SVEUČILIŠTE U RIJECI
POMORSKI FAKULTET U RIJECI

Marin Marinović

RAZVOJ ANDROID APLIKACIJE "KVIZ"

DIPLOMSKI RAD

Rijeka, 2014.

SVEUČILIŠTE U RIJECI

POMORSKI FAKULTET U RIJECI

RAZVOJ ANDROID APLIKACIJE "KVIZ"

DIPLOMSKI RAD

Kolegij: Operacijski sustavi

Mentor: dr.sc. Božidar Kovačić

Student: Marin Marinović

Studijski smjer: Elektroničke i informatičke tehnologije u pomorstvu

JMBAG: 0112039624

Rijeka, listopad 2014.

Student: Marin Marinović
Studijski program: Elektroničke i informatičke tehnologije u pomorstvu
JMBAG: 0112039624

IZJAVA

Kojom izjavljujem da sam diplomski rad s naslovom RAZVOJ ANDROID APLIKACIJE "KVIZ" izradio samostalno pod mentorstvom prof.dr.sc Božidar Kovačić.

U radu sam primijenio metodologiju znanstvenoistraživačkog rada i koristio literaturu koja je navedena na kraju diplomskog rada. Tuđe spoznaje, stavove, zaključke, teorije i zakonitosti koje sam izravno ili parafrazirajući naveo/la u diplomskom radu na uobičajen, standardan način citirao sam i povezao s fusnotama i korištenim bibliografskim jedinicama. Rad je pisan u duhu hrvatskoga jezika.

Suglasan sam s objavom diplomskog rada na službenim stranicama.

Student: Marin Marinović

SADRŽAJ

1. UVOD

Prije nego što se pristupi analiziranju i razmatranju problema ove tematike, neophodno je sadržajno determinirati pojam "Android" i "aplikacija".

Najednostavnija definicija Androida je, da je to operativni sustav kojeg koriste mobilni uređaji.Nastao je prvenstveno iz raloga da se funkcionalnost osobnog računala što je moguće brže približi mobilnim uređajima tj.da sve što radimo na računalu, prilagodimo i na mobilnim aplikacijama. Android je otvorenog koda, i to je jedna od prednosti koje su ga izdvojile od konkurencije, ali nije to naravno jedina prednost ovog operativnog sustava. Time se omogućava razvoj razvoj bogatih i naprednih aplikacija u relativno kratkom vremenu uz nerijetko besplatnu podršku zajednice.

U informatičkoj znanosti, aplikacija je skup računalnih programa dizajniranih za pomoć korisnicima u izvršavanju određenih zadataka.Sve aplikacije u Android-u su ravnopravne što znači da svaka aplikacija ima pristup istim resursima, pa je razvoj potpuno slobodan, tj.svaki korisnik može uređaj prilagoditi vlastitim potrebama.

Praktični dio ovog rada je aplikacija za Android, no prije svega treba se dobro upoznati s njegovim načinom funkcioniranja i strukturom aplikacije kako bi se mogle prikazati njegove mogućnosti.

Da bi tema bila što jednostavnije obrađena, a da potom budu obuhvaćeni svi važniji elementi, tematika diplomskog rada podjeljena je u pet međusobno povezanih cijelina.

U prvod dijelu "Uvod" dano je tek nekoliko uvodnih crta o pristupu i obradi završnog rada, determiniranju glavnih pojmova "Android-a" i "aplikacije" kako bi se što bolje pristupilo realizaciji cijelokupnog rada.

Drugi dio "Android operacijski sustav" sastoji se od četiri naslovne cijeline (podnaslova). Povijest Android operacijskog sustava prikazuje razvitak Android operacijskog sustava kroz povijest.Verzije Android operacisjkog sustava opisuju pet trenutno najkorištenijih verzija Android sustava. Arhitektura Android-a opisuje komponente od kojih se sastoji svaki android sustav. U "Android uređaji" navode se uređaji koji koriste Android operativni sustav.

U trećem djelu "Android razvojno okruženje" opisan je razvoj Android aplikacija korištenjem razvojnog okruženje Eclipse. Samo razvojno okruženje Eclipse nije dovoljno za razvoj Android aplikacija već je isti potrebno proširiti sa službenim razvojnim paketom Android SDK koji će također biti opisan. U istom poglavlju je opisna osnovna struktura Android aplikacija.

Potom u četvrtom djelu opisana je sama aplikacija te bitniji dijelovi koda. Objašnjen je i način kreiranja grafičkog sučelja aplikacije i njegov način funkcioniranja. Dan je i prikaz baze podataka sa potrebnim pojašnjenjima pojmova koje baza podataka sadrži. Također, navedene su i mogućnosti poboljšanja aplikacije.

U posljednjem djelu, "Zaključak" dana je sistematizacija cijelokupnog rada, kao i spoznaje do kojih je došlo.

2. OPERACIJSKI SUSTAV ANDROID

Android je najkorišteniji mobilni operacijski sustav koji je baziran na modificiranoj verziji Linux-a. Napravljen je od strane istoimene kompanije koju je kasnije kupio Google. Android OS se oslanja na programski jezik Java[1], te je programiranje vrlo slično Java programskom jeziku. Modularan je i prilagodljiv pa se prenosi (porta) na razne uređaje kao što su čitaći elektronskih knjiga, mobilni telefoni te prijenosnici. Android ima veliku aktivnu zajednicu programera koji iz dana u dan proširuju funkcionalnosti samih uređaja. Trenutno postoji preko milijun raznih aplikacija koje su korisnicima dostupni za preuzimanje.

Karakteristike Google Androida prema OHA[2]su: [1]

- otvorenost - programerima aplikacija omogućena je sloboda u smislu razvoja aplikacija, a proizvođačima uređaja korištenje i prilagođavanje platforme bez plaćanja autorskih prava,
- ravnopravnost aplikacija - ne postoji razlika između osnovnih jezgrenih aplikacija i aplikacija koje korisnik instalira. Također, svim aplikacijama je omogućen ravnopravni pristup resursima što daje izbor korisniku da prilagodi uređaj svojim individualnim potrebama,
- automatsko upravljanje životnim ciklusom aplikacije - omogućuje nadzor pokretanja i izvršavanja aplikacija na način da korisnik ne brine o gašenju određenih aplikacija prije pokretanja drugih,
- mogućnost razvoja novih inovativnih aplikacija,
- jednostavan i brz razvoj aplikacija - omogućen je bazom programskih biblioteka (eng. libraries) i raznih alata za samu izradu aplikacije,
- kvalitetni grafički prikaz i zvuk - podržana je 2D vektorska i 3D OpenGraphicLibrarygrafika, te je ugrađena podrška svih često korištenih audio i video formata,
- kompatibilnost sa većinom sadašnjih i budućih hardvera.

[1]Objektno orjentirani programski jezik
[2]Open Handset Alliance

2.1 Povijest Android operacijskog sustava

U listopadu 2003. godine kompaniju Android Inc. osnovali su Andy Rubin, Rich Miner, Nick Sears i Chris White kako bi razvijali programe za pametne mobilne uređaje koji bi uzimali u obzir korisničke postavke te njegovu lokaciju. Nakon dvije godine tihog rada kompaniju Android Inc. kupuje Google. Kupnja kompanije je značila ulazak Googla na tržište mobilnih uređaja. Dvije godine nakon kupovine Androida Inc., Google je zatražio stvaranje OHA udruženja (studeni 2007. godine), kojoj je temeljna zadaća pružanje podrške i daljnji razvoj sustava tj. stvaranje javnog standarda za mobilne uređaje.

Istog dana udruženje OHA predstavlja mobilnu platformu Android koja je otvorenog koda baziranu na Linux kernelu. Ovo je ujedno i prvo javno predstavljanje Androida kao operativnog sustava, a prvi komercijalni uređaj u koji je bio ugrađen Android OS bio je T-Mobile G1, tajvanskog proizvođača pametnih telefona HTC.

Google je želio da Android bude otvoren i besplatan, pa je tako veći dio Android koda objavljen pod open-source Apache licencom, što znači da svako tko želi da koristi Android može to učiniti downloadom kompletnog Android source koda. To je omogućilo mnogim proizvođačima da ugrade svoje dodatke u Android , te samim time se njihovi proizvodi razlikuju od proizvoda drugih proizvođača. Ovaj novi razvojni model je učinio Android veoma atraktivnim i postao je popularan među mnogim proizvođačima. Ovo se posebno odnosi na kompanije zahvaćene fenomenom Apple-ovog iPhone-a, veoma uspješnog proizvoda koji je unio revoluciju u industriju pametnih telefona. Primjeri tih kompanija su Motorola i Sony Ericsson, koji su godinama razvijali svojstvene mobilne operacijske sustava. Pojavom iPhona mnoge od ovih kompanija morale su pronaći način da svoje proizvode učine konkuretnim. Rješenje je pronađeno u Androidu, u kojem su mnoge kompanije koristile njegov operacijski sustav i nastavile razvijati svoj hardver. [2]

2.2 Verzije Android operacijskog sustava

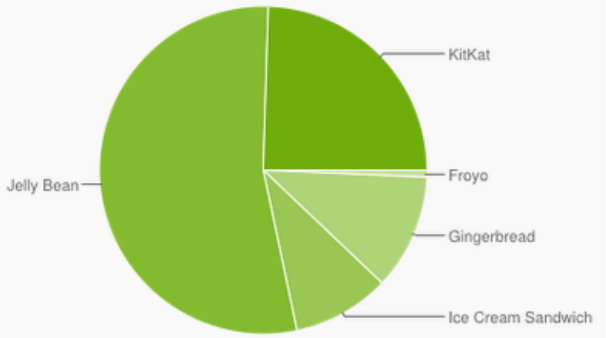
U travnju, 2010 godine izdana je verzija Androida Froyo. Neka od poboljšanja u odnosu na prijašnje verzije: tipkovnica podržava više jezika i brzo prebacivanje između njih, unaprijeđen je Bluetooth, glasovno biranje, dijeljenje kontakata itd.

Gingerbread verzija objavljena je u prosincu 2010.godine i donijela je mnoštvo novosti poput pojednostavljenja i ubrzanja korisničkog sučelja, intuitivnijeg i bržeg unosa teksta virtualnom tipkovnicom, unaprjeđeni copy-paste, status aplikacija i mogućnost ručnog zatvaranja aplikacija, internetsko telefoniranje, podršku prednju i stražnju kameru,podršku za nove senzore (barometar, žiroskop, rotacijski senzor itd).

U listopadu, 2011.godine na tržištu se pojavila verzija nazvana Ice Cream Sandwich. Donijela je poboljšano sučelje, bolji izbor posljednjih korištenih aplikacija, akcije zaključanog zaslona i face-unlock uz mnoga druga poboljšanja. Trenutno je najkorištenija verzija sa udjelom od gotovo 55%.

U listopadu, 2013.godine objavljena je verzija imena KitKat.Glavna karakteristika ovog mobilnog OS-a je prilagodba hardverski slabijim uređajima, s manje radne memorije i s manjom rezolucijom zaslona.Korisničko sučelje je ispolirano s novim ikonicama i prečicama. Jedna od novosti je prikaz preko cijelog zaslona. Ovo programerima omogućuje razvoj aplikacija za prikaz preko čitavog zaslona, tj.bez virtualnih tipki i statusnih traka. [3]

Version	Codename	API	Distribution
2.2	Froyo	8	0.7%
2.3.3 - 2.3.7	Gingerbread	10	11.4%
4.0.3 - 4.0.4	Ice Cream Sandwich	15	9.6%
4.1.x	Jelly Bean	16	25.1%
4.2.x		17	20.7%
4.3		18	8.0%
4.4	KitKat	19	24.5%



Slika 1. Verzije Android OS-a

Izvor :Platform versions, *www.developer.android.com* (9.10.2014)

2.3 Android uređaji

Android uređaji se proizvode u svim oblicima i veličinama. Android OS koriste sljedeći uređaji:

- Pametni telefoni
- Tableti
- Čitači e-knjiga
- Laptop
- Internet TV

2.3.1 Pametni telefoni

Pametni telefon (eng. smartphone) je mobilni telefon s većim mogućnostima za pohranu podataka i povezanost od običnog mobilnog telefona. Izraz „pametni“ se odnosi na mogućnost uporabe kao džepno računalo. Tipični pametni telefon ima zaslon na dodir visoke razlučivosti. Gotovo svi pametni telefoni korisnicima pružaju mogućnost instaliranja dodatnih aplikacija.

Temeljne funkcije android mobilnih uređaja: višezadaćne (eng. multitasking) funkcije, pristup internetu preko WiFi ili 3G mreže, multimedijske funkcije (kamera, videozapisi), kalendar, upravljanje kontaktima, GPS navigacija i mogućnost čitanja poslovnih dokumenata u različitim formatima kao što su PDF ili Microsoft Office. Na slici 2. prikazan je pametni telefon. [4]

Slika 2. Pametni telefon

Izvor : Samsung Galaxy S5,*http://www.smartphonehrvatska.com* (10.10.2014)

2.3.2 Tableti

Druga popularna kategorija uređaja su tableti (eng. tablet). Tablet računalo, ili jednostavnije tablet, je prijenosno računalo sa zaslonom, sklopovima i baterijom u jednoj cjelini. Tableti su opremljeni senzorima, uključujući kamere, mikrofon, brzinomjer i zaslon osjetljiv na dodir. Dodiri prstom ili olovkom gestama zamjenjuju računalni miš i tipkovnicu. Tablet može imati i dodatnu fizičku tipkovnicu, tada ona obično kontrolira osnovne značajke kao što su glasnoća zvučnika i snaga uređaja za mrežnu komunikaciju, a služi i za punjenje baterije. Tableti su u pravilu veći od pametnih telefona i obično imaju veličine zaslona od 7 inča (18 cm) ili veće.. Na slici 3.prikazan je primjer tableta. [5]

Slika 3. Tablet

Izvor : Sony Experia Z2, *http://www.smartphonehrvatska.com* (10.10.2014)

2.3.3 Čitači e-knjiga

Osim pametnih telefona i tableta, Adroid se sve više susreće i na uređajima posebne namjene, kao što su čitači e-knjiga (eng. e-book reader).Čitači e-knjiga je elektronički (digitalni) ekvivalent tiskanoj knjizi. E- knjiga može sadržavati i online časopise i digitalne knjige napravljene da budu slušane kao audio knjige. E- knjige su tehnologija koja se rapidno razvija i mijenja. Neke e-knjige se proizvode simultano s proizvodnjom tiskanih knjiga, iako se u mnogo slučajeva kasnije stavljaju u prodaju. Često se e- knjige proizvode nakon što su knjige već tiskane, obično skeniranjem, a ponekad se tekst unosi tipkovnicom.Slika 4. prikazuje čitač e-knjiga koji radi na Android operativnom sustavu. [6]

Slika 4. Čitač e-knjiga

Izvor : Prestigio PER5162B, *www.links.hr* (10.9.2014)

2.3.4 Prijenosna računala

Sve više u današnje vrijeme korisnici koriste prijenosna računala (eng. laptop), koji također podržava Android operativni sustav. Prijenosno računalo je osobno računalo relativno malih dimenzija koje čovjek može lako prenositi. Prijenosno računalo u istom kućištu objedinjuje tipične komponente osobnog računala, uključujući zaslon, tipkovnicu, pokazivački uređaj i zvučnike, a za napajanje koristi punjivu bateriju koja omogućava da računalo radi više sati bez napajanja iz električne mreže. Hardverska snaga prijenosnih računala manja je nego kod stolnih računala iste generacije i cijene jer prijenosna računala imaju veća ograničenja na fizičke dimenzije te koriste neke skuplje komponente. Na slici 5. je prikazano prijenosno računalo koje koristi Android operativni sustav.

Slika 5. Prijenosno računalo

Izvor : Lenovo A10, *www.shop.lenovo.com* (10.9.2014)

2.3.5 Android TV

Osim ovih popularnih mobilnih uređaja, Android pronalazi namjenu i području televizija. Švedska kompanija, People of Lava je razvila televiziju baziranu na Androidu, pod nazivom Scandinavia Android TV. Na slici 7. prikacana je televizija People of Lava. Google je također ušao u proizvodnju smart TV platforme, bazirane na Androidu i razvija je u suradnji sa kompanijama Intel, Sony i Logitech. Na slici 6. prikazana je Android televizija.

