati ankete

3.2. Optimizacija izrade nativne aplikacije za praćenje sportskog treninga

Izrada aplikacije počinje definiranjem izgleda i sadržaja početne aktivnosti koju korisnik prvu ugleda kad pokrene aplikaciju. Kako bi aplikacija bila što preglednija i jednostavnija za korištenje, početna aktivnost ove aplikacije (slika 7) prikazuje ime aplikacije, logo i 3 opcije za nastavak rada: unos kalorija, trening i princip treninga. Tu ujedno započinje XML kodiranje (programski isječak 4) i Java programiranje (programski isječak 5). Odabirom opcije *Unos kalorija*, korisnika aplikacija vodi kroz nekoliko jednostavnih parametara pomoću kojih će mu izračunati dnevni iznos potrebnih kalorija. Opcija *Trening* služi za vođenje zabilješki o podignutim kilažama tijekom treninga dok opcija *Princip treninga* sadrži tekst s jednostavnim objašnjenjem zašto je ovaj trening učinkovit te daje korisniku informaciju što, kada i koliko jesti.

Slika 7. Grafički prikaz početnog zaslona aplikacije

Elementi koje korisnik može odabrati su izrađeni u Photoshopu te dodani u aplikaciju kroz sučelje Eclipse-a unutar XML koda.

```
<RelativeLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    tools:context=".MainActivity" >

    <ImageButton
        android:id="@+id/imgBtn1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentTop="true"
        android:layout_centerHorizontal="true"
        android:layout_marginTop="70dp"
        android:src="@drawable/unos" />

    <ImageButton
        android:id="@+id/imgBtn2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_below="@+id/imgBtn1"
        android:layout_centerHorizontal="true"
        android:layout_marginTop="74dp"
        android:src="@drawable/trening" />

    <ImageButton
        android:id="@+id/imgBtn3"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_below="@+id/imgBtn2"
        android:layout_centerHorizontal="true"
        android:layout_marginTop="73dp"
```

```
        android:src="@drawable/princip" />
</RelativeLayout>
```

Programski isječak 4. XML kod početnog zaslona

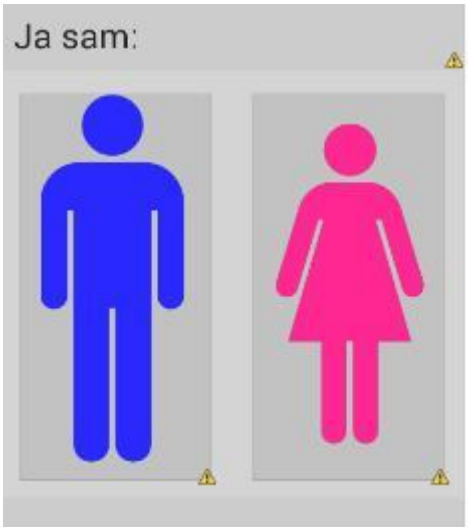
Kako bi elementi ispunjavali svoju namjenu, njima Java kodom moramo dodati funkcije koje će obavljati radnje temeljene na korisnikovim postupcima i unosima podataka.

```
unos = (ImageButton) findViewById(R.id.imgBtn1);
            trening = (ImageButton) findViewById(R.id.imgBtn2);
            princip = (ImageButton) findViewById(R.id.imgBtn3);

            unos.setOnClickListener(new OnClickListener() {

                  @Override
                  public void onClick(View arg0) {
                        // TODO Auto-generated method stub
                        startActivity(new Intent(MainActivity.this,
UnosKalorija1.class));
                  }
            });
                        trening.setOnClickListener(new
OnClickListener() {
                                    @Override
                  public void onClick(View arg0) {
                        // TODO Auto-generated method stub
                        startActivity(new Intent(MainActivity.this,
TreningActivity.class));
                  }
            });
                        princip.setOnClickListener(new
OnClickListener() {

                  @Override
                  public void onClick(View arg0) {
                        // TODO Auto-generated method stub
                        startActivity(new Intent(MainActivity.this,
InfoActivity.class));
                  }
```

```
        });
    }
```

Programski isječak 5. Java kod početnog zaslona

Kad korisnik poželi izračunati potreban iznos kalorija te pritisne gumb *Unos kalorija*, otvara se zaslon sa upitom za odabir spola (slika 8).



Slika 8. Grafički prikaz prve aktivnosti pod Unos kalorija

Pozadina izgleda u XML (programski isječak 6) kodu donosi dvije intuitivne opcije – slike standardnih simbola za muški i ženski rod koje služe kao aktivatori sljedeće aktivnosti u dijelu aplikacije za izračun kalorija.

```
<TextView
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:padding="10dp"
        android:text="Ja sam: "
        android:textColor="#232323"
        android:textSize="24dp" />

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:orientation="horizontal" >
```

```
        <ImageButton
            android:id="@+id/maleButton"
            android:layout_width="0dp"
            android:layout_height="275dp"
            android:layout_margin="10dp"
            android:layout_weight="1" />

        <ImageButton
            android:id="@+id/femaleButton"
            android:layout_width="0dp"
            android:scaleType="centerCrop"
            android:adjustViewBounds="true"
            android:src="@drawable/ic_launcher"
            android:layout_height="275dp"
            android:layout_margin="10dp"
            android:layout_weight="1" />
    </LinearLayout>
```

