Razvoj mobilne RDBMS aplikacije za primjenu u industriji

Šalamon, Lovro

Sveučilište
Sjever

Završni rad br. 477/EL/2020

Razvoj mobilne RDBMS aplikacije za primjenu u industriji

Lovro Šalamon, 1454/336

Varaždin, listopad 2020. godine

Sveučilište Sjever

Odjel za Elektrotehniku

Završni rad br. 477/EL/2020

Razvoj mobilne RDBMS aplikacije za primjenu u industriji

Student

Lovro Šalamon, 1454/336

Mentor

doc. dr. sc. Ladislav Havaš, dipl. ing. el.

Varaždin, listopad 2020. godine

Sveučilište Sjever
Sveučilišni centar Varaždin
104. brigade 3, HR-42000 Varaždin

Prijava završnog rada

Definiranje teme završnog rada i povjerenstva

ODJEL: Odjel za elektrotehniku

STUDIJ: preddiplomski stru ni studij Elektrotehnika

PRISTUPNIK: Lovro Šalamon | MATIČNI BROJ: 1454/336

DATUM: 24.09.2020 | KOLEGIJ: Baze podataka i SQL

NASLOV RADA: Razvoj mobilne RDBMS aplikacije za primjenu u industriji

NASLOV RADA NA ENGL. JEZIKU: Development of a mobile RDBMS application for industrial application

MENTOR: doc. dr. sc. Ladislav Havaš, dipl.ing.el. | ZVANJE: docent

ČLANOVI POVJERENSTVA:

1. doc. dr. sc. Dunja Srpak, dipl.ing.el.
2. mr. sc. Vladimir Šac, dipl. ing. el., pred.
3. doc. dr. sc. Ladislav Havaš, dipl.ing.el.
4. mr. sc. Ivan Šumiga, dipl. ing. el. v. pred. - pri uvni lan
5.

Zadatak završnog rada

BROJ: 477/EL/2020

OPIS:

U završnom radu je potrebno teorijski obraditi relacijski model podataka i objasniti konceptualni dijagram razvijanog mobilnog Android RDBMS sustava. U radu je potrebno ukratko opisati razvojno okruženje "Android studio", SQLite relacijsku bazu podataka, verzije Android operativnog sustava te ostale alate korištene kod izrade aplikacije.
Cilja rada je izraditi mobilnu RDBMS aplikaciju koja se može koristiti na Android operativnom sustavu i pokazati njenu primjenu u industrijskom okruženju. Sukladno navedenom, u tekstualnom dijelu završnog rada treba kreirati odgovaraju i sadržaj te napisati cjeloviti tekst završnog rada.

ZADATAK URUČEN: 28.09.2020 | POTPIS MENTORA:

Predgovor

Zahvaljujem se svim profesorima koji su mi omogućili ugodno studiranje i prenijeli mi potrebno znanje koje će mi pomoći u mom daljnjem usavršavanju.

Također, posebno se moram zahvaliti mentoru doc. dr. sc. Ladislavu Havašu što me znao usmjeriti, informirati, ispraviti i pomoći tijekom izrade završnog rada.

Zahvala ide i obitelji, prijateljima i kolegama koji su uvijek bili prisutni i spremni pomoći tijekom studiranja.

Sažetak

U završnom radu prikazan je razvoj i primjena relacijske baze podataka za Android sustav u industrijskom okruženju. U teorijskom dijelu opisuje se što je to relacijska baza podataka i sustav za upravljanje istom (engl. *Relational Database Management System (RDBMS)*). U radu je također analizirana korištena relacijska baza – SQLite i njezine značajke. Zatim su navedeni programi i alati koji su se koristili prilikom izrade mobilne aplikacije te postupak instalacije istih.

Kreiran je Android projekt; klasa za upis podataka u bazu, dohvat (ispis) podataka, ažuriranje podataka i brisanje podataka iz baze te dizajn aplikacije. Svrha aplikacije je upisivanje rezultata dijagnostičkih ispitivanja električnih strojeva u kreiranu mobilnu bazu podataka i prikaz generiranih rezultata iz CSV (engl. *Comma-separated values)* datoteke. Dakle, ona daje mogućnost upisa serijskog broja stroja, vrijednosti struje i vrijednosti napona. U razvijenoj aplikaciji isto tako postoji mogućnost ažuriranja podataka pomoću identifikacijskog broja (ID) koji se dodjeljuje prilikom svakog upisa. Ukoliko je iz nekog razloga podatke potrebno ukloniti, ta opcija je isto tako dostupna. Naravno, uvijek postoji prostor za nadogradnju. U ovom slučaju nadogradnju je moguće provesti na način da se doda mogućnost upisa drugih parametara (npr. brzina vrtnje motora), da se implementira opcija izlistavanja podataka na temelju jednog ili više parametara, te mogućnost spremanja upisanih rezultata u CSV datoteku.

Summary

In this project is shown development and application of Relational Database for Android operating system in an industrial environment. In a theoretical part is described what Relational Database and system for managing it (Relational Database Management System) are. In the project is also analized the Relational Database that has been used – SQLite and its features. Then the programs and tools that were used during the making of the mobile application are listed and is also shown the procedure for installing them.

An Android project is created; class for inserting data in the database, data retrieval, updating data and deleting data from the databse and design of the application. Purpose of the application is inserting the results of diagnostic tests from electrical machines in the created mobile database and also displaying the generated results from CSV (Comma-separated values) files. Therefore, the application gives the possibility of inserting serial number of the machine, current value and voltage value. In the developed application there is also the option to update the data using the identification number (ID) which is assigned at each insertion. If, from some reason, there is a need to delete the data, that option is also available. Of course, there is always a room for improvement. In this particular case an improvement can be carried out by adding the posibility for inserting other parameters (e.g. engine rotation speed), by implementing an option for listing the data out based on one or several parameters, and adding an option to save the inserted results in the CSV file.

Sadržaj

1. Uvod

Za početak je potrebno reći kako ljudska potreba za pametnim telefonima, aplikacijama, vođenjem bilješki i spremanjem podataka na pametnim telefonima raste iz dana u dan. Jednostavnije je i efikasnije u laboratoriju ili pogonu obaviti mjerenje i zapisati rezultate direktno na naš pametni telefon, nego li zapisivati rezultate na papir, pa zatim prepisivati na računalo. To ponajviše vrijedi za one koji su stalno u pokretu i obavljaju svoj posao na terenu. Stoga, reći ću u nekoliko rečenica što je to Android sustav i što je to aplikacija.

