Izrada aplikacije za Android operativni sustav

Javor, Božidar

Undergraduate thesis / Završni rad

2018

Degree Grantor / Ustanova koja je dodijelila akademski / stručni stupanj: **Josip Juraj Strossmayer University of Osijek, Faculty of Electrical Engineering, Computer Science and Information Technology Osijek / Sveučilište Josipa Jurja Strossmayera u Osijeku, Fakultet elektrotehnike, računarstva i informacijskih tehnologija Osijek**

Permanent link / Trajna poveznica: https://urn.nsk.hr/urn:nbn:hr:200:887960

Rights / Prava: In copyright / Zaštićeno autorskim pravom.

Download date / Datum preuzimanja: **2024-03-28**

Repository / Repozitorij:

Faculty of Electrical Engineering, Computer Science and Information Technology Osijek

zir.nsk.hr

Image not found or type unknown

SVEUČILIŠTE JOSIPA JURJA STROSSMAYERA U OSIJEKU

FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA OSIJEK

Sveučilišni studij

IZRADA APLIKACIJE ZA ANDROID OPERATIVNI SUSTAV

ZAVRŠNI RAD

Božidar Javor

Osijek, 2018.

SADRŽAJ

1. UVOD 1
1.1 Zadatak 2
2. ANDROID 2
2.1 Android verzije 4
2.2 Android uređaji 5
3. IZRADA ANDROID APLIKACIJE 7
3.1 Tehnologije izrade 9
3.1.1 Android studio 9
3.1.2 Android SDK 12
3.2 Algoritam sivog predviđanja 12
3.3 Izrada Android aplikacije 16
3.4 Slike iz funkcionalne aplikacije u nekoliko koraka 20
4. ZAKLJUČAK 22
5. SAŽETAK 23
6. ABSTRACT 24
LITERATURA 25
ŽIVOTOPIS 26
PRILOG 1:IZVORNI DIO APLIKACIJE 27

1. UVOD

Ovaj završni rad predstavlja opis sivih sustava, odnosno primjer predviđanja na temelju sivih sustava i izradu android aplikacije koja prikazuje sivo predviđanje. Sivo predviđanje je relativno nova metoda poznata pod nazivom (eng. *grey forecasting*) koja se temelji na teoriji sivih sustava (eng. *grey system theory*). Teorija sivih sustava vrlo je dugo poznata, od 1982. godine, te joj od tada raste popularnost a time i primjena. Danas se koristi u rješavanju raznih problema u različitim područjima.

Sivi sustavi su sustavi za koje nisu poznate sve informacije, odnosno sivi sustav se dijeli na sivi prostor stanja što je zapravo nepoznati dio sustava, te bijeli prostor stanja što čini poznati dio sustava. Cilj je kao što je ranije navedeno iz poznatog dijela informacija saznati ne poznati dio informacija, odnosno iz bijelog dijela sustava saznati sivi dio sustava. Sve je potkrijepljeno matematičkim aparatom.

Kako bi se izradila Android aplikacija potrebno je proći kroz pojedine teorijske dijelove od povijesti Androida i pojašnjenja pojedinih verzija, pa do Android uređaja i primjene. U radu su predstavljeni alati i tehnologije za izradu android aplikacije. Za izradu Android aplikacije potrebne su tehnologije koje su opće poznate svim programerima, a to je program „Android studio“ s svojim dodatcima koji su neophodni prilikom izrade Android aplikacije.

Android prvenstveno predstavlja operativni sustav koji je namijenjen za mobilne uređaje, s naznakom na približavanje računalnim funkcionalnostima na mobilne uređaje. Ta činjenica proizlazi iz potrebe da se operacije koje se obavljaju na računalima približe korisnicima na mobilnim uređajima.

Rad je predstavljen kroz teorijski dio pod nazivom „Android“ koji obuhvaća definiciju Androida te verzije Android sustava ali i uređaje na kojima je osposobljen operativni sustav Android. Nastavak rada predstavljen je kroz izradu Android aplikacije koja započinje tehnologijama prilikom izrade, što obuhvaća programsku podršku prilikom izrade te se nastavlja s opisivanjem algoritma za sivo predviđanje i samom izradom.

1.1 Zadatak

Zadatak završnoga rada je izraditi Android aplikaciju za sivo predviđanje na osnovu modela sivih sustava. Prvenstveno je potrebno opisati Android operativni sustav te načine i tehnologije potrebne za izradu Android aplikacije. Po tome je potrebno na temelju algoritma za sivo predviđanje izraditi Android aplikaciju za sivo predviđanje, odnosno predviđanje na osnovu modela sivih sustava.

2. ANDROID

Android je operativni sustav kojeg koriste mobilni uređaji. Nastao je prvenstveno iz razloga da se funkcionalnost osobnog računala što je moguće brže približi mobilnim uređajima, odnosno sve što radimo na računalu, prilagodimo i na mobilnim aplikacijama.

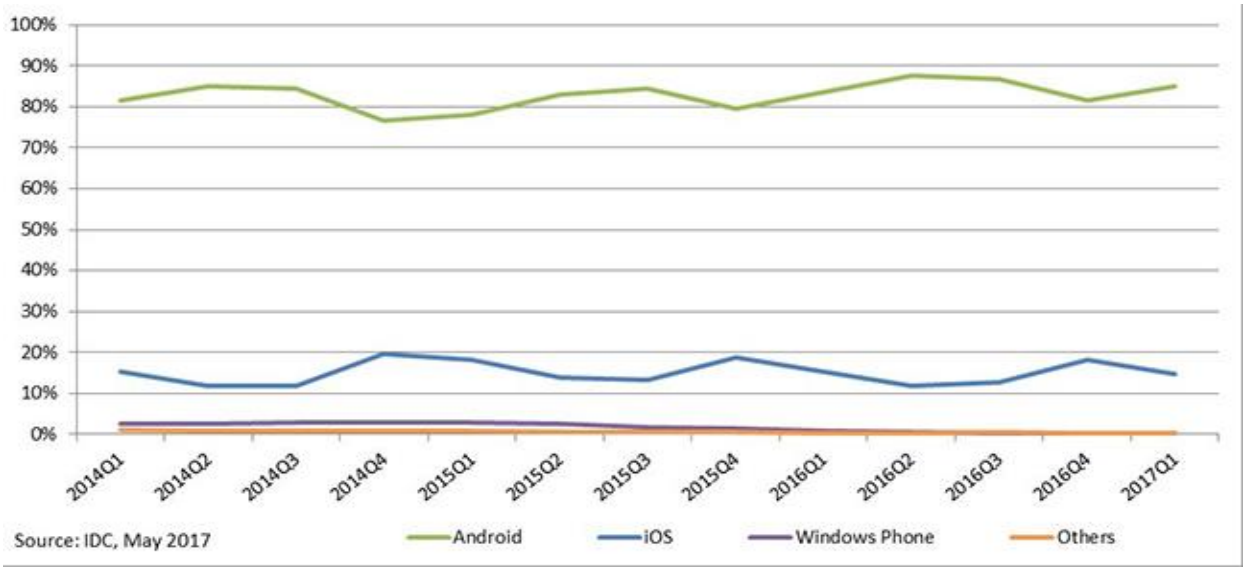


Slika 1. Prikaz Android Logo-a

Karakteristike Google Androida prema OHA(*eng*. *Open Handset Alliance*) su:

- otvorenost - programerima aplikacija omogućena je sloboda u smislu razvoja aplikacije, a proizvođačima je omogućeno besplatno korištenje.
- ravnopravnost aplikacija – nema razlike između aplikacije koju korisnik instalira i osnovne jezgrene aplikacije. Aplikaciji je omogućen pristup sredstvima i to omogućuje izbor korisniku da prilagodi uređaj svojim potrebama.
- mogućnost da se aplikacija pokrene i izvede, a da pri tome korisnik ne mora brinuti o tome je li druga aplikacija pokrenuta prije nego što je prethodna ugašena,
- omogućeno je brže razvijanje aplikacije pomoću baza programskih biblioteka i drugih alata za izradu,
- omogućen je bolji prikaz slike i zvuka te podrška za audio i video formate.

Prema mnogima, Androidi su mnogo primamljiviji zbog svoje jednostavnosti, ali i pristupačnosti, jer je u većini slučajeva Android uređaj povoljniji od npr. Apple uređaja. Slika 2 predstavlja omjer prodanih uređaja po operativnim sustavima. Iz slike je jasno vidljivo da na svim tržištima većinom prednjače Android uređaji, a Europa je najjača u tome s preko 70% [1].



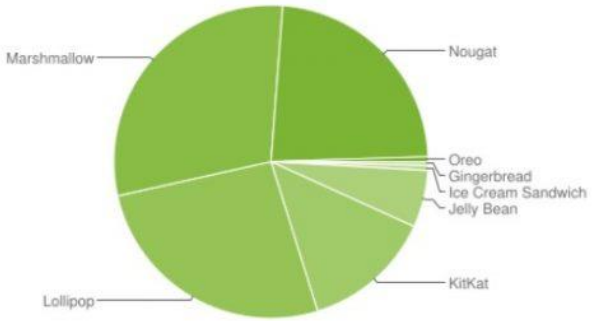
Slika 2. Prikaz konkurentnosti Android uređaja na svjetskom tržištu [6].

