Mobilna aplikacija Čovječe, ne ljuti se

Matijević, Josip

Undergraduate thesis / Završni rad

2017

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:200:093308

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
ELEKTROTEHNIČKI FAKULTET

Sveučilišni studij

MOBILNA APLIKACIJA „ČOVJEČE, NE LJUTI SE“

Završni rad

Josip Matijević

Osijek, 2017.

SADRŽAJ:

1. UVOD 1

1.1. Zadatak završnog rada 2

2. VAŽNE TEORIJSKE PODLOGE 3

2.1. Povijest Android sustava 3

2.2. Značajke Android platforme 5

2.3. Arhitektura Android platforme 7

2.4. Razvojno okruženje 9

2.5. Java 10

2.6. Povijest i upute za igranje igre „čovječe, ne ljuti se „ 11

2.6.1. Povijest igranja igre „čovječe, ne ljuti se „ 11

2.6.2. Pravila za igranje igre „čovječe, ne ljuti se“ 12

3. ANALIZA SUSTAVA 13

3.1. Ideja izrade aplikacije 13

4. PROGRAMSKO RJEŠENJE ZADATKA 14

4.1. Početni zaslon 14

4.2. Nova igra 16

4.3. Statistika 24

4.4. Bluetooth 27

5. ZAKLJUČAK 30

1. UVOD

U ovom radu prati se izrada mobilne aplikacije „Čovječe, ne ljuti se“. Ovo je poznata društvena igra uz koju su odrastale mnoge generacije. Igra je izrađena za sve standardnije Android sustave koji se nalaze na pametnim telefonima. Aplikaciju mogu igrati 2-4 igrača. Na početku se odabire koliko će igrača sudjelovati u igri. Svaki igrač prije početka dvoboja upisuje ime i bira željenu boju što je potrebno za praćenje statistike pojedinog igrača. Igrica će se moći igrati na jednom ili više uređaja. Igranjem na jednom uređaju svaki igrač prema odabranoj boji ima svoj red za igranje. Ako se igrači odluče igrati na više uređaja princip igrice je jednak kao da se igra na jednom mobilnom uređaju. Glavna razlika je ta što će se igrači povezivati pomoću Bluetooth-a i na taj način svaki će igrač igrati na svom mobilnom uređaju povezan u jednu zajedničku mrežu. Rezultati se prate statistički tako da se mogu vidjeti podatci poput ukupnog broja odigranih igara, ukupnog vremena igranja, prosječnog vremena po igri te statistika pojedinog igrača. Svi podatci se pohranjuju u bazu podataka. Prilikom izrade programa potrebna su znanja iz Jave i Android operacijskog sustava koja je moguće pronaći u literaturi dostupnoj na Internetu.

U drugom poglavlju rada opisana je povijest Android sustava, značajke i arhitektura Android platforme te razvojno okruženje i programski jezici potrebni za izradu aplikacije kao i upute za igranje igre „Čovječe, ne ljuti se“. Treće poglavlje obuhvaća analizu sustava, dok su u četvrtom poglavlju opisana rješenja pojedinih problema koje je bilo potrebno riješiti u okviru rada. Na kraju rada nalazi se zaključak u kojem je dan osvrt na postignute rezultate i mogućnost za daljnju nadogradnju.

1.1. Zadatak završnog rada

U okviru završnog rada potrebno je opisati značajke i arhitekturu Android operacijskog sustava te se upoznati sa alatima i tehnologijom potrebnom za izradu Android mobilne aplikacije. Izraditi mobilnu aplikaciju „Čovječe, ne ljuti se“ s mogućnošću odabira 2-4 igrača, njihovu personalizaciju i omogućiti igranje na više mobilnih uređaja preko Bluetooth-a. Na kraju je potrebno izraditi statistiku igre u kojoj će se pratiti ukupan broj odigranih igara, ukupno vrijeme, prosječno vrijeme po igri te statistiku pojedinog igrača.

2. VAŽNE TEORIJSKE PODLOGE

U ovom poglavlju rada bit će opisana povijest Android operacijskog sustava, značajke i arhitektura Android platforme, razvojno okruženje i programski jezici potrebni za izradu aplikacije te povijest i upute za igranje igre „Čovječe, ne ljuti se“.

2.1. Povijest Android sustava

Android operacijski sustav je sustav otvorenog koda i temelji se na Linux operacijskom sustavu za pametne telefone i tablete razvijen od Google-a.[1] Android Inc. su osnovali Andy Rubin, Rich Miner, Nick Sears i Chris White krajem 2003. godine kako bi se mogli razvijati programi za pametne mobilne uređaje. Godine 2005. Google odlučuje kupiti Android. Počelo se govoriti da će Google ući na tržište pametnih telefona. Uz Google-ovu pomoć Android Inc. na tržište lansira mobilnu platformu kojoj je temelj Linux jezgra. Zamisao platforme je da bude prilagodljiva svakom korisniku pojedinačno, a ne da se korisnici moraju prilagođavati platformi.
Open Handset Alliance je osnovana 5. studenog, uz inicijativu Google-a s ciljem stvaranja jedinstvenog standarda za mobilne uređaje. Okupile su se 34 tvrtke iz svih područja mobilne industrije.
Na isti dan Open Handset Alliance otkriva Android operacijski sustav otvorenog koda temeljen na Linux jezgri. Prvi mobilni uređaj koji se komercijalno proizvodio s ugrađenim Android OS bio je T-Mobile G1, od proizvođača mobilnih uređaja HTC.
Android je zamišljen kao projekt otvorenog koda (eng. open source project), koji je pod Apache licencom od 21. listopada 2008. godine. Da bi se Android koristio, Google mora ovjeriti uređaj kompatibilan s dokumentom za definiranje kompatibilnosti (eng. Compatibility Definition Document, CDD) nakon čega se može koristiti ime Android.
Sami uređaji moraju zadovoljavati kriterije zadane s CDD-om, inače nisu u mogućnosti pristupati aplikacijama zatvorenog koda. Aplikacije zatvorenog koda služe kako bi mogli instalirati aplikacije koje su dostupne na tržištu, ali nisu predviđene u osnovnom paketu aplikacija s kojim nam uređaj stiže. Jedine iznimke kod kojih nije moguće imati otvoren kod su verzije pod nazivom Honeycomb. Te verzije su predviđene za tablete, a ne mobilne uređaje. Zbog mogućnosti instalacije na mobilne uređaje (plan je da se ove verzije ne koriste za mobilne uređaje) išlo se s zatvorenim tipom koda.

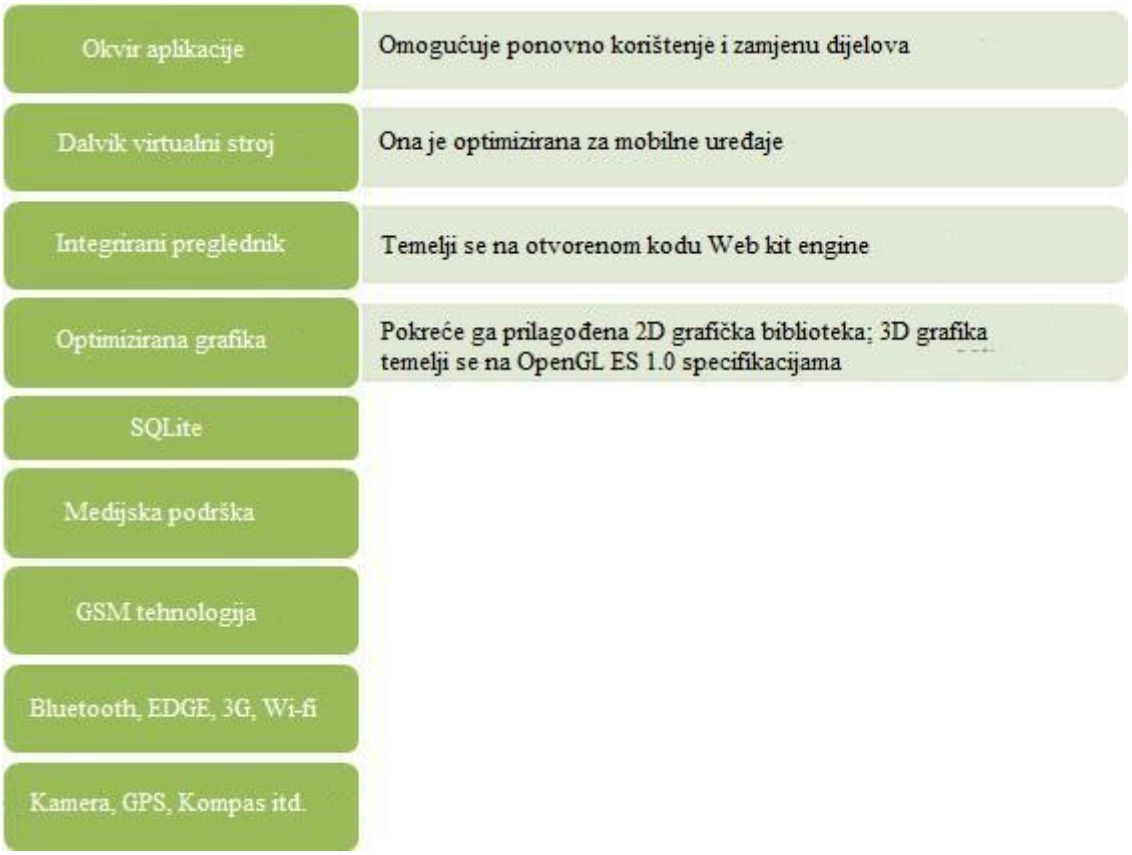
Povijest Android sustava:

- Android 1.0 izdan je u rujnu 2008. godine, za mobilni uređaj T-Mobile G1
- Android 1.1 izdan je veljači 2009. Radilo se o ažuriranoj verziji za T-Mobile G1
- Android 1.5 - Cupcake izdan je u travnju 2009. godine. Ova verzija je znatno poboljšala Android sustave. Omogućila je virtualnu tipkovnicu, snimanje i slikanje te mogućnost odlaska na YouTube. Od ove verzije više nije potrebno imati fizičku tipkovnicu.
- Android 1. - Donut izdan je u rujnu 2009. godine. Ova verzija je omogućila bolje korištenje glasovnog pretraživanja te poboljšanu verziju galerije slika i video zapisa.
- Android 2.0 - 2.1 - Eclair izdan je u listopadu 2009. godine. Ova verzija je donijela poboljšanja u vidu bolje mogućnosti korištenja kamere, zumiranje te korištenje bljeskalice, poboljšano korisničko sučelje, sinkronizaciju kontakata.
- Android 2.2. - FroYo izdan je u travnju 2010. godine. Ova verzija omogućava bolju kontrolu ekrana, izbor više jezika, bolju povezanost preko Bluetooth sustava.
- Android 2.3 - Gingerbread izdan je u prosincu 2010. godine. Ova verzija donijela je znatna pojednostavljenja, ali i znatna ubrzanja korisničkog sučelja, bolju mogućnost korištenja opcije kopiraj/zalijepi, mogućnost slikanja prednje i stražnje kamere, barometar, žiroskop, rotaciju zaslona.
- Android 3.x - Honeycomb izdan je isključivo za tablete. Ova verzija donijela je velika poboljšanja, ali nije imala prevelik uspjeh na tržištu.
- Android 4.0.x - Ice Cream Sandwich. Ova verzija donijela je znatna poboljšanja u vidu korisničkog sučelja, bolji izbornik aplikacija, mogućnost odabira načina zaključavanja. Inačica Android 4.0.x postigla je velik uspjeh na tržištu.
- Android 4.1.x - 4.3.x - Jelly Bean izdan je u srpnju 2012. godine. Android se širi na tržištu i postaje jači nego Apple-ov IOS. Donesena su znatna poboljšanja u slikanju i snimanju slika i video zapisa, poboljšanje Bluetooth-a, znatno bolja 2D/3D grafika.
- Android 4.4 - Kit Kat izdan je u listopadu 2013. godine. Uvodi se 64-bitni procesor i poboljšava vremešni Dalvik.
- Android 5 - Lollipop donosi racionalno korištenje baterije, pametne satove, omogućeno je razdvajanje poslovnih i privatnih podataka na mobilnim uređajima.
- Android 6 - Marshmallow omogućava bolje iskorištavanje SD kartice, poboljšan čitač otiska prsta, smrzavanje aplikacije prilikom zaključavanja ekrana.

- Android 7 – Nougat omogućava brisanje svih otvorenih aplikacija, spremanje podataka na jednostavan način, omogućava ponovnog slanja poruka s bilo kojeg izvora, prikazivanje svih obavijesti na jednome mjestu, jednostavno upravljanje obavijestima.

2.2. Značajke Android platforme

Android je operacijski sustav koji ima značajne sposobnosti pokretati veliki broj aplikacija za pametne telefone. Android aplikacije u velikoj mjeri olakšavaju svakodnevni život.[2] Slika 2.1. prikazuje specifikacije Android platforme.



Slika 2.1. Specifikacije Androida [3]

Ova platforma je zamišljena za rad na uređajima većeg zaslona. U tu grupu ulaze pametni telefoni. Biblioteka ovih uređaja temelji se na OpenGL ES 2.0 specifikacijama, a koriste 2D i 3D grafičku biblioteku. Kada se radi s bazama podataka, za pohranu koriste se SQLite relacije koje se pišu u C programskom jeziku. Drugi sustavi baze podataka tretiraju kao zaseban proces, ali kod Android platforme to je sastavni dio aplikacije. Povezivanje s drugim uređajima je moguće preko Bluetooth-a, WiMAX-a, IDEN-a, CDMA-a. Nije dozvoljeno povezivanje preko Ad hoc bežične mreže te preko proxy poslužitelja. Bluetooth podržava sučelje za upravljanje drugim uređajima (primjerice televizija, radio) te prijenos audio zapisa s jednog uređaja na drugi. Od

Android verzije 3.0 postoji podrška za spajanje uređaja za upravljanje (tipkovnica, miš, igraće palice), dok na prijašnjim verzijama podrška je bila napravljena od samih proizvođača takvih uređaja.

Na početku Android je podržavao samo 26 jezika. Pojavom Gingerbread-a povećava se broj podržanih jezika na 57.

Pristup Internetu temelji se na WebKit-u koji je kombiniran sa Google-ovim V8 JavaScript engine-om. Prije verzije Android Froyo nije bilo omogućeno koristiti uređaj kao izvorišnu točku za spajanje na Internet osim u slučaju ako je proizvođač uređaja ugradio tu funkciju na uređaj ili instalacijom neslužbenih aplikacija. Verzije, počevši od Android Froyo pa naviše imaju ovu opciju ugrađenu na samim uređajima kao standardnu opciju.

Java programski jezik je najzastupljeniji kod većeg broja aplikacija, ali kod samih uređaja ne postoji Java virtualni stroj (eng. Java Virtual Machine). Kako bi se omogućilo pokretanje Java aplikacija koristi se Dalvik virtualni stroj. Dalvik virtualni stroj je prilagođen za mobilne uređaje. Koristi se za optimalni rad virtualnog stroja za memoriju, vijek trajanja baterije i performanse samog uređaja. [4]

Android platforma podržava formate za slike .jpg, .gif, .png, .bmp. Video formati koje Android podržava su .3gp, .mp4 a od verzije 2.3.3 i nadalje .webm. Što se tiče audio formata Android podržava .3gp, .mp3, .mp4, .mid, .aac. Od verzije 3.1 i na slijedećim verzijama podržan je i .flac audio format.

Flash priključak (eng. Flash plugin) daje podršku za HTML progresivno skidanje podataka, Adobe Flash Streaming i HTTP Dynamic Streaming.

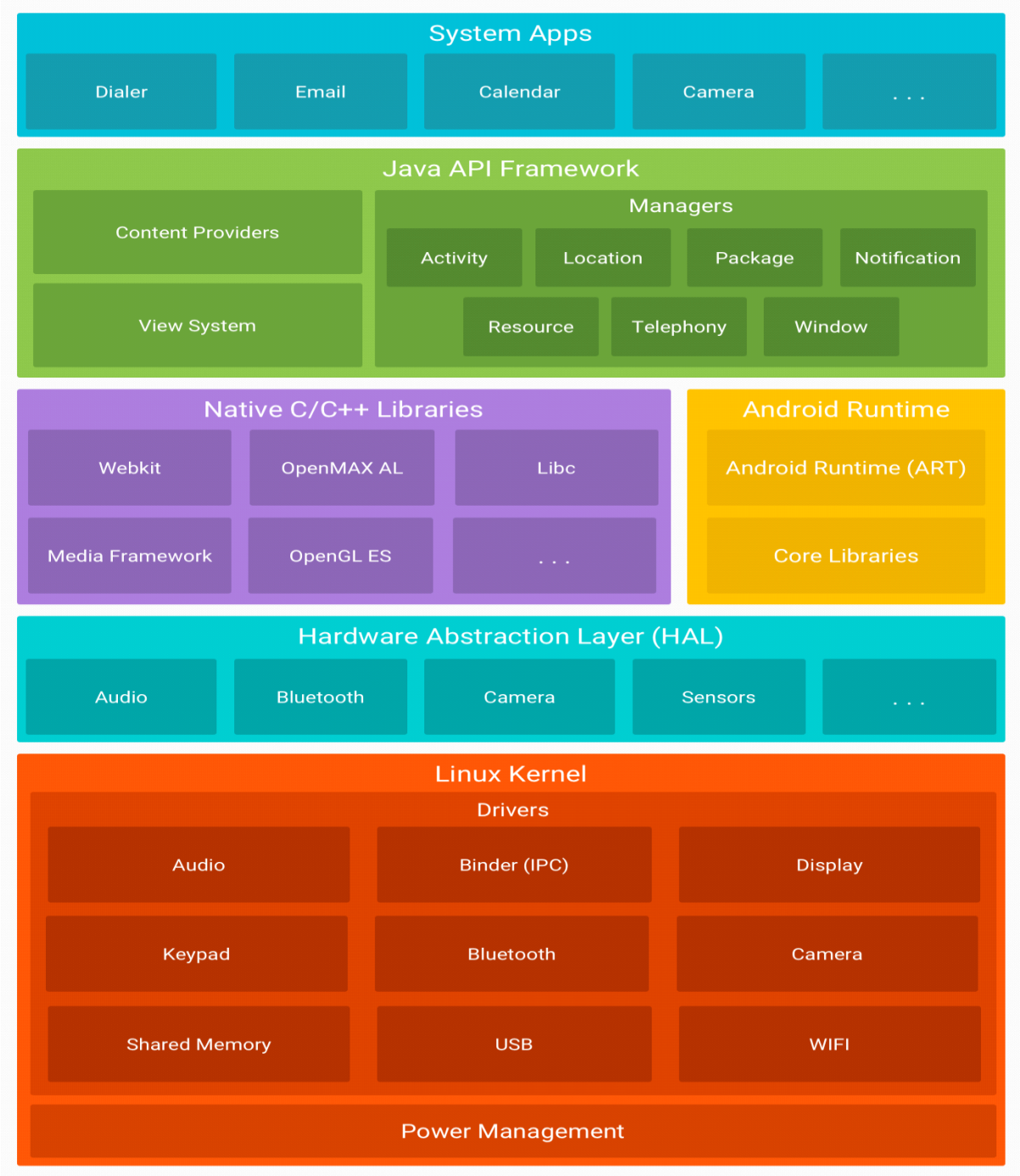
Od dodatnih hardvera moguće je koristiti žiroskop, GPS, magnetometar, ekran osjetljiv na dodir, senzore blizine i termo senzore te postoji mogućnost korištenja više dodira u isto vrijeme (eng. multi-touch).