Slika 6. Android TV

Izvor : Scandinavia TV, *www.peopleoflava.com* (10.9.2014)

2.4 Arhitektura Android operacijskog sustava

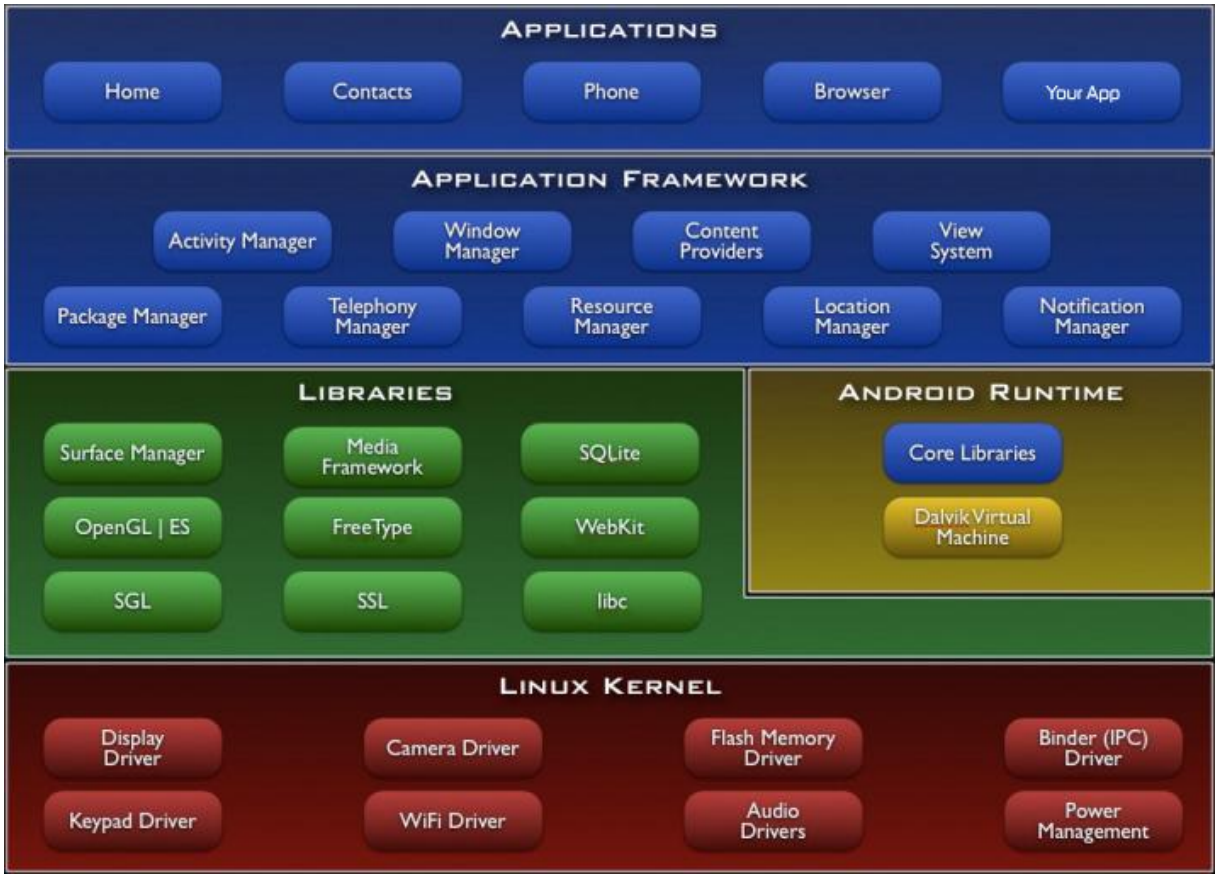
Android operacijski sustav sastoji je od nekoliko djelova i slojeva. Svaki pojedini sloj ima svoje karakteristike i ulogu, ali slojevi nisu jasno odvojeni već se najčešće preklapaju jedni sa drugima. Android operacijski sustav je baziran na Linux Kernel 2.6 jezgri napisanom u C ili C++ programskom jeziku. Linux je operacijski sustav otvorenog koda, pa aplikacije preko posrednika (eng. middleware) imaju mogućnost komunikacije te pokretanja drugih aplikacija kao što je slanje SMS poruka i email-ova, slikanja, primanja poziva i slično. Razlog njegove upotrebe je njegova laka prenosivost, sigurnost i značajke. 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.Iako su C i C++ programski jezici korišteni za radno okruženje (eng. framework), većina aplikacija je pisana u programskom jeziku Java koristeći Android SDK[3]. Aplikacije se mogu pisati i u C odnosno

[3]Software Development Kit

C++ programskom jeziku, ali tada se koristi Android NDK[4]. Ovakvim postupkom omogućuje se bolje raspolaganje resursima samog uređaja i korištenje knjižnica iz jezgre i radnog okruženja te se brzina aplikacije povećava i do 10 puta. Mana ovakvog pristupa je mnogo složenije pisanje koda. Arhitektura Android operacijskog sustava prikazana je na slici 7. [7]

2.4.1 Linux kernel

Linux 2.6 predstavlja jezgru na kojoj je baziran Android operacijski sustav. Sadrži upravljačke programe (eng. driver) od kojih su najvažniji programi za međuprocesnu komunikaciju (IPC3) koji služi za izmjenu podataka između različitih procesa, upravljački programi za upravljanje napajanjem (eng. power managment). Tu se također nalaze i upravljački programi za neke od osnovnih hardverskih komponenti samog sustava poput upravljačkog programa za ekran, kameru, tipkovnicu, Wi-Fi, zvuk i pohranu podataka.



Slika 7. Android arhitektura

Izvor : Android Architecture, *http://www.android-app-market.com* (10.9.2014)

[4]Native Code Development Kit

2.4.2 Biblioteke

Biblioteke sadrže sav kod koji osigurava glavne funkcionalnosti Android operativnog sustava.Na primjer biblioteka WebKit omogućava web pretraživanje.Pisane su u programskim jezicima C i C++. Neke od značajnijih su :

- BSD (eng. Berkeley SoftwareDistribution)- inačica implementacije standardne sistemske C biblioteke (libc), prilagođene za ugradbene uređaje,
- MediaLibraries- zasnovan na biblioteci OpenCORE koorporacije PecketVideo; biblioteke koje služe za reprodukciju i snimanje audio i video formata, kao i mirnih slika,
- SurfaceManager- komponenta biblioteke koji služi za iscrtavanje ploha po ekranu i upravlja radom prozora,
- LibWebCore- sadrži komponentu WebkitWebkit je pogonski model za internetski pretraživač otvorenog koda, koji također pogoni pretraživač Safari tvrtke Apple,
- SGL- mehanizam 2D grafike,
- 3D biblioteke- implementacija zasnovana na sučelju OpenGL ES (.OpenGraphicsLibrary for Embedded Systems) 1.0 API (Application Programming Interface). Biblioteke koriste sklopovsku 3D akceleraciju ili visoko optimozirani 3D programski raster,
- FreeType- biblioteka za iscrtavanje fontova,
- SQLite- služi za pohranu podataka u baze,
- SSL- inačica sigurnosnog mrežnog protokola otvorenog koda prilagođena ugradbenim sustavima.

2.4.3 Android radno okruženje

Android radno okruženje služi za pokretanje aplikacija. Nalazi se na istom sloju kao i biblioteke. Sastoji se od dvije komponente. Prva komponenta je tzv "Core libraries" odnosno knjižnica koja sadrži većinu jezgrenih knjižnica koje omogućuju pisanje Android aplikacija korištenjem Jave. Jezgrena knjižnica predstavlja sve esencijalne klase kao što su klase za manipulaciju kolekcijama, klase za komunikaciju sa okolinom i slično. Druga komponenta je Dalvik Virtual Machine (DVK), koja je dizajnirana specijalno za Android i optimizirana je za

uređaje s baterijama, limitiranom memorijom i procesorskom snagom. DVM je registarski bazirani virtualni stroj, dok je klasičn Java virtualni stroj (JVM) baziran na memoriji. Međukod (eng. bitecod) DVM transformira pomoću alata dk (koji je sastavni dio Android SDK-a) iz javinih klasnih datoteka (eng. Java class file) prevedenim javinim prevodiocem u novu klasu *.dex (eng. Dalvik Executable) formata.Međukod koji izvršava DVM nije javin međukod, nego upravo spomenuti dex oblik.Rezultat sevega je mogućnost višestrukog instanciranja samog virtualnog stroja, što znači da se svaka Android aplikacija pokreće kao zaseban proces.

2.4.4 Radni okvir aplikacija

Sloj aplikacijskih okvira (eng. aplication framework) napisan je u programskom jeziku Java i sadrži proširiv skup programskih komponeti kojeg koriste sve aplikacije uređaja. Ovaj sloj ujedinjuje različite karakteristike Android OS-a. Framework podržava mnogobrojne opensource jezike kao što su opensll, sqlite i libc.Također podržava i jezik Android jezgre.Sa gledišta sigurnosti, framework se bazira na UNIX file system ovlastima koja osiguravaju da aplikacije posjeduju samo one mogućnosti koje im je vlasnik telefona dao pri instalaciji aplikacije.

Neki od važnijih elementa su:

- upravljanje aktivnostima (ActivityManager)- upravljanje životnim ciklusom aplikacije,
- upravljanje programskim paketima (Packagemanager)- sadrži informaciju o aplikacijama instaliranim na sistemu,
- upravljanje prozorima (WindowManager)- upravljanje aplikacijskim prozorima,
- upravljanje pozivima (TelephoneManager)- sadrži API-je koji se koriste pri izradi aplikacija za upravljanje pozivima,
- pružaoci sadržaja (ContentProviders)- omogućuju zajedničko korištenje podataka od strane više aplikacija,
- upravljanje resursima (ResourceManager)- služi za skladištenje dijelova aplikacija koje nisu kod (npr. slike),

- sistem grafičkog prikaza (View sistem)- sadrži bazu gotovih grafičkih prikaza i alata (widget),
- upravljanje lokacijski zasnovanim servisima (LocationManager) i
- upravljanje notifikacijama (NotificationManager)- upravljanje notifikacijama i događajima.

2.4.5 Aplikacije

Aplikacijski sloj je posljednji sloj u arhitekturi sistema Android i čine ga korisničke aplikacije uređaja.Predstavlja sloj vidljiv krajnjem korisniku. Ovaj sloj obuhvaća sve aplikacije koje se isporučuju sa Android uređaja (Phone, Contacts, Browser..), kao i aplikacije koje se instaliraju sa GooglePlayStore. Svaka aplikacija koja se kreira nalazi se na ovom sloju.

3. ANDROID RAZVOJNI ALATI

Razvojni alati (eng. development kit) su skup programskih alata koji omogućavaju kreiranja aplikacija za određene skupine programa, programskog okruženja, hardvera, OS-s i sličnih platformi. Najčešće se sastoje od skupine gotovih programskih rješenja koji imaju mogućnost komunikacije sa specifičnim programom ili platformom. Uz navedene skupine sastoje se i od alata za dizajniranje sučelja, programa za lakše otkrivanje grešaka u samom programu te drugih sličnih alata koji pomažu programerima u pisanju aplikacija. [8]

3.1 Android SDK

Android SDK je programski paket koji sadrži alate za programiranje aplikacija u Java programskom jeziku. Android aplikacije mogu se razvijati na Windows, Mac ili Linux operativnim sustavima. Sve podatke o aplikaciji, uključujući i izvorni kod, metapodatke i manifest podatke o aplikaciji, SDK kod finaliziranja aplikacije pohranjuje u obliku arhive sa ekstenzijom .apk. SDK je besplatni alat i može se preuzeti za službene Android web stranice za programere *https://developer.android.com/sdk/index.html?hl=i*.

Neki od značajnijih paketa koje SDK sadrži jesu :

- razvojni alati - alati koji služe za prevođenje i testiranje. Najznačajniji alat iz ove skupine je ADTP[5] koji omogućuje pristup djelu u kojem se spremaju log zapisi,
- emulator - program koji služi za pokretanje aplikacija na računalu,
- DDMS[6] - služi za kontrolu, pronalaženje i uklanjanje grešaka u aplikaciji,
- AAPT[7] - služi za pohranjivanje aplikacije u oblik sa ekstenzijom .apk,
- Android aplikacijska programska sučelja,
- primjere koda za demonstrciju korištenja određenih sučelja,
- aplikacija za instaliranje i pokretanje emulatora.

[5]Android Development Tools Plugin
[6]Dalvik Debug Monitoring Service
[7]Android Asset Packaging Tool

3.1.1 Eclipse

Eclipse je platforma koja je dizajnirana za izgradnju aplikacijskih razvojnih alata. Razvijen je od strane Object Technilogy Internatinal kompanije. Eclipse platforma definira otvorenu arhitekturu te je neovisno o operacijskom sustavu. Baziran je na programskom jeziku Java. Sam dizajn platforme ne pruža veliku krajnju funkcionalnost, ali vrijednost platforme je u poticanju brzog razvoja integriranih mogućnosti koje se baziraju na modelu dodataka (eng. plug-in model).

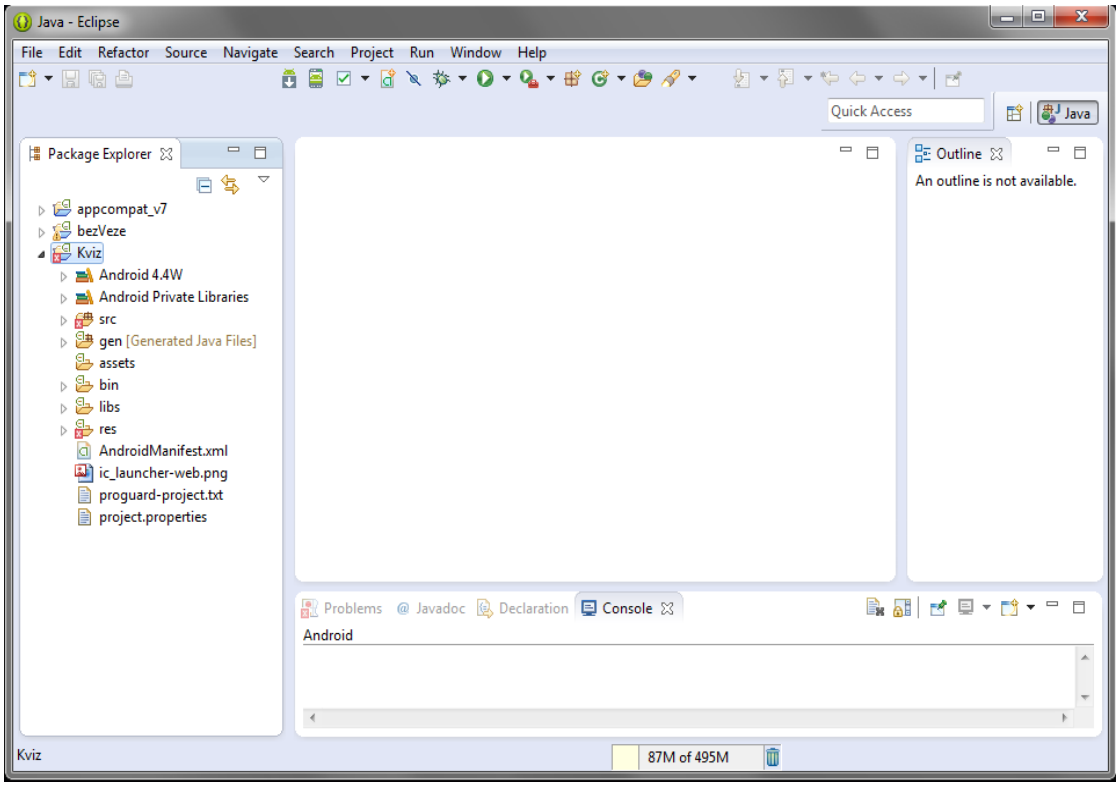
Jezgra Eclipse platforme je arhitektura za dinamičko otkrivanje, učitavanje i pokretanje dodataka.Sama platforma odrađuje logistiku traženja i pokretanje ispravnog koda, dok korisničko sučelje pruža standardni korisnički navigacijski model. Svaki dodatak može se orijentirati na mali broj zadataka koje mora izvršavati, zadaci mogu biti bilo što, definiranje, ispitivanje, animiranje, objavljivanje, prevođenje, izrada dijagrama ili nešto drugo, jer Eclipse ne postavlja nikakva ograničenja.