2.1 Android verzije

Android verzije poznate su prema tome što izlaze prema nazivima poslastica i slatkiša, ali i prema brojevima.

Posljednja verzija androida 8.0/8.1 Oreo predstavljena je 21.kolovoza.2017 godine. Glavna karakteristika Oreo Android verzije je „PROJECT TREBLE“, sustav koji omogućuje olakšanu i ubrzanu isporuku nadogradnje Android sustava. Neke karakteristike koje ga definiraju su: novi sustav brzih postavki, jednostavno korištenje ikona, duplo brže dizanje sustava te preuzimanje fontova i veći raspon boja za aplikacije [2]. Slika 3 predstavlja trenutni poredak u postotku (%) korištenje Android operativnog sustava prema verzijama.

Version	Codename	API	Distribution
2.3.3 - 2.3.7	Gingerbread	10	0.4%
4.0.3 - 4.0.4	Ice Cream Sandwich	15	0.5%
4.1.x	Jelly Bean	16	2.0%
4.2.x		17	3.0%
4.3		18	0.9%
4.4	KitKat	19	13.4%
5.0	Lollipop	21	6.1%
5.1		22	20.2%
6.0	Marshmallow	23	29.7%
7.0	Nougat	24	19.3%
7.1		25	4.0%
8.0	Oreo	26	0.5%



Slika 3. Prikaz Android operativnih sustava prema verzijama i trenutnom korištenju [7].

2.2 Android uređaji

Kod android uređaja, prvenstveno treba naglasiti da u to spadaju pametni telefoni koji čine jezgru kako na tržištu tako i na samoj razvojnoj platformi. No, u Android uređaje trenutno možemo svrstati :

- pametne telefone
- tablete
- hibride (tablet/laptop)
- laptop
- internet TV
- smart Watch (pametni satovi).



Slika 4. Prikaz Samsung Chromebook-a, laptop s Android operativnim sustavom [8].

Ovo su kategorije prema kojima Android odnosno Google izdaje svoje operativne sustave te iza čega stoji vrhunska podrška. Treba naglasiti da kada se radi o laptopima Google ima svoju verziju pod nazivom „Chromebook“ koja je rađena od različitih proizvođača prema raznim modelima.

Ovo nije prvi pokušaj da Google ujedinjuje razne proizvođače. Postoji stari projekt „Nexus“ koji godinama izdaje nove Android uređaje s tzv. „Stock Androidom“ odnosno čistim Android operativnim sustavom. Taj sustav nema sučelje zadanog proizvođača, iako svake godine kada izlazi proizvođač bude drugi. Google dodjeljuje zadanom proizvođaču mogućnost izrade mobilnog uređaja koji će biti novi Nexus. Tako do sada obitelj Nexus čine:

- Samsung
- Htc
- Lg
- Motorola
- Huawei
- Asus.

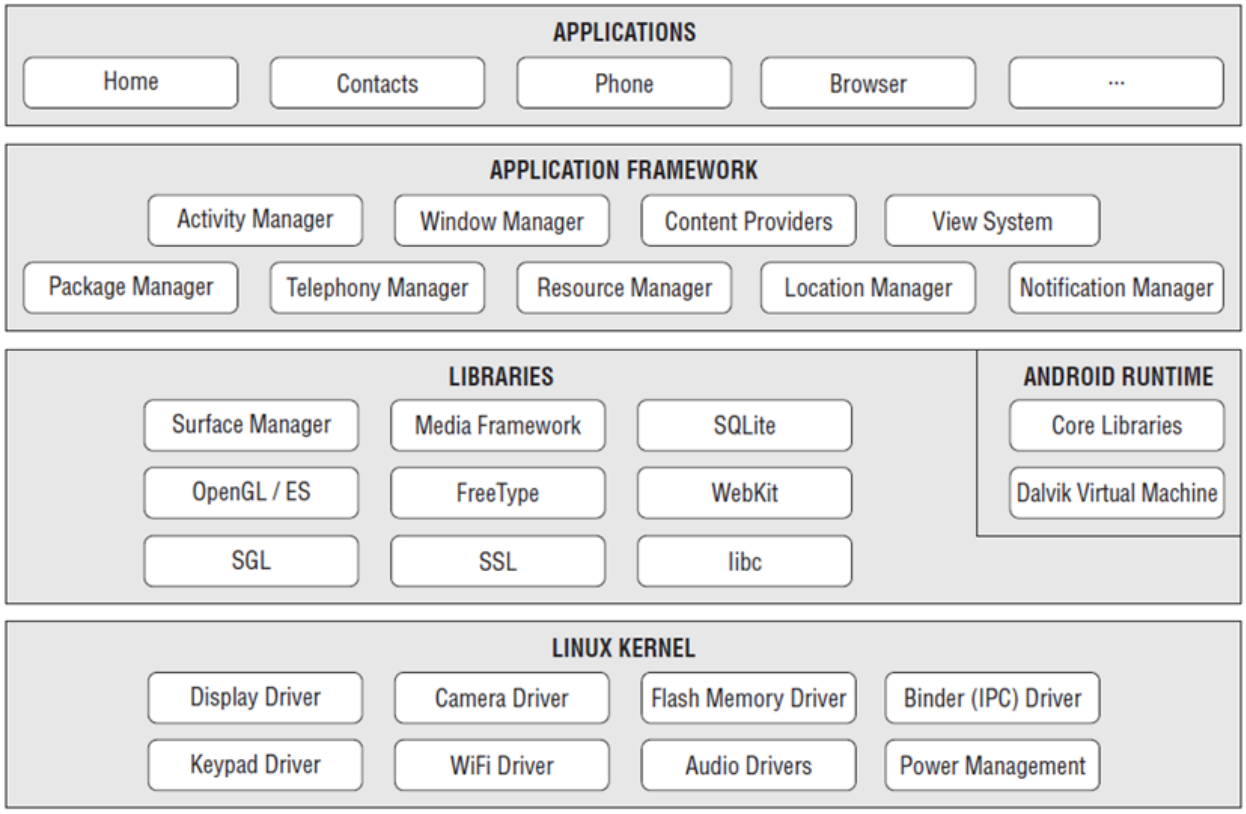


Slika 5. Android Google uređaj Nexus [9].

3. IZRADA ANDROID APLIKACIJE

Android operacijski sustav sastoji se od nekoliko dijelova i slojeva. Svaki pojedini sloj ima svoje karakteristike i ulogu, ali slojevi nisu jasno odvojeni već se najčešće preklapaju jedni s drugima. Android operacijski sustav je baziran na Linux Kernel. Linux je operacijski sustav otvorenog koda, pa aplikacije pomoću posrednika (eng. *middleware*) mogu komunicirati i pokretati druge aplikacije kao što je slanje SMS poruka , e-pošte, slikanja i primanja poziva.

Za sustav temeljen na Linuxu nije potrebno se potrebno brinuti o hardverskim karakteristikama pojedinih uređaja, pa se time omogućuje implementacija Androida na relativno velik raspon uređaja. Najveći broj aplikacija se piše u programskom jeziku Javi uz pomoć android SDK alata.



Slika 6. Prikaz Android arhitekture (Linux Kernel) [3].

Postupkom pisanja aplikacija u C++ programskom jeziku uz korištenje Android SDK alata omogućeno je jednostavnije iskorištavanje sredstava samog uređaja i korištenje biblioteka

te je povećana brzina aplikacija i do 10 puta. Mana ovakvog pristupa je mnogo složenije pisanje koda.

Ukratko su nabrojane biblioteke koje se nalaze iznad jezgre i služe za najvažnije funkcije:

- Surface Manager – omogućuje nadziranje za grafičko sučelje
- Media Framework – omogućuje korištenje audio i video formata
- SQLite – koristi se za rad s bazama podataka
- OpenGL | ES – omogućuje rad s 3D prikazom
- FreeType – omogućuje rad s fontovima
- WebKit – koristi se kod web preglednika
- SGL – prisutna kod većine aplikacija
- SSL (Secure Sockets Layer) – osigurava komunikaciju na internetu
- libc – omogućuje rad u linuxu

Java čini glavni programski jezik prilikom izrade. Sun Micro systems tvrtka i njeni inženjeri objavili su krajem 1995. godine programski jezik pod nazivom Java. Osnovna prednost Jave u odnosu na druge programske jezike je u tome što nije potrebna prilagodba za operacijske sustave koji podržavaju JVM (Java virtual machine) dok se obični programi pisani npr. u C-u moraju prilagoditi operacijskim sustavima u kojem se izvršavaju [3].

Java programski jezik zamišljen je na način da se ne prave gotovo nikakve preinake prilikom korištenja aplikacija na operacijskim sustavima kao što su Windows, Linux itd.

Ideja Jave kao programskog jezika zamišljena je na način da bude pristupačna zajednici, samim time je zajednica pridonijela velikom broju alata koji omogućuju lakše korištenje Jave. Alati nužni za razvoj aplikacija dostupni su putem Java development kit-a JDK koji je besplatan. Najbitnije u razvoju aplikacije je testiranje aplikacije tijekom izrađivanja, a za programere je najvažniji tzv.unit test [3,4].