Glasovna pretraga postoji od prvih Android verzija, ali tek od verzije 2.2 omogućeno je pozivanje, pisanje te navigacija pomoću glasa.

Sigurnost na Android platformama omogućava da si svaki korisnik osobno odredi koja će mu biti sigurnosna lozinka. Ima različitih mogućnosti od toga da nema nikakvu lozinka do otiska prsta ili brojevnih kombinacija kao i mogućnost ostavljanja određenog uzorka koji je potrebno ponoviti prilikom svakog novog otključavanja mobilnog uređaja.

2.3. Arhitektura Android platforme

Android platforma je zamišljena kao platforma otvorenog koda, koji se temelji na Linux bazi te su sve osnovne funkcije zapisane C/C++ programskim jezikom. Platforma je kreirana za pametne mobilne uređaje. U dijagramu na slici 2.2. vide se glavne komponente Android platforme.



Slika 2.2. Komponente Android platforme [5]

Arhitektura Androida promatra se kao jedan stog i ima nekoliko različitih razina. Na samom dnu nalazi se Linux jezgra (eng. kernel). Ova razina omogućava nisku razinu iskorištene memorije samog uređaja te omogućava funkciju nizanja. Također omogućava međuprocesorsku komunikaciju te izmjenu podataka između različitih procesa. Preko ovog sloja kontrolira se i samo napajanje.

Slijedeći sloj je apstraktni sloj hardvera (eng. hardware abstraction layer, HAL) koji osigurava standardno sučelje koje omogućava kompatibilnost samog uređaja s visokom razinom Java API radnog okruženja.

HAL ima više biblioteka koje se nalaze u modulima. Svaki od tih modula osigurava implementaciju za svaki od specifičnih tipova hardverskih komponenti, kao što su kamera ili Bluetooth. Kada radno okruženje API pozove određeni hardware uređaja, sustav otvara biblioteku za tu komponentu.

Nakon ovog sloja slijedi Android Runtime sloj. Ovaj sloj koristi se za pokretanje više virtualnih uređaja pri čemu se koristi niska razina memorije za pokretanje aplikacija na uređaju. Bytecode format napravljen je specijalno za Android i izrađen je na način da ostavlja što je moguće manji memorijski otisak. Izradom alata, kao što je Jack, prevođenje koda Java izvora u DEX bytecode omogućuje pokretanje programa na Android platformi. [6]

Jack je Android alatna traka koja pretvara Java kod u Android DEX bytecode. Zamjenjuje prethodnu Android alatnu traku koja je sadržavala više alata. Alati prijašnje trake su ProGuard, dx, javac.[7]

Android runtime ima neke od slijedećih mogućnosti:

- Ispred vremena (eng. Ahead-of-time) i na vrijeme (eng. just-in-time) prevođenje koda
- Optimizacija prikupljanja otpada
- Bolja mogućnost otklanjanja pogreške

Na istoj razini se stoga nalaze izvorne C/C++ biblioteke. Velika većina Android sustava izrađena je pomoću izvornog koda. Ovom kodu potrebne su izvorne biblioteke pisane u C te u C++ programskom jeziku. Android platforma koristi Java radno okruženje da otkrije funkcije nekih od izvornih biblioteka u aplikaciji. Na primjer, može se pristupiti OpenGL ES preko Android okvira Java OpenGL API za unos crteža te upravljanje unesenim crtežima u samoj aplikaciji. U koliko se razvija aplikacija koja zahtjeva C ili C++ kod, preporuča se korištenje Android NDK kako bi se pristupilo izvornim biblioteke platforme.

Slijedeći sloj u stogu je Java API radno okruženje. Najvažnije značajke Androida su dostupne kroz API blokove. Ovi blokovi su pisane Java programskim jezikom. Preko API blokova

omogućenom je stvaranje Android aplikacija. Bez postojanja ovih blokova nije moguće stvaranje Android aplikacije.

Ovi blokovi omogućavaju izgradnju aplikacijskog korisničkog sučelja. Prošireni sustav prikaza koji omogućava kreiranja mreža, blokova za pisanje i tipki.

Jednostavno upravljanje voditeljem resursa (eng. Resource Managerom) koji dozvoljava pristup grafici i projekt (eng. layout) datotekama iako to nisu kodovi.

Upravitelj obavijesti (eng. Notification Manager) omogućuje prikaz svih upozorenja koji se javljaju u aplikaciji. Voditelj aktivnosti (eng. Activity Manager) omogućava upravljanjem životnog vijeka aplikacije.

Pružatelj sadržaja (eng. Content Providers) daju mogućnost svakoj aplikaciji da pristupi podatcima druge aplikacije. Primjer toga je pristup datotekama drugog uređaja. Glavna karakteristika ovog sloja je omogućavanje programerima kompletan pristup svim API okvirima (eng. Framework) koje koristi aplikacija.

Na vrhu stoga nalaze se aplikacije sustava (eng. System Apps). Na ovoj razini obuhvaćene su sve aplikacije koje su instalirane zajedno sa Android sustavom i tvornički su obuhvaćene prilikom lansiranja mobilnog uređaju na tržište. U te aplikacije se ubrajaju aplikacije za slanje poruka, uspostavu poziva, kalkulator, kalendar i ostale aplikacije koje se dobiju kupnjom samog uređaja. Kako je Android platforma otvorenog tipa svaka osoba može svoj uređaj prilagoditi sebi. To znači da svaka aplikacija koja se tvornički nalazi na mobilnom uređaju može biti zamijenjena bilo kojom aplikacijom koja ima istu namjenu kao i trenutno instalirana aplikacija. U slučaju da je na uređaju instalirano više aplikacija koje imaju jednako svojstvo (npr. više aplikacija za obradu teksta), Android sustav će korisniku ponuditi sve aplikacije koje se nalaze na uređaju tako da korisnik izabere aplikaciju koja njemu najviše odgovara za obavljanje željenog posla.

2.4. Razvojno okruženje

Aplikacija za igru „Čovječe ne ljuti se“ razvijana je u programu Android Studio. Izbor ovog razvojnog okruženja u odnosu na Eclipse je taj što je Android Studio znatno napredniji od Eclipse i znatno je pojednostavljena njegova upotreba te rad sa grafikama. Prilikom izrade aplikacije koristi se programski jezik Java uz korištenje paketa SDK. Razvojno okruženje omogućava korištenje biblioteka, ispravljanje pogrešaka (eng. debugger), emulatora za testiranje aplikacija te razne alate za upravljanje radom u bazama podataka. Od 2015. godine Eclipse IDE je zamijenjen s Android Studio i Google ga je službeno prozvao zastarjelim kako bi se

usredotočio na što bolji razvitak Android Studija. Programeri pri razvoju aplikacija koriste Javu te XML datoteke za razvoj grafike i pomoću Java Development Kita te Apache Ant stvaraju Android aplikaciju. U slučaju da programeri ciljaju mobilne uređaje koji su stariji i ne podržavaju nove nadogradnje omogućava se izrada aplikacija preko ostalih razvojnih programa. Korištenjem mobilnog uređaja za testiranje aplikacija potrebno je izraditi apk dokument. Ovaj dokument sadrži sve potrebne kodove i grafike za uspješnu instalaciju aplikacije. [8]

2.5. Java

Java je objektno orijentirani programski jezik razvijen u kompaniji Sun Microsystems. Java se bazira na C++ i neovisna je od operacijskog sustava. Ima pojednostavljenu sintaksu, stabilnije runtime sustave i pojednostavljenu kontrolu memorije. [9]
Programeri imaju mogućnost izbora Java platformi ovisno o poslu kojim se bave.

- Java tehnologija u osobnim računalima (Java SE)

  Omogućava razvoj aplikacija za desktop na siguran način. Moguće je koristiti Javu na slijedećim platformama: Macintosh, Linux, Microsoft Windows i Sun Solaris.

- Java tehnologija u srednjim i velikim poslovnim sustavima (Java EE)

  Omogućen je na što jednostavniji način razvijati složene probleme. Kao što sam naziv kaže namijenjen je srednjim i velikim poslovnim sustavima.

- Java tehnologija u mobilnim i malim uređajima (Java ME)

  Omogućen je razvoj mobilnih aplikacija na siguran i otporan način. Pojednostavljen je način pisanja koda kako bi se dobilo fleksibilno okruženje u kojemu će svaki programer imati mogućnost izrade i podrške mrežnih i lokalnih aplikacija.