Eclipse platforma koristi model zajedničkog radnog mjesta (eng. worbench) u koje integriraju svi alati koristeći dobro definirane točke proširenja (eng. extensionpoints). Na taj način platforma pruža korisniku jedinstven izgled svih alata i pruža integrirano upravljanje svih resursa (projekata, direktorija, datoteka) kojima manipuliraju ti alati.

Sama platforma izgrađena je od više slojeva dodataka od kojih svaki pruža svoja proširenja za njihovo prilagođavanje, a pružaju točke proširenja za svoju prilagodbu. Platforma je strukturirana u podsustave koji su implementirani kao jedan ili više dodataka, a svi podsustavi izgrađeni su na malom pokretačkom stroju (eng. runtimeengine).

Zahvaljujući arhitekturi Eclipse platforme programer ne mora brinuti o operacijskom sustavu na kojem se izvodi i na taj način omogućava programeru da se usredotoči na problem koji rješava, a ne na implementacije na različitim sustavima. Radna površina Eclipsea vrlo je slična ostalim programima za programiranje. Sastoji se od niza prozora koji se mogu razmještati po sučelju ovisno o potrebama programera. Slika 8. prikazuje najkorištenije dijelove Eclipsea. U sredini se nalazi prostor za uređivanje koda (eng. editor) koji zauzima najveći dio prikaza. S lijeve je strane prozor za upravljanje objektima (eng. package explorer) u kojem se nalazi cijela hijerarhija objekata u svim projektima trenutno otvorenima u programu. Ispod prostora za uređivanje koda nalazi se prozor za praćenje i zapisivanje procesa koji se odvijaju u Eclipseu, kao i prozor za čitanje mogućih grešaka (eng. log), a

iznad alatna traka. Svaka od kartica na vrhu editora sadrži kod jedne klase, activityja ili nekog drugog dijela razvijane aplikacije.



Slika 8. Eclipse platforma

Izvor : Autor

3.1.2 Android emulator

Emulator je virtualni uređaj koji ima ulogu testiranja aplikacije bez stvarnog uređaja.Temelji se na strojnom emulatoru otvorenog koda QEMU (eng. Open source processor emulator). Prednosti emulatora su opcije testiranja aplikacija na različitim virtualnim uređajima koji mogu imati različite postavke, različitim verzijama operativnog sustava, različitim procesorskim snagama, memorijama, veličinama ekrana i slično. Nedostaci emulatora su nepodržavanje funkcionalnosti kao što su WiFi, GPS, IR itd. Na slici 9.prikazan je Android emulator.

Slika 9. Android emulator

Izvor : Autor

3.2 Aplikacijske komponente

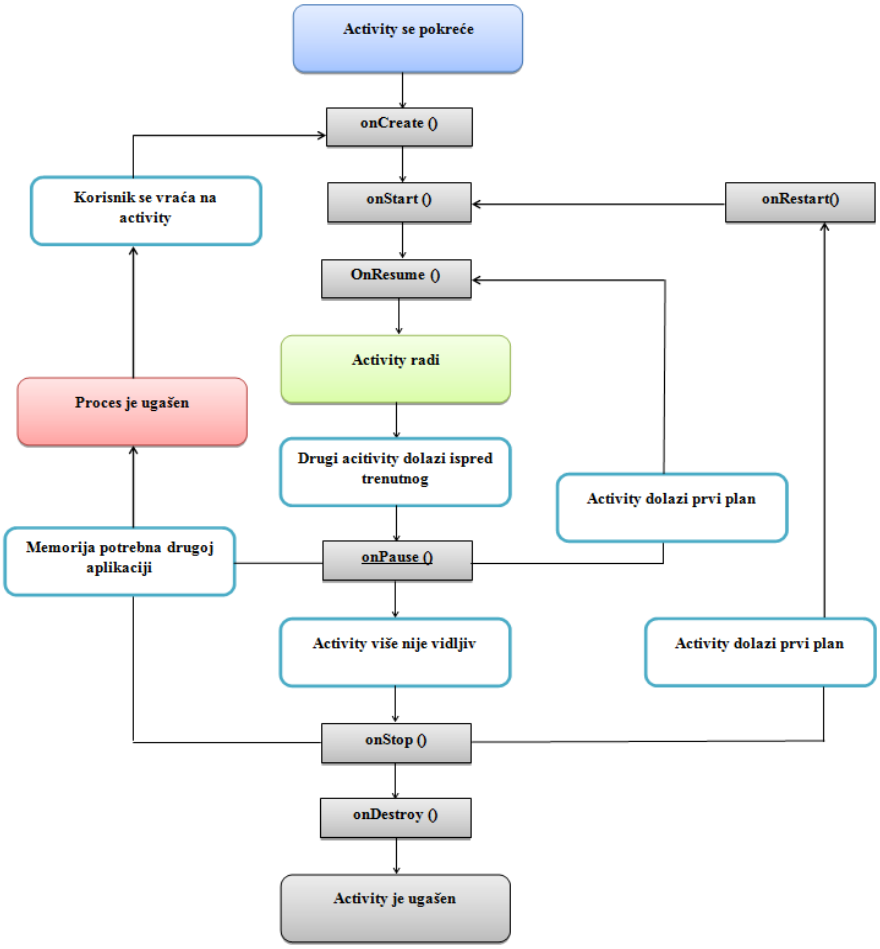
Aplikacijske komponente su na Android platformi najvažniji elementi od kojih se sastoje aplikacije. Svaka komponenta ima svoj životni ciklus, te omogućava sustavu rad s aplikacijom na sebi svojstven način. Postoje četri osnovne komponente : [9]

- aktivnost,
- namijera,
- usluga,
- pružatelj sadržaja.

3.2.1 Aktivnosti

Aktivnost (engl. activity) je zaslon koji korisnik vidi u određenom trenutku na svom uređaju. Tipična aplikacija ima nekoliko aktivnosti i korisnik može prelaziti sa jedne aktivnosti na drugu. Uzmimo za primjer web lokaciju. Baš kao što web lokaciju čini nekoliko web stranica, tako aplikaciju čini nekoliko aktivnosti. Svaka od tih aktivnosti je nezavisan dio aplikacije iako rade zajedno i dio su iste aplikacije. Android aplikacija obično sadrži jednu glavnu aktivnost (npr. početni zaslon) i nekoliko aktivnosti koje su međusobno povezane na način da se iz prve aktivnosti, na zahtjev korisnika, pokreću ostale aktivnosti. Aktivnost je definirana određenom java klasom, a korisničko sučelje učitava iz određene XML datoteke.

Pokretanje neke aktivnosti može biti jako zahtjevan posao. Ujedno može značiti pokretanje novog procesa, alociranje memorije za sve objekte korisničkog sučelja, učitavanje svih objekta od XML koda i slično. Pošto treba relativno puno resursa za pokretanje aktivnosti bilo bi šteta tu aktivnost odbaciti nakon što korisnik napusti zaslon. Zbog svega navedenoga, životnim ciklusom upravlja Activity Menager. Activity Menager je odgovoran za stvaranje, uništavanje i upravljanje aktivnostima. Kada korisnik starta aplikaciju prvi puta, Activity Menager će stvoriti aktivnost koja će nam se prikazati na ekranu. Kasnije, kada korisnik promijeni promijeni aktivnost Activity Menager će početnu aktivnost staviti na čekanje. Tako će, ukoliko se korisnik odluči vratiti na prijašnju aktivnost, aktivnost brže pokrenuti. Ukoliko korisnik nije pristupio duže vrijeme starim aktivnostima te iste aktivnosti će se izbrisati iz memorije (što je i jedan od ciljeva razvoja Android aplikacije) da bi se oslobodila memorija za aktivne aktivnosti. Ovaj mehanizam ima za cilj poboljšanje brzine korisničkog sučelja i korištenje same aplikacije.Na slici 10. prikazan je životni ciklus aktivnosti.



Slika 10. Životni ciklus aktivnosti

Izvor : Autor prema [10]

Životnim ciklusom aktivnosti upravlja se sljedećim naredbama:

- onCreate (Bundle)- poziva se prilikom pokretanja aplikacije,
- onStart ()- označava početak prikaza aplikacije korisniku,
- onResume ()- poziva se prilikom početka ili nakon nastavka interakcije s korisnikom,
- onPause ()- poziva se prilikom prelaska u pozadinski način rada,
- onStop ()- poziva se u slučaju dužeg perioda nekorištenja aplikacije,
- onRestart ()- označava ponovno pozivanje aplikacije iz zaustavljenog stanja,
- onDestroy ()- poziva se trenutak prije gašenja aplikacije,
- onSaveInstanceState (Bundle)- opcionalna metoda koja se poziva u slučaju očuvanja stanja i
- onRestoreInstanceState (Bundle)- poziva se prilikom ponovnog pokretanja aktivnosti iz stanja prethodno pohranjenog onSaveInstanceState () metodom.

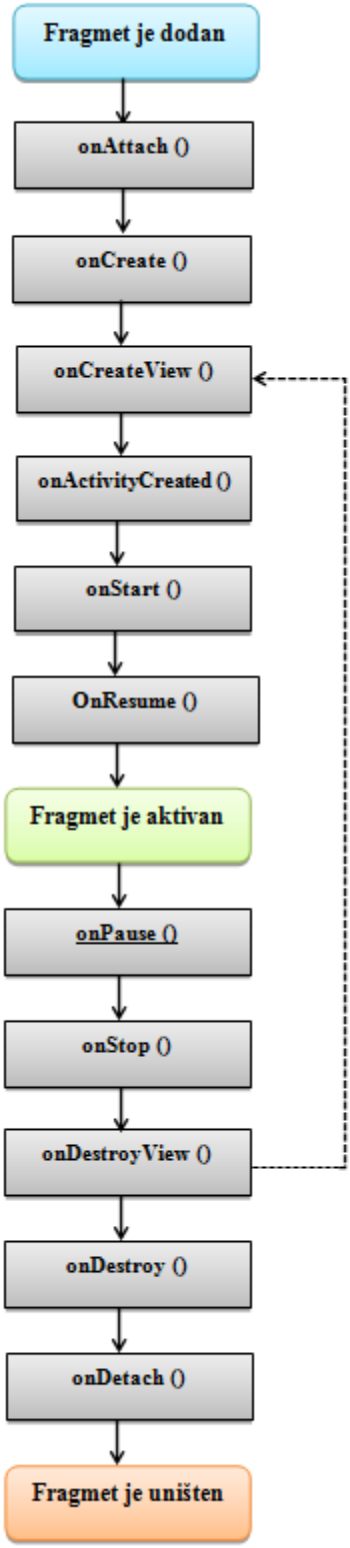
3.2.1.1 Fragmenti

Fragment (eng. fragmet) predstavlja dio korisničkog sučelja u aktivnosti. Možemo ga zamisliti kao mini-aktivnost koja ima svoj životni ciklus, ali za razliku od standardnih aktivnosti ne može se prikazivati zasebno, tj. da bi se prikazao fragment, mora se dodati u standardnu aktivnost. Fragment možemo dodati odnosno ukloniti dok se aktivnost izvodi, te on izravno utječe na životni ciklus aktivnosti. Uzmimo za primjer da je aktivnost pauzirana, tada je i fragment u njoj pauziran, a kada je uništena i fragmenti su uništeni. Dok je aktivnost pokrenuta, fragmenti u aktivnosti se mogu dodavati odnosno uklanjati.Fragment kao i aktivnost je definiran određenom java klasom, a korisničko sučelje učitava iz određene XML datoteke.Na slici 11.prikazan je životni ciklus fragmenata.

Glavne karakteristika fragmenta

- enkapsuliraju dijelove korisničkog sučelja
- mogu se jednostavno ponovo koristiti
- svaki ima svoj životni ciklus
- dinamički se mogu dodavati i uklanjati
- sučelje se lakše prilagođaja uređaju

- za razliku od aktivnosti ne trebaju se registrirati u manifestu



Slika 11. Životni ciklus fragmenta

Izvor : Autor prema [11]

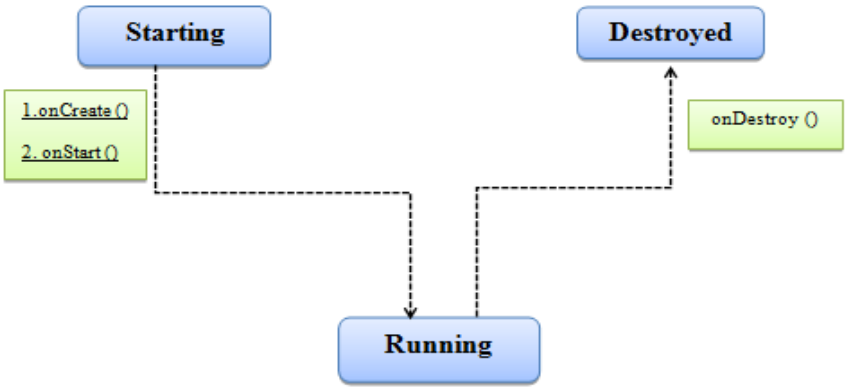
3.2.2 Namjere

Namjere (eng. Intents) su poruke koje se šalju između glavnih blokova aplikacije. Njihova funkcija je da pokreću aktivnost, govore uslugama da se pokrenu ili zaustave. Namjere su asinkrone što znači da kod koji šalje namjere ne mora čekati da budu izvršene.

Namjera se dijeli na implicitnu i eksplicitnu. Kod eksplicitne namjere pošiljatelj jasno govori koja komponenta treba primiti poruku. U implicitnoj namjeri pošiljatelj zadaje vrstu primatelja. Na primjer, aktivnost može poslati naredbu da se otvori web stranica. U tom slučaju će se sve aplikacije koje mogu otvoriti web stranicu „natjecati“ da obave taj posao. Ukoliko postoje aplikacije koje se natječu, sustav traži od korisnika da izabere koju želi upotrijebiti. Ova vrsta razmjene poruka omogućava da se s prilagođenom zamijeni bilo koja aplikacija u sustavu.

3.2.3 Usluge

Usluge ili servisi (eng. services) obavljaju funkciju u pozadini aplikacije i nemaju korisničko sučelje. Imaju mogućnost obavljanja istih poslova kao i aktivnosti, ali bez prikazivanja. Korisne su za akcije koje se obavljaju u određeno vrijeme, bez obzira što je prikazano na zaslonu. Primjer je slušanje glazbe dok je istovremeno otvorena druga aplikacija. Nasuprot aktivnosti usluge imaju mnogo jednostavniji životni ciklus. Usluge mogu biti pokrenute ili zaustavljene. Na slici 12.prikazan je životni ciklus usluga.



Slika 12. Životni ciklus usluga

Izvor : Autor prema [9]

3.2.4 Pružatelji sadržaja

Pružatelji sadržaja (eng. content providers) vrše funkciju razmjene podataka između aplikacija.Android izvršava svaku aplikaciju u posebnom izoliranom okruženju (eng. Sandbox). Podaci ovih aplikacija su potpuno izolirani od podataka drugih aplikacija. Primjer korištenja je “contactsprovider”, pružatelj sadržaja podataka o određenom kontaktu. Ovakvo odvajanje korisničkog sučelja i pohrane podataka nudi veću fleksibilnost samog sustava. Na primjer, korisnik može instalirati alternativnu aplikaciju adresara, koji zatim koristi iste podatke kao i podrazumijevana aplikacija. Pružatelji sadržaja su jednostavna sučelja sa standardnim metodama Insert (), update (),delete(), query (). Radi se s metodama koje se često rabe za rad s bazama podataka.