3.1 Tehnologije izrade

Prilikom izrade završnog rada potrebno je naglasiti korištenje programa „Android studio“ koji služi kao glavni program i glavno okruženje tijekom izrade Android aplikacije. Prilikom prvoga korištenja potrebno je učitati sve bitne dodatke kako bi se omogućio rad putem virtualne mašine koja će simulirati događanja tijekom same izrade.

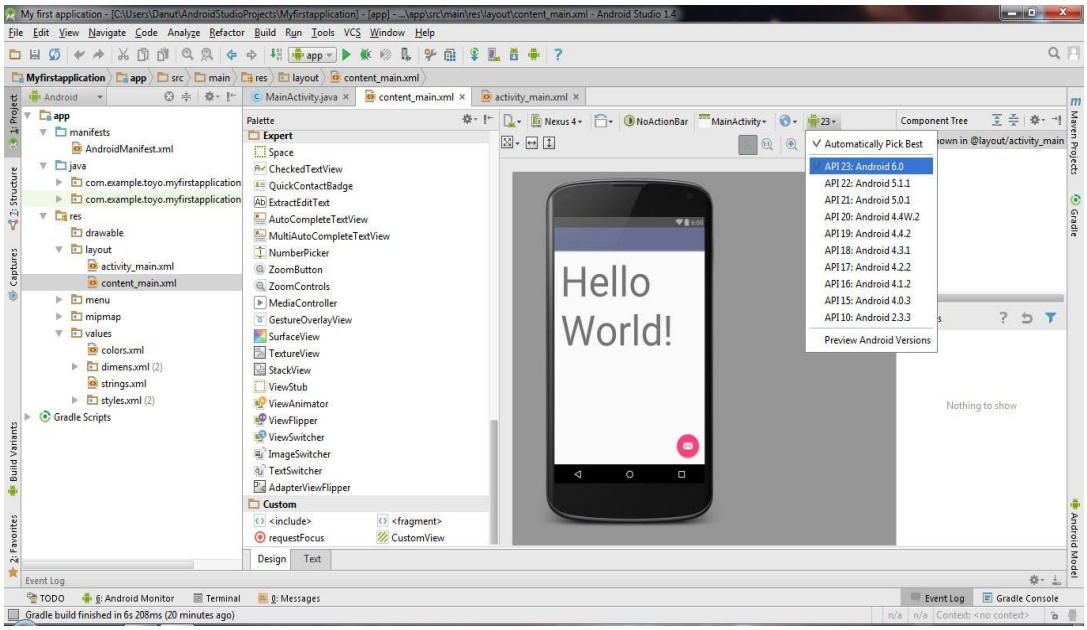
3.1.1 Android studio

Android studio predstavlja službeni i najzastupljeniji alat za razvijanje android aplikacija. Dostupan je za Windows, macOS i Linux operacijske sustave. Prva verzija Android studia 1.0 krenula je s radom u prosincu 2014 godine, a trenutna verzija 3.0 je objavljena u listopadu 2017 godine.



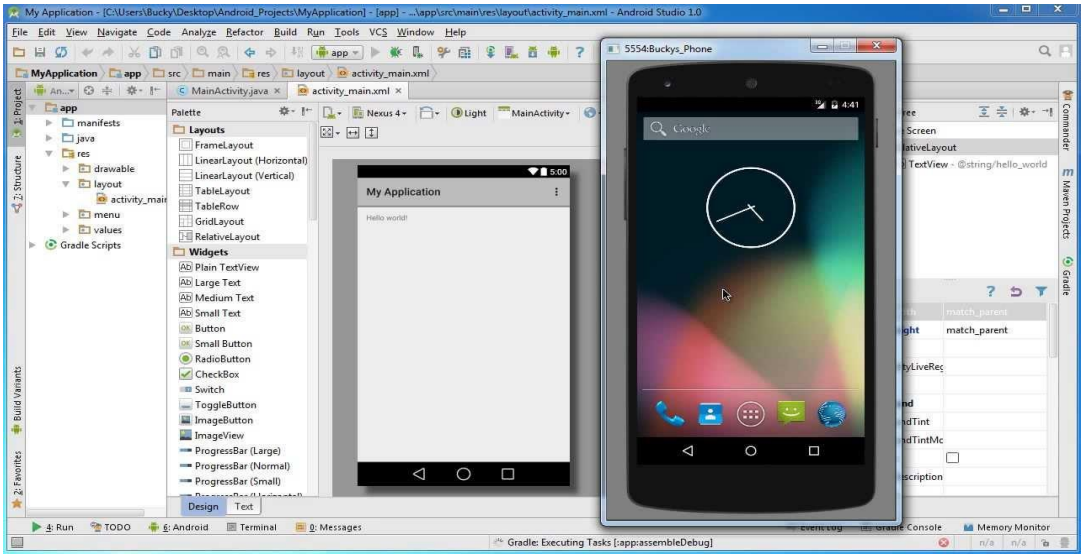
Slika 7. Android studio Logo.

Slika 8. prikazuje izgled rada u Android studiu. Vrlo je teško pričati samo o Android studiju, a ne spomenuti ostale programe i nadogradnje kao i razvojne alate. Prilikom rada odnosno kreacije aplikacije stvara se tzv. Android paket koji obuhvaća kompajlirani java kod sa svim skupovima podataka te datotekama resursa koje su potrebne za razvoj.



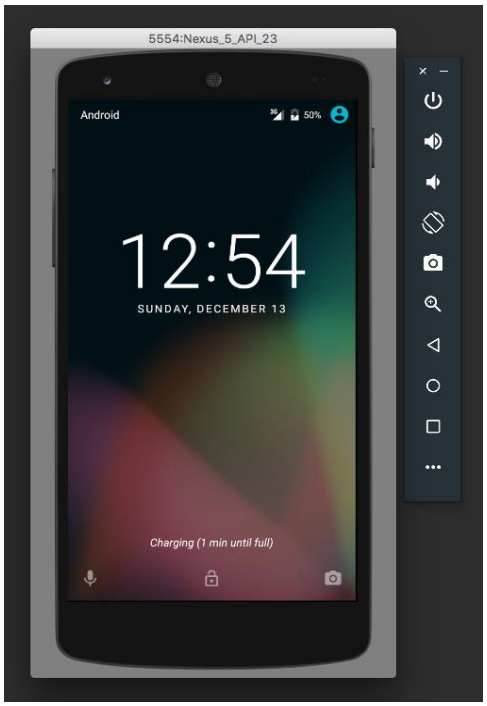
Slika 8. Prikaz Android studio okruženja.

Android studio je poprilično snažan alat za sve vrste izrade aplikacija, ali ponajprije za Android upravo zbog velike podrške prilikom korištenja Jave. Slika 9. prikazuje pokrenuti Android projekt u Android studiu. Kada se započne nove projekt u Android studiu ponuđen je i odabir konkretnog uređaja odnosno u ponudi s ostalim razvojnim alatima dolaze većinom Nexus uređaji kao okosnica tome što se odmah dobije željena rezolucija prilikom čega može započeti razvoj.



Slika 9. Prikaz započetog projekta u Android studio okruženju s pokrenutom konzolom

Mnogi smatraju vrlo složenim izradu Android aplikacija upravo zbog toga što se koristi Java. Početkom izrade dolazi do početnog prozora koji je predstavljen kao na mobilnom uređaju, odnosno u gornjem kutu prozorčića može se primjetiti da je ikona baterije ili pojedinih drugih svojstava, ali je omogućeno i samo konzolno upravljanje, odnosno pojedine funkcije kao na pravom uređaju što je prikazano na slici 10.



Slika 10. Android konzola tijekom izrade, SDK.

Iz navedenoga se vidi da je konzola u potpunosti virtualno omogućen prikaz Android operativnog sustava, što znači da će kao i na mobilnom uređaju, tijekom izrade biti omogućena aplikacija, te simulirati sva događanja koja se događaju i na Android uređajima. Znači svaka promjena na konzoli prikazat će stvarnu promjenu kao na uređaju.

Svaka aplikacija je svojevrsni unikat jer svaka nova aplikacija koja se izrađuje imat će svoj izgled i svoj dizajn, upravo je to ono što aplikacije čini privlačnima i posebnima. Mnogo je dodatnih alata u Android studiu razvojnom okruženju koje svojevrsno pomažu tijekom izrade, pa tako postoji i niz dodataka(*eng plugins*) za samo kreiranje i olakšano kreiranje aplikacija.

Prvobitan dio koda tzv. „import“ dio koda, učitava zadane biblioteke koje će omogućiti razvoj aplikacije i time omogućiti pristup potrebnim parametrima koji su potrebni za programiranje aplikacije. Kako bi sve bilo omogućeno, odnosno kako bi konzola bila funkcionalna potrebno je ono najvažnije a to je Android SDK (eng. *Software Development Kit*).

3.1.2 Android SDK

Pilikom izrade android aplikacija u Javi najvažniji alat za programiranje je Android SDK. Android SDK sadrži mnoštvo alata koji omogućavaju lakše uklanjanje pogrešaka, simulator za testiranje aplikacije te android aplikacije koje služe kao ogledni primjer. Android SDK može raditi na svim operacijskim sustavima. Za izrađivanje Android aplikacija najviše se upotrebljava program Android studio. Osim Android studija pisanje koda aplikacije za Android moguće je izvršiti u bilo kojem programu za uređivanje teksta.