Programski isječak 6. XML kod prve aktivnosti pod Unos kalorija

```
muski = (ImageButton) findViewById(R.id.maleButton);
            zenski = (ImageButton) findViewById(R.id.femaleButton);
                        daljeIntent = new Intent(UnosKalorija1.this,
UnosKalorija2.class);
                        muski.setOnClickListener(new OnClickListener()
{
                                    @Override
                    public void onClick(View arg0) {
                        // TODO Auto-generated method stub
                        daljeIntent.putExtra(SPOL, MUSKO);
                        startActivity(daljeIntent);
                    }
            });
                        zenski.setOnClickListener(new OnClickListener()
{
                                    @Override
                    public void onClick(View arg0) {
                        // TODO Auto-generated method stub
                        daljeIntent.putExtra(SPOL, ZENSKO);
```

```
                startActivity(daljeIntent);
            }
});
```

Programski isječak 7. Java kod prve aktivnosti pod izbornikomUnos kalorija

U Java kodu su slike također jasno definirane (programski isječak 7) jer će, ovisno o odabiru korisnika, iduća aktivnost na temelju odabranog spola bez znanja korisnika primijeniti određene formule koje u kombinaciji s tjelesnim parametrima (slika 9) i tjelesnom aktivnošću korisnika (slika 10) računaju potreban dnevni unos kalorija.

Slika 9. Grafički prikaz druge aktivnosti pod izbornikom Unos kalorija

Format teksta definiran kroz sljedeći XML kod (programski isječak 8) se ponavlja još nekoliko puta na sličan način za ostale opcije koje se prikazuju na zaslonu mobilnog uređaja.

```
<LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:orientation="horizontal"
        android:padding="10dp" >

        <TextView
            android:layout_width="0dp"
            android:layout_height="wrap_content"
```

```
            android:layout_weight="0.35"
            android:text="Kilograms"
            android:textColor="#232323"
            android:textSize="24dp" />

        <EditText
            android:id="@+id/kgEditTxt"
            android:layout_width="0dp"
            android:layout_height="wrap_content"
            android:layout_weight="0.65"
            android:hint="65 kg"
            android:inputType="number"
            android:textColor="#232323"
            android:textSize="24dp" />
    </LinearLayout>
```

Programski isječak 8. XML kod druge aktivnosti pod izbornikom Unos kalorija

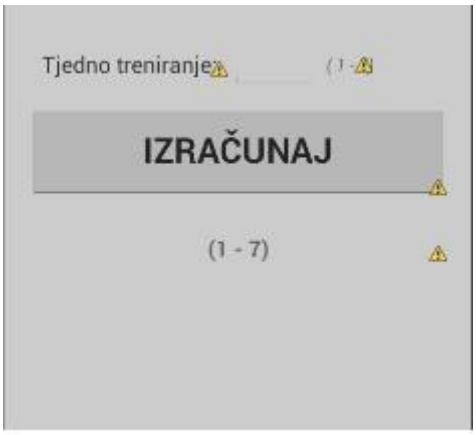
Kroz sljedeći kod (programski isječak 9) aplikacija pamti unesene tjelesne parametre korisnika te ih uvrštava u jednu od formula baziranih na odabiru spola korisnika iz prethodne aktivnosti.

```
public class UnosKalorija2 extends Activity {

      EditText kgEditTxt, visinaEditTxt, dobEditTxt;
      Button daljeButton;

      int spol;

      @Override
      protected void onCreate(Bundle savedInstanceState) {
            // TODO Auto-generated method stub
            super.onCreate(savedInstanceState);
            setContentView(R.layout.unos_kalorija2);

            spol = getIntent().getIntExtra(UnosKalorija1.SPOL, 1);

            kgEditTxt = (EditText) findViewById(R.id.kgEditTxt);
            visinaEditTxt = (EditText)
findViewById(R.id.visinaEditTxt);
```

```
            dobEditTxt = (EditText) findViewById(R.id.dobEditTxt);

            daljeButton = (Button) findViewById(R.id.daljeBtn);
            daljeButton.setOnClickListener(new OnClickListener() {

                  @Override
                  public void onClick(View arg0) {
                        // TODO Auto-generated method stub
                        float kg = 0, visina = 0, dob = 0;

                        try { kg =
Float.parseFloat(kgEditTxt.getText().toString());
                              visina =
Float.parseFloat(visinaEditTxt.getText()
                                          .toString());
                              dob =
Float.parseFloat(dobEditTxt.getText().toString());
                        } catch (NumberFormatException e) {
                              e.printStackTrace();
                        }

                        float rezultat = 0;

if (spol == UnosKalorija1.MUSKO) {
                              rezultat = 66 + (13.7f * kg) + (5f *
visina) - (6.76f * dob);
                        } else {
                              rezultat = 655 + (9.6f * kg) + (1.8f *
visina) - (4.7f * dob); }

                        Log.w("REZULTAT", "rezultat je " + rezultat);

                        Intent intent = new Intent(UnosKalorija2.this,
UnosKalorija3.class);
                        intent.putExtra("rez", rezultat);
                        startActivity(intent);
```

Programski isječak 9. Java kod druge aktivnosti pod izbornikom Unos kalorija

Završna aktivnost u dijelu aplikacije za izračun kalorija traži od korisnika da unese broj treninga koje trenutno obavlja tjedno ili ih planira obavljati te na temelju toga množi prethodno dobivene rezultate sa određenim faktorom. Ako je fizička aktivnost veća, potrebno je više kalorija kako bi se održavala tjelesna masa i obratno.