4. ANDROID APLIKACIJA

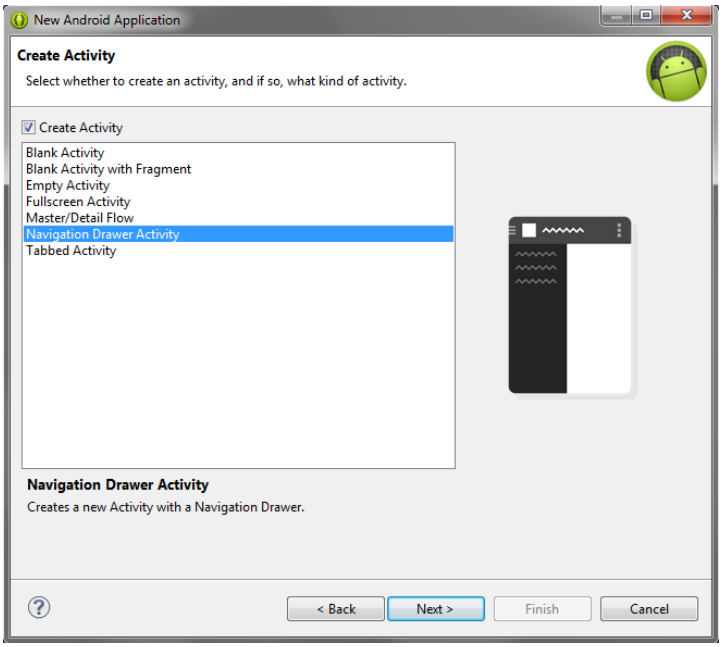
Aplikacija je zamišljena kao kviz provjere znanja iz područja informatike i elektronike. Pokretanjem aplikacije korisniku se pruža mogućnost odabira igranja kviza, informacija o kvizu te izlaz iz kviza. Ako se izabere opcija "kviz", ponuđuju se opcije igranja kviza te prikaza rezultata. Prilikom odabira igranja kviza na ekranu se pojavljuje pitanje sa tri odgovora od kojih je samo jedan točan odgovor. Nakon odabira odgovora korisniku se pritiskom na tipku "dalje" ponuđuje slijedeće pitanje. Nizom od deset pitanja kviz završava te se korisniku prikazuje postignuti rezultat i ocijena od jedan do pet i od njega se zahtijeva da upiše svoje ime i prezime. Rezultat se zatim sprema u bazu podataka. Također, u bazi podataka spremljena su i pitanja. Opcija "rezultati" prikazuje deset najboljih rezultata kviza. Informacije o kvizu sadrže ime i prezime mentora i autora, kolegij te studijski smjer. Pritiskom na "izlaz" korisnik gasi aplikaciju.

4.1 Priprema razvojnog okruženja

Da bi se pristupilo razvoju aplikacije potrebno je instalirati Eclipse. Uz instalaciju Eclipsea na računalo na kojem će se razvijati aplikacija, potrebno i instalirati Android SDK alate te ih implementirati u Eclipse. Potrebno je osigurati i testno okruženje. U ovom je slučaju korišten emulator za Android i mobilni uređaj Explorer A310 s verzijom Androida 2.3.5. Za testiranje na mobilnom uređaju bilo je potrebno osigurati i kabel za spajanje s računalom preko USB ulaza.

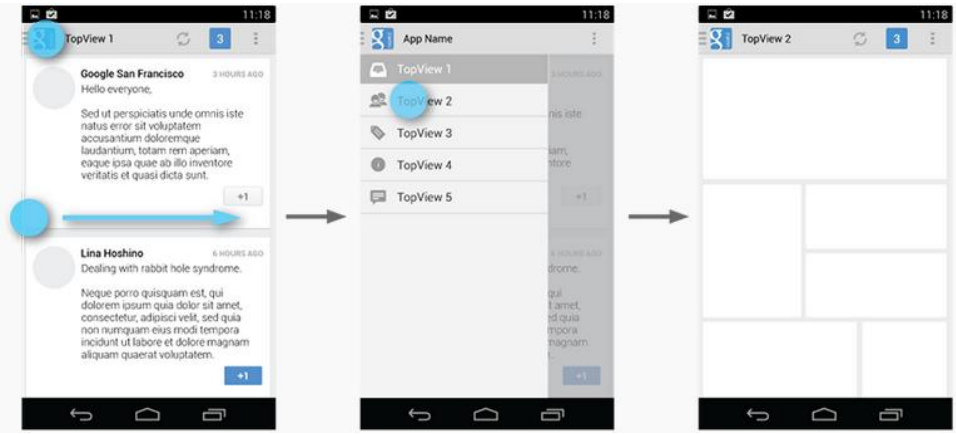
4.2 Razvoj aplikacije

Nakon što se odaber ime projekta odnosno aplikacije i izabere minimalna verzija operativnog sustava za aplikaciju, programeru se nudi opcija izrade aktivnosti (slika 13). U ovom radu izabrana je "Navigation drawer" aktivnost. Time se automatski kreira padajući izbornik sa tri ili više opcija. Izbornik se otvara pritiskom na ikonu u gornjem lijevom kutu ekrana, odnosno "klizanjem" prsta od desne strane ekrana prema lijevoj što je i prikazano na slici 14.



Slika 13. Biranje početne vrste aktivnosti

Izvor : Autor



Slika 14. "Navigation drawer" aktivnost"

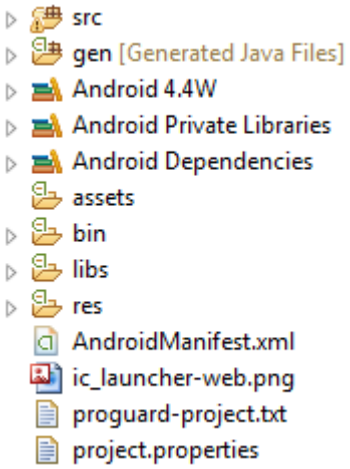
Izvor : Navigation drawer, *www.developer.android.com* (28.10.2014)

Kada se stvori android projekt automatski se generiraju i njegove datoteke (slika 15.) :

- src - direktorij koji sadrži sve java klase datoteka aplikacije. U njoj se nalazi kod aplikacije,
- gen - sadrži 2 automatski kreirane datoteke :
    1. BuildConfig.java - bazična konfiguracija aplikacije,

2. R.java - osigurava pristup resursima iz aplikacije. Svaki resurs ima svoj specifični ID koji je ovdje spremljen,

- android biblioteke - sadrži biblioteke klasa potrebne za samu Android aplikaciju,
- assets - sadrži resurse koje aplikacija koristi, kao što su tekstualne datoteke, baze podataka, HTML datoteke,
- libs - sadrži android-support-v4, tj. najnoviju verziju android API-a, koja je kompatibilna sa prijašnjim verzijama,
- bin - sadrži datoteke napravljene od Android Development Tools-a. Također sadrži APK format koji se koristi za instalaciju aplikacije na Android operacijski sustav,
- res - direktoriji koji sadrže slike različite rezolucije (npr. mala rezolucija za mobitel, srednja rezolucija za tablet uređaj)
- androidmanifest.xml - ovdje se bilježe sve aktivnosti koje aplikacija ima, dozvole koje su potrebne aplikaciji za rad, definira se ime paketa i dr.



Slika 15. Komponente Android projekta

Izvor : Autor

4.3 Fragmenti aplikacije

Aplikacija "Kviz" sastoji se od 5 fragmenata. Fragmenti odnosno zasloni sa kojima korisnik interakterira jesu :

- Fragment "kviz meni" - zaslon koji sadržava dva gumbića : gumbić "Započni kviz" čijim odabirom započinje kviz, te gumbić "Rezultati" čijim se odabirom prikaže lista deset najboljih rezultata.

- Fragmenti "kviz" - zasloni koji sadržavaju pitanje, tri odgovora od kojih je samo jedan točan te gumbić "Dalje" čijim se odabirom prikazuje novo pitanje.
- Fragment "upis podataka" - zaslon koji se pojavi nakon što korisnik odgovori na niz od deset pitanja. Sadrži rezultat igre odnosno ocijenu od jedan do pet, polja za upis imena i prezimena te gumbić "Spremi rezultat" čijim se odabirom rezultat sprema u bazu podataka.
- Fragment "rezultati" - zaslon koji sadrži deset najboljih rezultata sa pripadajućim imenom, prezimenom te ocijenom od jedan do pet. Rezultati su poredani od najbolje ocijene prema najlošijoj ocijeni.
- Fragment "info" - zaslon koji sadrži svrhu i informacije o aplikaciji.

4.3.1 Fragment kviz

Naredbom View rootView = inflater.inflate(R.layout.*kviz_menu*, container, false); zadano je grafičko sučelje iz mape layout koje se prikazuje za fragment "Kviz meni", konkretno zadana je datoteka kviz_menu.xml.

```
public class KvizMenu extends Fragment{
    //Definiranje varijabli
    View v;
    Button start;
    Button results;

    public KvizMenu(){}
    //Dio životnog ciklusa fragmenta koji sadrži glavni dio programa
    @Override

    public View onCreateView(LayoutInflater inflater, ViewGroup
container,
            Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

        //Definiranje grafičkog izgleda
        View rootView = inflater.inflate(R.layout.kviz_menu, container,
false);
        v = rootView;

        INIT();
        EVENTS();

        return rootView;
    }
```

Da bi gumbići dobili svoj izgled u funkciji "INIT" (inicijalizacija) postavlja se njihov pogled na Id koji je definaran u XML datoteci. Izgled se postavlja pomoću naredbe findViewById koja traži izgled aktivnosti prema nazivu deklaracije za određeni gumbić u XML datototeci. Važan dio je dodjeljivanje funkcije gumbiću u funkciji "EVENTS" (događaji). U ovom slučaju gumbićima se dodjeljuje funkcija kada se pritisne na određeni gumbić. Naredbom setOnClickListener(new View.OnClickListener() { } (tzv. osluškivači događaja) definirano je da gumbić bude osjetljiv na dodir, odnosno da reagira na dodir. Osluškivač događaja je koncept android platforme koji nam omogućava reagiranje na događaje kada se dogode, te po potrebi i obrađivanje istih. Koncept se temelji na povezivanju osluškivača na događaj nekog od prethodno navedenih elemenata te time i na izmjeni izvršavanja njegove funkcionalnosti. Pritsikom na gumbić "Započni kviz" pokreće se nova funkcija "START", a pritiskom na gumbić "Rezultati" pokreće se nova funkcija "RESULTS".

```
//Dodjeljivanje funkcije objektima
    public void INIT(){
        start = (Button) v.findViewById(R.id.QUIZ_START);
        results = (Button) v.findViewById(R.id.QUIZ_RESULTS);
    }
    //Pritiskom na tipku start pokreće se funkcija START()
    public void EVENTS(){
        start.setOnClickListener(new View.OnClickListener() {

            @Override
            public void onClick(View v) {
                START();
            }
        });
        //Pritiskom na tipku rezultati pokrećese funkcija RESULTS()
        results.setOnClickListener(new View.OnClickListener() {

            @Override
            public void onClick(View v) {
                RESULTS();
            }
        });
```

Funkcija "START" naredbom Fragment fg = new Quiz_start() stvara novi fragment "Kviz" čime započinje igra, dok funkcija "RESULTS" naredbom Fragment fg = new Quiz_results stvara novi fragment koji ispisuje 10 najboljih rezultata. Naredbom public void NEW_FRAGMENT(Fragment fg){ } izvršava se zamjena postojećeg fragmenta sa novim fragmentom.

```
    //Stvaramo novi fragment Quiz_start()
    public void START(){
        Fragment fg = new Quiz_start();
        NEW_FRAGMENT(fg);
    }
    //Stvaramo novi fragment Quiz_results()
    public void RESULTS(){
        Fragment fg = new Quiz_results();
        NEW_FRAGMENT(fg);
    }
    //Vrši se transakcija postojećeg fragmenta sa novim
    public void NEW_FRAGMENT(Fragment fg){
        android.support.v4.app.FragmentTransaction fragmentTransaction =
getFragmentManager().beginTransaction();
        fragmentTransaction.replace(R.id.container,fg );
```

Koristi se dva načina dizajniranja korisničkog sučelja: deklarativno i proceduralno. Deklarativni se odnosi na pisanje XML[8] koda, dok se proceduralni odnosi na pisanje Java koda. Deklerativni način se mnogo više koristi iz razloga svoje jednostavnosti, ali i zbog toga jer se njegovim upotrebljavanjem odvaja prikaz same aplikacije od glavnog java koda.

Sukladno tome postoje različite vrste rasporeda elemenata tj. objekata unutar zaslona:

- linearni raspored (eng. Linear layout) - predstavlja osnovni raspored elemenata odnosno kontrola gdje su svi osnovni elementi raspoređeni u samo jednom redu ili stupcu. Za složenije aplikacije ovakav raspored elemenata nije uvijek najbolje rješenje, ali za jednostavnije aplikacije je najbolje rješenje.
- apsolutni raspored (eng. Apsolute layout) - donosi potpunu slobodu postavljanja elemenata
- tablični raspored (eng. Table layout) - koristi se kod razvrstavanja elemenata u više redova ili stupaca tablice. Na taj se način mogu dobiti pravilni rasporedi osnovnih elemenata na pogledu
- relativni raspored (eng. Relative layout) - elementi se postavljaju na različita mjesta na dostupnom prostoru za prikazivanje, ali točno mjesto nije određeno apsolutnim koordinatama kao kod apsolutnog rasporeda, nego je pozicija elemenata definirana relativnim koordinatama prema drugoj osnovnoj kontroli
- okvirni raspored (eng. Frame layout) - koristi se za rezerviranje mjesta za naknadno postavljanje elemenata na način da se elementi postavljaju prema gornjem lijevom uglu ekrana

[8] Exstensable Markup Language

- listajući pregled (eng. Scroll view) - upotrebljava se kada na fizičkim dimenzijama ekrana nije moguće prikazati sve potrebne osnovne elemente na pregledan način tj. bez međusobnog preklapanja i jasnog razdvajanja.

Kako bi svaki od upotrebljenih elemenata bilo moguće koristiti u Java klasi u kojoj će se pozvati ova XML datoteka svakom od elemenata treba pridodati svoj ID preko kojeg ćemo u Java klasi prepoznavati i pozivati određeni element. Dodjeljivanje ID-a se vrši kod svakog od elemenata te se isti dodaje tako da se unutar prostora elementa doda sljedeći dio koda : android:id="@+id/željeni_id".

Naredbama android:layout_width="*match_parent*" i android:layout_height="*wrap_content*" određuje se veličina gumbića, konkretno gumbić će se grafički prilagoditi veličini ekrana uređaja odnosno biti će raširen preko cijelog zaslona.Također, naredbom android:text="*@string/start*" povezali smo gumbić sa string.xml datotekom (vidi prilog B) u kojoj je definiran njegov tekstualni sadržaj.

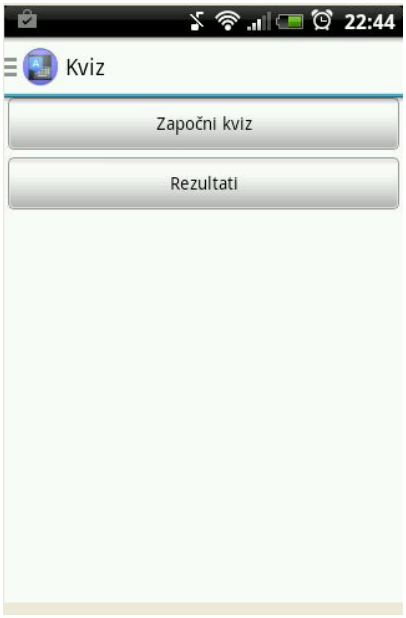
XML datoteka koja definira grafičko sučelje fragmenta izgleda ovako:

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical" >

    <Button

                android:id="@+id/QUIZ_START"
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:text="@string/start"
                 />

    <Button
                android:id="@+id/QUIZ_RESULTS"
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:text="@string/results"
                />
</LinearLayout>
```



Slika 16. Zaslon fragmenta "Kviz meni"

Izvor : Autor

4.3.2 Fragment Kviz

Kao i kod prethodnog fragmenta grafički izgled definiran je pomoću naredbe

View rootView = inflater.inflate(R.layout.*kviz*, container, false).