Android sustav je operativni sustav koji se koristi na mobilnim uređajima (pametnim telefonima, tabletima). Njegov zadatak je pružati funkcije kakve otprilike imaju računala tj. da rad koji izvodimo na računalu prilagodimo i na mobilnim aplikacijama. On je otvorenog koda što omogućava razvoj mnoštva aplikacija uz često besplatnu podršku zajednice. Razvojni programer svoju aplikaciju može i naplaćivati uz plaćanje naknade za uslugu distribucijskom partneru.

Aplikacija je računalni program koji korisniku omogućava izvršavanje jednog ili više zadataka. U Android sustavu korisnik može svojim potrebama prilagoditi koje će aplikacije koristiti.

Praktični dio završnog rada je Android aplikacija čiju funkciju ću temeljito obrazložiti u nastavku.

U prvom dijelu „uvod“ dano je nekoliko informacija o čemu će se kroz ovaj završni rad govoriti. Ukratko su objašnjeni pojmovi „Android“ i „aplikacija“ za što bolji i jasniji pristup realizaciji rada.

U drugom dijelu „Operativni sustav Android“ pobliže je prikazana tema Android sustava. Taj dio se sastoji od dva podnaslova. Jedan obuhvaća verzije Androida, a drugi govori o tome kako je Android sustav nastao.

U trećem dijelu „Relacijska baza podataka“ je obrađena tema relacijske baze podataka. Također je stavljen fokus na korištenu SQLite relacijsku bazu podataka; gdje se koristi, njezine prednosti i nedostaci.

U dijelu „Alati korišteni kod izrade aplikacije“ govori se o programu pomoću kojega se kodira i razvija aplikacija - Android Studio. Isto tako prikazan je Android emulator na kojem prikazujemo i testiramo našu aplikaciju.

U petom dijelu „Android aplikacija“ fokus je stavljen na izrađenu aplikaciju. Detaljno je opisano sve vezano za aplikaciju; od kreiranja projekta do objašnjenja svakog dijela koda. Također je prikazan način dobivanja grafičkog sučelja.

U pretposljednjem dijelu „Primjer rada aplikacije“ aplikacija je pokrenuta na uređaju Huawei P20 i prikazano je kako ona funkcionira.

U zaključku je dan osvrt na cjelokupni rad.

2. Operativni sustav Android

Android operativni sustav je stvoren od istoimene tvrtke koja je kasnije kupljena od strane Google-a. Google je kompanija koja se bazira na proizvodnji i prodaji svojih uređaja i mrežnih usluga (npr. Google disk). Bazira se na prilagođenoj verziji Linuxa. Linux je operativni sustav otvorenog koda koji je osmislio Linus Benedict Torvalds na način da je krenuo učiti kako programirati Intel x86 linije mikroprocesora. Ono što se često naziva „Linux“ je zapravo ime za jezgru računalnog operativnog sustava sličnog Unixu [1]. Android je dostupan na telefonima (različitih proizvođača), tablet računalima, pametnim televizorima, pametnim satovima, pa čak i automobilima.

Slika 1: Prikaz Android sustava na različitim uređajima [2]

Prema stanju u srpnju 2019. godine, Android ima više od 2.5 milijarde aktivnih korisnika [3]. Trenutno ima oko 2.96 milijuna razvijenih aplikacija za Android koje dolaze od strane Google-a, ali i od strane slobodnog razvoja aplikacija za Android prema stanju u lipnju 2020. godine [4]. Prilikom instalacije aplikacije na Android uređaj dobiva se aplikacija napisana za Android sustav. Međutim Android aplikaciju je moguće pokrenuti i na Windows operativnom sustavu, ali tada je potreban softver „Android emulator“ koji stvara simulaciju i pokreće aplikaciju. Mnoštvo aktivnih programera svakodnevno radi na sustavu, njegovoj modernizaciji i poboljšanju.

Karakteristike Google Androida su [5]:

- otvorenost – izvorni kod je dostupan javnosti, pa tako bilo tko može raditi na nadogradnji operativnog sustava, a programeri aplikacija zbog toga mogu dodati dodatne značajke svojim aplikacijama,
- skladište – koristi se SQLite relacijska baza podataka za svrhu skladištenja podataka,
- medijska podrška – sadrži podršku za određene audio/video formate (npr. MPEG-4, JPEG, PNG, GIF, MP3, WAV…),

- podrška za streaming medije – RTP/RTPS streaming, HTML progresivno preuzimanje, Adobe Flash streaming (RTMP), HTTP Dynamic Streaming za slanje audio i video datoteka,
- multitouch,
- web pretraživač,
- video poziv – ne podržava lokalne video pozive, ali je moguće pomoću drugih aplikacija
- višezadaćnost – nama poznati „multitasking“ koji je moguć zbog posebnog upravljanja dodavanjem memorije
- pristupačnost – ugrađen „text to speech“ za osobe s problemom vida ili sljepoćom
- glasovne značajke – glas aktivira navigaciju, poziv, pisanje…
- vanjska pohrana – postoje utori za MicroSD kartice s dodatnom vanjskom memorijom

2.1. Verzije Android sustava

Kao što smo već rekli Android 1.0 na tržištu se nametnuo krajem 2008. godine. Neke od značajki koje su ga tada krasile bile su: Gmail, Google Maps, Youtube, trgovina aplikacija, web preglednik, WiFi, Bluetooth itd [5].

Krajem listopada 2009. godine izdan je Android 2.0. Dodana je mogućnost sinkronizacije kontakata preko višestrukih računa, uvedene su mogućnosti automatskog brisanja SMS i MMS poruka ukoliko dođe do memorijskog ograničenja [5].

Početkom 2011. godine javlja se Android 3.0 koji je namijenjen samo za tablet uređaje. To je ujedno i razlog zašto nije bio dostupan izvorni kod te inačice [5].

Krajem 2011. godine pojavljuje se Android 4.0 koji je dostupan mobilnim i tablet uređajima. Tom inačicom postignuta je bolja integracija s društvenim mrežama, poboljšanja kamere i galerije, podrška za NFC… [5]

Android 5.0 predstavljen je 25. lipnja 2014. godine, a glavna odlika je napravljeni redizajn sučelja. Uz to što je promjena uglavnom vizualna, to je prva inačica koja podržava 64-bitnu arhitekturu procesa i aplikacija [5].

5. listopada 2015. godine objavljen je Android 6.0 koji dolazi sa značajkom „drijemanja“ koja pokušava održati sustav u stanju spavanja ako je zaslon isključen određeni period vremena, pa je time produljen vijek trajanja baterije. Također se javljaju neke manje promjene kao što je uključivanje fotoaparata duplim pritiskom tipke za otključavanje i sl. [5] .

Kod Androida 7.0, koji je objavljen 22. kolovoza 2016. godine, dodane su nove funkcije kao što je mogućnost prebacivanja aplikacija dvostrukim pritiskom na tipku za pregled, tipka „obriši sve“, opcija „brze postavke“. Dodan je i novi set emotikona i mogućnost slanja GIF-ova sa zadane tipkovnice [5].