Slika 10. Grafički prikaz treće aktivnosti pod izbornikom Unos kalorija

XML kod ovdje ne donosi mnoge novitete (programski isječak 10), nego se pomoću njega prikazuje polje za unos posljednjeg potrebnog parametra za izračun kalorija te gumb za konačni izračun.

```
<LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_gravity="center_vertical"
        android:gravity="center_vertical"
        android:orientation="horizontal"
        android:padding="10dp" >

        <TextView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:paddingRight="5dp"
            android:text="Tjedno treniranje: "
            android:textSize="15sp" />

        <EditText
            android:id="@+id/ex_editext"
            android:layout_width="60dp"
```

```
            android:layout_height="wrap_content"
            android:digits="0123456789."
            android:inputType="number"
            android:textSize="15sp" />

        <TextView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:paddingLeft="5dp"
            android:text="( 1 - 7)"
            android:textColor="#80000000"
            android:textSize="12sp"
            android:textStyle="italic" />
    </LinearLayout>
```

Programski isječak 10. XML kod treće aktivnosti pod izbornikom Unos kalorija

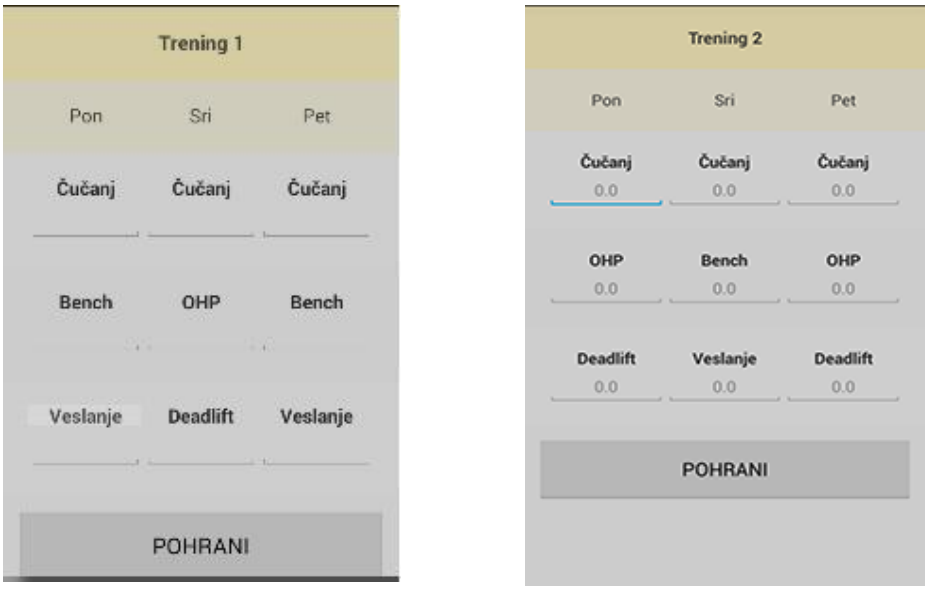
Za razliku od XML-a, u Java kodu se događa mnogo toga (programski isječak 11). Posljednja aktivnost preuzima rezultat iz prethodne aktivnosti dobiven iz formule s unesenim spolom, visinom, kilažom i dobi korisnika. Taj rezultat množi sa određenim faktorom koji ovisi o trenutnoj ili planiranoj tjelesnoj aktivnosti korisnika te prikazuje konačni iznos potrebnih kalorija.

```
public class UnosKalorija3 extends Activity {

      private EditText mEditText;
      private Button mCalculateBtn;

      private float mRez;

      private float mCalories;

      @Override
      protected void onCreate(Bundle savedInstanceState) {
            // TODO Auto-generated method stub
            super.onCreate(savedInstanceState);
            setContentView(R.layout.unos_kalorija3);
```

```
            mEditText = (EditText) findViewById(R.id.ex_editext);
            mCalculateBtn = (Button) findViewById(R.id.btn_calculate);

            mCalculateBtn.setOnClickListener(new OnClickListener() {

                  @Override
                  public void onClick(View v) {
                        int num =
Integer.valueOf(mEditText.getText().toString());

                        switch (num) {
                        case 0:
                        case 1:
                        case 2:
                              mCalories = mRez * 1.375f;
                              break;
                        case 3:
                        case 4:
                        case 5:
                              mCalories = mRez * 1.55f;
                              break;
                        case 6:
                        case 7:
                              mCalories = mRez * 1.725f;
                              break;
                        default:
                              break;
                        }

      ((TextView)findViewById(R.id.cal_txtview)).setText("Potrebno vam
je " + (int) mCalories + " kalorija.");
                  }
            });
            mRez = getIntent().getFloatExtra("rez", 0);

      }
}
```

Programski isječak 11. Java kod treće aktivnosti pod izbornikom Unos kalorija

Sljedeća aktivnost do koje se dolazi preko glavnog izbornika u glavnoj aktivnosti je dnevnik treninga (slika 11).