```
    public static class Quiz_start extends Fragment{
        //Definiranje varijabli
        Activity a;
        View v;

        int BUNDLE_QUESTION_NUMBER;

        Button next;
        TextView question;
        RadioButton rda, rdb, rdc;
        RadioGroup grp;

        Question currentQ;
        ArrayList<Question> questions;
        Database data;
        Database.App qData;

        int maxQ,count,score;
        String name,surname;

        int idStatico;
        private Bundle savedInstanceState;

        public Quiz_start(){}
```

```
//Dio životnog ciklusa fragmenta koji sadrži glavni dio programa
        @Override
        public View onCreateView(LayoutInflater inflater, ViewGroup
container,
                Bundle savedInstanceState) {
            super.onCreate(savedInstanceState);

            //Definiranje grafičkog izgleda
            View rootView = inflater.inflate(R.layout.kviz, container,
false);

            a = getActivity();
            v = rootView;
             if (savedInstanceState != null) {
                 // Restore last state for checked position.
            BUNDLE_QUESTION_NUMBER =
savedInstanceState.getInt("QUESTION_NUMBER");
             }

            this.INIT();
            this.EVENTS();

            //Funkcija koja pokreće kviz
            this.START();

            return rootView;
        }
```

Da bi objekti dobili svoj izgled moramo postaviti njihov pogled, kao i u prethodnom fragmentu na Id koji je definaran u XML datoteci. To se postiže u funkciji "INIT" pomoću naredbe findViewById koja traži izgled aktivnosti prema nazivu deklaracije u XML datototeci.

Fragment "Kviz" sastoji se od:

- pitanja na koje korisnik treba odgovoriti,
- radio grupe - komponenta koja sadrži više radio-dugmića, tj. omogućava izbor jednog dugmića između više ponuđenih mogućnosti,
- tri radio dugmića - korisnik može odabrati samo jedan od njih,
- dugmića "Dalje" - njegovim odabirom na zaslonu se prikazuje novo pitanje te se pokreće funkcija provjere odgovora.

Razlika između radio grupe i radio dugmića je u tome što se radio grupa promatra kao jedna cjelina, a u drugom slučaju se postavljaju kao zasebna dugmad, s tim da jedno mora biti izabrano.

U istoj se funkciji osim postavljanja pogleda na Id, ograničava maksimalni broj pitanja u kvizu na deset, trenutno pitanje se postavlja na negativnu vrijednost te se rezultat postavlja na nulu.

```
        public void INIT(){
            next = (Button) v.findViewById(R.id.QUIZ_NEXT);
            question = (TextView) v.findViewById(R.id.QUIZ_QUESTION);

            grp=(RadioGroup) v.findViewById(R.id.QUIZ_RADIO);
            rda = (RadioButton) v.findViewById(R.id.radio0);
            rdb = (RadioButton) v.findViewById(R.id.radio1);
            rdc = (RadioButton) v.findViewById(R.id.radio2);
            //Maksimalni broj pitanja koje odgovaram
            //Trenutno pitanje = -1
            //Rezultat = 0
            maxQ = 10;
            count = -1;
            score = 0;
```

Kao i u prethodnom fragmentu u funkciji "EVENT" daje se funkcija gumbiću naredbom next.setOnClickListener(new View.OnClickListener(){ } . Pritiskom na gumbić pokreće se nova funkcija "CHECK_ANSWER". Funkcija "CHECK_ANSWER" provjerava točnost odgovora tj. naredbom currentQ.getANSWER().equals(answer.getText()) uspoređuje odabrani odgovor sa točnim odgovorom te se if-else petljom u slučaju točnog odgovora rezultat povećava dok u slučaju netočnog odgovora rezultat ostaje isti. Samim izvršavanjem funkcije provjere odgovora povećava se zbroj pitanja (npr. kada je započeo kviz zbroj je -1, a pritiskom na tipku "Dalje" zbroj postaje jednak nuli) Funkcijom "CLEAN" briše se prethodno pitanje odnosno osigurava se da pitanje ne ostane na zaslonu uređaja. Također se pokreće i funkcija "START".

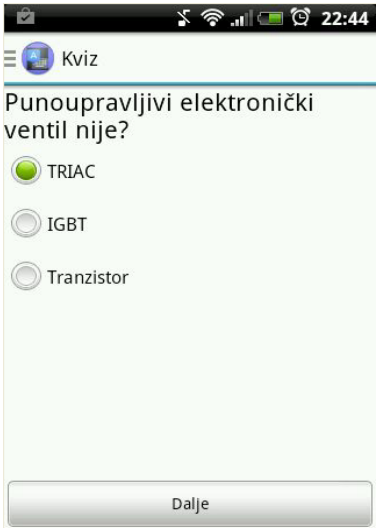
```
public void CHECK_ANSWER(){
            RadioButton answer=(RadioButton)
v.findViewById(grp.getCheckedRadioButtonId());

            //ako je odgovor točan povečava rezultat
            if (currentQ.getANSWER().equals(answer.getText())){
                score++;
            }
            else {

            }
            count++;
            this.CLEAN();
            this.START();
        }
```

Funkcija "START" kao što samo ime govori pokreće kviz. Ako je zbroj pitanja manji od nule (jedino u slučaju početka kviza), baza podataka se funkcijom "EMPTY_DATABASE" prazni, zatim se funkcijom "FILL_DATABASE" baza podataka popunjava pitanjima te se funkcijom "GET_QUESTIONS" pitanja dohvaćaju iz baze podataka odnosno funkcijom "MIX_QUESTIONS" pitanja mješaju da ne budu uvijek istim redosljedom. U slučaju da je zbroj pitanja manji od maksimalnog broja pitanja izvršava se funkcija "SET_QUESTION" koja zamjenjuje postojeće pitanje novim pitanjem. U slučaju da zbroj pitanja ne zadovoljava niti jedan navedeni uvijet briše se prethodno pitanje i stvara se novi fragment "Upis podataka".

```
//Funckija koja se brine o tijeku kviza
        public void START(){
            //Ako je zbroj pitanja manji od 0
            if (count<0){
                currentQ = new Question;
                //Prazni se baza podataka
                EMPTY_DATABASE();
                //Puni se baza podataka
                FILL_DATABASE();
                //Dohvaća pitanja
                GET_QUESTIONS();
                //Mjea pitanja
                MIX_QUESTIONS();
                //Povećava se zbroj pitanja
                count++;
            }
            //Ako je zbroj manji od maximalnog broja pitanja
            if (count<maxQ){
            //Postavlja novo pitanje
            SET_QUESTION();
            }//Ako nije ništa od navedenog
            else{
                //Brisanje pitanja
                CLEAN();
                //Stvaranje novog fragmenta
                Fragment fg = new Quiz_result();
                NEW FRAGMENT(fg);
```



Slika 17. Zaslon fragmenta "Kviz"

Izvor : Autor

4.3.3 Fragment "Upis podataka"

Naredbom View rootView = inflater.inflate(R.layout.*kviz_rezultati*, container, false) zadano je grafičko sučelje iz mape layout koje se prikazuje za fragment "Upis podataka".

```
//Fragmenta za ispis rezultata nakon završetka kviza
    public static class Quiz_result extends Fragment{
        //Definiranje varijabli
        View v;
        Activity a;

        int score,maxQ;
        String text,name,surname;
        float rating;

        TextView result_text;
        RatingBar result_bar;
        Button save;
        AutoCompleteTextView input_name,input_surname;

        Database data;
        Database.App rData;

        public Quiz_result(){
        }
        @Override
        public View onCreateView(LayoutInflater inflater, ViewGroup
container,
                Bundle savedInstanceState) {
            super.onCreate(savedInstanceState);
```

```
//Definiranje grafičkog izgleda
            View rootView = inflater.inflate(R.layout.kviz_rezultati,
container, false);
            a = getActivity();
            v = rootView;
            INIT();
            EVENTS();
            SHOW_RESULT();
            return rootView;
        }
```

Da bi objekti dobili svoj izgled u funkciji "INIT" kao i u prethodnim fragmentima postavlja se njihov pogled na Id koji je definaran u XML datoteci.

U funkciji "EVENTS" gumbiću se pomoću naredbe save.setOnClickListener(new View.OnClickListener() dodjeljuje funkcija. Pritiskom na gumbić pokreće se funkcija "CHECK_INPUTS" koja provjerava jesu li u polja za upis imena odnosno prezimena upisani podaci. U slučaju upisa podataka, isti se spremaju u bazu podataka te se korisnika preusmjeruje na "Kviz meni" dok se u slučaju praznih polja korisniku ne dozvoljava da spremi svoje rezultate.

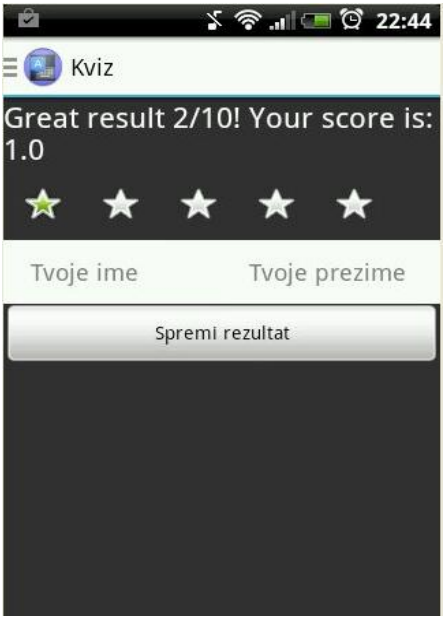
```
public void EVENTS(){
            save.setOnClickListener(new View.OnClickListener() {

                @Override
                public void onClick(View v) {
                    CHECK_INPUTS();
```

Funkcija "SHOW RESULT" koja se poziva u fragmentu "Upis podataka" prikazuje učinak igrača, broj točno odgovorenih pitanja te ocijenu od jedan do pet (tzv "rating bar").

```
//Ispisuje rezultat na zaslon sa rating bar-om
        public void SHOW_RESULT(){
            text = "Great result " + score + "/" + maxQ + "! Your score
is: "+ rating;
            result_text.setText(text);

            result_bar.setRating(rating);
        }
```



Slika 18. Zaslon fragmenta "Upis rezultata"

Izvor : Autor

4.3.4 Fargment "Rezultati"i "Info"

Fragment "Rezultati" i "Info" (prilog A) su zasloni aplikacije koji prikazuju 10 najboljih rezultata aplikacije "Kviz", odnosno informacije o istoj. Zaslon "Rezultati" sastoji se od liste rezultata i jednog gumbića čijim se pritiskom korisnik vraća na "Kviz meni", dok se zaslon "Info" sastoji od prikaza kolegija, imena i prezimena mentora i autora aplikacije, studijskog smjera te loga Pomorskog fakulteta u Rijeci.

Slika 19. Zasloni fragmenata "Rezultati" i "Kviz"

Izvor : Autor

4.4 Baza podataka

Aplikacija koristi SQLitle bazu podataka koja je izrađena u Eclipse razvojnoj okolini. SQLite je relacijska baza podataka temeljena na maloj C programskoj biblioteci. Glavne značajke SQLite baze podataka su:

- nema potrebe za odvojenim poslužiteljem ili sistemom da bi radio. SQLite pristupa direktno svojim podacima,
- nulta konfiguracija - nema potrebu za prilagođavanjem i administracijom,
- cijela baza podataka smještena u jednoj datoteci,
- podržava nekoliko terabajta velike baze i nekoliko gigabajta velike nizove podataka,
- jednostavno korisničko sučelje.

U bazi podataka pohranjuju se pitanja sa odgovorima te rezultati samog kviza. Dvije stalne tablice čine ovu bazu podataka (prilog A) :

- tablica pitanja - pohranjuje pitanje sa tri moguća odgovora te točan odgovor. Atribut QUIZ_COLUMN_ID označava primarni ključ koji pruža jedinstven indentifikator za svaki redak. Vrijednost atributa QUIZ_COLUMN_QUESTION je pitanje, atributi QUIZ_COLUMN_A, QUIZ_COLUMN_B, QUIZ_COLUMN_C predstavljaju tri moguća odgovora na pitanje, dok atribut QUIZ_COLUMN_ANSWER predstavlja točan odgovor,
- tablica rezultata - pohranjuje rezultate kviza. Atribut RESULT_COLUMN_ID označava primarni ključ, vrijednost atributa RESULT_COLUMN_NAME i RESULT_COLUMN_SURNAME je ime odnosno prezime dok atribut RESULT_COLUMN_RESULT predstavlja sam rezultat.

Kao ilustraciju tablice rezultata, na tablici 1. može se vidjeti izgled tablice rezultata.

RESULT_COLUMN_ID	RESULT_COLUMN_NAME	RESULT_COLUMN_SURNAME	RESULT_COLUMN_RESULT
1.	Marin	Marinović	10
2.	Pomorski	Fakultet	9

Tablica 1. Baza podataka "Rezultati"

4.5 Mogućnosti poboljšanja aplikacije

Postoje različite mogućnosti poboljšanja aplikacije. U prvom planu bismo trebali navesti poboljšanje grafičkog izgleda same aplikacije. Pod poboljšanje grafičkog izgleda prvenstveno se podrazumijeva promijena teme, font i boja slova, dodijeljivanje dugmićima odgovarajuće slike koja prvenstveno zavisi o samoj funkciji dugmića.

Osim promijene grafičkog izgleda aplikacije također se možemo bazirati na promjeni pitanja, tj. da pitanja stavimo u različite kategorije (teme) na koje se odnose (npr. jakost magnetskog polja u kategoriju elektronike...). Vezano uz promijenu pitanja također možemo omogućiti korisniku da prilikom odgovora na određeno pitanje pojavi TOČNO ili NETOČNO tj. dali je ponuđeni odgovor ispravan ili pogrešan. Osim svrstavanja pitanja po kategorijama i pružanje mogućnosti da li je korisnik ispravno ili pogrešno odgovorio na određeno pitanje također postoji mogućnost i ubacivanja slika u pitanja, kao i određeno vrijeme za odgovor pojedinog pitanja (npr. 30 s).

Jedna od bitnijih načina poboljšanja je mogućnost igranja više korisnika istovremeno, kao jednog od oblika natjecanja.

5. ZAKLJUČAK

Pokretni uređaji postali su svakodnevni alati koje koristi veliki broj ljudi, različitih godina i zanimanja. Oni više ne predstavljaju luksuz, već su dostupni svima. U oblasti pokretnih uređaja, Android je vodeći operacijski sustav već nekoliko godina, što ga čini prvim izborom svakog korisnika koji bi htjeli da koriste jedan ovakav uređaj. Činjenica da je to open-source operacijski sustav, čini ga idealnom platformom za kreiranje najrazličitijih aplikacija, i korištenje svih funkcionalnosti koje najnovije tehnologije pružaju. U korist sustavu ide i bogata baza biblioteka te funkcionalni alati za izradu aplikacija. Također, Android ima vrlo dobru podršku za Eclipse kao jednog od najpopularnijih programa za razvoj. Uz aktivna poboljšanja razvojne okoline za pisanje aplikacija i programskih biblioteka, Android predstavlja operacijski sustav koji obećava ostaviti značajni trag u svijetu razvoja programske podrške, ne samo za pokretne uređaje već za male uređaje općenito.

U ovom diplomskom radu opisan je razvoj Android aplikacije "Kviz". Kroz razvoj aplikacije prikazana je upotreba nekih od funkcija i metoda koje se koriste kod izrada Android aplikacija. Pojašnjeni su pojmovi vezani za Android, kao što su njegova arhitektura, aplikacijske komponente koje čine samu aplikaciju te je pojašnjen program za razvoj aplikacija Eclipse. Također, ovim se diplomskim radom pokazalo kako je bez ikakvog prijašnjeg znanja programskog jezika u kratkom vremenu moguće napraviti funkcionalnu aplikaciju.

LITERATURA

[1] Čarapina, M: "Operacijski sustav Android i načini povezivanja s poslužiteljem", Fakultet elektrotehnike i računarstva, Zagreb, srpanj 2009.

[2] online: http://hr.wikipedia.org/wiki/Android_(operacijski_sustav) (9.10.2014)

[3] online: http://hr.wikipedia.org/wiki/Dosadašnje_inačice_sustava_Androida (9.10.2014)

[4] online: http://pametni-telefoni.blogspot.com/2012/08/sta-su-to-pametni-smart-telefoni.html (10.10.2014)

[5] online http://hr.wikipedia.org/wiki/Tablet (10.10.2014)

[6] online: http://hr.wikipedia.org/wiki/E-knjiga (10.10.2014)

[7] online: http://www.tutorialspoint.com/android/android_architecture.htm (12.10.2014)

[8] Dujmović, A: "Upravljanje videonadzorom cestovnog prometa mobilnim uređajem na Android platformi", Sveučilište u Dubrovniku, Odjel za elektrotehniku i računarstvo, Dubrovnik, rujan 2010.

[9] Gargenta, M., Nakamura, M: Learning Android: Develop Mobile Apps Using Java and Eclipse, O'Reilly Media, 2014

[10] online: http://developer.android.com/reference/android/app/Activity.html (18.10.2014)

[11] online: http://developer.android.com/guide/components/fragments.html (18.10.2014)

[12] Kušek, M., Topolnik, M.: Uvod u programski jezik Java, Fakultet elektrotehnike i računarstva, Zagreb, 2008.

[13] Glasinović, F.: Android aplikacija kao GNSS kontroler, Geodetski fakultet, Zagreb, rujan 2012.

POPIS SLIKA

POPIS TABLICA

PRILOG A

NavigationDrawerFragment.java