21. kolovoza sljedeće godine predstavljen je Android 8.0 sa modularnom arhitekturom koja olakšava i ubrzava isporuku ažuriranja sustava korisnicima Android uređaja. Također, redizajnirane su brze postavke, ikone su prilagodljive, omogućena je odgoda obavijesti te automatska izmjena svijetle i tamne teme [5].

Kod Androida 9.0 koji se pojavio 7. ožujka 2018. poboljšane su performanse i dodane su nove sigurnosne značajke. Novi sustav upravljanja baterijom, prečaci na radnje unutar aplikacija, novi sustav navigacije sučeljem samo su neke od novosti Androida 9.0 [5] .

Stabilna verzija nove inačice Androida 10.0 se pojavila 3.9.2019. gdje je dodano mnoštvo novih značajki kao što je podešavanje pozadinskog zamagljivanja nakon slikanja, podrška za preklopne telefone, podrška za razne audio i video formate… [5]

Android 11.0 je posljednja i glavna inačica operativnog sustava Android.

Tablica 1: Verzije Android sustava

Inačica	Kodno ime	Razina API-ja
1.0		1
1.5	Cupcake	3
1.6	Donut	4
2.0 / 2.1	Eclair	5-7
2.2	Froyo	8
2.3.x	Gingerbread	9-10
3.x	Honeycomb	11-13
4.0.x	Ice Cream Sandwich	14-15
4.1 / 4.2 / 4.3	Jelly Bean	16-18
4.4	KitKat	19-20
5.0/5.1	Lollipop	21-22
6.0 / 6.0.1	Marshmallow	23
7.0 / 7.1	Nougat	24-25
8.0 / 8.1	Oreo	26-27
9.0	Pie	28
10.0	Q	29
11.0	R	30

2.2. Povijest Android operativnog sustava

U listopadu 2003. godine Andy Rubin, Rich Miner, Nick Sears i Chris White su osnovali Android Inc. kako bi razvijali programe za pametne mobilne uređaje [6]. Prvobitno je Android trebao biti platforma za digitalne kamere, no to tržište je bilo premalo. Zanimljiva činjenica je da je Android koji se razvijao za mobilne uređaje bio istog koda kao i ovaj za digitalne kamere. Nakon dvije godine tihog rada (znalo se samo da se radi o softveru za mobilne uređaje) Google je odlučio kupiti tvrtku Android Inc. Oni su isto tako zaposlili prvobitnog osnivača Androida da nastavi daljnji razvoj platforme. Rubin i ostali osnivači su tako nastavili razvijati Android za novog vlasnika. U studenom 2007. godine nakon što je osnovan OHA, otkrivaju mobilnu platformu otvorenog koda baziran na Linux kernelu – Android. To je ujedno bilo i prvo predstavljanje Androida kao operativnog sustava. Google je donio odluku da to bude operativni sustav otvorenog koda, dostupan svim proizvođačima mobilnih uređaja, kako bi popularnost platforme bila što veća.

Prvi uređaj u koji je bio ugrađen bio je HTC T-Mobile G1. Tako je Android postao konkurencija ostalim mobilnim platformama kao što su: iOs, Windows Phone, Symbian…

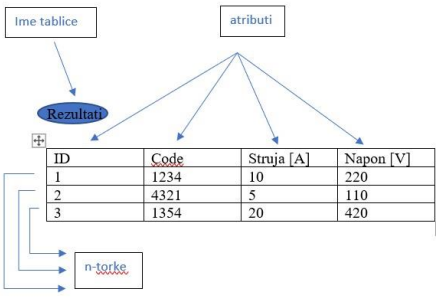
Još od samih početaka je Android zamišljen kao projekt otvorenog koda te je od 21. listopada 2008. godine dostupan cjeloviti kod pod Apache licencom. Od toga su drugi proizvođači profitirali jer svatko može ugraditi svoje dodatke u Android, pa se samim tim razlikuju od ostalih proizvođača. Međutim, proizvođačima nije dopuštena upotreba imena „Android“ ukoliko Google ne certificira uređaj kao kompatibilan prema CDD-u (engl. *Compatibility Definition Document*). Tako je Google osjetio da mogu lako zaraditi novac iz ostalih usluga koje koriste operativni sustav umjesto da naplaćuju platformu proizvođačima. Uz to, dodali su mnoštvo svojih proizvoda i usluga operativnom sustavu uključujući Google Maps, Youtube, Google Browser…

Nakon pojave iOs-a tj. IPhone-a 2007. godine, koji je donio jednu vrstu revolucije na mobilne uređaje, mnogi proizvođači su morali posegnuti za boljim rješenjem kako bi ostali konkurentni. Tad je od velike koristi bio Android čiji softver su kompanije koristile, a nastavile su razvijati vlastiti hardver. Google je tako počeo otkrivati informacije o operativnom sustavu za mobilne uređaje i njegovim daljnjim planovima. Razvijali su program koji bi „obični“ mobilni uređaj napravio pametnim i više svjesnim svoje okoline. Pa tako i dan danas svakim ažuriranjem Android postaje pametniji i bolji u smislu performansi. [6]

Slika 2: Razvoj Android sustava [7]

3. Relacijska baza podataka

Relacijska baza podataka sastoji se od skupa povezanih tablica odnosno relacija. Relacijska shema je definicija jedne relacije. Dakle ona (slika 3) obuhvaća naziv relacije (Rezultati) i skup atributa (ID, Code, Struja, Napon) koji čine tu relaciju. Relacijska shema baze podataka je skup relacijskih shema svih relacija koje se nalaze u toj bazi. Relacijska struktura daje uvid u realan svijet i podatke određenih događaja. Relacijska baza podataka je trenutno najkorišteniji model baze podataka. [8]



Slika 3: Primjer korištene relacijske sheme

Upravljanje relacijskom bazom podataka vrši Sustav za upravljanje bazom podataka – (engl. *RDBMS (Relational Database Management System)*). To je sustav koji služi za pohranjivanje podataka u obliku međusobno povezanih tablica i organiziranje podataka u bazi podataka. Ovaj sustav obavlja dvije grupe poslova:

1. Definiranje baze podataka
2. Rad s podacima

Time se zapravo obuhvaća:

- Ažurnost pohranjenih podataka
- Izgradnja pristupa podatcima
- Mogućnost sigurnosnog arhiviranja
- Mogućnost kontrole s jednog mjesta (unošenje, uređivanje, pretraživanje, prikazivanje, sortiranje i filtriranje podataka)
- Stvaranje izvještaja
- Najmanja redundancija (nema gomilanja podataka)
- Postojanost pohranjenih podataka
- Točnost pohranjenih podataka