Slika 11. Prikaz dnevnika za parne i neparne dane treninga

Program treninga traje 3 mjeseca te sadrži 3 treninga tjedno. Za svaki trening u tjednu su dane vježbe te mogućnost unosa podignutih kilaža kako bi se uspješno pratila krivulja napretka te kako bi se nastavilo sa istom kilažom na idućem treningu u slučaju eventualnog neuspjeha.

Svaki tjedan sadrži 9 polja za unos te gumb za pohranu unesenih vrijednosti. Primjer polja za unos prikazan je na slici ispod (programski isječak 12).

```
<TextView
    android:id="@+id/txt1"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_gravity="center_horizontal"
    android:gravity="center_horizontal"
    android:text="@string/_u_anj"
    android:textColor="#FA232323"
    android:textSize="15sp"
```

```
            android:textStyle="bold" />

        <EditText
            android:id="@+id/ett1"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_gravity="center_horizontal"
            android:digits="0123456789."
            android:gravity="center_horizontal"
            android:inputType="number"
            android:textColor="#77232323"
            android:textSize="15sp" />
```

Programski isječak 12. XML kod aktivnosti Trening

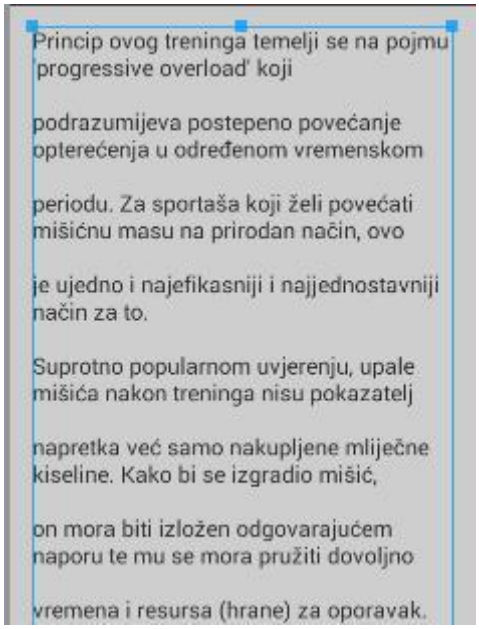
Java kod koncipiran je tako da omogućuje pamćenje svake unešene vrijednosti u obliku decimalne brojke pritiskom na gumb „Pohrani“ (programski isječak 13).

```
public void onClick(View v) {
                        float ett1 = Float

      .valueOf((et1.getText().toString().length() > 0) ? et1
                                                .getText().toString() :
"0");
                        float ett2 = Float

      .valueOf((et2.getText().toString().length() > 0) ? et2
                                                .getText().toString() :
"0");
                        float ett3 = Float

      .valueOf((et3.getText().toString().length() > 0) ? et3
                                                .getText().toString() :
"0");
                        float ett4 = Float

      .valueOf((et4.getText().toString().length() > 0) ? et4
                                                .getText().toString() :
"0");
                        float ett5 = Float

      .valueOf((et5.getText().toString().length() > 0) ? et5
                                                .getText().toString() :
"0");
                        float ett6 = Float

      .valueOf((et6.getText().toString().length() > 0) ? et6
                                                .getText().toString() :
"0");
```

```
                        float ett7 = Float

      .valueOf((et7.getText().toString().length() > 0) ? et7
                                                .getText().toString() :
"0");
                        float ett8 = Float

      .valueOf((et8.getText().toString().length() > 0) ? et8
                                                .getText().toString() :
"0");
                        float ett9 = Float

      .valueOf((et9.getText().toString().length() > 0) ? et9
                                                .getText().toString() :
                                    "0");
```

Programski isječak 13. Manji dio Java koda za aktivnost Trening

U trećoj aktivnosti dostupnoj preko početnog zaslona otvara se tekst sa osnovnim objašnjenjima kako i zašto ova vrsta treninga funkcionira te koncept prehrane koji će pružiti rezultate svakome tko ga se pridržava (slika 12). U Java kodu se otvara nova aktivnost kao što je viđeno u prethodnim slučajevima, a sav sadržaj se unosi preko XML datoteke unutar aplikacije.