3.2 Algoritam sivog predviđanja

Ova android aplikacija mora služiti u svrhu predviđanja stanja sustava sivim modelom, što znači da je potrebno integrirati algoritam te prema tome izraditi adekvatno sučelje koje će moći na temelju prethodnih podataka ali i trenutnih podataka previdjeti nadolazeće stanje. Ovo predviđanje vrlo je zastupljeno na temelju pitanja prilikom održavanja tehničkih sustava.

Postoje razni postupci i metode koje se mogu primijeniti, ali svaka ta metoda ima određene prednosti i nedostatke.

Sivo predviđanje je relativno nova metoda poznata pod nazivom (eng. *grey forecasting*) koja se temelji na teoriji sivih sustava (eng. *grey system theory*). Teorija svih sustava vrlo je dugo poznata, od 1982. godine, te joj od tada raste popularnost a time i primjena. Danas se koristi u rješavanju raznih problema u različitim područjima.

Sivi sustavi su sustavi za koje nisu poznate sve informacije, odnosno sivi sustav se dijeli na sivi prostor stanja što je zapravo nepoznati dio sustava, te bijeli prostor stanja što čini poznati dio sustava. Cilj je kao što je ranije navedeno iz poznatog dijela informacija saznati ne poznati dio informacija, odnosno iz bijelog dijela sustava saznati sivi dio sustava. Sve je potkrijepljeno matematičkim aparatom.

Kako bi došli do sivog predviđanja potrebno je koristiti model sivog sustava koji se predstavlja diferencijalnom jednadžbom prvoga reda s jednom nepoznanicom. Cilj je kroz algoritam odnosno matematički model načiniti aplikaciju, preslikavanjem matematičkog dijela u programski kod, kojom bi se vršilo sivo predviđanje za pojedine parametre unutar nekog sustava.

Kako bi sve bilo moguće potreban je algoritam za sivo predviđanje. Kada se radi o praksi ponekada je teško doći do podataka koji su prikupljeni u jednakim intervalima, tj. problem nastaje prilikom prikupljanja podataka koji u velikom broju slučaja nisu jednaki.

Kada se radi o izvršavanju algoritma sivog predviđanja nad podacima s nejednakim intervalima prikupljanja podataka, stvara se potreba za prilagođavanjem ulaznih parametara odnosno podataka kako bi se sveli na jednake vremenske intervale.

Zadatak je sada kreirati aplikaciju na temelju algoritma za sivo predviđanje te prikazati rezultate koji će pokazati točnost zadanog algoritma kojega je potrebno prikazati kroz Java programski jezik kako bi ga bilo moguće implementirati unutar aplikacije za Android operativni sustav. Za primjer su uzeti podaci o količini otpada u Slavonskom brodu te prema tim podacima i formulama za GM(1,1) model napravljeni su sljedeći proračuni:

Primjer. Sivo predviđanje količine otpada u Slavonskom Brodu.

U Slavonskom Brodu 2013-te godine količina otpada iznosila je 26406 tona, prema tome količina otpada po stanovniku iznosila je 1.22 kg/st./dan.

Količina otpada po stanovniku = 26406 x 1000 / (59141 x 365) = 1.22 kg/st./dan

Pretpostavlja se da količina otpada koji proizvede jedan stanovnik Slavonskog Broda iznosi 0.85 – 0.9 kg/st./dan.

Tablica 1 Količina otpada na području Grada Savonskog Broda „Vijuš Jug“od 2009. do 2013. godine.

Godina	2009.	2010.	2011.	2012.	2013.
Komunalni otpad (tona)	35537.8	29339.1	27723.2	26024.2	26406.8

U navedenom primjeru provedeno je predviđanje za podatke od 2009. do 2013. godine kako bi se predvidjela količina otpada u 2014. godini.

1. Izvorni niz: $x^{(0)} = (35537.8, 29339.1, 27723.2, 26024.2, 26406.8\ )$
2. AGO niz: $x^{(1)} = (35537.8, 64876.9, 92600.1, 118624.3)$
3. Srednji niz: $z^{(1)} = (50207.35, 78738.5, 105612.2)$
4. Matrica Y: $Y = \begin{bmatrix} 29339.1 \\ 27723.2 \\ 26024.2 \end{bmatrix}$
5. Matrica B: $\begin{bmatrix} -50207.35 & 1 \\ -78738.5 & 1 \\ -105612.2 & 1 \end{bmatrix}$
6. Koeficijenti a, b pomoću jednadžbe $\hat{a} = (B^T B)^{-1} B^T Y$ gdje je $\hat{a} = \begin{bmatrix} a \\ b \end{bmatrix}$.

$$B^T = \begin{bmatrix} -50207.35 & -78738.5 & -105612.2 \\ 1 & 1 & 1 \end{bmatrix}$$

$$B^T B = \begin{bmatrix} 19874466165.11 & -234558.05 \\ -234558.05 & 3 \end{bmatrix}$$

$$(B^T B)^{-1} = \begin{bmatrix} 0 & 0 \\ 0 & 4.31 \end{bmatrix}$$

$$\hat{a} = \begin{bmatrix} 0.06 \\ 32370.84 \end{bmatrix}$$

Dakle, $a = 0.06, b = 32370.84$.

7. Jednadžba predviđanja.

$$\hat{x}^{(0)}(k+1) = (1 - e^{a}) \left(x^{(0)}(1) - \frac{b}{a} \right) e^{-ak}$$

$$\hat{x}^{(0)}(k+1) = (1 - e^{0.06}) \left(35537.8 - \frac{32370.84}{0.06} \right) e^{-0.06k}, k = 1,2,3,4$$

Tablica 2 Izračunata predviđena vrijednost.

Godina	2009.	2010.	2011.	2012.	2013.	2014.
Prava vrijednost	35537.8	29339.1	27723.2	26024.2	26406.8	
Predviđena vrijednost	35537.8	29349.3	27640.1	26030.48	24514.6	23086.97

8. Računanje greške:

   Koristeći formulu za grešku GM(1,1) modela $e(k) = \left|\frac{x^{(0)}(k) - \hat{\hat{x}}^{(0)}(k)}{x^{(0)}(k)}\right| \times 100\%$ dobivamo greške:

   Tablica 3 Izračunata pogreška predviđanja.

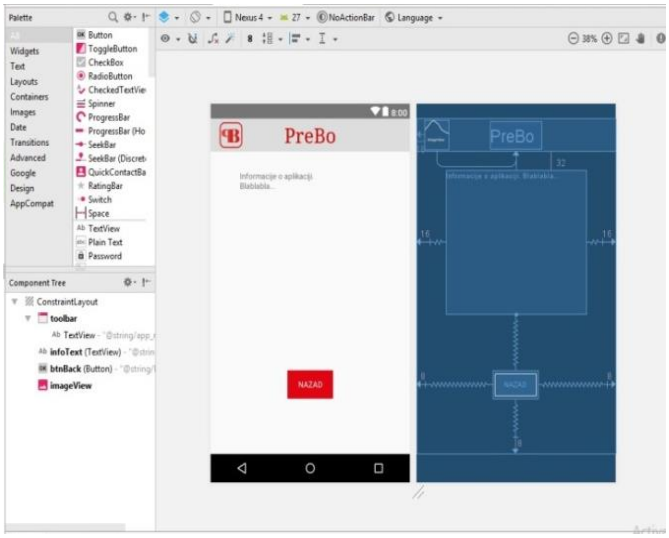
Godina	2009.	2010.	2011.	2012.	2013.
Pogreška	0%	0.188%	0.3%	0.02%	7.17%

9. Izračunata prosječna pogreška: $\frac{0\%+0.188\%+0.3\%+0.02\%+7.17\%}{5} = 1.54\%$

3.3 Izrada Android aplikacije

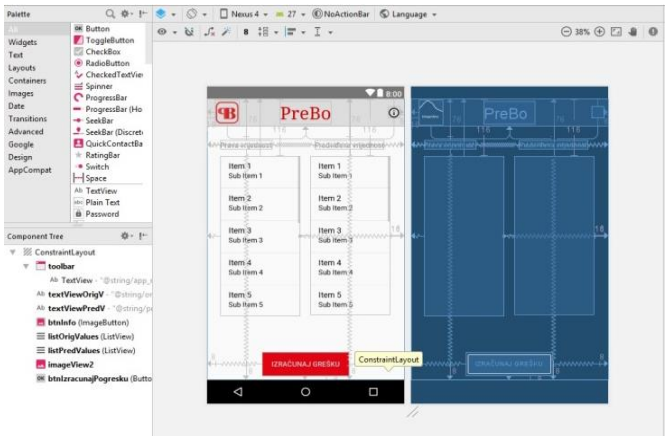
Prikaz izrade aplikacije

Slika 11 prikazuje proces izrade front-end dijela aplikacije koji se odnosi na informacije o aplikaciji i njezinoj upotrebi.