- Trajno očuvanje integriteta pohranjenih podataka (sve vrijednosti podataka su konkretne u situacijama kad postoje greške u aplikacijama)

3.1. SQLite relacijska baza podataka

SQLite je relacijska baza podataka temeljena na C programskoj biblioteci [9]. U programskoj biblioteci nalaze se potprogrami koji nude rješenja tematski povezanim problemima. Glavna ideja SQLite je riješiti se server-klijent arhitekture i skladištiti sve podatke direktno na mobilni uređaj. Tako se SQLite koristi već dugi niz godina kada je potrebno skladištiti podatke na uređaj, a ne na server. Neka od velikih imena koja koriste SQLite su: Apple, Google, Adobe…

Glavne značajke SQLite su:

- Transakcije funkcioniraju po načelu ACID (atomicity, consistency, isolation, and durability)
- Podatci ostaju postojani nakon pada sustava
- Cijela baza podataka nalazi se u jednoj datoteci na disku
- Bazni podatci mogu se dijeliti između računala
- Baze mogu biti velike nekoliko terabajta
- Jednostavno korisničko sučelje

SQLite baza podataka zahtjeva minimalnu administraciju što je čini odličnom za primjenu u mobilnim telefonima. Svoju primjenu nalazi svugdje gdje je potreban automatski rad. Najbolje ju je koristiti kod manjih projekata gdje može ispuniti sve potrebe aplikacije. U svim ostalim slučajevima javlja se nekoliko problema.

SQLite koristi SQL stoga ima sve značajke standardne SQL baze podataka. Često se developeri znaju pomučiti kada u pitanje dođe testiranje aplikacije baze podataka, a SQLite je vrlo dobra za testiranje. Isto tako kada se izradi aplikacija, SQLite baza dolazi integrirana sa platformom. Nije potrebno instalirati nikakvu „knjižnicu“ za pristup podacima.

Tako uz mnoštvo prednosti mora postojati i nekoliko nedostataka. Svi podaci su dostupni na jednom mjestu, pa postoji mogućnost gubitaka ili krađe tih podataka. Problem sigurnosti otvaranja datoteka rješava se kriptiranjem gdje samo korisnici koji imaju ključ mogu te podatke čitati.

Drugi problem se javlja zbog toga što baza s vremenom raste. SQLite podupire baze veličine oko tridesetak terabajta, a u nekim slučajevima dolazi do potrebe za većim kapacitetom.

Time dolazimo do trećeg problema koji govori o kompleksnosti baze. Povećavanje kapaciteta baze iziskuje više analitičkog rada.

SQLite bi se isto tako trebao izbjegavati kod web stranica na kojima dolazi do učestalog upisa podataka ili ako web stranica zahtjeva više servera. [10]

3.2. Konceptualni ER dijagram

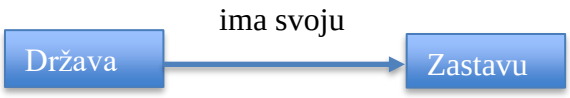
ER (engl. *Entity-relationship*) model služi za izradu manje precizne sheme koja je apstrakcija realnog svijeta. Autor ER dijagrama je dr. sc. Pin-Shan 1976. godine [11]. Prvi korak kod izrade je identifikacija entiteta, atributa i veza. Entiteti su događaji koji nas zanimaju, atributi su svojstva entiteta, a veze su odnosi među entitetima. Dakle entitet može biti neki objekt (parfem, stol…), živo biće (student, kupac, lav…) ili neki događaj (utakmica, obrana rada…). Isto tako je važno da entitet predstavlja imenica u jednini. Atribut koji je specifičan za svaki entitet predstavlja primarni ključ. Odnosno, specifičnih atributa može biti i više, ali samo jedan se bira kao primarni ključ. Funkcionalnost veze između entiteta može biti:

- 1 prema 1 (1:1)
- 1 prema više (1:N)
- Više prema 1 (N:1)
- Više prema više (N:M)

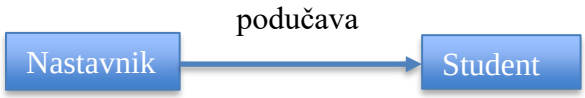
Što se grafičkog prikaza tiče, tipovi entiteta u ER dijagramu su prikazani kao pravokutnici, veze kao romboidi, atributi kao elipse, a krakovi veza imaju oznaku „1“ ili „N“ zavisno od vrsti veze.

Primjeri veze:

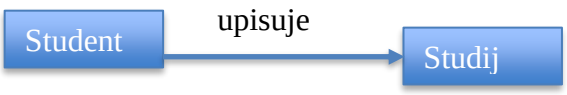
- 1 prema 1 (1:1)



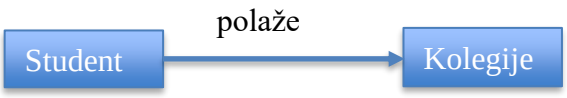
- 1 prema više (1:N)



- Više prema 1 (N:1)



- Više prema više (N:M)

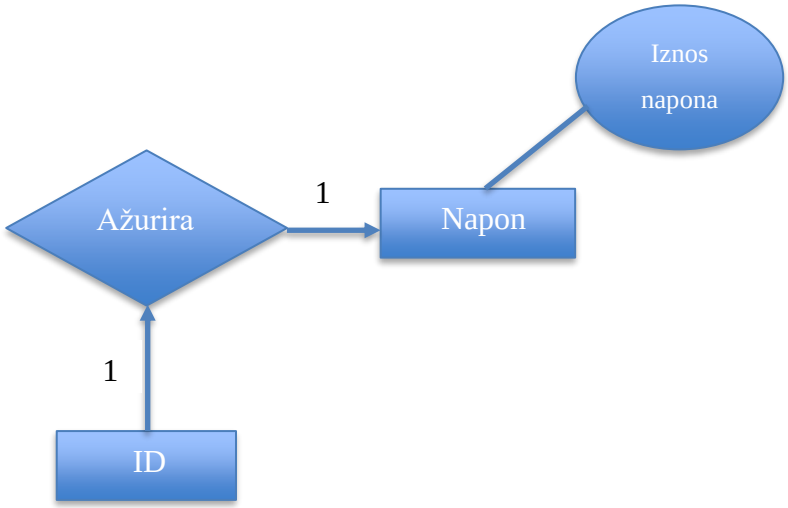


Postoji više rješenja problema i načina prikaza, no u svakom slučaju je vrlo jednostavan za nacrtati i razumjeti od projektanata baze podataka do korisnika. U gotovo svim slučajevima najjednostavniji model je i najbolji model. Uglavnom je to zbog toga što su jednostavniji modeli više fleksibilni, lakše ih je implementirati i lakše ih je razumjeti.