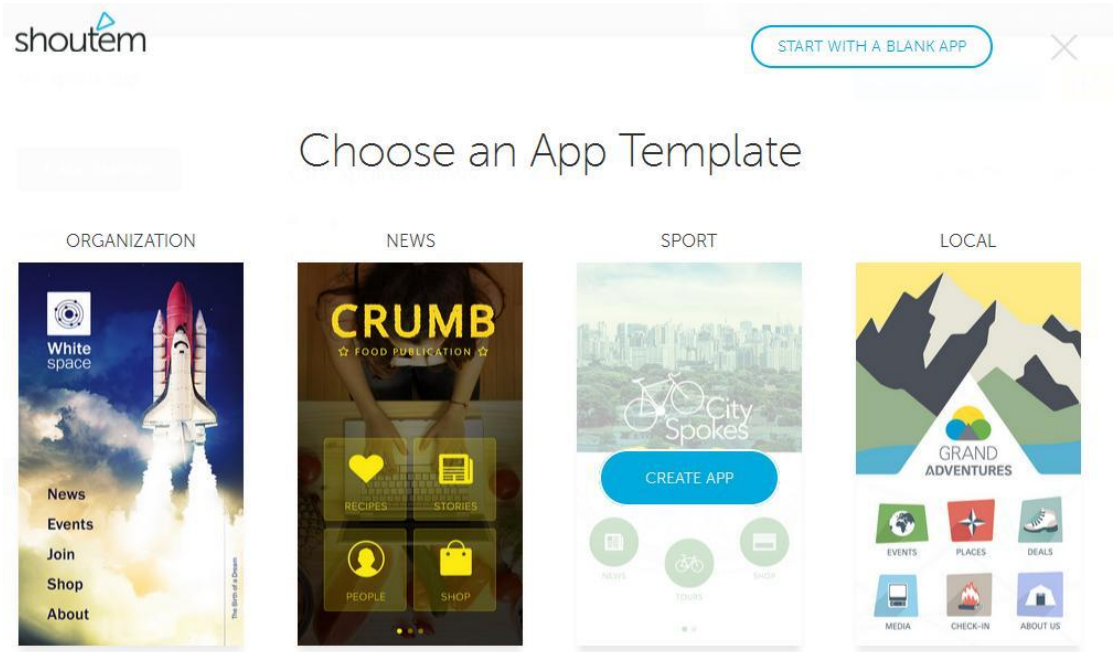


Slika 12. Grafički prikaz aktivnosti Princip treninga

3.3. Izrada aplikacije u ShoutEm web servisu

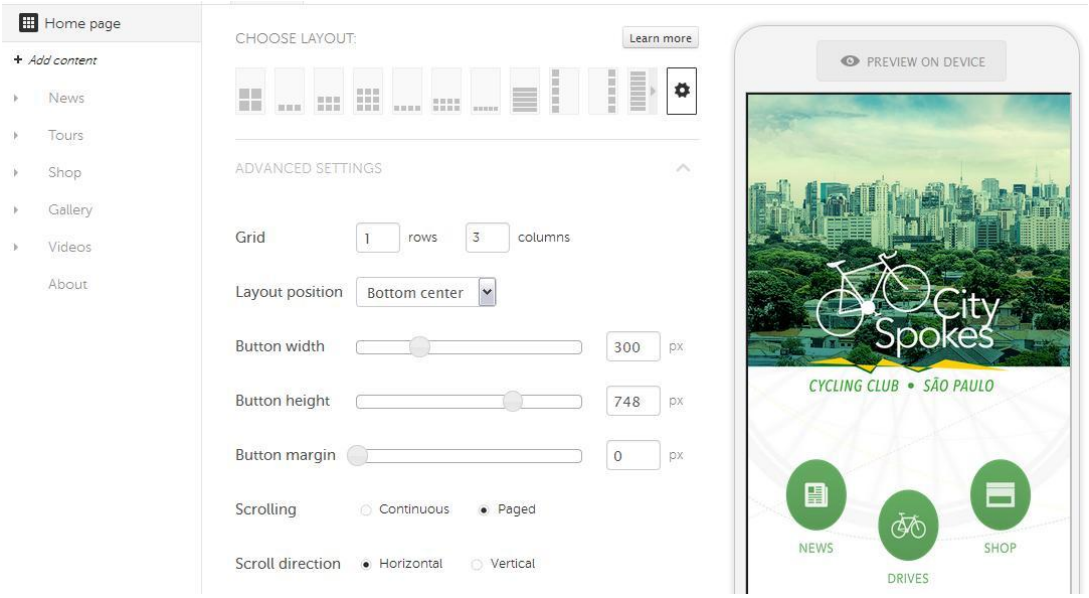
ShoutEm je jedan od projekata tvrtke za mobilni razvoj Five Minutes Ltd sa uredima u SAD-u i Zagrebu. Predstavlja online servis za brzu i jednostavnu izradu mobilnih aplikacija za iOS, Android i HTML5 u samo nekoliko minuta. Nudi široku paletu modula za pomoć malim i srednjim poduzećima te pojedincima koji žele stvoriti vlastite personalizirane aplikacije za njihov sadržaj ili društvene mreže te je ujedno ovim grupama prvenstveno namijenjen. S obzirom da danas većina tvrtki želi uz kvalitetnu web stranicu imati i svoju aplikaciju kako bi povećali prihode (npr. Amazon, Ebay) te da se puno pojedinaca želi okušati u izradi aplikacija za uređaj koji posjeduju, a nemaju dovoljno znanja u programiranju, ovakvi servisi predstavljaju savršeno rješenje.

Početni zaslon odaje dojam kvalitetnog i ozbiljnog alata za izradu aplikacija te prvo nudi odabir platforme kojoj je aplikacija namijenjena te odabir predloška za aplikaciju (slika 13).