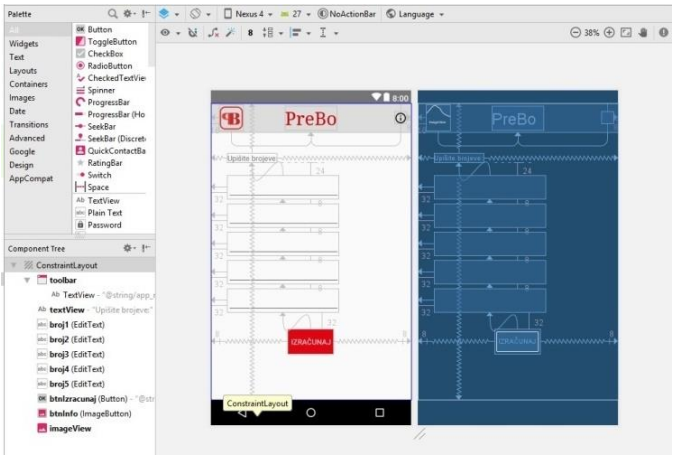
Slika 11. Prikaz izrade info dijela aplikacije.

Slika 12 prikazuje proces izrade front-end dijela aplikacije koji se odnosi na izračun aplikacije.



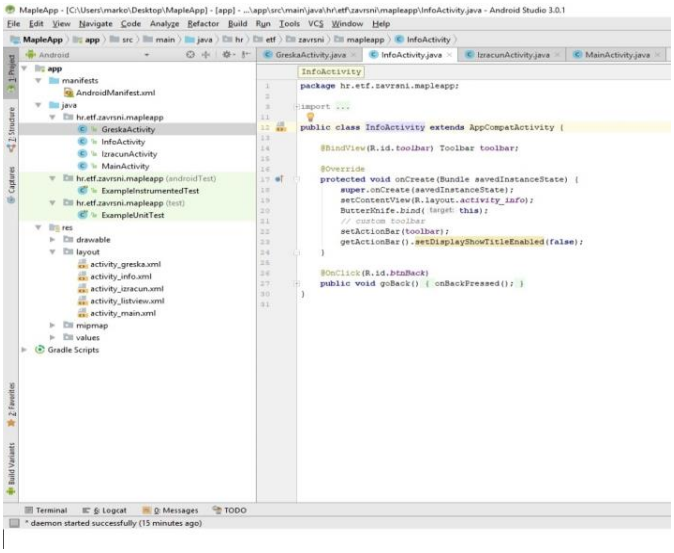
Slika 12. Prikaz izračuna u front-end dijelu aplikacije

Slika 13 prikazuje front-end dio unošenja 5 brojeva kako bi se mogao izvršiti daljnji postupak računanja.



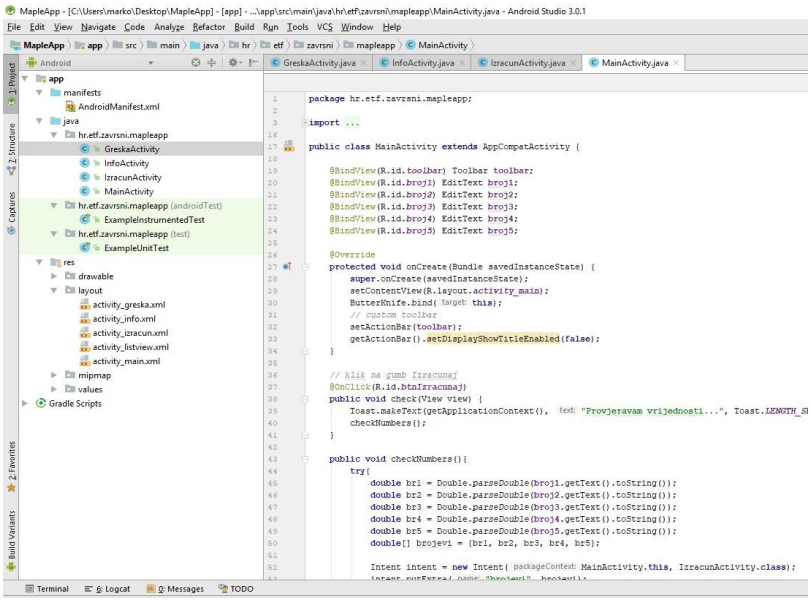
Slika 13. Prikaz unošenja brojeva u front-end dijelu aplikacije

Slika 14 prikazuje Back-end dio procesa izrade aplikacije prilikom koje sam kodirao info dio koji se odnosi na opis aplikacije i njehe svrhe.



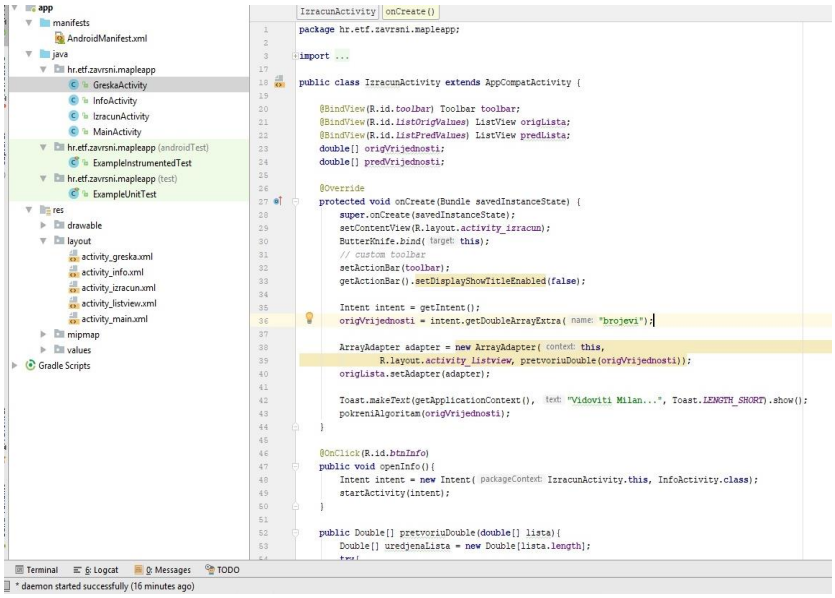
Slika 14. Back-end dio info dijela aplikacije

Slika 15 prikazuje back-end dio prilikom kojeg sam kodirao početni zaslon koji se prikazuje otvaranjem aplikacije te omogućuje unos brojeva kako bi se izračun mogao izvršiti.



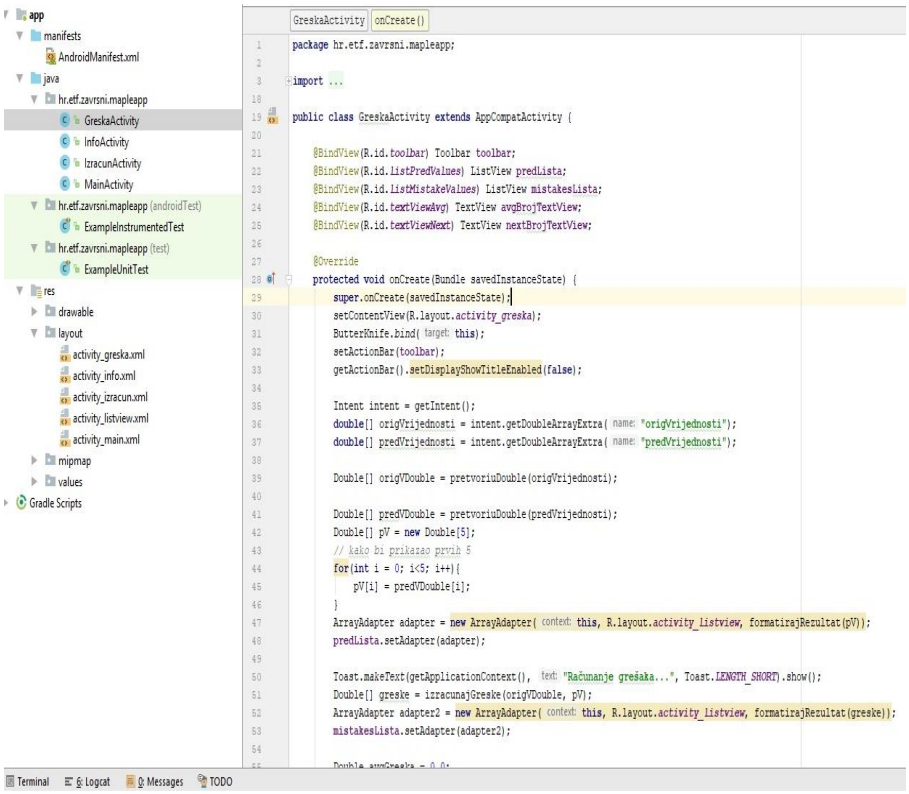
Slika 15. Back-end dio početnog zaslona aplikacije

Slika 16 prikazuje proces izrade back-end dijela aplikacije koji služi za izračunavanje prave vrijednosti i predviđene vrijednosti. Nakon unošenja brojeva otvara se mogućnost izračuna pogreške koji će biti prikazan na slici 17.