Pažnju treba obratiti na sljedeće:

1. Svaki entitet se smije pojaviti samo jednom u dijagramu
2. Svakom entitetu, atributu i vezi treba dati ime
3. Provjeriti jesu li veze između entiteta zaista potrebne
4. Mogu se koristiti boje za jasnije označivanje bitnijih dijelova dijagrama

Na slici 4 je prikazan dio ER dijagrama korištene aplikacije.



Slika 4: Primjer ER dijagrama aplikacije

4. Alati korišteni kod izrade aplikacije

4.1. Android Studio

Android Studio je integrirano razvojno okruženje za razvoj Android platforme [12]. Omogućava pisanje, analiziranje, oponašanje i otklanjanje neispravnosti na Android aplikacijama. Podržava 3 programska jezika za razvijanje aplikacija:

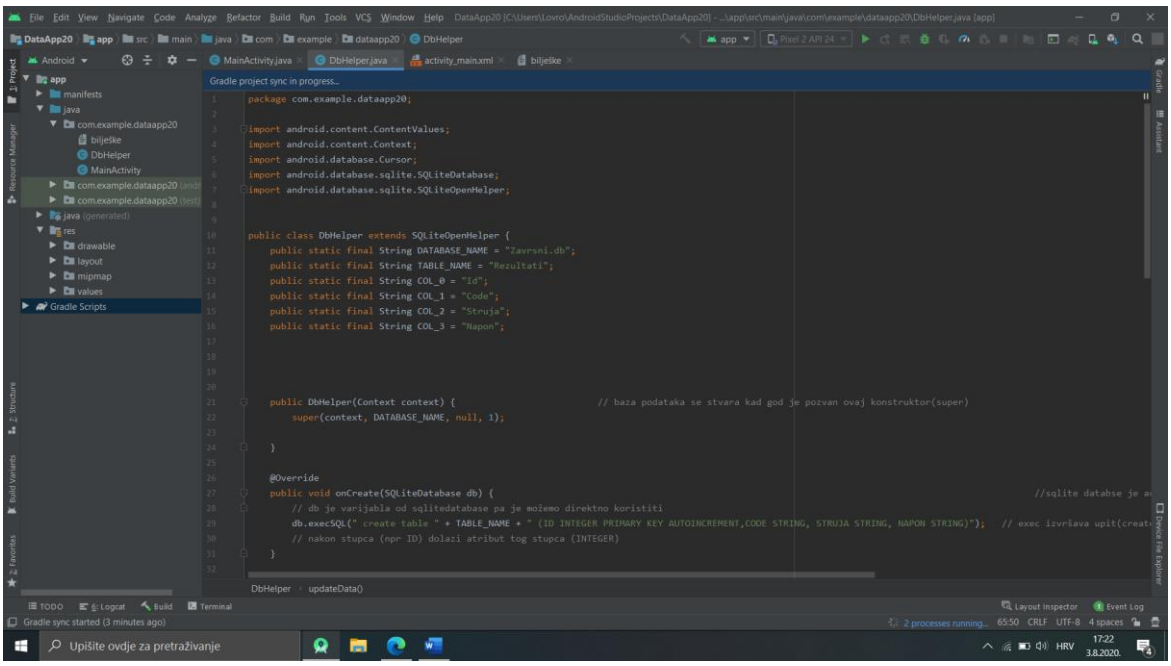
1. Kotlin
2. Java
3. C++

Slika 5: Logo Android Studia [13]

Android Studio nudi mnogobrojne funkcionalnosti kao što su:

- Fleksibilan Gradle sustav za izgradnju (alat za nadgledanje procesa razvoja)
- Emulator brojnih mogućnosti
- Specifično razvojno okruženje koje omogućava razvijanje za sve Android uređaje
- Promjena koda i izgleda sučelja bez resetiranja aplikacije
- Predlošci za kod i GitHub integracija (GitHub uočava promjene u kodu)
- Alati za performanse, provjeru kompatibilnosti i ostale probleme

Preuzimanje je besplatno, a posljednje stabilno izdanje 4.0.1 izdano je 14. srpnja 2020. godine. Za razvoj aplikacija Android Studio zahtijeva instalaciju Java Development Kit-a (JDK). Podržan je za sustav Linux, Windows i Mac OS X.

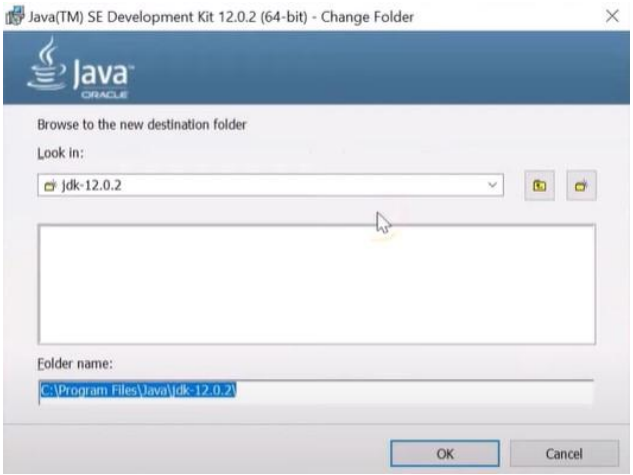


Slika 6: Izgled projekta u Android Studiu

4.2. Instalacija Android Studia

Prije nego što krenemo s instalacijom Android Studia podrazumijeva se da imamo instaliran JDK. Njega je moguće preuzeti na sljedećoj adresi:
https://www.oracle.com/java/technologies/javase-downloads.html