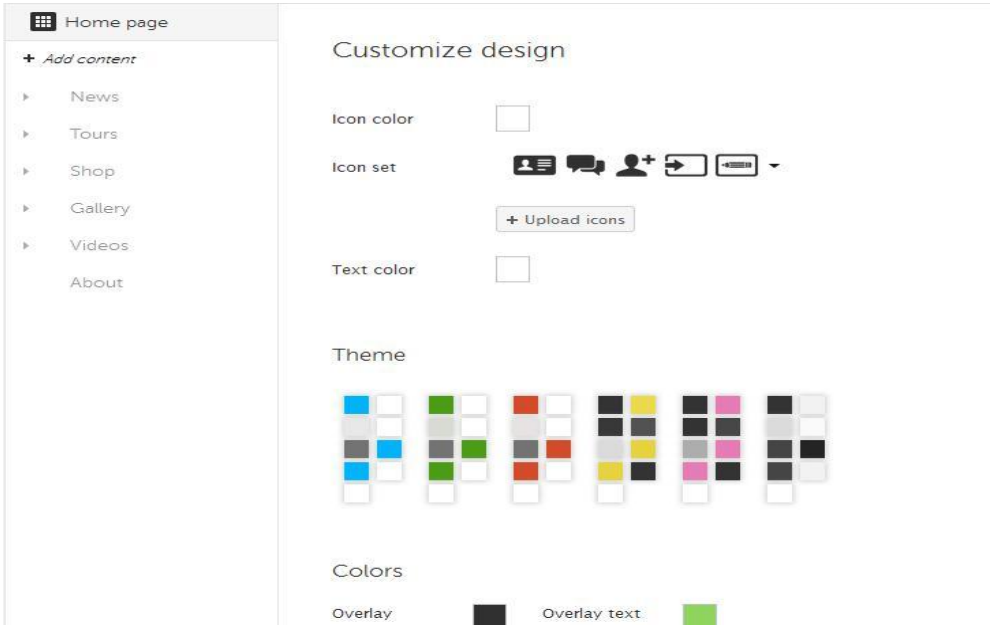
Slika 13. Odabir predloška za aplikaciju

Nakon odabira predloška ShoutEm servis nudi izbornik u kojem je moguće modificirati dobiven predložak s već gotovim opcijama (slika 14). Glavne značajke su opširno uređivanje početnog zaslona na kojem je moguće brzo i lako mijenjati dimenzije objekata poput slika i gumbova.



Slika 14. Opcije za podešavanje izgleda početnog zaslona

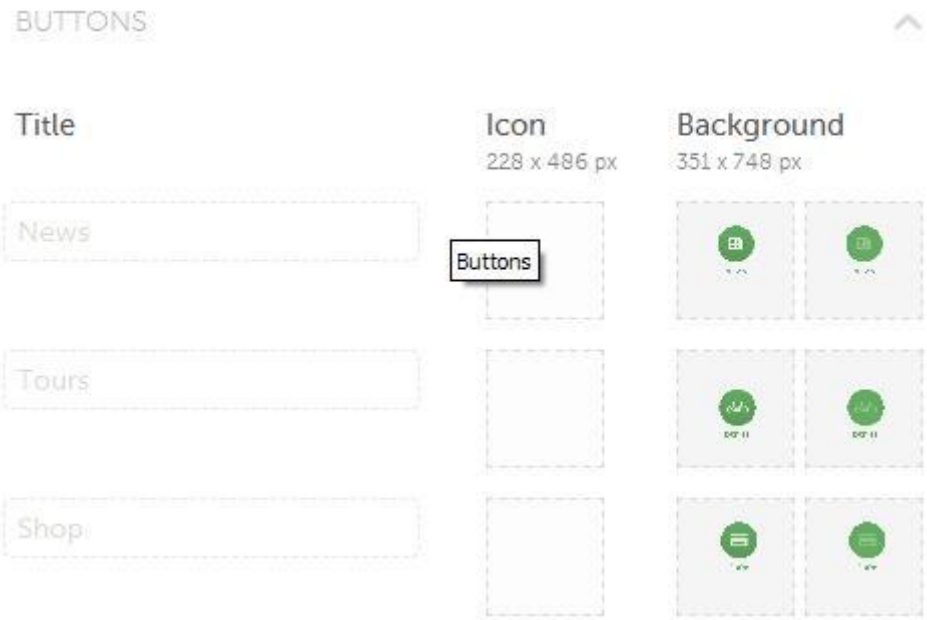
Dodatne opcije uključuju mijenjanje boja i fontova te promjena slika i ikona u samoj aplikaciji bilo sa ponuđenim sadržajem, bilo s vlastitim slikama.



Slika 15. Podešavanje boja i fonta

ShoutEm u početku impresionira ponuđenim mogućnostima od kojih su najdojmljivije mogućnosti vezane uz društvene mreže te jednostavno objavljivanje novih vijesti, no ubrzo na površinu izlaze njegove ogromne mane.

Koliko mogućnosti pruža u smislu grafičkog uređivanja (slika 15), toliko mu nedostaje u uređivanju sadržaja. Osim zadanih kategorija, nemoguće je napraviti bilo što unikatno u smislu dodavanja vlastitih gumbova i slika za njih. Podesive veličine elemenata, posebice gumbova se automatski primjenjuju na sve elemente te vrste pa je nemoguće biti originalan i imati raznovrsne elemente u aplikaciji rađenoj u ShoutEm.