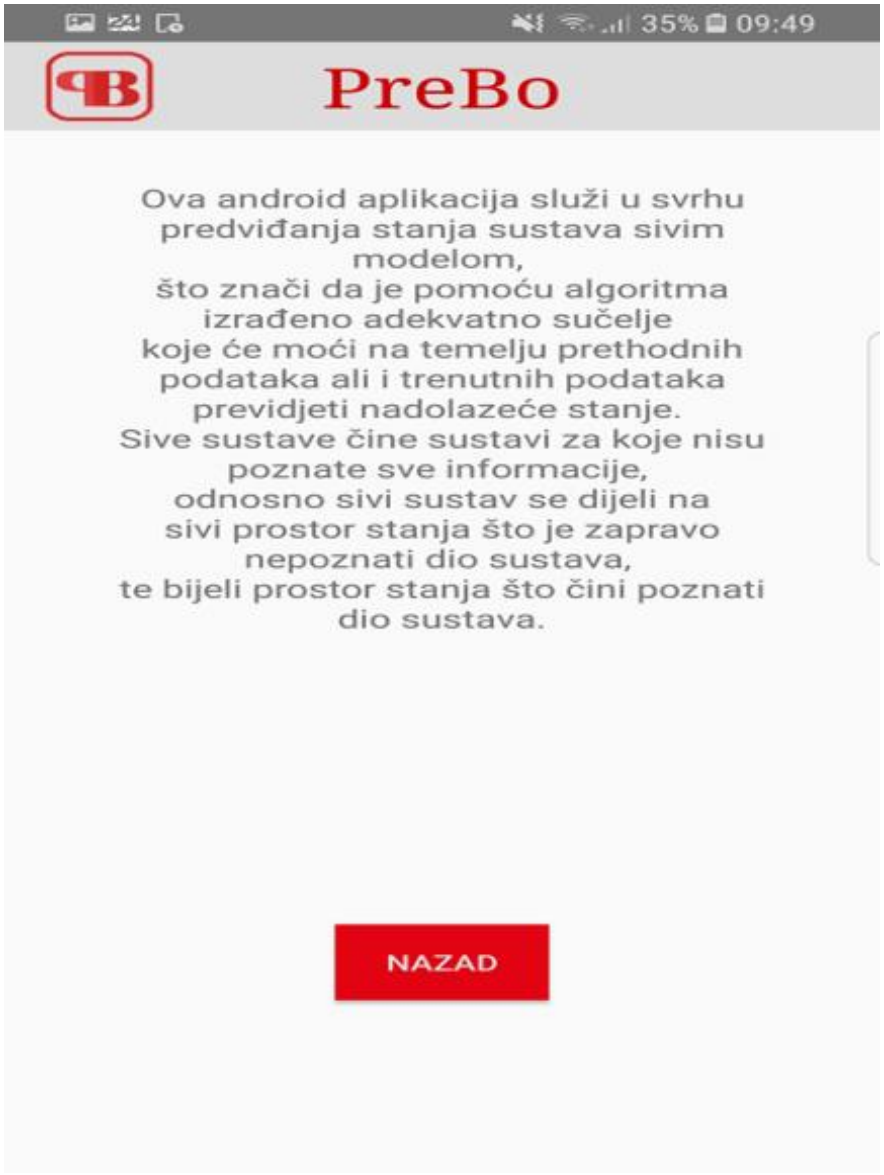
Slika 16. Back-end dio izračuna prave vrijednosti i predviđene vrijednosti

Slika 17 prikazuje back-end dio aplikacije koja opisuje proces prilikom koje se računa greška. Nakon pritiska na gumb izračunaj grešku otvara se novi zaslon koji ispisuje prosječnu grešku te predviđenu slijedeću vrijednost.

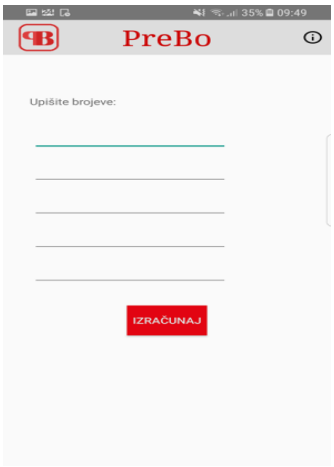


Slika 17. Back-end dio izračunavanja prosječne pogreške i predviđene sljedeće vrijednosti

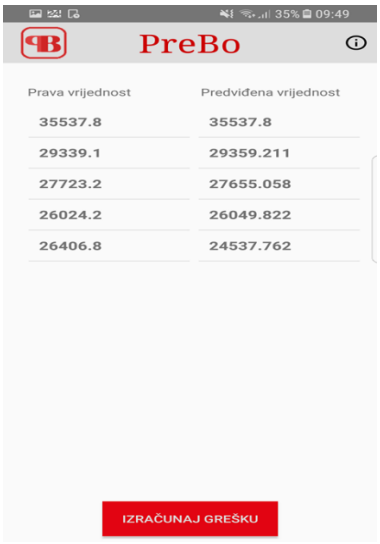
3.4 Slike iz funkcionalne aplikacije u nekoliko koraka



Slika 18. Info aplikacije



Slika 19. Unos brojeva



Slika 20. Unešeni brojevi



Slika 21. Izračun pogreške i predviđena vrijednost

4. ZAKLJUČAK

Ovaj rad predstavlja upute za izradu Android aplikacije s primjenom za sivo predviđanje. Teorijski dio dokumenta prikazuje pojedina načela koja su bitna kako bi se općenito predstavio Android operativni sustav. Navedeni su pojedini modeli sivog predviđanja te opisana funkcionalnost ali i oprez kod pojedinih aspekata poput nejednakog vremenskog intervala prilikom uzimanja ulaznih podataka.

Aplikacija je obavljena uz pomoć alata "Android studio" koji uz dodatne alate poput "Android SDK" omogućavaju korisnicima izradu aplikacija za Android operativni sustav.

Izrada aplikacije prikazana je kroz praktični dio od samoga izgleda odnosno dizajna do implementacije algoritma ali i jednostavnih stvari poput definiranja biblioteka i sl.

5. SAŽETAK

U ovom radu prikazani su alati "Android studio" i "SDK" kao temelj za izradu pravovaljane aplikacije. Opisan je postupak računanja sivog predviđanja. Prikazana je izrada Android aplikacije u navedenim alatima. Prikazan je svaki korak tijekom izrade kao i prilikom implementacije java programa, odnosno algoritma za sivo predviđanje. Funkcionalnost aplikacije prikazana je na primjeru izračuna za sivo predviđanje otpada u Slavonskom Brodu.

Ključne riječi: Algoritam, Android, Android studio, Java, SDK

6. ABSTRACT

Title: Creating Android operating system application

This paper presents the “Android studio” and “SDK” tools as the basics for creating a legitimate application. The gray prediciton calculation procedure is described. Creating of Android application is shown using the listed tools. Each step during the design is presented as well as implementing the Java program, or the gray prediction algorithm. The functionality of the application is shown in the calculation example for gray prediction of garbage in Slavonski Brod.

Keywords : Algorithm,Android,Android studio,Java,SDK

LITERATURA

[1] Logo Android operacijskog sustava, lipanj, 2016.

http://hr.wikipedia.org/wiki/Android_(operacijski_sustav)

[2] Android verzije, lipanj, 2016.

https://developer.android.com

[3] Prikaz android arhitekture, lipanj, 2016.
http://www.tutorialspoint.com/android/android_architecture.html

[4] Predviđanje održavanja tehničkog sustava procjenom stanja, Predviđanje stanja sustava sivim predviđanjem(pdf document).

[5] Grey systems: Modeling and prediction, Kun-li Wen, 2004. godine by Yangsky.

[6] Prikaz konkurentnosti Android uređaja na svjetskom tržištu, travanj, 2018.
https://www.idc.com/promo/smartphone-market-share/os

[7] Prikaz Android operativnih sustava prema verzijama i trenutnom korištenju, travanj 2018.
http://mobiclass.csc.ncsu.edu/2012/12/find-indystate-android-os-ice-cream.html

[8] Prikaz Samsung Chromebook-a, laptop s Android operativnim sustavom, travanj 2018.
http://www.upgrademag.com/web/2014/03/21/samsung-pldt-enable-smes-collaborate-work-cloud/

[9] Android Google uređaj Nexus, travanj 2018.
https://www.livemint.com/Leisure/iJdAuCEgv6xkZTTGpRo8cN/Are-Google-Nexus-and-Samsung-Galaxy-devices-the-only-safe-An.html

[10] Prikaz Android arhitekture, travanj 2018.

http://shubhamjain.space/wp-content/uploads/2016/09/android-architecture.png

ŽIVOTOPIS

Božidar Javor rođen je 16.srpnja.1991.godine . Pohađao je osnovnu školu Antun Mihanović u Slavonskom Brodu u vremenu od 1998-2006. Srednju Tehničku školu u Slavonskom Brodu upisuje 2006. godine s naznakom smjera elektrotehničar. Srednju Tehničku školu završava 2010. godine te upisuje Elektrotehnički fakultet u Osijeku s naznakom smjera informatika. Krajem 2015-te godine odlazi u inozemstvo i zapošljava se u informatičkoj firmi kao tehničar koji održava rad na računalima.