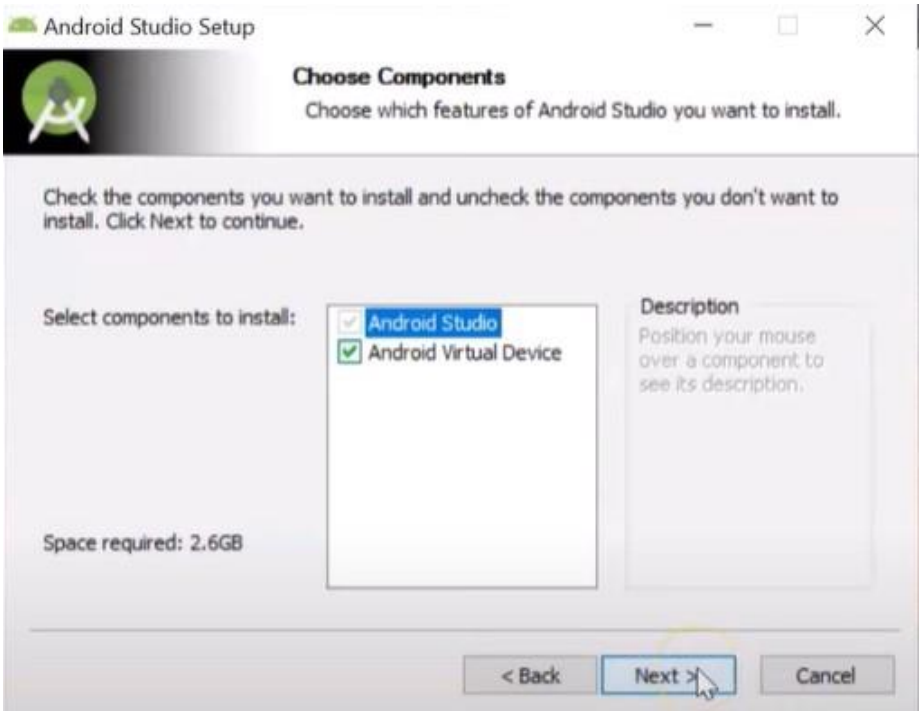
Pojavljuje se jednostavan čarobnjak kod kojeg je samo potrebno odabrati direktorij gdje ćemo instalirati JDK (slika 7).



Slika 7: Instalacija Java Development Kit-a

Slijedi preuzimanje Android Studia koje je vrlo jednostavno. Potrebno je putem web preglednika otvoriti stranicu: https://developer.android.com/studio

Pojavljuje se jednostavan čarobnjak čije upute vrlo jednostavno slijedimo (slika 8). Čak i ukoliko nemamo instaliran JDK, Android Studio pokreće analizu i detektira sve što nam nedostaje kako bismo započeli razvoj aplikacije i pokreće preuzimanje istog.



Slika 8: Instalacija Android Studia

4.3. Android emulator

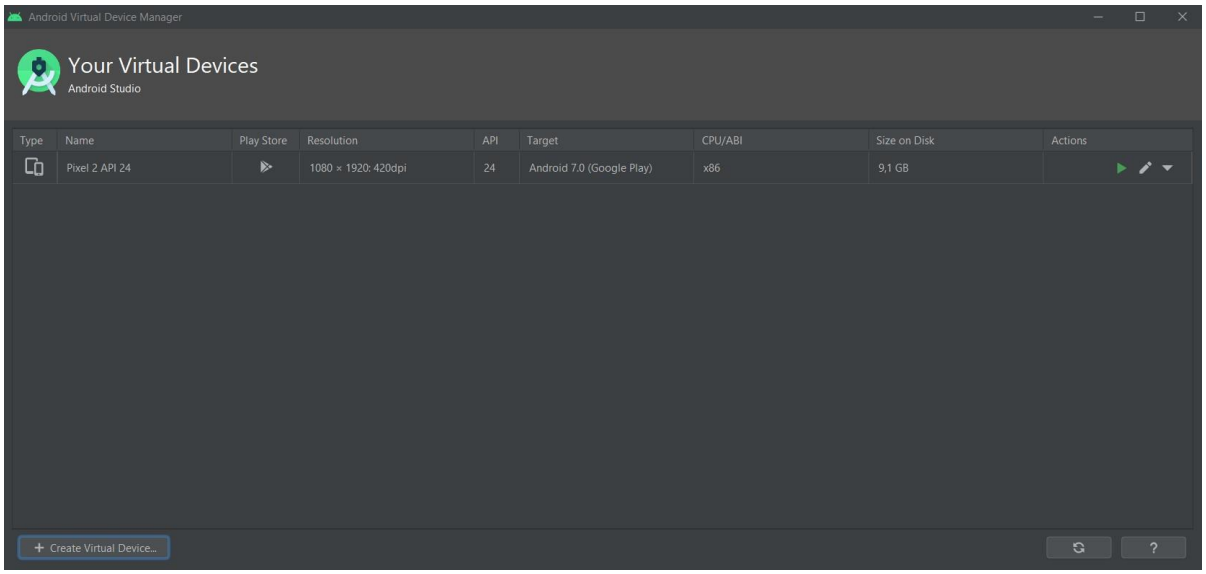
Emulator je virtualni uređaj odnosno stvara virtualno okruženje u kojem se testira aplikacija bez stvarnog uređaja (slika 9). Za razliku od samog Android Studia njemu je za pokretanje potrebno instalirati Android Software Development Kit (SDK) i definirati Android Virtual Device (AVD) odnosno uređaj na kojem želimo pokrenuti aplikaciju (TV, mobilni uređaj, pametni sat, tablet).

Prednost mu je što je moguće testirati aplikaciju na mnogobrojnim uređajima sa različitim Android operativnim sustavima, različitim procesorima, memorijama… Nedostatak mu je što ne podržava WiFi, GPS, Bluetooth…