```
package com.example.helloworld5;

import android.support.v7.app.ActionBarActivity;
import android.app.Activity;
import android.support.v7.app.ActionBar;
import android.support.v4.app.Fragment;
import android.support.v4.app.ActionBarDrawerToggle;
import android.support.v4.view.GravityCompat;
import android.support.v4.widget.DrawerLayout;
import android.content.SharedPreferences;
import android.content.res.Configuration;
import android.os.Bundle;
import android.preference.PreferenceManager;
import android.view.LayoutInflater;
import android.view.Menu;
import android.view.MenuInflater;
import android.view.MenuItem;
import android.view.View;
import android.view.ViewGroup;
import android.widget.AdapterView;
import android.widget.ArrayAdapter;
import android.widget.ListView;
import android.widget.Toast;

/**
 * Fragment used for managing interactions for and presentation of a
navigation drawer.
 * See the <a
href="https://developer.android.com/design/patterns/navigation-
drawer.html#Interaction">
 * design guidelines</a> for a complete explanation of the behaviors
implemented here.
 */
public class NavigationDrawerFragment extends Fragment {

    /**
     * Remember the position of the selected item.
     */
    private static final String STATE_SELECTED_POSITION =
"selected_navigation_drawer_position";

    /**
     * Per the design guidelines, you should show the drawer on launch
until the user manually
     * expands it. This shared preference tracks this.
     */
    private static final String PREF_USER_LEARNED_DRAWER =
"navigation_drawer_learned";

    /**
     * A pointer to the current callbacks instance (the Activity).
     */
    private NavigationDrawerCallbacks mCallbacks;

    /**
```

```
     * Helper component that ties the action bar to the navigation drawer.
     */
    private ActionBarDrawerToggle mDrawerToggle;

    private DrawerLayout mDrawerLayout;
    private ListView mDrawerListView;
    private View mFragmentContainerView;

    private int mCurrentSelectedPosition = 0;
    private boolean mFromSavedInstanceState;
    private boolean mUserLearnedDrawer;

    public NavigationDrawerFragment() {
    }

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

        // Read in the flag indicating whether or not the user has
demonstrated awareness of the
        // drawer. See PREF_USER_LEARNED_DRAWER for details.
        SharedPreferences sp =
PreferenceManager.getDefaultSharedPreferences(getActivity());
        mUserLearnedDrawer = sp.getBoolean(PREF_USER_LEARNED_DRAWER,
false);

        if (savedInstanceState != null) {
            mCurrentSelectedPosition =
savedInstanceState.getInt(STATE_SELECTED_POSITION);
            mFromSavedInstanceState = true;
        }

        // Select either the default item (0) or the last selected item.
        selectItem(mCurrentSelectedPosition);
    }

    @Override
    public void onActivityCreated (Bundle savedInstanceState) {
        super.onActivityCreated(savedInstanceState);
        // Indicate that this fragment would like to influence the set of
actions in the action bar.
        setHasOptionsMenu(true);
    }

    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup container,
            Bundle savedInstanceState) {
        mDrawerListView = (ListView) inflater.inflate(
                R.layout.fragment_navigation_drawer, container, false);
        mDrawerListView.setOnItemClickListener(new
AdapterView.OnItemClickListener() {
            @Override
            public void onItemClick(AdapterView<?> parent, View view, int
position, long id) {
                selectItem(position);
            }
        });
        mDrawerListView.setAdapter(new ArrayAdapter<String>(
                getActionBar().getThemedContext(),
                android.R.layout.simple_list_item_1,
```

```
                android.R.id.text1,
                new String[]{
                        getString(R.string.title_section1),
                        getString(R.string.title_section2),
                        getString(R.string.title_section3),
                }));
        mDrawerListView.setItemChecked(mCurrentSelectedPosition, true);
        return mDrawerListView;
    }

    public boolean isDrawerOpen() {
        return mDrawerLayout != null &&
mDrawerLayout.isDrawerOpen(mFragmentContainerView);
    }

    /**
     * Users of this fragment must call this method to set up the
navigation drawer interactions.
     *
     * @param fragmentId   The android:id of this fragment in its
activity's layout.
     * @param drawerLayout The DrawerLayout containing this fragment's UI.
     */
    public void setUp(int fragmentId, DrawerLayout drawerLayout) {
        mFragmentContainerView = getActivity().findViewById(fragmentId);
        mDrawerLayout = drawerLayout;

        // set a custom shadow that overlays the main content when the
drawer opens
        mDrawerLayout.setDrawerShadow(R.drawable.drawer_shadow,
GravityCompat.START);
        // set up the drawer's list view with items and click listener

        ActionBar actionBar = getActionBar();
        actionBar.setDisplayHomeAsUpEnabled(true);
        actionBar.setHomeButtonEnabled(true);

        // ActionBarDrawerToggle ties together the the proper interactions
        // between the navigation drawer and the action bar app icon.
        mDrawerToggle = new ActionBarDrawerToggle(
                getActivity(),                    /* host Activity */
                mDrawerLayout,                    /* DrawerLayout object */
                R.drawable.ic_drawer,             /* nav drawer image to
replace 'Up' caret */
                R.string.navigation_drawer_open,  /* "open drawer"
description for accessibility */
                R.string.navigation_drawer_close  /* "close drawer"
description for accessibility */
        ) {
            @Override
            public void onDrawerClosed(View drawerView) {
                super.onDrawerClosed(drawerView);
                if (!isAdded()) {
                    return;
                }

                getActivity().supportInvalidateOptionsMenu(); // calls
onPrepareOptionsMenu()
            }

            @Override
```

```
            public void onDrawerOpened(View drawerView) {
                super.onDrawerOpened(drawerView);
                if (!isAdded()) {
                    return;
                }

                if (!mUserLearnedDrawer) {
                    // The user manually opened the drawer; store this flag
to prevent auto-showing
                    // the navigation drawer automatically in the future.
                    mUserLearnedDrawer = true;
                    SharedPreferences sp = PreferenceManager
                            .getDefaultSharedPreferences(getActivity());
                    sp.edit().putBoolean(PREF_USER_LEARNED_DRAWER,
true).commit();
                }

                getActivity().supportInvalidateOptionsMenu(); // calls
onPrepareOptionsMenu()
            }
        };

        // If the user hasn't 'learned' about the drawer, open it to
introduce them to the drawer,
        // per the navigation drawer design guidelines.
        if (!mUserLearnedDrawer && !mFromSavedInstanceState) {
            mDrawerLayout.openDrawer(mFragmentContainerView);
        }

        // Defer code dependent on restoration of previous instance state.
        mDrawerLayout.post(new Runnable() {
            @Override
            public void run() {
                mDrawerToggle.syncState();
            }
        });

        mDrawerLayout.setDrawerListener(mDrawerToggle);
    }

    private void selectItem(int position) {
        mCurrentSelectedPosition = position;
        if (mDrawerListView != null) {
            mDrawerListView.setItemChecked(position, true);
        }
        if (mDrawerLayout != null) {
            mDrawerLayout.closeDrawer(mFragmentContainerView);
        }
        if (mCallbacks != null) {
            mCallbacks.onNavigationDrawerItemSelected(position);
        }
    }

    @Override
    public void onAttach(Activity activity) {
        super.onAttach(activity);
        try {
            mCallbacks = (NavigationDrawerCallbacks) activity;
        } catch (ClassCastException e) {
            throw new ClassCastException("Activity must implement
NavigationDrawerCallbacks.");
```

```
        }
    }

    @Override
    public void onDetach() {
        super.onDetach();
        mCallbacks = null;
    }

    @Override
    public void onSaveInstanceState(Bundle outState) {
        super.onSaveInstanceState(outState);
        outState.putInt(STATE_SELECTED_POSITION, mCurrentSelectedPosition);
    }

    @Override
    public void onConfigurationChanged(Configuration newConfig) {
        super.onConfigurationChanged(newConfig);
        // Forward the new configuration the drawer toggle component.
        mDrawerToggle.onConfigurationChanged(newConfig);
    }

    @Override
    public void onCreateOptionsMenu(Menu menu, MenuInflater inflater) {
        // If the drawer is open, show the global app actions in the action
bar. See also
        // showGlobalContextActionBar, which controls the top-left area of
the action bar.
        if (mDrawerLayout != null && isDrawerOpen()) {
            inflater.inflate(R.menu.global, menu);
            showGlobalContextActionBar();
        }
        super.onCreateOptionsMenu(menu, inflater);
    }

    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
        if (mDrawerToggle.onOptionsItemSelected(item)) {
            return true;
        }

        if (item.getItemId() == R.id.action_example) {
            Toast.makeText(getActivity(), "Example action.",
Toast.LENGTH_SHORT).show();
            return true;
        }

        return super.onOptionsItemSelected(item);
    }

    /**
     * Per the navigation drawer design guidelines, updates the action bar
to show the global app
     * 'context', rather than just what's in the current screen.
     */
    private void showGlobalContextActionBar() {
        ActionBar actionBar = getActionBar();
        actionBar.setDisplayShowTitleEnabled(true);
        actionBar.setNavigationMode(ActionBar.NAVIGATION_MODE_STANDARD);
        actionBar.setTitle(R.string.app_name);
    }
```

```
    private ActionBar getActionBar() {
        return ((ActionBarActivity) getActivity()).getSupportActionBar();
    }

    /**
     * Callbacks interface that all activities using this fragment must
implement.
     */
    public static interface NavigationDrawerCallbacks {
        /**
         * Called when an item in the navigation drawer is selected.
         */
        void onNavigationDrawerItemSelected(int position);
    }
}
```

MarinApp.java

```
package com.example.helloworld5;

import android.app.Activity;
import android.support.v7.app.ActionBarActivity;
import android.support.v7.app.ActionBar;
import android.support.v4.app.Fragment;
import android.support.v4.app.FragmentManager;
import android.content.Context;
import android.os.Build;
import android.os.Bundle;
import android.view.Gravity;
import android.view.LayoutInflater;
import android.view.Menu;
import android.view.MenuItem;
import android.view.View;
import android.view.ViewGroup;
import android.support.v4.widget.DrawerLayout;
import android.widget.ArrayAdapter;
import android.widget.TextView;

public class MarinApp extends ActionBarActivity implements
        NavigationDrawerFragment.NavigationDrawerCallbacks {

    /**
     * Fragment managing the behaviors, interactions and presentation of
the
     * navigation drawer.
     */
    private NavigationDrawerFragment mNavigationDrawerFragment;

    /**
     * Used to store the last screen title. For use in
     * {@link #restoreActionBar()}.
     */
    private CharSequence mTitle;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.marin_app);
```

```
        mNavigationDrawerFragment = (NavigationDrawerFragment)
getSupportFragmentManager()
                .findFragmentById(R.id.navigation_drawer);
        mTitle = getTitle();

        // Set up the drawer.
        mNavigationDrawerFragment.setUp(R.id.navigation_drawer,
                (DrawerLayout) findViewById(R.id.drawer_layout));
    }

    @Override
    public void onNavigationDrawerItemSelected(int position) {
        // update the main content by replacing fragments
        FragmentManager fragmentManager = getSupportFragmentManager();
        fragmentManager
                .beginTransaction()
                .replace(R.id.container,
                        PlaceholderFragment.newInstance(position +
1)).commit();
    }

    public void onSectionAttached(int number) {
        switch (number) {
        case 1:
            mTitle = getString(R.string.title_section1);
            break;
        case 2:
            mTitle = getString(R.string.title_section2);
            break;
        case 3:
            mTitle = getString(R.string.title_section3);
            break;
        }
    }

    public void restoreActionBar() {
        ActionBar actionBar = getSupportActionBar();
        actionBar.setNavigationMode(ActionBar.NAVIGATION_MODE_STANDARD);
        actionBar.setDisplayShowTitleEnabled(true);
        actionBar.setTitle(mTitle);
    }

    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        if (!mNavigationDrawerFragment.isDrawerOpen()) {
            // Only show items in the action bar relevant to this screen
            // if the drawer is not showing. Otherwise, let the drawer
            // decide what to show in the action bar.
            getMenuInflater().inflate(R.menu.marin_app, menu);
            restoreActionBar();
            return true;
        }
        return super.onCreateOptionsMenu(menu);
    }

    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
        // Handle action bar item clicks here. The action bar will
        // automatically handle clicks on the Home/Up button, so long
        // as you specify a parent activity in AndroidManifest.xml.
        int id = item.getItemId();
```

```
        if (id == R.id.action_settings) {
            return true;
        }
        return super.onOptionsItemSelected(item);
    }

    /**
     * A placeholder fragment containing a simple view.
     */
    public static class PlaceholderFragment extends Fragment {
        /**
         * The fragment argument representing the section number for this
         * fragment.
         */
        private static final String ARG_SECTION_NUMBER = "section_number";

        /**
         * Returns a new instance of this fragment for the given section
number.
         */
        public static PlaceholderFragment newInstance(int sectionNumber) {
            PlaceholderFragment fragment = new PlaceholderFragment();
            Bundle args = new Bundle();
            args.putInt(ARG_SECTION_NUMBER, sectionNumber);
            fragment.setArguments(args);
            return fragment;
        }

        public PlaceholderFragment() {
        }

        @Override
        public View onCreateView(LayoutInflater inflater, ViewGroup
container,
                Bundle savedInstanceState) {
            View rootView = inflater.inflate(R.layout.fragment_marin_app,
                    container, false);

            //// Deklarirana varijabla fragment, koja ce sadrzavati sam
fragment
            Fragment fragment = null;
            //getArguments() dohvaca sekciju meni-a
            switch (getArguments().getInt(
                    ARG_SECTION_NUMBER)){
                case 0:{
                    break;
                }
                case 1:{
                    fragment = new KvizMenu();
                    break;
                }
                case 2:{
                    fragment = new Info();
                    break;
                }
                case 3:{
                    System.exit(0);
                    break;
                }
                default:
                    break;
```

```
            }
            //ako postoji fragment stvori ga
            if (fragment != null){
                //fragment.setRetainInstance(true);
                FragmentManager fragmentManager = getFragmentManager();
                fragmentManager.beginTransaction()
                        .replace(R.id.container, fragment).commit();
            }
            else {
            }
            return rootView;
        }

        @Override
        public void onAttach(Activity activity) {
            super.onAttach(activity);
            ((MarinApp) activity).onSectionAttached(getArguments().getInt(
                    ARG_SECTION_NUMBER));
        }
    }

}
```

KvizMenu.java

```
package com.example.helloworld5;
//Uključivanje svih potrebnih djelova (biblioteka) za izvođenjee aplikacije
import java.util.ArrayList;
import java.util.Collections;
import java.util.List;

import android.app.Activity;
import android.app.FragmentTransaction;
import android.content.Context;
import android.os.Bundle;
import android.support.v4.app.Fragment;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.ArrayAdapter;
import android.widget.AutoCompleteTextView;
import android.widget.Button;
import android.widget.ListView;
import android.widget.RadioButton;
import android.widget.RadioGroup;
import android.widget.RatingBar;
import android.widget.TextView;

public class KvizMenu extends Fragment{
    //Definiranje varijabli
    View v;
    Button start;
    Button results;

    public KvizMenu(){}
    //Dio životnog ciklusa fragmenta koji sadrži glavni dio programa
    @Override

    public View onCreateView(LayoutInflater inflater, ViewGroup container,
            Bundle savedInstanceState) {
```

```
        super.onCreate(savedInstanceState);

        //Definiranje grafičkog izgleda
        View rootView = inflater.inflate(R.layout.kviz_menu, container,
false);
        v = rootView;