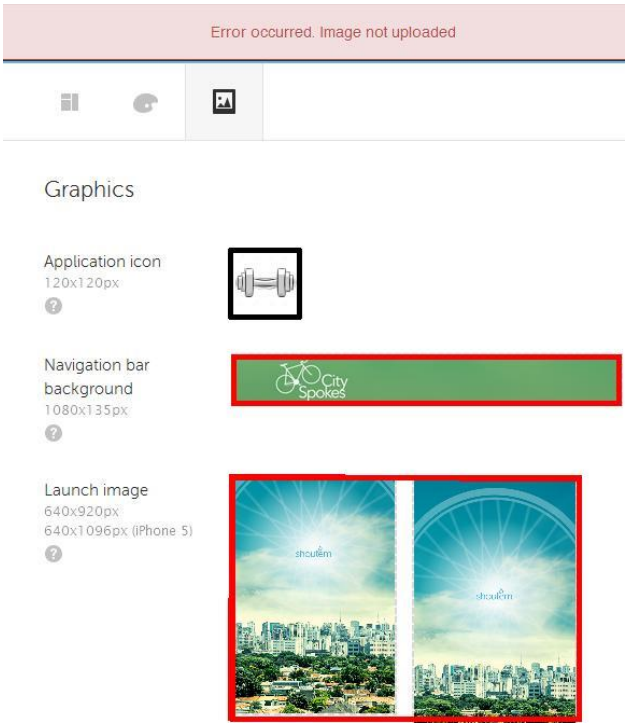


Slika 16. Jednoličnost gumbova

Kako je prikazano na slici iznad, prvi dojam o nebrojenim mogućnostima je bio netočan jer su mogućnosti izrade nečeg originalnog gotovo nikakve u usporedbi sa varijacijama koje mi je pružila Java u kombinaciji sa XML-om (slika 16).

Puno problema je pružio i upload već pripremljenih slika koje su korištene u izradi nativne Android aplikacije. Naime, tijekom modifikacije ShoutEm aplikacije istim slikama kako bi se pokušala izraditi ista ili barem slična aplikacija na 2 načina radi usporedbe, ShoutEm je u potpunosti zakazao (slika 17). Jedina slika od svih pripremljenih koju je prihvatio je ikona aplikacije, dok je za sve ostale slike objavio grešku prilikom uploada slika različitih formata, što dovodi do zaključka kako ShoutEm

ima interni problem sa formatima. Na slici ispod u crnom okviru prikazana je uspješno uploadana slika, a u crvenim okvirima prikazane su sve lokacije gdje su trebale biti uploadane slike, no nisu prihvaćene.



Slika 17. Nemogućnost uploada slika

Još jedan veliki nedostatak ShoutEm naspram izrade aplikacije u Javi je što se objava aplikacija, kako bi se mogle skinuti na uređaje, naplaćuje. To i ne bi bio veliki problem, ali s obzirom da neograničen broj preuzimanja nudi samo najskuplji paket te da su mogućnosti samog servisa zapravo nepostojeće u odnosu na programirane aplikacije, svaka cijena je prevelika u odnosu na Google Play.

Na kraju donosim, po meni, najveći nedostatak ShoutEm servisa, a to je potpuna nemogućnost programiranja ikakvih radnji. Ovo predstavlja manu koja ShoutEm čini neozbiljnim te nedostojnim pažnje bilo koga tko želi da njegova aplikacija radi nešto izvan postojećih predložaka. Tako sam nakon nekoliko dana pokušavanja odustao od ShoutEm servisa u kojem nije bilo moguće napraviti ništa (što bi nalikovalo nativnoj aplikaciji iz prvog dijela eksperimentalnog rada) osim uvidjeti njegove (ne)mogućnosti.

4. Rezultati i rasprava

Induktivna metoda je dokazala da je Android nativna aplikacija najbolji izbor jer ima najveći broj potencijalnih korisnika (slika 2).

Metodom analize dokazala se prednost izrade grafičkog sučelja u XML-u nad Java objektima.

Statistička metoda analize pokazala je da postoji potreba za aplikacijom koja će sadržavati sve bitne informacije na jednom mjestu.

Eksperimentalna metoda izrade aplikacija te usporedba izrade Android aplikacije pomoću programiranja i ShoutEm servisa, dokazala se ogromna prednost klasičnog načina izrade aplikacija nad jednostavnim servisima. Java i XML pružaju neograničene mogućnosti grafičke modifikacije te programerskih rješenja dok ShoutEm ne može parirati niti u jednom segmentu.

Eksperimentalna metoda je također pokazala da je korištenje „wrap_content“ i „match_parent“ mogućnosti superiorna u prilagodbi aplikacija za sve veličine uređaja u odnosu na strogo definiranje veličina elemenata pomoću vrijednosti u pikselima.

5. Zaključci

Izrada nativne aplikacije je složen, dugotrajan proces koji zahtijeva detaljnu pripremu prije same realizacije te maksimalnu koncentraciju i posvećenost, ali su rezultati nagrađujući. Neograničene mogućnosti grafičke modifikacije i programiranja vlastite aplikacije su jedino čemu bi ozbiljan *developer* trebao posvetiti pažnju.