Slika 9: Android emulator za uređaj Pixel 2, API 24

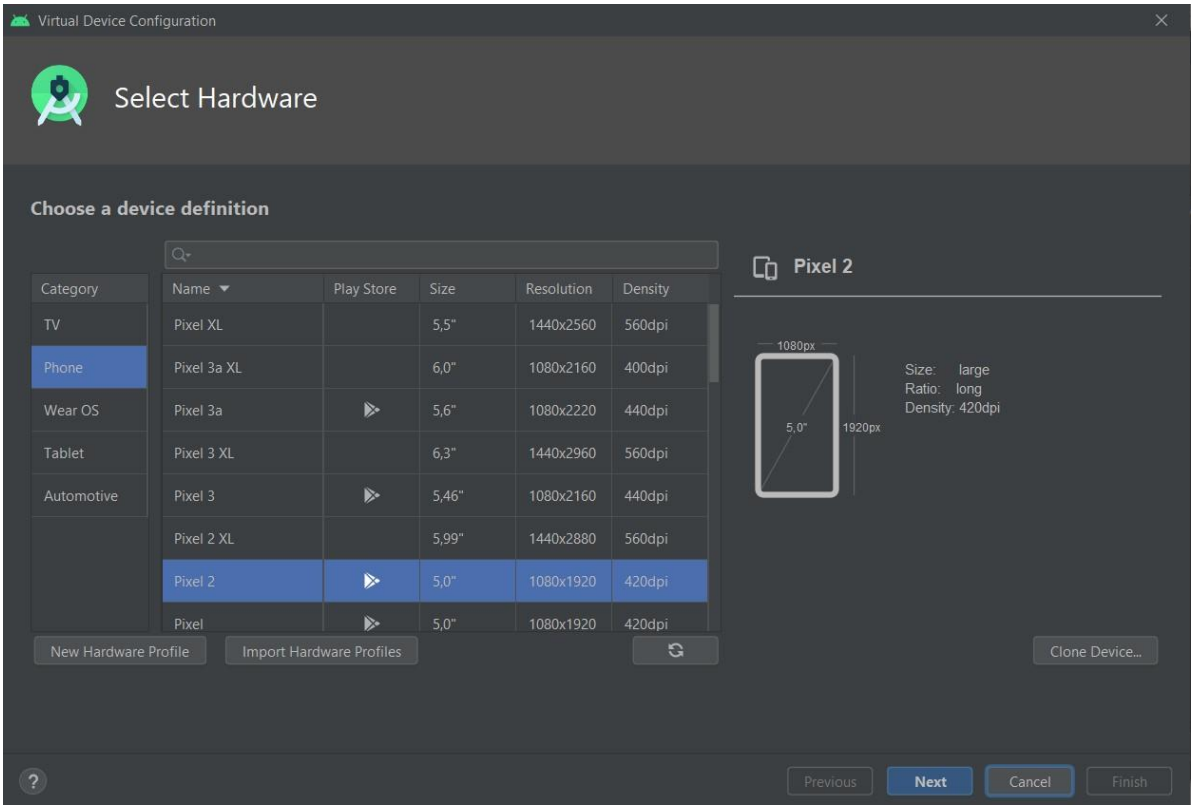
4.4. Instalacija Android emulatora

Nakon instalacije Android Studia potrebno je otvoriti AVD Manager. Otvara nam se čarobnjak na kojem su nam prikazani već instalirani virtualni uređaji i mogućnost instalacije drugih. Potrebno je odabrati opciju **Create Virtual Device** (slika 10).



Slika 10: Prikaz početnog prozora AVD Managera

Nakon što smo odabrali opciju stvaranja novog virtualnog uređaja pojavljuje se prozor na kojem imamo popis virtualnih uređaja i biramo onog kojeg želimo koristiti. Zatim se pojavljuje prozor na kojem se odabire sistemska slika odnosno verzija Android sustava (slika 11).

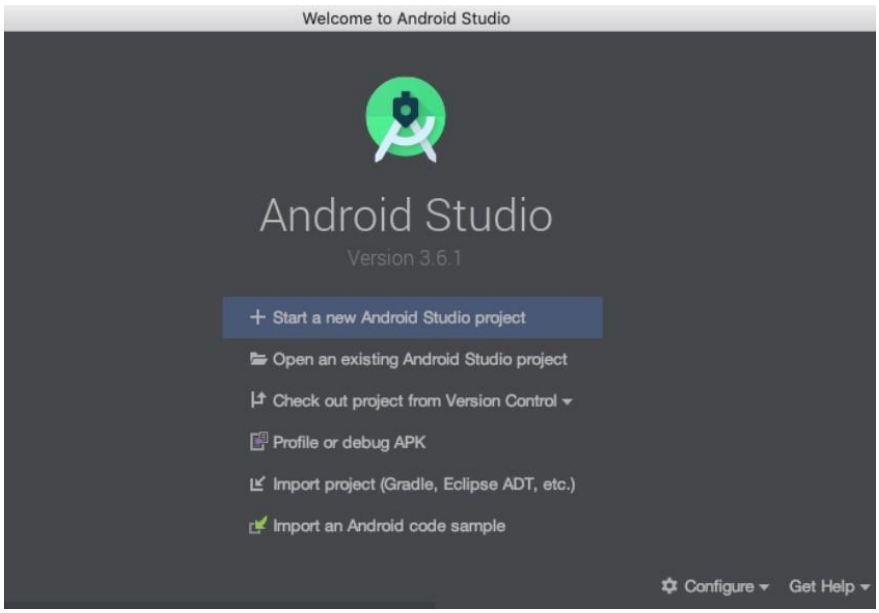


Slika 11: Odabir virtualnog uređaja

5. Android aplikacija

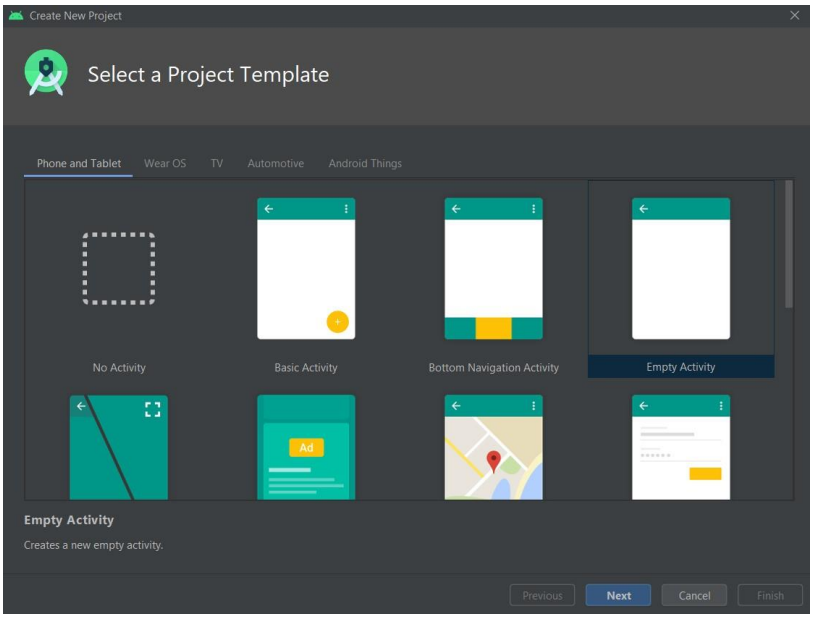
5.1. Kreiranje projekta

1. U samom početku pojavljuje se prozor *Welcome to Android Studio* (slika 12) gdje odabiremo **Start a new Android Studio project**. Ukoliko već imamo otvoren projekt, odaberemo **File > New > New project**.