Iako Sun drži prava na ime Java, Javu je moguće naći potpuno besplatno na njihovim Internet poslužiteljima. Glavni problem kod klasičnih programa je taj što se operacijski sustav mora prilagođavati razvojnom okruženju. Kod Jave to nije potrebno zbog toga što je Java izrađena na način da se ne mijenja struktura operacijskog sustava. Java sadrži veliki broj klasa koje omogućavaju jednostavan rad i čine ju najboljim izborom za izradu aplikacija. Pokazala se najbolja u problemima u kojima je potrebna velika brzina rada programa. Visok stupanj sigurnosti i pouzdanosti su jedne od glavnih karakteristika Jave uz jednostavnu detekciju i ponuđene jednostavne savjete kako otkloniti pojedini problem u kodu.

Zbog navedenih karakteristika Java je od svog lansiranja na tržište do danas prikupila između 7 i 10 milijuna korisnika diljem svijeta. [10]
Jedan od glavnih razloga velikog broja korisnika Jave krije se u Google-ovom sustavu Android koji koristi Javu u razvojnom okruženju za stvaranje aplikacija za pametne mobitele.

2.6. Povijest i upute za igranje igre „čovječe, ne ljuti se „

2.6.1. Povijest igranja igre „čovječe, ne ljuti se „

Igra iz koje se razvila popularna igra „Čovječe, ne ljuti se“, zvala se Pachisi. Pachisi se izvorno igrao u Indiji. Prvi pisani dokazi o ovoj igri datiraju iz 6. stoljeća. Igra se zasnivala na strategiji u kojoj je bilo nužno surađivati s još nekim od igrača da bi se osigurala pobjeda. Kako je Indija bila Engleska kolonija, engleski časnici i vojnici su se zainteresirali za igru i ona je prenesena u Englesku. Igra je doživjela znatne promjene u odnosu na originalnu igru iz Indije. Igra „Čovječe, ne ljuti“ je patentirana je 1896. godine pod nazivom Ludo. Igra je prilagođena za sve uzraste, a prvenstveno je namijenjena za mlađe uzraste. Gdje se bacanjem kocke i uz malo sreće svi mogu zabaviti. Prvo izdanje igre čovječe, ne ljuti se prikazano na slici 2.3.

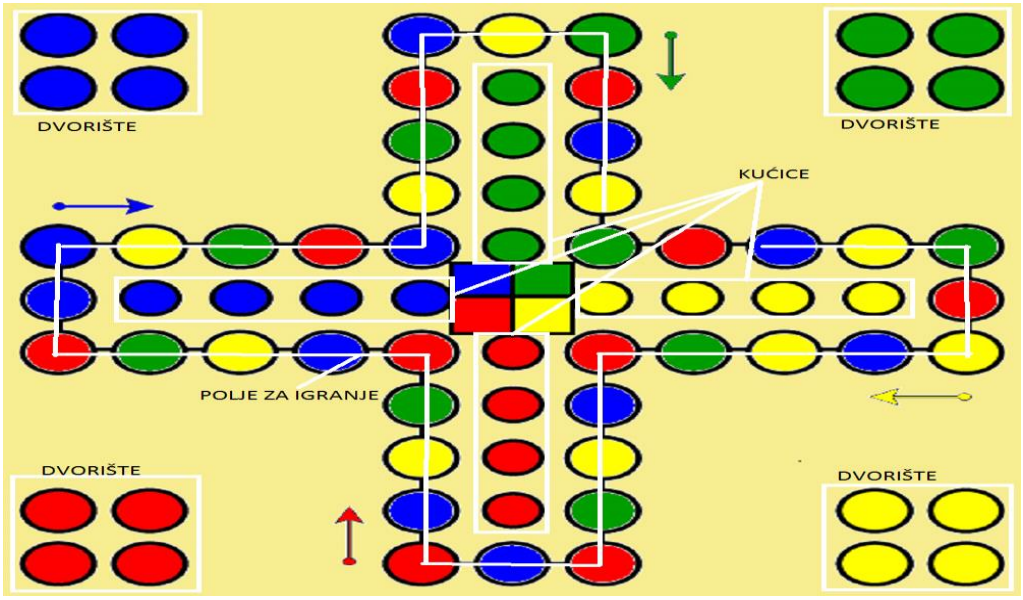


Slika 2.3. Prvo izdanje igre čovječe, ne ljuti se. [11]

2.6.2. Pravila za igranje igre „čovječe, ne ljuti se“

- Igru mogu igrati 2-4 igrača.
- Svaki igrač na početku unosi svoje ime i odabire željenu boju.
- Svaki igrač ima 4 igrače figurice koje se nalaze u dvorištu.
- Igra se igra u smjeru kazaljke na satu.
- Igrač započinje igru bacanjem kockice 3 puta, ako igrač u 3 bacanja dobije broj 6 smije izaći iz svog dvorišta sa svojom figuricom na igraču ploču te ponovno baciti kockicu i pomaknuti figuru za broj koji je dobio na kocki.
- Svaki igrač ima pravo bacati kockicu 3 puta dok ne izađe iz svog dvorišta na polje sa svojim pijunom, nakon toga baca u svakom krugu kockicu samo jednom po krugu.
- Ako se dva igrača nađu na istom polju, igrač koji drugi stiže na to polje jede igrača koji je prije njega bio na tom polju i pojedeni igrač se vraća natrag u svoje dvorište.
- Ako se igrač nalazi pred kućicom i nije u mogućnosti ući mora preskočiti krug do ponovnog bacanja.
- Pijuni se moraju slagati jedan iza drugoga u kućici, preskakanje u kućici nije dozvoljeno.
- Važno je naglasiti da su pijuni zaštićeni od jedenja samo u kućici dok na polju nema sigurnih mjesta.
- Pobjednik je onaj koji prvi složi sve pijune u kućicu.
- Udruživanje je dozvoljeno, ali samo jedan igrač može izaći kao pobjednik iz igre.

Grafički prikaz pojedinih dijelova igre prikazan je na slici 2.4.



Slika 2.4. Prikaz pojedinih dijelova objašnjenih u pravilima igre

3. ANALIZA SUSTAVA

U ovom poglavlju opisana je ideja izrade aplikacije, njen izgled te kratak opis cijelog sustava od pokretanja aplikacije do svih važnih točaka kojima je obuhvaćena aplikacija. Važan je dijagram toka aplikacije koja će na jednostavan način dati uvid u opisane dijelove aplikacije. Na kraju će biti kratak opis uređaja na kojemu će se vršiti ispitivanje aplikacije.

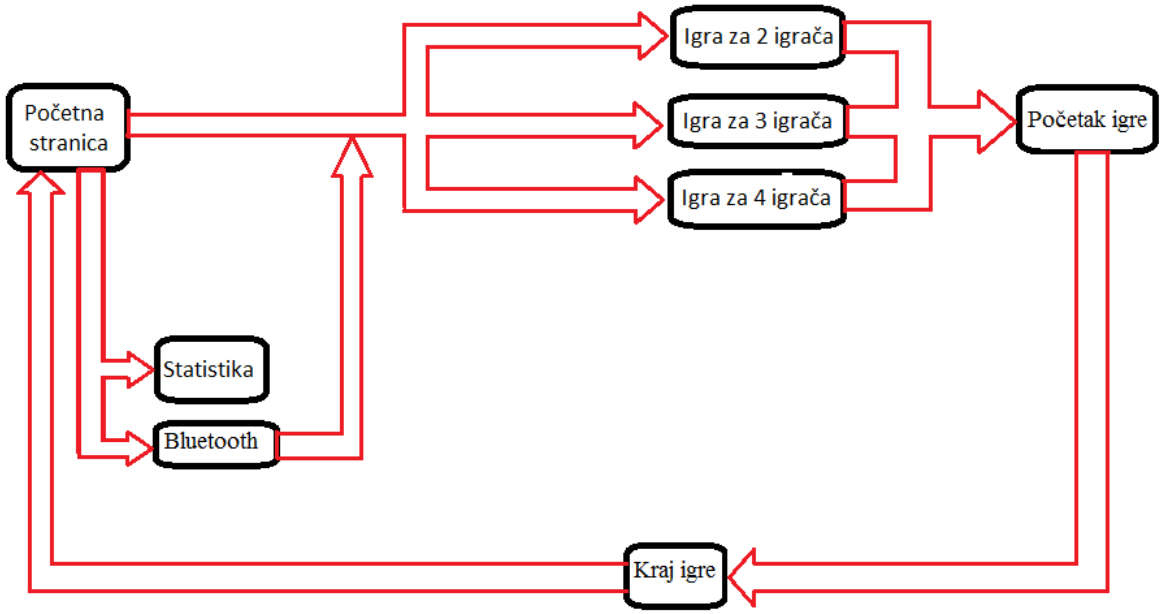
3.1. Ideja izrade aplikacije

Pokretanjem aplikacije na ikonu koja se nalazi na zaslonu mobilnog uređaja ulazi na početnu stranicu aplikacije. Početna stranica omogućava izbor broja igrača koji će sudjelovati u igri. Igrači mogu birati igru od 2-4 igrača. Nadalje, svaki igrač može klikom na gumb statistiku dobiti uvid o svakom igraču koji je sudjelovao u igri. Također se u statistici nalazi i statistika cjelokupne igre. Igrači mogu odabrati igru preko Bluetooth-a. Igra ostaje potpuno ista, ali je jedina razlika ta što svako može igrati na svom uređaju bez „guranja“ na jednom uređaju.

Odabirom broja igrača omogućava se upis podataka (personalizacija) pojedinog igrača. Prilikom personalizacije odabire se i boja s kojom igrači žele igrati. Pritiskom na gumb „Igraj“, spremaju se podatci u datoteku i pokreče se igra. Nakon što jedan od igrača pobjedi, igra se prekida i svi podatci o pobjedniku se spremaju u datoteku. Isto se događa ako igrači igraju preko Bluetooth-a nakon završetka igre.