PRILOG 1:IZVORNI DIO APLIKACIJE

```
package hr.etf.zavrsni.mapleapp;

import android.content.Intent;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.widget.ArrayAdapter;
import android.widget.ListView;
import android.widget.TextView;
import android.widget.Toast;
import android.widget.Toolbar;

import java.math.RoundingMode;
import java.text.DecimalFormat;

import butterknife.BindView;
import butterknife.ButterKnife;
import butterknife.OnClick;

public class GreskaActivity extends AppCompatActivity {

    @BindView(R.id.toolbar) Toolbar toolbar;
    @BindView(R.id.listPredValues) ListView predLista;
    @BindView(R.id.listMistakeValues) ListView mistakesLista;
    @BindView(R.id.textViewAvg) TextView avgBrojTextView;
    @BindView(R.id.textViewNext) TextView nextBrojTextView;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_greska);
        ButterKnife.bind(this);
        setActionBar(toolbar);
        getActionBar().setDisplayShowTitleEnabled(false);

        Intent intent = getIntent();
        double[] origVrijednosti =
intent.getDoubleArrayExtra("origVrijednosti");
        double[] predVrijednosti =
intent.getDoubleArrayExtra("predVrijednosti");

        Double[] origVDouble = pretvoriuDouble(origVrijednosti);

        Double[] predVDouble = pretvoriuDouble(predVrijednosti);
        Double[] pV = new Double[5];
        // kako bi prikazao prvih 5
        for(int i = 0; i<5; i++){
            pV[i] = predVDouble[i];
        }
        ArrayAdapter adapter = new ArrayAdapter(this,
R.layout.activity_listview, formatirajRezultat(pV));
        predLista.setAdapter(adapter);

        Toast.makeText(getApplicationContext(), "Računanje grešaka...",
Toast.LENGTH_SHORT).show();
        Double[] greske = izracunajGreske(origVDouble, pV);
        ArrayAdapter adapter2 = new ArrayAdapter(this,
R.layout.activity_listview, formatirajRezultat(greske));
        mistakesLista.setAdapter(adapter2);

        Double avgGreska = 0.0;
```

```
        Double nextValue = 0.0;
        for(int i = 0; i<greske.length; i++){
            avgGreska += greske[i];
        }
        avgGreska = avgGreska / 5;
        nextValue = predVDouble[5];

        avgBrojTextView.setText(avgBrojTextView.getText() +
String.format("%.2f", avgGreska) + "%");
        nextBrojTextView.setText(nextBrojTextView.getText() + ""
+String.format("%.2f", nextValue));
    }

    private Double[] izracunajGreske(Double[] origV,Double[] predV) {
        Double[] greske = new Double[5];
        for(int i = 0; i < 5; i++){
            // formula za greške
            greske[i] = Math.abs((origV[i] - predV[i])/origV[i]) * 100;
        }
        return greske;
    }

    private String[] formatirajRezultat(Double[] predVDouble) {
        DecimalFormat dcf = new DecimalFormat("#.###");
        dcf.setRoundingMode(RoundingMode.CEILING);
        String[] formatirano = new String[predVDouble.length];
        for (int i =0; i < predVDouble.length; i++) {
            formatirano[i] = dcf.format(predVDouble[i]);
        }
        return formatirano;
    }

    public Double[] pretvoriuDouble(double[] lista){
        Double[] uredjenaLista = new Double[lista.length];
        try{
            for(int i = 0; i<lista.length; i++){
                uredjenaLista[i] = lista[i];
            }
        }
        catch (Exception e){
            Toast.makeText(getApplicationContext(), "Dogodila se pogreška",
Toast.LENGTH_SHORT).show();
        }
        return uredjenaLista;
    }

    @OnClick(R.id.btnInfo)
    public void openInfo(){
        Intent intent = new Intent(GreskaActivity.this, InfoActivity.class);
        startActivity(intent);
    }
}



package hr.etf.zavrsni.mapleapp;

import android.content.Intent;
import android.os.Bundle;
import android.support.v7.app.AppCompatActivity;
import android.widget.ArrayAdapter;
import android.widget.ListView;
import android.widget.Toast;
import android.widget.Toolbar;
```

```
import java.math.RoundingMode;
import java.text.DecimalFormat;

import butterknife.BindView;
import butterknife.ButterKnife;
import butterknife.OnClick;

public class IzracunActivity extends AppCompatActivity {

    @BindView(R.id.toolbar) Toolbar toolbar;
    @BindView(R.id.listOrigValues) ListView origLista;
    @BindView(R.id.listPredValues) ListView predLista;
    double[] origVrijednosti;
    double[] predVrijednosti;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_izracun);
        ButterKnife.bind(this);
        // custom toolbar
        setActionBar(toolbar);
        getActionBar().setDisplayShowTitleEnabled(false);

        Intent intent = getIntent();
        origVrijednosti = intent.getDoubleArrayExtra("brojevi");

        ArrayAdapter adapter = new ArrayAdapter(this,
                R.layout.activity_listview,
pretvoriuDouble(origVrijednosti));
        origLista.setAdapter(adapter);

        Toast.makeText(getApplicationContext(), "Vidoviti Milan...",
Toast.LENGTH_SHORT).show();
        pokreniAlgoritam(origVrijednosti);
    }

    @OnClick(R.id.btnInfo)
    public void openInfo(){
        Intent intent = new Intent(IzracunActivity.this, InfoActivity.class);
        startActivity(intent);
    }

    public Double[] pretvoriuDouble(double[] lista){
        Double[] uredjenaLista = new Double[lista.length];
        try{
            for(int i = 0; i<lista.length; i++){
                uredjenaLista[i] = lista[i];
            }
        }
        catch (Exception e){
            Toast.makeText(getApplicationContext(), "Dogodila se pogreška",
Toast.LENGTH_SHORT).show();
        }
        return uredjenaLista;
    }

    public void pokreniAlgoritam(double[] origVrijednosti){
        // origVrijednosti uvijek ima 5
        // agoNiz - 4 broja
        double[] agoNiz = {origVrijednosti[0],
                    origVrijednosti[0] + origVrijednosti[1],
                    origVrijednosti[0] + origVrijednosti[1] +
origVrijednosti[2],
```

```
                    origVrijednosti[0] + origVrijednosti[1] +
origVrijednosti[2] + origVrijednosti[3]};

        // srednjiNiz - 3 broja
        double[] srednjiNiz = {(agoNiz[0] + agoNiz[1])/2,
                               (agoNiz[1] + agoNiz[2])/2,
                               (agoNiz[2] + agoNiz[3])/2};

        double[][] matricaY = {{origVrijednosti[1]}, {origVrijednosti[2]},
{origVrijednosti[3]}};

        double[][] matricaB = {{-srednjiNiz[0], 1}, {-srednjiNiz[1], 1}, {-
srednjiNiz[2], 1}};

        double[][] matricaTransB = transponirajMatricu(matricaB);
        double[][] tempMatrica1 = pomnoziMatrice(matricaTransB, matricaB);
        double[][] tempMatrica2 = inverzMatrice(tempMatrica1);
        double[][] tempMatrica3 = pomnoziMatrice(tempMatrica2,
matricaTransB);
        double[][] tempMatrica4 = pomnoziMatrice(tempMatrica3, matricaY);
        // tempMatrica4 se sastoji od 2 reda i 1 stupca (a/b)
        // a,b - koeficijenti
        double a = tempMatrica4[0][0];
        double b = tempMatrica4[1][0];

        predVrijednosti = predvidiVrijednosti(origVrijednosti, a, b);

        Double[] predVDouble = pretvoriuDouble(predVrijednosti);
        Double[] pV = new Double[5];
        // kako bi prikazao prvih 5
        for(int i = 0; i<5; i++){
            pV[i] = predVDouble[i];
        }
        ArrayAdapter adapter = new ArrayAdapter(this,
                R.layout.activity_listview, formatirajRezultat(pV));
        predLista.setAdapter(adapter);
    }

    private String[] formatirajRezultat(Double[] predVDouble) {
        DecimalFormat dcf = new DecimalFormat("#.###");
        dcf.setRoundingMode(RoundingMode.CEILING);
        String[] formatirano = new String[predVDouble.length];
        for (int i =0; i < predVDouble.length; i++) {
           formatirano[i] = dcf.format(predVDouble[i]);
        }
        return formatirano;
    }

    private double[][] inverzMatrice(double[][] m) {
        // kvadratna matrica - jednak broj stupaca i redaka
        double sk = 1/((m[0][0] * m[1][1]) - (m[0][1] * m[1][0]));
        double[][] tmpMatrica = {{m[1][1], -m[0][1]},
                                 {-m[1][0], m[0][0]}};
        double[][] rezMatrica = {{sk * tmpMatrica[0][0], sk *
tmpMatrica[0][1]},
                                 {sk * tmpMatrica[1][0], sk *
tmpMatrica[1][1]}};
        return rezMatrica;
    }

    private double[][] pomnoziMatrice(double[][] matricaTransB, double[][]
matricaB) {

        int tranRows = matricaTransB.length;
        int tranColumns = matricaTransB[0].length;
```

```
        int bRows = matricaB.length;
        int bColumns = matricaB[0].length;

        if (tranColumns != bRows) {
            throw new IllegalArgumentException("Broj redaka transponirane: "
+ tranColumns +
                    " ne odgovara broju stupaca B matrice: " + bRows + ".");