Android platformu odabrao sam iz razloga što se većina korisnika koristi Androidom, čime je tržište veće. Aplikacija nema strogo definiranu ciljanu dobnu skupinu jer je trening u određenoj mjeri primjenjiv na sve dobne skupine. Java programiranje i XML kodiranje također su mi već bili približeni kroz obrazovanje te mi je to znanje olakšalo eksperimentalni dio ovog rada.

Anketirani korisnici su mi kroz svoje odgovore pokazali u kojem smjeru trebam raditi aplikaciju te je ona na kraju tako i koncipirana. Jednostavnim rječnikom aplikacija korisnika uči najbitnije stvari o treningu, prehrani te mu nudi mogućnost konstantne revizije potrebnog unosa kalorija i dnevnik treninga.

S druge strane, prilikom pokušaja izrade aplikacije u ShoutEm servisu uočeni su veliki nedostaci koje nisam mogao zaobići. Nemogućnost modificiranja aplikacije izvan zadanih predložaka te nepostojanje kodiranja onemogućili su ShoutEm-u da ozbiljnije konkurira programerima. Time je pokušaj izrade aplikacije u ShoutEm-u identične onoj u Javi propao i Java se potvrdila kao dominantan način za izradu aplikacija.

Zaključujem da je Java programiranje zapravo jedini način izrade složenih i kvalitetnih Android aplikacija te se namjeravam posvetiti isključivo tome, dok ShoutEm može poslužiti samo onima koji imaju sreću da se njihova vizija aplikacije podudara sa ShoutEm predlošcima.

6. Literatura

[1] ***Java Technology: The Early Years, Sun Microsystems, dostupno od svibnja 2008.

[2] Dr. Alan Kay; Object-Oriented Programming, 2003.

[3] Gosling, James; McGilton, Henry; The Java Language Environment, svibanj 1996.

[4] Jelovic, Dejan; Why Java will always be slower than C++, dostupno na www.jelovic.com

[5] Gosling, James; Joy, Bill; Steele, Guy; Bracha, Gilad; The Java Language Specification, 2nd Edition, 2006.

[6] Bray, Tim; Paoli, Jean; Maler, Eve; Extensible Markup Language (XML) 1.1 (2nd edition), W3C, 2006.

[7] Quin, Liam R.E.; *What is XML?*, dostupno na: http://www.w3.org/standards/xml/core, 14.05.2014.

[8] Clark, James; DeRose, Steve; XML Path Language (Xpath) 1.0, W3C, 1999.

[9] ***Google Launches Android, an Open Mobile Platform, Google Operating System, 5.11.2007.

[10] ***Touch Devices | Android Open Source, Source.android.com, dostupno od 15.9.2012.

[11] ***What is Android?, Android Developers. 21.7.2009.

[12] Deleon, Walter; *A brief history of Android*, 2014., dostupno na: http://technoblimp.com/2013/09/14/a-brief-history-of-android/

[13] ***http://developer.android.com/about/dashboards/index.html, Android Developers, Dashboards, dostupno 12.8.2014.

[14] Gassée, Jean-Louis; The Silly Web vs. Native Apps Debate, The Silly Web vs. Native Apps Debate, 17.9.2012.

[15] Budiu , Raluca; *Hybrid apps*, dostupno na: http://www.nngroup.com/articles/mobile-native-apps/, 22.05.2014.

[16] Janssen; Cory; *Definition - What does Native Mobile App mean?*, 2014., dostupno na: http://www.techopedia.com/definition/27568/native-mobile-app

[17] Rouse ,Margaret; *Web application (Web app)*, dostupno na: http://searchsoftwarequality.techtarget.com/definition/Web-application-Web-app, 2014.

[18] ***Android device activation numbers reach 1 billion worldwide, 3.9.2013., dostupno na Phandroid.com

[19] Google; About Google Play,. Dostupno od 8.3.2012.

[20] ***http://developer.android.com/guide/practices/screens_support.html, Android Developer, *Supporting Multiple Screens*, dostupno 26.05.2014.

[21] ***http://developer.android.com/guide/topics/ui/declaring-layout.html, Android Developers, Layouts,

[22] Rogers, Rick; Lombardo, John; Mednieks, Zigurd; Meike, Blake; Android Application Development: Programming with the Google SDK, O'Reilly Media Inc, 2009.

[23] Burnette, Ed; *Hello, Android*, 3rd edition, The Pragmatic Bookshelf, Raleigh, North Carolina, 2010.

[24] Sierra, Kathy; Bates, Bert; *Head First Java*, 2nd edition, O'Reilly Media Inc., Sebastopol, California, 2005.

[25] Nakamura, Nasumi; Gargenta, Marko; *Learning Android*, 2nd edition, O'Reilly Media, Sebastopol, California, 2014.

Popis slika:

Izvori slika

Slika 2: http://www.idc.com/prodserv/smartphone-os-market-share.jsp

Slika 3:
Https://developer.android.com/about/dashboards/index.html?utm_source=ausdroid.net

Slika 4: http://www.tutorialspoint.com/android/android_architecture.htm

Slika 5: http://developer.android.com/guide/practices/screens_support.html

Slika 6: http://www.stellarsurvey.com/

Slika 13-17: http://www.shoutem.com/

Popis programskih isječaka