Aplikacija će biti izrađena u Android Studiju, a testirana na mobilnom uređaju Samsung Galaxy S5.

Na slici 3.1. prikazan je dijagram toka aplikacije.



Slika 3.1. Dijagram toka aplikacije

4. PROGRAMSKO RJEŠENJE ZADATKA

U ovom poglavlju opisan je postupak izrade Android aplikacije. Zbog kompleksnosti koda i velikog broja projekcija (engl. layout) ovdje su izdvojeni samo određeni dijelovi koda koji su bitni za rad aplikacije.

Aplikacija se sastoji od 11 klasa i 5 projekcija.

Prvi korak bila je izrada početnog zaslona iz kojeg se ulazi u izbornik gdje se vrši personalizacija, izbor boje i odabir broja igrača. Nakon toga izrađena je igra, potom baza podataka za spremanje svih podataka potrebnih za izradu statistike, a zatim je napravljen tablični prikaz svih podataka iz baze. Potom je opisan način izrade i rada igranja preko Bluetooth-a.

4.1. Početni zaslon

Grafika u Android aplikacijama izrađuje se u XML jeziku, tako su i u aplikaciji „Čovječe ne ljuti se“ početna projekcija (engl. layout) kao i sve ostale izrađene u XML jeziku. Izgled početnog zaslona prikazan je na slici 4.1.



Slika 4.1. Početni zaslon aplikacije

```
<Button
    android:id="@id/nova_igra"
    style="@style/button_text"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:background="@drawable/red_button"
    android:onClick="Nova"
    android:text="NOVA IGRA"/>


<Button
    android:id="@id/statistika"
    style="@style/button_text"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:background="@drawable/red_button"
    android:onClick="Statistika"
    android:text="STATISTIKA"/>
<Button
    android:id="@id/prijatelj"
    style="@style/button_text"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:background="@drawable/red_button"
    android:onClick="Prijatelj"
    android:text="POZOVI PRIJATELJA"/>
```

Programski kod 4.1. XML kod početnog zaslona

Početni zaslon ima 3 tipke koje na jednostavan način omogućavaju odabir željene radnje. Iz koda se vidi sličnost između te tri tipke, ali svaka ima svoj identifikacijski broj pomoću kojega se poziva kod u programu i omogućava izvršavanje potrebne radnje. Npr. pritiskom na tipku *NOVA IGRA* prelazi se iz početne projekcije na projekciju preko koje igrač odabire boju, broj igrača i upisuje ime igrača te započinje novu igru.

Naredba *style* prikazuje o kojem se tipu podataka radi, a u ovom slučaju radi se o *button_text* tipu podataka.

Naredbe *android: layout_width* i *android: layout_height* omogućavaju određivanje širine i dužine same tipke. Ovaj parametar se može podešavati mišem prilikom postavljanja grafičkog prikaza ili u XML kodu unošenjem željene veličine tipke.

Naredba *android:background* omogućava podešavanje pozadine tipke. U aplikaciji je odabrana crvena boja pozadine pomoću *@drawable/red_button*.

Naredba *android:onClick* određuje da se u akciju kreće pritiskom na tipku, dok se pomoću naredbe *android:text* postavlja naziv tipke.

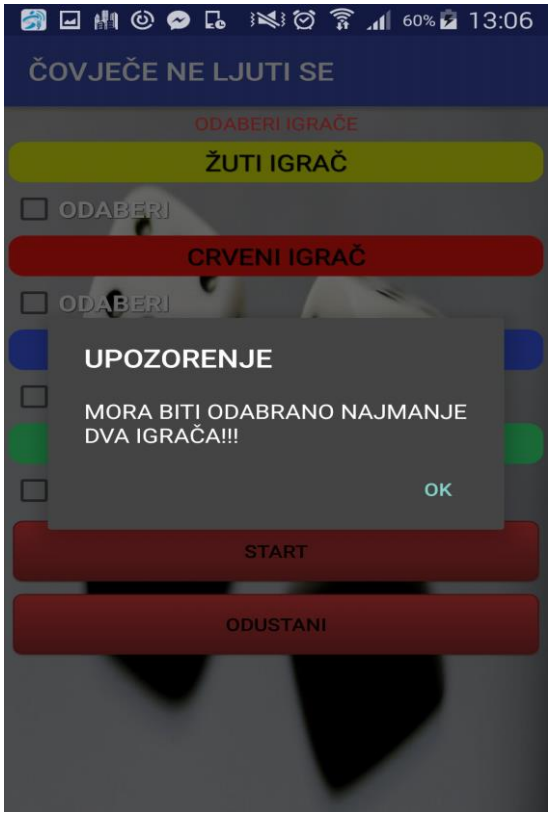
4.2. Nova igra

U projekciji *nova igra* igračima je omogućeno odabrati broj igrača i boju te upisati imena sudionika igre. Izgled zaslona prikazan je na slici 4.2.



Slika 4.2. Odabir postavki nove igre

Nakon odabira svih potrebnih parametara pritiskom na tipku *START* započinje igra. Igru mogu igrati najmanje dva, a najviše četiri igrača. U slučaju da nije odabran dovoljan broj igrača ispisuje se poruka koja je prikazana na slici 4.3., a kod za provjeru i ispis upozorenja prikazan je u programskom kodu 4.2.



Slika 4.3. Upozorenje da se mora odabrati minimalno dva igrača

```
start = (Button) findViewById(R.id.button_start);
odustani = (Button) findViewById(R.id.button_odustani);
Log.d("button", "button");
start.setOnClickListener((arg0)  › {

        int i = 0;
        //provjera da li su odabrana najmanje dva igrača

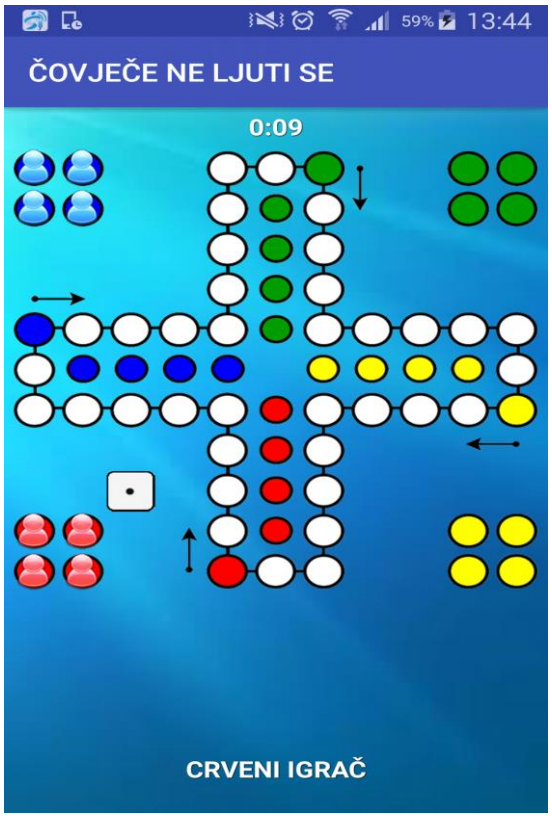
        i = igrac.size();
        //u slučaju da je izabrano manji od dva igrača iskače alert
        if (i < 2) {
            alertDialog("MORA BITI ODABRANO NAJMANJE DVA IGRAČA!!!", "UPOZORENJE");
```

Programski kod 4.2. Provjera i ispis upozorenja da treba odabrati minimalno dva igrača

U programskom kodu 4.2. naredba *start* omogućava pokretanje igre, ali samo u slučaju kada je zadovoljena varijabla i koja je na početku postavljena u 0 (*int i = 0*).

U slučaju da je i >= 2 započinje se postavljanje igrače ploče i omogućava se igranje igre. U slučaju kada je i < 2 ispisuje se u petlji pomoću funkcije *alertDialog* poruka u kojoj se upozorava sudionike igre da ih je premalo, odnosno da za početak igre moraju biti odabrana najmanje dva igrača.

Nakon što se uspješno prođe provjera broja igrača slijedi postavljanje ploče za igranje i postavljanje figura na ploču, a izgled zaslona prikazan je na slici 4.4., dok je kod postavljanja figurica na početne pozicije prikazan u programskom kodu 4.3.