        }
        double[][] rezMatrica = new double[tranRows][bColumns];
        for (int i = 0; i < tranRows; i++) {
            for (int j = 0; j < bColumns; j++) {
                for (int k = 0; k < tranColumns; k++) {
                    rezMatrica[i][j] += matricaTransB[i][k] * matricaB[k][j];
                }
            }
        }
        return rezMatrica;
    }

    private double[][] transponirajMatricu(double[][] matricaB) {
        int m = matricaB.length;
        int n = matricaB[0].length;
        double[][] transB = new double[n][m];
        for (int i = 0; i < m; i++)
            for (int j = 0; j < n; j++)
                transB[j][i] = matricaB[i][j];
        return transB;
    }

    private double[] predvidiVrijednosti(double[] origVrijednosti, double a,
double b){
        double[] predVrijednosti = new double[6];
        // prva vrijednost se prepisuje
        predVrijednosti[0] = origVrijednosti[0];
        for(int k = 1; k <= origVrijednosti.length; k++){
            predVrijednosti[k] = (1 - Math.exp(a))*(origVrijednosti[0] -
b/a)*Math.exp(-a * k);
        }
        return predVrijednosti;
    }

    @OnClick(R.id.btnIzracunajPogresku)
    public void otvoriGreskaActivity(){
        Intent intent = new Intent(IzracunActivity.this,
GreskaActivity.class);
        intent.putExtra("origVrijednosti", origVrijednosti);
        intent.putExtra("predVrijednosti", predVrijednosti);
        startActivity(intent);
        finish();
    }
}



package hr.etf.zavrsni.mapleapp;

import android.content.Intent;
import android.os.Bundle;
import android.support.v7.app.AppCompatActivity;
import android.widget.ArrayAdapter;
import android.widget.ListView;
import android.widget.Toast;
import android.widget.Toolbar;

import java.math.RoundingMode;
```

```
import java.text.DecimalFormat;

import butterknife.BindView;
import butterknife.ButterKnife;
import butterknife.OnClick;

public class IzracunActivity extends AppCompatActivity {

    @BindView(R.id.toolbar) Toolbar toolbar;
    @BindView(R.id.listOrigValues) ListView origLista;
    @BindView(R.id.listPredValues) ListView predLista;
    double[] origVrijednosti;
    double[] predVrijednosti;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_izracun);
        ButterKnife.bind(this);
        // custom toolbar
        setActionBar(toolbar);
        getActionBar().setDisplayShowTitleEnabled(false);

        Intent intent = getIntent();
        origVrijednosti = intent.getDoubleArrayExtra("brojevi");

        ArrayAdapter adapter = new ArrayAdapter(this,
                R.layout.activity_listview,
pretvoriuDouble(origVrijednosti));
        origLista.setAdapter(adapter);

        Toast.makeText(getApplicationContext(), "Vidoviti Milan...",
Toast.LENGTH_SHORT).show();
        pokreniAlgoritam(origVrijednosti);
    }

    @OnClick(R.id.btnInfo)
    public void openInfo(){
        Intent intent = new Intent(IzracunActivity.this, InfoActivity.class);
        startActivity(intent);
    }

    public Double[] pretvoriuDouble(double[] lista){
        Double[] uredjenaLista = new Double[lista.length];
        try{
            for(int i = 0; i<lista.length; i++){
                uredjenaLista[i] = lista[i];
            }
        }
        catch (Exception e){
            Toast.makeText(getApplicationContext(), "Dogodila se pogreška",
Toast.LENGTH_SHORT).show();
        }
        return uredjenaLista;
    }

    public void pokreniAlgoritam(double[] origVrijednosti){
        // origVrijednosti uvijek ima 5
        // agoNiz - 4 broja
        double[] agoNiz = {origVrijednosti[0],
                    origVrijednosti[0] + origVrijednosti[1],
                    origVrijednosti[0] + origVrijednosti[1] +
origVrijednosti[2],
                    origVrijednosti[0] + origVrijednosti[1] +
origVrijednosti[2] + origVrijednosti[3]};
```

```
        // srednjiNiz - 3 broja
        double[] srednjiNiz = {(agoNiz[0] + agoNiz[1])/2,
                               (agoNiz[1] + agoNiz[2])/2,
                               (agoNiz[2] + agoNiz[3])/2};

        double[][] matricaY = {{origVrijednosti[1]}, {origVrijednosti[2]},
{origVrijednosti[3]}};

        double[][] matricaB = {{-srednjiNiz[0], 1}, {-srednjiNiz[1], 1}, {-
srednjiNiz[2], 1}};

        double[][] matricaTransB = transponirajMatricu(matricaB);
        double[][] tempMatrica1 = pomnoziMatrice(matricaTransB, matricaB);
        double[][] tempMatrica2 = inverzMatrice(tempMatrica1);
        double[][] tempMatrica3 = pomnoziMatrice(tempMatrica2,
matricaTransB);
        double[][] tempMatrica4 = pomnoziMatrice(tempMatrica3, matricaY);
        // tempMatrica4 se sastoji od 2 reda i 1 stupca (a/b)
        // a,b - koeficijenti
        double a = tempMatrica4[0][0];
        double b = tempMatrica4[1][0];

        predVrijednosti = predvidiVrijednosti(origVrijednosti, a, b);

        Double[] predVDouble = pretvoriuDouble(predVrijednosti);
        Double[] pV = new Double[5];
        // kako bi prikazao prvih 5
        for(int i = 0; i<5; i++){
            pV[i] = predVDouble[i];
        }
        ArrayAdapter adapter = new ArrayAdapter(this,
                R.layout.activity_listview, formatirajRezultat(pV));
        predLista.setAdapter(adapter);
    }

    private String[] formatirajRezultat(Double[] predVDouble) {
        DecimalFormat dcf = new DecimalFormat("#.###");
        dcf.setRoundingMode(RoundingMode.CEILING);
        String[] formatirano = new String[predVDouble.length];
        for (int i =0; i < predVDouble.length; i++) {
           formatirano[i] = dcf.format(predVDouble[i]);
        }
        return formatirano;
    }

    private double[][] inverzMatrice(double[][] m) {
        // kvadratna matrica - jednak broj stupaca i redaka
        double sk = 1/((m[0][0] * m[1][1]) - (m[0][1] * m[1][0]));
        double[][] tmpMatrica = {{m[1][1], -m[0][1]},
                                 {-m[1][0], m[0][0]}};
        double[][] rezMatrica = {{sk * tmpMatrica[0][0], sk *
tmpMatrica[0][1]},
                                 {sk * tmpMatrica[1][0], sk *
tmpMatrica[1][1]}};
        return rezMatrica;
    }

    private double[][] pomnoziMatrice(double[][] matricaTransB, double[][]
matricaB) {

        int tranRows = matricaTransB.length;
        int tranColumns = matricaTransB[0].length;
        int bRows = matricaB.length;
        int bColumns = matricaB[0].length;
```

```
        if (tranColumns != bRows) {
            throw new IllegalArgumentException("Broj redaka transponirane: "
+ tranColumns +
                    " ne odgovara broju stupaca B matrice: " + bRows + ".");
        }
        double[][] rezMatrica = new double[tranRows][bColumns];
        for (int i = 0; i < tranRows; i++) {
            for (int j = 0; j < bColumns; j++) {
                for (int k = 0; k < tranColumns; k++) {
                    rezMatrica[i][j] += matricaTransB[i][k] * matricaB[k][j];
                }
            }
        }
        return rezMatrica;
    }

    private double[][] transponirajMatricu(double[][] matricaB) {
        int m = matricaB.length;
        int n = matricaB[0].length;
        double[][] transB = new double[n][m];
        for (int i = 0; i < m; i++)
            for (int j = 0; j < n; j++)
                transB[j][i] = matricaB[i][j];
        return transB;
    }

    private double[] predvidiVrijednosti(double[] origVrijednosti, double a,
double b){
        double[] predVrijednosti = new double[6];
        // prva vrijednost se prepisuje
        predVrijednosti[0] = origVrijednosti[0];
        for(int k = 1; k <= origVrijednosti.length; k++){
            predVrijednosti[k] = (1 - Math.exp(a))*(origVrijednosti[0] -
b/a)*Math.exp(-a * k);
        }
        return predVrijednosti;
    }

    @OnClick(R.id.btnIzracunajPogresku)
    public void otvoriGreskaActivity(){
        Intent intent = new Intent(IzracunActivity.this,
GreskaActivity.class);
        intent.putExtra("origVrijednosti", origVrijednosti);
        intent.putExtra("predVrijednosti", predVrijednosti);
        startActivity(intent);
        finish();
    }
}



package hr.etf.zavrsni.mapleapp;

import android.content.Intent;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.widget.Toolbar;

import butterknife.BindView;
import butterknife.ButterKnife;
import butterknife.OnClick;

public class InfoActivity extends AppCompatActivity {
```

```
    @BindView(R.id.toolbar) Toolbar toolbar;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_info);
        ButterKnife.bind(this);
        // custom toolbar
        setActionBar(toolbar);
        getActionBar().setDisplayShowTitleEnabled(false);
    }

    @OnClick(R.id.btnBack)
    public void goBack(){
        onBackPressed();
    }
}
```