        INIT();
        EVENTS();

        return rootView;
    }


    //Dodjeljivanje funkcije objektima
    public void INIT(){
        start = (Button) v.findViewById(R.id.QUIZ_START);
        results = (Button) v.findViewById(R.id.QUIZ_RESULTS);
    }
    //Pritiskom na tipku start pokreće se funkcija START()
    public void EVENTS(){
        start.setOnClickListener(new View.OnClickListener() {

            @Override
            public void onClick(View v) {
                START();
            }
        });
        //Pritiskom na tipku rezultati pokrećese funkcija RESULTS()
        results.setOnClickListener(new View.OnClickListener() {

            @Override
            public void onClick(View v) {
                RESULTS();
            }
        });
    }
    //Stvaramo novi fragment Quiz_start()
    public void START(){
        Fragment fg = new Quiz_start();
        NEW_FRAGMENT(fg);
    }
    //Stvaramo novi fragment Quiz_results()
    public void RESULTS(){
        Fragment fg = new Quiz_results();
        NEW_FRAGMENT(fg);
    }
    //Vrši se transakcija postojećeg fragmenta sa novim
    public void NEW_FRAGMENT(Fragment fg){
        android.support.v4.app.FragmentTransaction fragmentTransaction =
getFragmentManager().beginTransaction();
        fragmentTransaction.replace(R.id.container,fg );

fragmentTransaction.setTransition(FragmentTransaction.TRANSIT_FRAGMENT_OPEN
);
        fragmentTransaction.commit();
    }
    //Klasa Question služi za postavljanje cijelokupnog pitanja sa
odgovorima u jedan objekt
    public static class Question{
            //Definiranje varijabli
```

```
            private int ID;
            private String QUESTION;
            private String OPTA;
            private String OPTB;
            private String OPTC;
            private String ANSWER;
            public Question()
            {
                ID=0;
                QUESTION="";
                OPTA="";
                OPTB="";
                OPTC="";
                ANSWER="";
            }
            public Question(String qUESTION, String oPTA, String oPTB,
String oPTC,
                    String aNSWER) {

                QUESTION = qUESTION;
                OPTA = oPTA;
                OPTB = oPTB;
                OPTC = oPTC;
                ANSWER = aNSWER;
            }

            }
            public String getQUESTION() {
                return QUESTION;
            }
            public String getOPTA() {
                return OPTA;
            }
            public String getOPTB() {
                return OPTB;
            }
            public String getOPTC() {
                return OPTC;
            }
            public String getANSWER() {
                return ANSWER;
            }



    }
    //Klasa User služi za postavljanje cijelokupnog korisnika sa
imenom,prezimenom i rezultatom u jedan objekt
    public static class User{
        String name,surname;
        float result;

        public User(String n,String un,float r){
            this.name = n;
            this.surname = un;
            this.result = r;
        }
        public String getName(){
            return this.name;
        }
        public String getSurname(){
```

```
            return this.surname;
        }
        public float getResult(){
            return this.result;
        }
    }

    public static class Quiz_start extends Fragment{
        //Definiranje varijabli
        Activity a;
        View v;

        int BUNDLE_QUESTION_NUMBER;

        Button next;
        TextView question;
        RadioButton rda, rdb, rdc;
        RadioGroup grp;

        Question currentQ;
        ArrayList<Question> questions;
        Database data;
        Database.App qData;

        int maxQ,count,score;
        String name,surname;

        int idStatico;
        private Bundle savedInstanceState;

        public Quiz_start(){}

        //Dio životnog ciklusa fragmenta koji sadrži glavni dio programa
        @Override
        public View onCreateView(LayoutInflater inflater, ViewGroup
container,
                Bundle savedInstanceState) {
            super.onCreate(savedInstanceState);

            //Definiranje grafičkog izgleda
            View rootView = inflater.inflate(R.layout.kviz, container,
false);

            a = getActivity();
            v = rootView;
             if (savedInstanceState != null) {
                 // Restore last state for checked position.
            BUNDLE_QUESTION_NUMBER =
savedInstanceState.getInt("QUESTION_NUMBER");
             }

            this.INIT();
            this.EVENTS();

            //Funkcija koja pokreće kviz
            this.START();

            return rootView;
        }
        @Override
        public void onAttach(Activity activity) {
```

```
            super.onAttach(activity);

        }
        @Override
        public void onActivityCreated(Bundle savedInstanceState) {
            super.onActivityCreated(savedInstanceState);
        if (savedInstanceState != null) {
                // Restore last state for checked position.
            BUNDLE_QUESTION_NUMBER =
savedInstanceState.getInt("QUESTION_NUMBER");
            }
    }
        @Override
        public void onSaveInstanceState(Bundle outState) {
            super.onSaveInstanceState(outState);
            outState.putInt("QUESTION_NUMBER", 2);
         }
        //Dodjeljivanje funkcije objektima
        public void INIT(){
            next = (Button) v.findViewById(R.id.QUIZ_NEXT);
            question = (TextView) v.findViewById(R.id.QUIZ_QUESTION);

            grp=(RadioGroup) v.findViewById(R.id.QUIZ_RADIO);
            rda = (RadioButton) v.findViewById(R.id.radio0);
            rdb = (RadioButton) v.findViewById(R.id.radio1);
            rdc = (RadioButton) v.findViewById(R.id.radio2);
            //Maksimalni broj pitanja koje odgovaram
            //Trenutno pitanje = -1
            //Rezultat = 0
            maxQ = 10;
            count = -1;
            score = 0;

            data = new Database(a,null,null,1);
            //Spremamo objekt database u varijablu data
            qData = data.new App(a,null,null,1);
        }
        //Brisanje zaslona
        public void CLEAN(){
            question.setText("");
            rda.setText("");
            rdb.setText("");
            rdc.setText("");
        }
        //
        //Prilikom pritiska na tipku dalje pokreće se funckija
CHECK_ANSWER()
        public void EVENTS(){
            next.setOnClickListener(new View.OnClickListener() {

                @Override
                public void onClick(View v) {
                    CHECK_ANSWER();
                }
            });

        }
        //Provjera točnosti rezultata
        public void CHECK_ANSWER(){
            RadioButton answer=(RadioButton)
v.findViewById(grp.getCheckedRadioButtonId());
```

```
            //Ako je točan rezultat povećava rezultat
            if (currentQ.getANSWER().equals(answer.getText())){
                score++;
            }
            else {

            }
            //Zbroj pitanja se povećava
            count++;
            //Brisanje zaslona
            this.CLEAN();
            //Nastavlja se kviz
            this.START();
        }
        //Funkcija koja postavlja pitanje na zaslon
        public void SET_QUESTION(){
            currentQ = questions.get(count);

            question.setText(currentQ.getQUESTION());
            rda.setText(currentQ.getOPTA());
            rdb.setText(currentQ.getOPTB());
            rdc.setText(currentQ.getOPTC());
        }
        //Briše bazu podataka ukoliko već postoji
        public void EMPTY_DATABASE(){
            qData.emptyQuestions();
        }
        //Dohvaća pitanja iz baze
        public void GET_QUESTIONS(){
            questions = qData.getQuestions();
        }
        //Mješa pitanja dohvaćena iz baze, kako nebi bila uvijek istim
redoslijedom
        public void MIX_QUESTIONS(){
            Collections.shuffle(this.questions);
        }
        //Prilikom pokretanja aplikacije priprema se baza podataka sa
pitanjima
        public void FILL_DATABASE(){
            Question q = new Question("Produetak .doc u nazivu datoteke
označava?","Tekstualnu datoteku","Slikovnu datoteku","Audio
datoteku","Tekstualnu datoteku");
            qData.addQuestion(q);

            q = new Question("Što od navedenog ne spada u
Hardware?","Matična ploča,"Windows","Tipkovnica","Windows");
            qData.addQuestion(q);

            q = new Question("Koliko centimetara ima 15 inčni
monitor?","37.1","38.1","39.1","38.1");
            qData.addQuestion(q);

            q = new Question("Koliko piksela ima slika na monitoru čijaa je
rezolucija 800x600?","420000","450000","480000","480000");
            qData.addQuestion(q);

            q = new Question("Koliko tranzistora ima jedan suvremeni
tranzistor?","Oko sto tisuća,"Oko milijun","Oko milijardu","Oko
milijardu");
            qData.addQuestion(q);
```

```
            q = new Question("Koji od navedenih nizova ima 3
bita?","010","101","110","101");
            qData.addQuestion(q);

            q = new Question("Kako računalo kodira slovo
A?","01000001","11000101","10000000","01000001");
            qData.addQuestion(q);

            q = new Question("Koja od navedenih ekstenzija nije
ispravna?",".docx",".xls",".ddoc",".ddoc");
            qData.addQuestion(q);

            q = new Question("Izbaci uljeza!","NPN","PNP","PNN","PNN");
            qData.addQuestion(q);

            q = new Question("Punoupravljivi elektronički ventil
nije?","TRIAC","IGBT","Tranzistor","TRIAC");
            qData.addQuestion(q);

            q = new Question("Nadopuni rečenicu: Gubici u tranzistoru
dijele se na gubitke _________ i gubitke
prekapčanja.","Polarizacije","Vođenja","Oporavljanja","Vođenja");
            qData.addQuestion(q);

            q= new Question("Nadopuni rečenicu: Elekronički sklopovi dijele
se na ________ i digitalne.","Cjelovite","Jeftine","Analogne","Analogne");
            qData.addQuestion(q);

            q= new Question("Android operacijski sustav predstavljen
je?","U Studenom, 2007","U Lipnju, 2007","U Srpnju, 2007","U Studenom,
2007");
            qData.addQuestion(q);

            q= new Question("Android je prvi _______ operacijski
sustav.","Otvoreni","Zatvoreni","Mobilni","Otvoreni");
            qData.addQuestion(q);

            q= new Question("Koji od navedenih materijala najbolje provodi
struju?","Aluminij","Zlik","Zlato","Zlato");
            qData.addQuestion(q);

            q= new Question("Koji od navedenih materijala ne provodi
struju?","Volfram","Konstantan","Porculan","Porculan");
            qData.addQuestion(q);


            q= new Question("Najbolji elektriki vodič
je?","Srebro","Bakar","Zlato","Srebro");
            qData.addQuestion(q);

            q= new Question("_________ je mjerna jedinica za jakost
magnetskog polja.","Henri","Weber","Tesla","Tesla");
            qData.addQuestion(q);
        }
        //Funckija koja se brzine o tijeku kviza
        public void START(){
            //Ako je zbroj pitanja manji od 0
            if (count<0){
                currentQ = new Question;
                //Prazni se baza podataka
                EMPTY_DATABASE();
```

```
                //Puni se baza podataka
                FILL_DATABASE();
                //Dohvaća pitanja
                GET_QUESTIONS();
                //Mješa pitanja
                MIX_QUESTIONS();
                //Povećava se zbroj pitanja
                count++;
            }
            //Ako je zbroj manji od maximalnog broja pitanja
            if (count<maxQ){
            //Postavlja novo pitanje
            SET_QUESTION();
            }//Ako nije ništa od navedenog
            else{
                //Brisanje pitanja
                CLEAN();
                //Stvaranje novog fragmenta
                Fragment fg = new Quiz_result();
                NEW_FRAGMENT(fg);
            }

        }
        //Stvaranje novog fragmenta nakon odgovora na zadnje pitanje
        //za ispis rezultata te traženje od korisnika da upiše ime i
prezime za spremanje rezultata
        public void NEW_FRAGMENT(Fragment fg){
            //Prkeo bundle-a šaljemo rezultat i ostale informacije o
korisniku i igri
            Bundle bundle = new Bundle();
            bundle.putInt("score", score);
            bundle.putInt("maxQ", maxQ);
            bundle.putString("name", name);
            bundle.putString("surname", surname);
            ////Vrši se transakcija postojećeg fragmenta sa novim
            fg.setArguments(bundle);
            android.support.v4.app.FragmentTransaction fragmentTransaction
= getFragmentManager().beginTransaction();
            fragmentTransaction.replace(R.id.container,fg );

fragmentTransaction.setTransition(FragmentTransaction.TRANSIT_FRAGMENT_OPEN
);
            fragmentTransaction.commit();
        }
    }

    //Fragment za ispis svih rezultata iz baze poredanih prema najboljim
korisnicima
    public static class Quiz_results extends Fragment{
        //Definiranje varijabli
        View v;
        Activity a;

        ListView results;
        Button  menu;

        ArrayList<User> users;
        List<String> results_array = new ArrayList<String>();

        Database data;
        Database.App rData;
```

```
        @Override
        public View onCreateView(LayoutInflater inflater, ViewGroup
container,
                Bundle savedInstanceState) {
            super.onCreate(savedInstanceState);
            //Definiranje grafičkog izgleda
            View rootView = inflater.inflate(R.layout.svi_rezultati,
container, false);
            a = getActivity();
            v = rootView;

            INIT();
            EVENTS();
            GET_RESULTS();
            SET_RESULTS();

            return rootView;
        }
        @Override
        public void onAttach(Activity activity) {
            super.onAttach(activity);

        }
        //Dodjeljivanje funkcije objektima
        public void INIT(){
            results = (ListView) v.findViewById(R.id.QUIZ_RESULTS);
            menu = (Button) v.findViewById(R.id.QUIZ_MENU_);


            data = new Database(a,null,null,1);
            rData = data.new App(a,null,null,1);
        }
        //Pritiskom na tipku spremi rezultat, vraćamo se na početnifragment
        public void EVENTS(){
            menu.setOnClickListener(new View.OnClickListener() {

                @Override
                public void onClick(View v) {
                    Fragment fg = new KvizMenu();
                    NEW_FRAGMENT(fg);
                }
            });
        }
        //Dohvaća rezultate iz baze podataka poredane po najboljim igračima
        public void GET_RESULTS(){
            users = rData.getResults();
        }
        //Ispisuje na zaslon rezultate
        public void SET_RESULTS(){
            String resultFormat = "";
            //Maksimalan broj rezultata ograničen je na 10
            for (int i=0;i<users.size();i++){

                resultFormat = i+1 + ") " +users.get(i).getName() + " " +
users.get(i).getSurname() + ": " + users.get(i).getResult();
                results_array.add(resultFormat);
            }

            ArrayAdapter<String> content_adapter = new
ArrayAdapter<String>(
```

```
                    a,
                    android.R.layout.simple_list_item_1,
                    results_array );

            results.setAdapter(content_adapter);
        }
        //Vrši se transakcija postojećeg fragmenta sa novim
        public void NEW_FRAGMENT(Fragment fg){
            android.support.v4.app.FragmentTransaction fragmentTransaction
= getFragmentManager().beginTransaction();
            fragmentTransaction.replace(R.id.container,fg );

fragmentTransaction.setTransition(FragmentTransaction.TRANSIT_FRAGMENT_OPEN
);
            fragmentTransaction.commit();
        }
    }
    //Fragmenta za ispis rezultata nakon završetka kviza
    public static class Quiz_result extends Fragment{
        //Definiranje varijabli
        View v;
        Activity a;

        int score,maxQ;
        String text,name,surname;
        float rating;

        TextView result_text;
        RatingBar result_bar;
        Button save;
        AutoCompleteTextView input_name,input_surname;

        Database data;
        Database.App rData;

        public Quiz_result(){
        }
        @Override
        public View onCreateView(LayoutInflater inflater, ViewGroup
container,
                Bundle savedInstanceState) {
            super.onCreate(savedInstanceState);

            //Definiranje grafičkog izgleda
            View rootView = inflater.inflate(R.layout.kviz_rezultati,
container, false);
            a = getActivity();
            v = rootView;
            INIT();
            EVENTS();
            SHOW_RESULT();
            return rootView;
        }
        @Override
        public void onAttach(Activity activity) {
            super.onAttach(activity);

        }
        //Dodjeljivanje funkcije objektima
        public void INIT(){
            score = this.getArguments().getInt("score");
```