Slika 4.4. Početne postavke ploče i figurica

```
private void setPocetnePozicije() {
    for (Igrac i : getIgarci_u_igri()) {

        if (i.getBojaIgraca().equalsIgnoreCase(BojaIgraca.CRVENA.toString())) {
            i.getFigura1().setXY(xx - 5 * korak, yy + 5 * korak);
            i.getFigura2().setXY(xx - 4 * korak, yy + 5 * korak);
            i.getFigura3().setXY(xx - 5 * korak, yy + 4 * korak);
            i.getFigura4().setXY(xx - 4 * korak, yy + 4 * korak);
        } else if (i.getBojaIgraca().equalsIgnoreCase(BojaIgraca.ZELENA.toString())) {
            i.getFigura1().setXY(xx + 5 * korak, yy - 5 * korak);
            i.getFigura2().setXY(xx + 4 * korak, yy - 5 * korak);
            i.getFigura3().setXY(xx + 5 * korak, yy - 4 * korak);
            i.getFigura4().setXY(xx + 4 * korak, yy - 4 * korak);

        } else if (i.getBojaIgraca().equalsIgnoreCase(BojaIgraca.PLAVA.toString())) {
            i.getFigura1().setXY(xx - 5 * korak, yy - 5 * korak);
            i.getFigura2().setXY(xx - 4 * korak, yy - 5 * korak);
            i.getFigura3().setXY(xx - 5 * korak, yy - 4 * korak);
            i.getFigura4().setXY(xx - 4 * korak, yy - 4 * korak);

        } else {
            i.getFigura1().setXY(xx + 5 * korak, yy + 5 * korak);
            i.getFigura2().setXY(xx + 4 * korak, yy + 5 * korak);
            i.getFigura3().setXY(xx + 5 * korak, yy + 4 * korak);
            i.getFigura4().setXY(xx + 4 * korak, yy + 4 * korak);

        }
    }
}
```

Programski kod 4.3. Metoda postavljanja figura na početne pozicije

Ova metoda prilikom prvog iscrtavanja postavlja figurice na početne pozicije. Prilikom odabira igrača popunjava se varijabla „i“ koja označava broj koliko igrača sudjeluje u igri. Za svakog igrača prema odabranoj boji postavljaju se figurice na početnu poziciju. U if funkciji provjerava se da li je igrač odabrao određenu boju i na osnovu te usporedbe postavljaju se figurice ili preskače boju ako ona nije odabrana. Naredba *i.getFigura* unutar if funkcije uzima dužinu i širinu ekrana i na osnovu toga izračunava poziciju na koju će postaviti figurice. Na programskom kodu 4.4. prikazan je kod kojim se poziva kockica.

```
public void bacenaKocka() {

    if (kocka.getBrojKocke() == 0) { // prvo bacanje
        kocka.vracanjeBacanjaNaPocetak();
        kocka.bacanje();

    } else if (kocka.getBrojKocke() != 6 && igrac.nastavakBacanja() && kocka.getPokusajbacanja() < 3) {

        kocka.novoBacanje();
        kocka.bacanje();

        if (kocka.getPokusajbacanja() == 3) {
            provjeraIgraca();
        }
    } else {
        provjeraIgraca();
    }

}
```

Programski kod 4.4. Metoda koja poziva kockicu

Ova metoda poziva klasu kockica svaki puta kada je igrač na potezu. Metoda vraća vrijednost kockice svaki puta kada se ostvari dodir na kocku te na osnovu dodira vraća se slučajno generiran broj od 1-6 na zaslon ekrana u obliku vrijednosti broja prikazanog na kockici. Također ova metoda prati je li je igrač dobio broj 6 i broj pokušaja bacanja. Svaki igrač ima pravo tri pokuš*a*ja dobivanja šestice za početak igre. U slučaju da dobije šesticu omogućava se pomicanje figure i ponovno bacanje, dok u slučaju ako u tri bacanja ne dobije šesticu kocku baca slijedeći igrač.

```
private int[] provjeraIstePozicijeIgraca(String boja, int koraci) {

    int[] ret = new int[2];
    ret[0] = 0;
    ret[1] = 0;
    for (Igrac igr : igarci_u_igri) {
        if (igr.getBojaIgraca() != boja) {
            ret[1] = igr.provjeraKolizije(koraci);
            if (ret[1] > 0) {
                ret[0] = igr.getPoredak();
                break;
            }
        }
    }
    return ret;
}
```

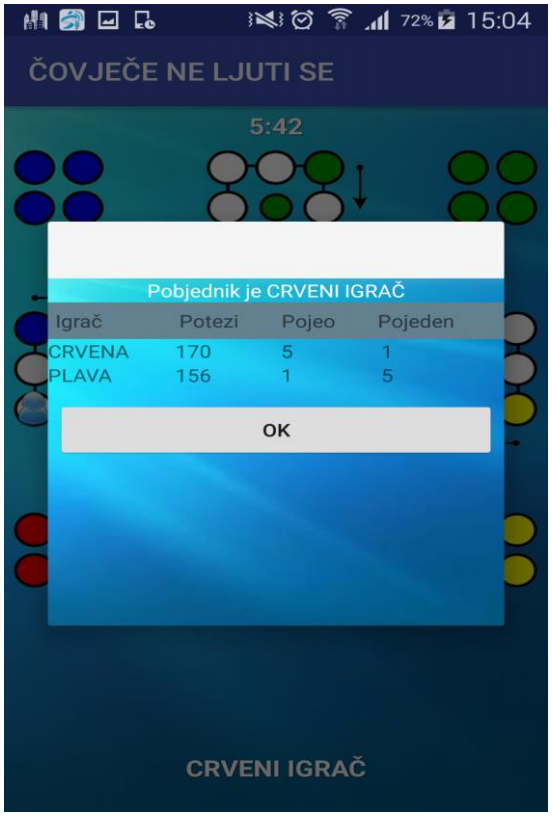
Programski kod 4.5. Provjera sudara figurica

Ova metoda provjerava nalaze li se figurice na istoj poziciji.

U slučaju da se figurice iste boje trebaju naći na istoj poziciji program ne dopušta pristup figurici koja druga dolazi na poziciju.

Isto tako ako se dvije figurice različitih boja trebaju naći na istoj poziciji, figurica koja druga stiže na istu poziciju izbacuje onu figuricu koja se već nalazi na toj poziciji.

Izgled zaslona na kraju igrice prikazan je na slici 4.5., dok je kod za kreiranje tablice rezultata unutar dijaloga prikazan na programskom kodu 4.6.



Slika 4.5. Ispis pobjednika i statistika igre nakon završetka igre

```
if (igr.getBojaIgraca().equalsIgnoreCase("crvena")) {

    trCrvena.addView(igrac);
    trCrvena.addView(potezi);
    trCrvena.addView(pojeo);
    trCrvena.addView(pojeli);
    tl.addView(trCrvena, new TableLayout.LayoutParams(
            TableRow.LayoutParams.WRAP_CONTENT,
            TableRow.LayoutParams.WRAP_CONTENT));
} else if (igr.getBojaIgraca().equalsIgnoreCase("plava")) {
    trPlava.addView(igrac);
    trPlava.addView(potezi);
    trPlava.addView(pojeo);
    trPlava.addView(pojeli);
    tl.addView(trPlava, new TableLayout.LayoutParams(
            TableRow.LayoutParams.WRAP_CONTENT,
            TableRow.LayoutParams.WRAP_CONTENT));
} else if (igr.getBojaIgraca().equalsIgnoreCase("zelena")) {

    trZelena.addView(igrac);
    trZelena.addView(potezi);
    trZelena.addView(pojeo);
    trZelena.addView(pojeli);
    tl.addView(trZelena, new TableLayout.LayoutParams(
            TableRow.LayoutParams.WRAP_CONTENT,
            TableRow.LayoutParams.WRAP_CONTENT));
} else if (igr.getBojaIgraca().equalsIgnoreCase("zuta")) {

    trZuta.addView(igrac);
    trZuta.addView(potezi);
    trZuta.addView(pojeo);
    trZuta.addView(pojeli);
    tl.addView(trZuta, new TableLayout.LayoutParams(
            TableRow.LayoutParams.WRAP_CONTENT,
            TableRow.LayoutParams.WRAP_CONTENT));
}
```

Programski kod 4.6. Kreiranje tablice rezultata unutar dijaloga

U kodu se može vidjeti način na koji se kreiraju pogledi na rezultate igre nakon pobjede jednog od igrača.

Kod ispisa pobjednika koristi se funkcija *pobjednik.set-text* (Pobjednik je *+igrač.getIme());*. Za kreiranje pogleda koristi se funkcija *.addView* te se u zagradi upisuje naziv vrijednosti koju želimo ispisati u dijalog. Korištenjem boje igrača ispisuje se naziv svakoga igrača ili ako nije upisano ime koristi se automatski postavljeno ime (CRVENI, ZELENI, ŽUTI ILI PLAVI IGRAČ). Za kreiranje redaka i tablični ispis koristi se funkcija *TableRow*i.

U slučaju da igrač želi prekinuti igru ranije, na zaslonu mobilnog uređaja pojavit će se poruka u kojoj se pita da li želite izaći iz igre. Upit za potvrdu izlaska iz igre prikazan je na slici 4.6.



Slika 4.6. Upit da li igrač želi izaći iz igre

```
public void onBackPressed(){
    AlertDialog.Builder builder = new AlertDialog.Builder(Igra.this);
    builder.setMessage("Želite li izaći iz igre?");
    builder.setCancelable(true);
    builder.setPositiveButton("Ok", (dialog, id) -> { finish(); });
    builder.setNegativeButton("Canel", (dialog, id) -> { dialog.cancel(); });
    AlertDialog alert = builder.create();
    alert.show();
}
```

Programski kod 4.7. Alter Dialog za ispis upita za zahtjevom izlaska iz igre

Programski kod ispisuje pomoću naredbe *builder.setMessage („Želite li izaći iz igre“).* U slučaju da igrač pritisne Ok preko naredbe *finish();* izlazi se iz igre u suprotnom pritiskom na Cancel preko naredbe *dialog.cancel();* dialog se zatvara i igra se nastavlja.

4.3. Statistika

Za izradu i rad sa bazama podataka u Android aplikacijama koristi se SQLite baza podataka. Prvo je bilo potrebno kreirati klasu koja će upravljati bazama podataka (engl. data base handler). U toj klasi bilo je potrebno napraviti protezanje (engl. extends) pomoću konstruktora *SQLiteOpenHelper*. Ovaj konstruktor služi za upravljanje, kreiranje te za osvježavanje podataka u tablicama, a prikazan je na programskom kodu 4.8., dok je programski kod za kreiranje tablica prikazan na programskom kodu 4.9.

```
public class DatabaseHandler extends SQLiteOpenHelper {
```

Programski kod 4.8. Kreiranje konstruktora za rad sa bazama podataka

```
@Override
public void onCreate(SQLiteDatabase db) {
    String CREATE_CONTACTS_TABLE = "CREATE TABLE " + TABLE_IGRA + "("
            + colId_igre + " integer primary key autoincrement, "
            + colDatum + " date,"
            + colVrijeme + " TEXT" + ")";


    db.execSQL(CREATE_CONTACTS_TABLE);

    String CREATE_CONTACTS_TABLE2 = "CREATE TABLE " + TABLE_IGRACI + "("
            + colId + " integer primary key autoincrement , "
            + colId_igreFK + " integer not null , "
            + colIgrac + " TEXT, "
            + colPobjeda + " TEXT NOT NULL CHECK ("+colPobjeda+" IN ('D', 'N')), "
            + colPojeo + " integer  , "
            + colPojeden + " integer  , "


            + " FOREIGN KEY("+colId_igreFK+") REFERENCES " + TABLE_IGRA + "(id))";
    System.out.println(CREATE_CONTACTS_TABLE2);

    db.execSQL(CREATE_CONTACTS_TABLE2);

}
```

Programski kod 4.9. Kreiranje tablica

U ovom primjeru koda prikazan je način kreiranja dvije tablice koje prikazuju statistike pojedinih igrača te same igre. Naredba *CREATE_TABLE* omogućava kreiranje tablica. Tablica

TABLE_IGRA sadržava stupce *colld_igre,colDatum, te colVrijeme,* dok se istim postupkom kreira tablica *TABLE_IGRACI* te se popunjava na isti način stupcima. Svaki stupac u produžetku ima naziv parametara kojima se popunjavaju stupci. Kod ažuriranja tablice prikazan je na programskom kodu 4.10.

```
public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {
    // Drop older table if existed

    System.out.println("onUpgrade");
    // Create tables again

    String CREATE_CONTACTS_TABLE2 = "CREATE TABLE " + TABLE_IGRACI + "("
            + colId + " integer primary key autoincrement , "
            + colId_igreFK + " integer not null , "
            + colIgrac + " TEXT, "
            + colPobjeda + " TEXT NOT NULL CHECK ("+colPobjeda+" IN ('D', 'N')), "
            + colPojeo + " integer  , "
            + colPojeden + " integer  , "

            + " FOREIGN KEY("+colId_igreFK+") REFERENCES " + TABLE_IGRA + "(id))";
    System.out.println(CREATE_CONTACTS_TABLE2);

    db.execSQL(CREATE_CONTACTS_TABLE2);
}
```

Programski kod 4.10. Metoda ažuriranja tablice

Ova metoda je bitna zbog toga što se podatci osvježavaju nakon svake završene igre. Prvo je potrebno obrisati postojeću tablicu (ako ona postoji) te se nakon toga ažurira nova tablica.

```
String upitIgraci =
 " select   ime_igraca " +
",          sum(pojeo) pojeo " +
",          sum(pojeden) pojeden " +
",          sum(case when pobjeda='D' then 1 else 0 end) pobjeda " +
",          sum(substr(vrijeme_igranja,0,instr(vrijeme_igranja,':'))" +
        " *60+substr(vrijeme_igranja,instr(vrijeme_igranja,':')+1))" +
        " as ukupno_vrijeme_po_igracu " +
        "from " +
              " (select   ime_igraca " +
              " ,         pojeo " +
              " ,         pojeden " +
               ",         pobjeda " +
               ",         (select  vrijeme_igranja from igra iga where igi.id_igre=iga.id)" +
        " vrijeme_igranja " +
        "  from     igraci igi )t "+
          "group by ime_igraca;";
```

Programski kod 4.11. Upit na statistiku za prikaz podataka o igračima

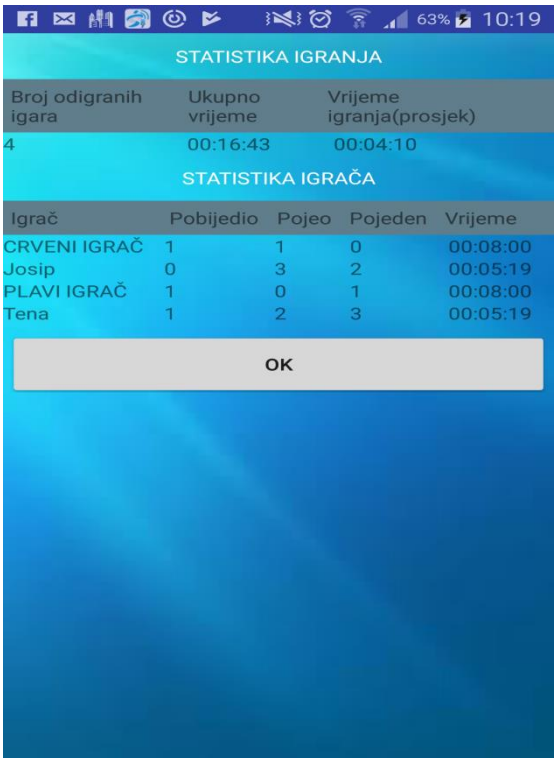
U ovome dijelu koda može se vidjeti kako se izvršava upit na bazu podataka pomoću kojega se vraćaju podatci u obliku tablice s podatcima svakog sudionika igre. Ovaj upit vraća vrijednosti ime igrača, koliko je puta igrač nekoga „pojeo“ ili bio „pojeden“, je li je igrač ikada pobijedio u igri te ukupno vrijeme koje je igrač proveo igrajući aplikaciju. Naredba *from igraci* određuje da se podatci vade iz tablice te pomoću *group by ime_igraca* izvršava sortiranje igrača po abecedi od A-Z. Kako bi se u statistici ispisalo i ukupno vrijeme igranja bilo je potrebno povezati tablice *IGRA* i *IGRAČI* zbog toga što se vrijeme igranja nalazi u tablici *IGRA*. Za kreiranje veze korišteni su *igi.id_igre=iga.id* te se ovim povezivanjem omogućio prikaz ukupnog vremena u tablici *IGRAČI*.

```
int ukupno = rs.getInt(rs.getColumnIndex("ukupno"));
int vrijeme = rs.getInt(rs.getColumnIndex("ukupno_vrijeme"));
try {
    prosjecno = (vrijeme) / ukupno;
} catch (java.lang.ArithmeticException e) {
    prosjecno = 0;
}
```

Programski kod 4.12. Računanje prosječnog vremena igranja igre

Potrebno je prvo prikupiti podatke o ukupnom broju igara te o ukupnom vremenu igranja igre. To izvršavamo pomoću naredbe *int ukupno* i *int vrijeme* koje preko indeksa *ukupno* i *ukupno_vrijeme* svake kolone računa prosječno vrijeme igranja igre. Prosječno vrijeme se računa matematičkim izrazom *prosjecno=(vrijeme)/ukupno*, gdje vrijeme označava sumu svega vremena kada je aplikacija korištena te sumu ukupnog broja igara.

Ovaj podatak se mijenja nakon svake odigrane igre tako da se broj odigranih igara povećava za 1, na vrijeme se dodaje onoliko vremena koliko je igrana ta jedna igra. Slika 4.7. prikazuje kako izgleda statistika igre nakon odigrane igre.



Slika 4.7. Izgled statistike nakon odigrane igre

4.4. Bluetooth

Zadatak je bio izraditi i omogućiti povezivanje 2-4 igrača pomoću Bluetooth-a. Zbog složenosti zadatka ovaj dio nije izrađen. Igra preko Buetooth-a odvijala bi se na isti način kao i na jednom uređaju s razlikom da bi svakom igraču bilo omogućeno da igru igra na svom uređaju. Za početak bi se trebao odrediti jedan glavni uređaj tj. domaćin (eng. host). Nakon toga domaćin bi putem Bluetooth-a kreirao igru i mrežu. Svaki igrač koji sudjeluje u igri vidio bi na svom uređaju statistiku svakog pojedinog igrača koji je sudjelovao u igri kao i onih koji su do tada sudjelovali u igranju aplikacije.

Kako bi se omogućio rad sa Bluetooth-om potrebno je kreirati novu projekciju (eng. layout) koja će sadržavati mogućnost odabira igrača kao i kod igre na jednom uređaju, ali bi dodatno sadržavala listu pregleda (eng. ListView) u kojem bi se ispisivali svi uređaji s kojima je moguće uspostaviti vezu. U projekciji bi se nalazila tipka za pretraživanje uređaja u blizini koji imaju uključen Bluetooth, tipka za uključivanje Bluetooth-a, ako nije uključen, tipka za otkrivanje uređaja, tipka za pokretanje igre i tipka za gašenje Bluetooth-a.

Nakon inicijalizacije svih tipki napravila bi se provjera da li je na uređaju moguće uspostaviti Bluetooth vezu što je prikazano u programskom kodu 4.13.

```
if (mBluetoothAdapter == null) {
    out.append("device not supported");
```

Programski kod 4.13. If petlja koja provjerava da li je uređaj u mogućnosti otvoriti Bluetooth

Ako uređaj prođe provjeru dopušta mu se uključivanje Bluetootha. Pritiskom na tipku uključi Bluetooth isti bi se automatski pokrenuo na mobilnom uređaju.

```
button1.setOnClickListener(new View.OnClickListener() {
    public void onClick(View v) {
        if (!mBluetoothAdapter.isEnabled()) {
            Intent enableBtIntent = new Intent(BluetoothAdapter.ACTION_REQUEST_ENABLE);
            startActivityForResult(enableBtIntent, REQUEST_ENABLE_BT);
```

Programski kod 4.14. Uključivanje Bluetooth-a na uređaju

Na programskom kodu 4.13. i 4.14. mogu se vidjeti kodovi koji vrijede za bilo koju aplikaciju i sastavni su dio svake Bluetooth konekcije te ih je moguće naći u svakoj aplikaciji koja omogućava ovakav tip komunikacije.