```
            maxQ = this.getArguments().getInt("maxQ");
            name = this.getArguments().getString("name");
            surname = this.getArguments().getString("surname");

            rating = ((float) score/(float) maxQ) * 5;

            result_text = (TextView) v.findViewById(R.id.QUIZ_RESULT);
            result_bar = (RatingBar) v.findViewById(R.id.QUIZ_RATING);
            save = (Button) v.findViewById(R.id.QUIZ_SAVE);
            input_name = (AutoCompleteTextView)
v.findViewById(R.id.QUIZ_NAME);
            input_surname = (AutoCompleteTextView)
v.findViewById(R.id.QUIZ_SURNAME);

            result_bar.setNumStars(5);
            result_bar.setStepSize(0.01f);

            data = new Database(a,null,null,1);
            rData = data.new App(a,null,null,1);

        }
        //Pritiskom na tipku spremi rezultat, pregledavamo da li je upisano
ime i prezime te ako je rezultat se sprema u bazu
        public void EVENTS(){
            save.setOnClickListener(new View.OnClickListener() {

                @Override
                public void onClick(View v) {
                    CHECK_INPUTS();

                }
            });
        }
        //Ispisuje rezultat na zaslon sa rating bar-om
        public void SHOW_RESULT(){
            text = "Great result " + score + "/" + maxQ + "! Your score is:
"+ rating;
            result_text.setText(text);

            result_bar.setRating(rating);
        }
        //Provjerava formu za upis imena i prezimena (Ime i prezime mora
biti ispunjeno ukoliko želimo spremiti rezultat)
        public void CHECK_INPUTS(){
            if (input_name.getText().toString() != null &&
input_name.getText().toString().length() != 0){
                if (input_surname.getText().toString() != null &&
input_surname.getText().toString().length() != 0){
                    SAVE_RESULT();
                    Fragment fg = new KvizMenu();
                    NEW_FRAGMENT(fg);
                }
            }
        }
        //Spremanje rezultata u bazu podataka
        public void SAVE_RESULT(){
            KvizMenu.User user = new User(input_name.getText().toString(),
input_surname.getText().toString(), rating);
            rData.addResult(user);
        }
        //Vrši se transakcija postojećeg fragmenta sa novim
```

```
        public void NEW_FRAGMENT(Fragment fg){
            android.support.v4.app.FragmentTransaction fragmentTransaction
= getFragmentManager().beginTransaction();
            fragmentTransaction.replace(R.id.container,fg );

fragmentTransaction.setTransition(FragmentTransaction.TRANSIT_FRAGMENT_OPEN
);
            fragmentTransaction.commit();
        }
    }

}
```

Info.java

```
package com.example.helloworld5;

import android.app.Activity;
import android.os.Bundle;
import android.support.v4.app.Fragment;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;

public class Info extends Fragment{
    public Info(){}

    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup container,
            Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);


        View rootView = inflater.inflate(R.layout.info, container, false);

        return rootView;
    }
    @Override
    public void onAttach(Activity activity) {
        super.onAttach(activity);

    }
}
```

Database.java

```
package com.example.helloworld5;
import java.util.ArrayList;

import com.example.helloworld5.KvizMenu.Question;
import com.example.helloworld5.KvizMenu.User;

import android.content.ContentValues;
import android.content.Context;
import android.database.Cursor;
import android.database.sqlite.SQLiteDatabase;
import android.database.sqlite.SQLiteDatabase.CursorFactory;
import android.database.sqlite.SQLiteOpenHelper;
```

```
//Klasa koja se brine za pristup bazi podataka, popunjavanje baze te
dohvaćanje iz baze
public class Database extends SQLiteOpenHelper {

    private static final int DATABASE_VERSION = 1;
    private static final String DATABASE_NAME = "kviz.db";

    public static final String SEARCH_COLUMN_ID = "id";
    public static final String SEARCH_COLUMN_TEXT = "text";
    public static final String SEARCH_COLUMN_COUNTER = "counter";

    public static final String TABLE_QUIZ = "quiz";
    public static final String QUIZ_COLUMN_ID = "id";
    public static final String QUIZ_COLUMN_QUESTION = "question";
    public static final String QUIZ_COLUMN_A = "a";
    public static final String QUIZ_COLUMN_B = "b";
    public static final String QUIZ_COLUMN_C = "c";
    public static final String QUIZ_COLUMN_ANSWER = "answer";

    public static final String TABLE_RESULT = "result";
    public static final String RESULT_COLUMN_ID = "id";
    public static final String RESULT_COLUMN_NAME = "name";
    public static final String RESULT_COLUMN_SURNAME = "surname";
    public static final String RESULT_COLUMN_RESULT = "result";

    SQLiteDatabase CURR_DATABASE;

    public Database(Context context, String name,
            CursorFactory factory, int version) {
        super(context, DATABASE_NAME, factory, DATABASE_VERSION);
    }

    @Override
    public void onCreate(SQLiteDatabase db) {



    }

    @Override
    public void onUpgrade(SQLiteDatabase db, int oldVersion, int
newVersion) {
        db.execSQL("DROP TABLE IF EXISTS " + TABLE_QUIZ);
          onCreate(db);
    }

    //Klasa unutar klase Database koja se brine za dohvaćanje, spremanje i
ispis rezultata kviza
    public class App extends SQLiteOpenHelper{

        public App(Context context, String name, CursorFactory factory,
                int version) {
            super(context, DATABASE_NAME, factory, DATABASE_VERSION);
            // TODO Auto-generated constructor stub
        }
        @Override
        public void onCreate(SQLiteDatabase db) {
            // TODO Auto-generated method stub
            String CREATE_QUIZ_TABLE ="CREATE TABLE "+ TABLE_QUIZ +
                    "(" + QUIZ_COLUMN_ID +
```

```
                    " INTEGER PRIMARY KEY," + QUIZ_COLUMN_QUESTION +
                    " TEXT," + QUIZ_COLUMN_A +
                    " TEXT," + QUIZ_COLUMN_B +
                    " TEXT," + QUIZ_COLUMN_C +
                    " TEXT," + QUIZ_COLUMN_ANSWER +
                    " TEXT)";
        db.execSQL(CREATE_QUIZ_TABLE);

        String CREATE_RESULT_TABLE = "CREATE TABLE " + TABLE_RESULT +
                "(" + RESULT_COLUMN_ID +
                " INTEGER PRIMARY KEY," + RESULT_COLUMN_NAME +
                " TEXT, " + RESULT_COLUMN_SURNAME +
                " TEXT," + RESULT_COLUMN_RESULT +
                " REAL)";
        db.execSQL(CREATE_RESULT_TABLE);

        }
        @Override
        public void onUpgrade(SQLiteDatabase db, int oldVersion, int
newVersion) {
            db.execSQL("DROP TABLE IF EXISTS " + TABLE_QUIZ);
              onCreate(db);

        }
        //Dodaje pitanje u bazu
        public void addQuestion(Question question) {

            ContentValues values = new ContentValues();

            values.put(QUIZ_COLUMN_QUESTION, question.getQUESTION());
            values.put(QUIZ_COLUMN_A, question.getOPTA());
            values.put(QUIZ_COLUMN_B, question.getOPTB());
            values.put(QUIZ_COLUMN_C, question.getOPTC());
            values.put(QUIZ_COLUMN_ANSWER, question.getANSWER());

            SQLiteDatabase db = this.getWritableDatabase();

            db.insert(TABLE_QUIZ, null, values);
            db.close();
    }
        //Prazni tablicu sa pitanjima
        public void emptyQuestions(){
            SQLiteDatabase db = this.getWritableDatabase();
            try
            {
                db.delete(TABLE_QUIZ, QUIZ_COLUMN_QUESTION+ " > 0", null);
            }
            catch(Exception e)
            {
                e.printStackTrace();
            }
            finally
            {
                db.close();
            }
        }
        //Dohvaća max 30 pitanja iz baze
        public ArrayList<Question> getQuestions() {
            String query = "Select * FROM " + TABLE_QUIZ + " order by " +
QUIZ_COLUMN_QUESTION + " DESC limit 30";
```

```
            ArrayList<Question> q = new ArrayList<Question>();
            int i=-1;
            SQLiteDatabase db = this.getWritableDatabase();

            Cursor cursor = db.rawQuery(query, null);
            if(cursor!=null && cursor.getCount()>0){
            while (cursor.moveToNext()) {
                if(cursor!=null && cursor.getCount()>0){
                i++;

                q.add(new
Question(cursor.getString(1),cursor.getString(2),cursor.getString(3),cursor
.getString(4),cursor.getString(5)));
                }
            }
            }
                db.close();
            return q;
        }
        //Dodaje još jedan rezultat u bazu
        public void addResult(User user){
            ContentValues values = new ContentValues();

            values.put(RESULT_COLUMN_NAME, user.getName());
            values.put(RESULT_COLUMN_SURNAME, user.getSurname());
            values.put(RESULT_COLUMN_RESULT, user.getResult());

            SQLiteDatabase db = this.getWritableDatabase();

            db.insert(TABLE_RESULT, null, values);
            db.close();

        }
        //Dohvaća max 10 rezultata poredanih prema najboljim igračima
        public ArrayList<User> getResults() {
            String query = "Select * FROM " + TABLE_RESULT + " ORDER BY " +
RESULT_COLUMN_RESULT + " DESC limit 10";

            ArrayList<User> r = new ArrayList<User>();
            SQLiteDatabase db = this.getWritableDatabase();

            Cursor cursor = db.rawQuery(query, null);
            if(cursor!=null && cursor.getCount()>0){
            while (cursor.moveToNext()) {
                if(cursor!=null && cursor.getCount()>0){

                r.add(new
User(cursor.getString(1),cursor.getString(2),cursor.getFloat(3)));
                }
            }
            }
                db.close();
            return r;
        }


    }
```

PRILOG B

fragment_navigation_drawer.xml

```
<ListView xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="#cccc"
    android:choiceMode="singleChoice"
    android:divider="@android:color/transparent"
    android:dividerHeight="0dp"
    tools:context="com.example.helloworld5.NavigationDrawerFragment" />
```

fragment_marin_app.xml

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    tools:context="com.example.helloworld5.MarinApp$PlaceholderFragment" >

    <TextView
        android:id="@+id/section_label"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content" />

</RelativeLayout>
```

marin_app.xml

```
<!-- A DrawerLayout is intended to be used as the top-level content view
using match_parent for both width and height to consume the full space
available. -->
<android.support.v4.widget.DrawerLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/drawer_layout"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context="com.example.helloworld5.MarinApp" >

    <!--
         As the main content view, the view below consumes the entire
         space available using match_parent in both dimensions.
    -->

    <FrameLayout
        android:id="@+id/container"
        android:layout_width="match_parent"
        android:layout_height="match_parent" />

    <!--
```

```
        android:layout_gravity="start" tells DrawerLayout to treat
        this as a sliding drawer on the left side for left-to-right
        languages and on the right side for right-to-left languages.
        If you're not building against API 17 or higher, use
        android:layout_gravity="left" instead.
    -->
    <!--
        The drawer is given a fixed width in dp and extends the full
height of
        the container.
    -->

    <fragment
        android:id="@+id/navigation_drawer"
        android:name="com.example.helloworld5.NavigationDrawerFragment"
        android:layout_width="@dimen/navigation_drawer_width"
        android:layout_height="match_parent"
        android:layout_gravity="start" />

</android.support.v4.widget.DrawerLayout>
```

kviz_menu.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical" >


    <Button
                android:id="@+id/QUIZ_START"
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:text="@string/start"
                 />
    <Button
                android:id="@+id/QUIZ_RESULTS"
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:text="@string/results"
                />

</LinearLayout>
```

kviz.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical" >
 <TextView
            android:id="@+id/QUIZ_QUESTION"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:text="Large Text"
            android:textAppearance="?android:attr/textAppearanceLarge" />
```

```
        <RadioGroup
            android:id="@+id/QUIZ_RADIO"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_weight="0.04" >
            <RadioButton
                android:id="@+id/radio0"
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:checked="true"
                android:text="RadioButton" />
            <RadioButton
                android:id="@+id/radio1"
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:text="RadioButton" />
            <RadioButton
                android:id="@+id/radio2"
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:text="RadioButton" />
        </RadioGroup>
        <Button
                android:id="@+id/QUIZ_NEXT"
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:text="@string/str_next" />
</LinearLayout>
```

kviz_rezultati.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    android:background="#333" >

    <TextView
            android:id="@+id/QUIZ_RESULT"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:text="Čestitamo"
            android:textColor="#fff"
            android:textAppearance="?android:attr/textAppearanceLarge" />
    <RatingBar
            android:id="@+id/QUIZ_RATING"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:numStars="5"
            android:stepSize="1.0"
            android:rating="0.0"/>
       <LinearLayout
         android:orientation="horizontal"
         android:layout_width="match_parent"
        android:layout_height="50dip"
         android:background="#fff">


    <RelativeLayout
```

```
            android:layout_width="0dip"
            android:layout_height="fill_parent"
            android:layout_weight="10">

           <AutoCompleteTextView
        android:id="@+id/QUIZ_NAME"
        android:layout_width="match_parent"
            android:layout_height="match_parent"
        android:ems="10"
        android:hint="Tvoje ime"
        android:background="#00000000"
        android:paddingLeft="20dip"
        android:textColor="#00BFFF"
        android:inputType="text">

        <requestFocus />
</AutoCompleteTextView>
    </RelativeLayout>
    <RelativeLayout
            android:layout_width="0dip"
            android:layout_height="fill_parent"
            android:layout_weight="10"
            android:background="@drawable/border_left">
           <AutoCompleteTextView
        android:id="@+id/QUIZ_SURNAME"
        android:layout_width="match_parent"
            android:layout_height="match_parent"
        android:ems="10"
        android:hint="Tvoje prezime"
        android:background="#00000000"
        android:paddingLeft="20dip"
        android:textColor="#00BFFF"
        android:inputType="text"></AutoCompleteTextView>

    </RelativeLayout>

 </LinearLayout>
 <LinearLayout
         android:orientation="horizontal"
         android:layout_width="match_parent"
        android:layout_height="fill_parent"
         android:background="#333">
<Button
        android:id="@+id/QUIZ_SAVE"
        android:layout_width="match_parent"
            android:layout_height="50dip"
        android:text="Spremi rezultat"
        />

</LinearLayout>
</LinearLayout>
```

svi_rezultati.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical" >
```

```
    <Button
                android:id="@+id/QUIZ_MENU_"
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:text="Natrag"
    />
    <ListView
        android:id="@+id/QUIZ_RESULTS"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:entries="@array/search_array"

    />

</LinearLayout>
```

info.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical" >
 <TextView
    android:layout_width="fill_parent"
android:layout_height="wrap_content"
android:text="Kviz je izrađen u svrhu diplomskoga rada!"
android:textColor="#00BFFF"
android:textSize="20sp"
android:paddingBottom="20sp"/>
        <TextView
    android:layout_width="fill_parent"
android:layout_height="wrap_content"
android:text="Kolegij: Operacijski sustavi"
android:textColor="#00BFFF"
android:textSize="20sp"
android:padding="5sp"/>
<TextView
    android:layout_width="fill_parent"
android:layout_height="wrap_content"
android:text="Mentor: dr. sc. Božidar Kovačić "
android:textColor="#00BFFF"
android:textSize="20sp"
android:padding="5sp"/>
<TextView
    android:layout_width="fill_parent"
android:layout_height="wrap_content"
android:text="Student: Marin Marinović"
android:textColor="#00BFFF"
android:textSize="20sp"
android:padding="5sp"/>
<TextView
    android:layout_width="fill_parent"
android:layout_height="wrap_content"
android:text="Studijski smjer: EITP"
android:textColor="#00BFFF"
android:textSize="20sp"
android:padding="5sp"/>
  <ImageView
        android:id="@+id/main_logo_bar"
```

```
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_centerVertical="true"
        android:layout_marginLeft="5dp"
        android:src="@drawable/pfri" />
</LinearLayout>
```

string.xml

```
<?xml version="1.0" encoding="utf-8"?>
<resources>

    <string name="app_name">Kviz</string>
    <string name="title_section1">Kviz</string>
    <string name="title_section2">Info</string>
    <string name="title_section3">Izadi</string>
    <string name="navigation_drawer_open">Open navigation drawer</string>
    <string name="navigation_drawer_close">Close navigation drawer</string>
    <string name="action_example"></string>
    <string name="action_settings"></string>
    <string name="start">Započni kviz</string>
    <string name="results">Rezultati</string>
    <string name="str_next">Dalje</string>
    <string name="menu">Meni</string>
<string-array name="search_array">
    </string-array>
</resources>
```