Nakon što je Bluetooth uključen uređaj postaje vidljiv pritiskom na tipku otkrivanje uređaja. Nakon toga bi se pojavio dijalog koji postavlja pitanje da li odobravamo da uređaj bude vidljiv drugim uređajima na 120 sekundi.

Prihvaćanjem postavki može se početi s pretragom uređaja u blizini pritiskom na tipku pretraga uređaja. Za otkrivanje uređaja koristila bi se jednostavna naredba *startDiscovery()*. Pozivom ove naredbe pokrenula bi se pretraga okoline i svakih 12 sekundi pristizali bi rezultati pretrage sa nazivima uređaja koji će se ispisivati u listi pregleda.

U rezultatima pretrage potrebno je prihvatiti željene igrače te s njima stvoriti vezu. Veza bi se ostvarivala pomoću naredbe *BluetoothSocket*. Budući da se spajaju na uređaj domaćina ostali igrači su klijenti. Nakon spajanja izradio bi se protokol na koji način se stvaraju informacije za prijenos podataka te kako ih aplikacija shvaća. Pomoću naredbi *InputStream* i *OutputStream* omogućilo bi se primanje i slanje podataka unutar mreže.

Prilikom izrade protokola važno je odrediti koje su informacije važne za mogućnost igranja aplikacije preko Bluetooth-a. Neke od važnijih informacija su pozicija figurica, broj dobiven bacanjem, koja figurica je izvršila pomak, da li je figurica stala na mjesto drugog igrača (pojela ga). Izrada protokola za prijenos ovih informacija radi se u obliku niz znakova koji bi glasio +DXXX/. Plus označava početak naredbe, D vrsta naredbe koja se prenosi, XXX broj

naredbe,/označava završetak naredbe. Iz toga proizlazi da bi naredba za dobiveni broj bacanja bila npr. +D102/. Na ovaj način bi bilo potrebno izraditi naredbe za sve informacije potrebne za rad aplikacije.

5. ZAKLJUČAK

U sklopu završnog rada mobilna aplikacija „Čovječe ne ljuti se“ uspješno je odrađen dio problema vezanih uz zadatak završnog rada.

Uspješno je izrađena aplikacija koja omogućava igranje igre na jednom uređaju s mogućnosti personalizacije, odabira boje, te broja igrača koji sudjeluju u igri.

Uspješno je izrađena statistika koja prikazuje sve podatke koji su navedeni u zadatku. Za izradu statistike korištena se SQLite baza podataka.

Zbog složenosti postupka kreiranja Bluetooth-a ovaj dio zadatka nije uspješno odrađen. Detaljno je opisan postupak izrade Bluetooth-a te tijek rada u prethodnom poglavlju.

Završni rad obuhvaća detaljan opis važnih teorijskih podloga o povijesti Android sustava, značajke Android platforme, arhitekturu Android platforme kao i razvojno okruženje. Dane su osnovne teorijske podloge o programu u kojem je aplikacija izrađena te osnove o programskom jeziku Java.

Tijek razvoja aplikacije opisan je dosta detaljno uz prilaganje kodova i slika te njihovog objašnjenja u nastavku.

Prilikom izrade aplikacije bilo je važno najprije se upoznati s Android okruženjem, Java programiranjem, objektno orijentiranim programiranjem te snalaženje u Android Studio-u korištenom za razvoj programa. Za stjecanje znanja o Androidu, Javi te Android Studio-u korištena je literatura pronađena na Internetu jer je u današnje vrijeme programiranje za Android vrlo popularno i ljudi širom svijeta dijele svoja znanja i iskustva putem Interneta.

Problemi koji su se pojavljivali tijekom izrade aplikacije otklonjeni su istraživanjem i učenjem o načinu njihova rješavanja. Proces izrade završnog rada i aplikacije trajao je dosta dugo, ali je stjecanje znanja i vještina potrebnih za savladavanje ovoga problema bilo jako zanimljivo. Znanja stečena izradom ovoga završnoga rada dobar su temelj za daljnje usavršavanje i uporabu u nastavku obrazovanja i kasnije tijekom zaposlenja.

Najveća mana aplikacije je nemogućnost korištenja Bluetooth-a za povezivanje više igrača, ali osim toga sve ostalo funkcionira u skladu s zahtjevima.

Aplikaciju je moguće pokretati na raznim dimenzijama uređaja zbog toga što su projekcije (engl. layoute) za normalne veličine zaslona izradio i projekcije za male i velike rezolucije zaslona. U budućoj nadogradnji moguće je izraditi funkcionalni Bluetooth koji će podići igru na malo bolju razinu.

LITERATURA

[1] Hello, Android Introducing Google's Mobile Development Platform, Ed Bumette, 28.6.2017.

[2] Značajke Android platforme, https://www.engineersgarage.com/articles/what-is-android-introduction , 22.5.2017.

[3] Specifikacije Androida, https://www.engineersgarage.com/articles/what-is-android-introduction , 22.5.2017.

[4] Dalvik virtualni stroj, https://www.javatpoint.com/dalvik-virtual-machine , 7.5.2017.

[5] Komponente Android Platforme, https://developer.android.com/guide/platform/index.html , 7.5.2017.

[6] Android Studio, https://en.wikipedia.org/wiki/Android_software_development , 7.5.2017.

[7] Jack, https://source.android.com/source/jack , 28.6.2017.

[8] Razvojno okruženje, https://en.wikipedia.org/wiki/Android_software_development , 20.6.2017.

[9] Java, http://www.algebra.hr/edukacija/java/ , 7.5.2017.

[10] Broj korisnika Jave, https://plumbr.eu/blog/java/how-many-java-developers-in-the-world, 7.5.2017.

[11] Prvo izdanje igre „Čovječe, ne ljuti se“, https://www.google.hr/search?q=first+edition+of+ludo+game&source=lnms&tbm=isch&sa=X&ved=0ahUKEwi6hNXC7t_TAhWpAsAKHSmpBGsQ_AUIBigB&biw=1600&bih=756#imgrc=PYRAMn8oyTLKrM:, 8.5.2017.

SAŽETAK

Završni rad na temu mobilna aplikacija „Čovječe ne ljuti se“ sadrži sljedeće zadatke: upoznavanje s Android operacijskim sustavima, kratak opis svih potrebnih znanja o Androidu kao uvod u konačno rješenje zadatka. Teorijski dio se odnosi općenito na Android sustave, njihov razvoj kroz povijest, arhitekturu i ostale značajke Android sustava. Obuhvaća teorijska znanja iz Java programskog jezika, te kratki opis razvojnog okruženja Android Studija u kojem je aplikacija izrađena. U praktičnom dijelu izrađena je aplikacija koja omogućava odabir boje i broja igrača te personalizaciju svakog pojedinog igrača. Nadalje, izrađena je statistika koja prati rezultate svakog pojedinog igrača. Igranje igre preko Bluetooth-a, zbog kompleksnosti ovoga područja i teškog usklađivanja svih protokola i parametara kad je u igru uključen veći broj igrača, nije uspješno odrađeno. Svi dijelovi ovog završnog rada su detaljno opisani te svaki čitatelj jasno može shvatiti kako je ovaj rad napravljen. U opisima su dani osnovni kodovi i slike koje su potrebne za izradu aplikacije. Aplikacija radi bez problema i stabilna je. Daljnja poboljšanja igre mogu biti u vidu osposobljavanja Bluetooth-a te popravljanje sitnih grafičkih pogrešaka.

Ključne riječi:

Čovječe ne ljuti se, Android, Android Studio, Java, Bluetooth

ABSTRACT

In this thesis, it is given introduction to Android operating system, short description of all indispensable knowledge about Android as a preface to final solution of the task. Theoretical part is consisted of basic information about Android operating system, their historical development, architecture and other features. It contains theoretical knowledge about Java programming language and short description of Android Studio development environment in which this application was developed. As a practical part, mobile application was developed, which allows 1-4 players to play popular game “Ludo”. Furthermore, statistics which follows results of each player has been made. However, playing this Game over Bluetooth, when there are more players involved, was not successfully done because of complexity of this department and difficult with coordinating all protocols and parameters. All parts of this work are described in details and every reader can understand how this project was made. Basic parts of programming code needed for developing this app are provided in description of this solution. Application works without any problems.. Further improvements can be considered with Bluetooth implementation and fixing minor graphic mistakes.

Keywords:

Ludo, Android, Android Studio, Java, Bluetooth

ŽIVOTOPIS

Josip Matijević rođen 16.07.1955. godine u Požegi. Osnovnu školu Vladimir Nazor pohađao je u Trenkovu. Završio je srednju Tehničku školu u Požegi, smjer mehatronika. Sudjelovao je na tri Sajma inovacija, gospodarstva i tehničkog stvaralaštva mladih INVENTUM. Rezultati s natjecanja su dva treća mjesta te jedno drugo mjesto. U trećem razredu srednje škole pohađao je stručnu praksu u europskom projektu odrađivanja prakse u stranoj zemlji. U sklopu projekta pohađao je dva tjedna praksu u tvrtci Yulon u Ljubljani, Slovenija. Nakon završetka srednje škole upisuje Elektrotehnički fakultet u Osijeku.

Potpis

_____