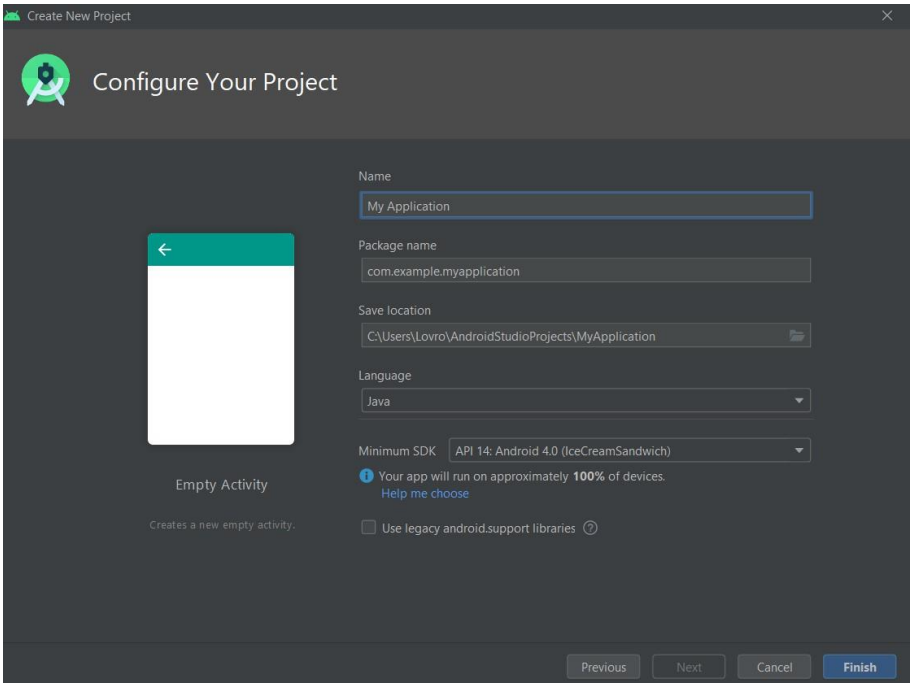
Slika 12: Kreiranje projekta 1. korak

2. U sljedećem koraku biramo **Empty activity** i kliknemo **Next** (slika 13).



Slika 13: Kreiranje projekta 2. korak

3. U posljednjem koraku potrebno je upisati ime aplikacije, mjesto gdje želimo spremiti naš projekt, programski jezik kojim se želimo koristiti i verziju Android operativnog sustava. Kliknemo **Finish** (slika 14).

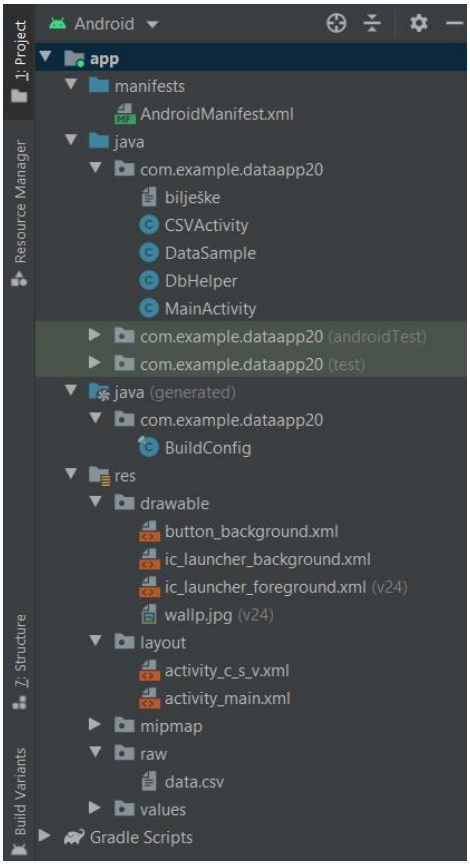


Slika 14: Kreiranje projekta 3. korak

5.2. Razvoj aplikacije

Nakon stvaranja projekta dolazi do automatskog generiranja njegovih datoteka (slika 15):

- Androidmanifest.xml – ovdje se bilježi aktivnost aplikacije,
- java – direktorij koji sadrži sve java klase datoteka aplikacije. U njoj se nalazi kod aplikacije,
- java (generated) – tu se nalazi BuildConfig odnosno bazična konfiguracija aplikacije,
- res – u ovom direktoriju se nalaze sve naše slike tj. grafički prikaz aplikacije

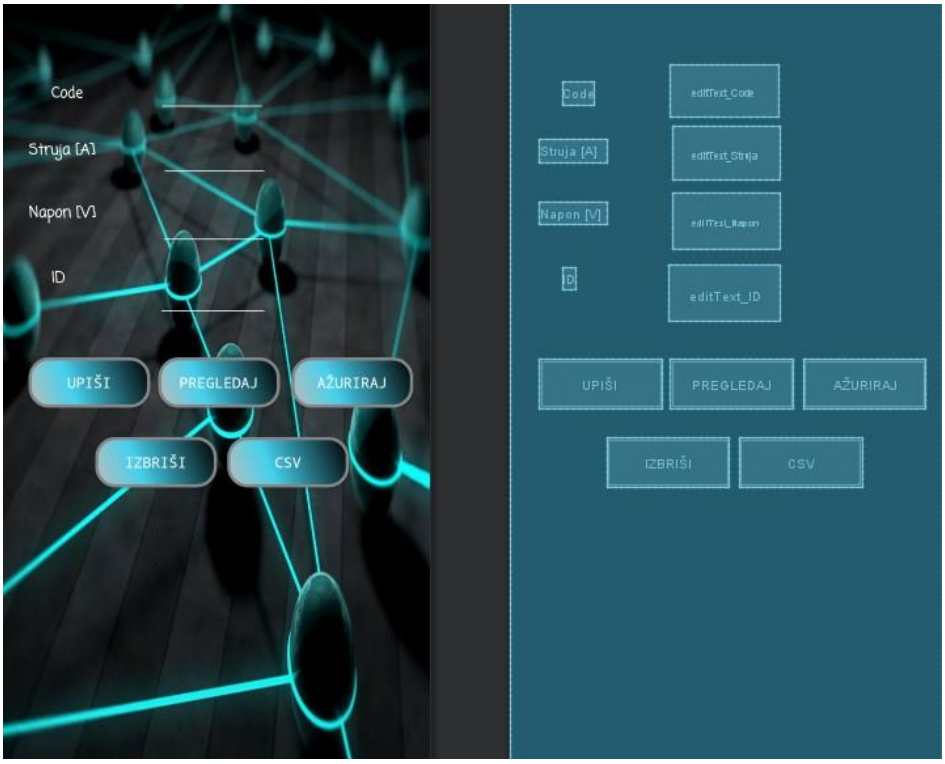


Slika 15: Struktura projekta

5.2.1. Kreiranje početnog zaslona

Najpraktičnije bi bilo da se početni zaslon izradi prije pisanja koda. Tako su u ovom slučaju na početni zaslon stavljena 4 gumba pomoću kojih upisujemo, pregledavamo, ažuriramo i brišemo podatke iz baze (slika 16). Zatim su nam potrebna četiri EditText-a u koje upisujemo naše vrijednosti i četiri TextView-a koji prikazuju koju vrijednost upisujemo. Dodajemo i sliku

wallp.jpg u projektnu mapu *drawable* koju da bi prikazali na zaslonu moramo napisati jednostavnu naredbu *android:background="@drawable/wallp"* gdje sa „@“ biramo iz koje mape uzimamo sliku.



Slika 16: Početni zaslon

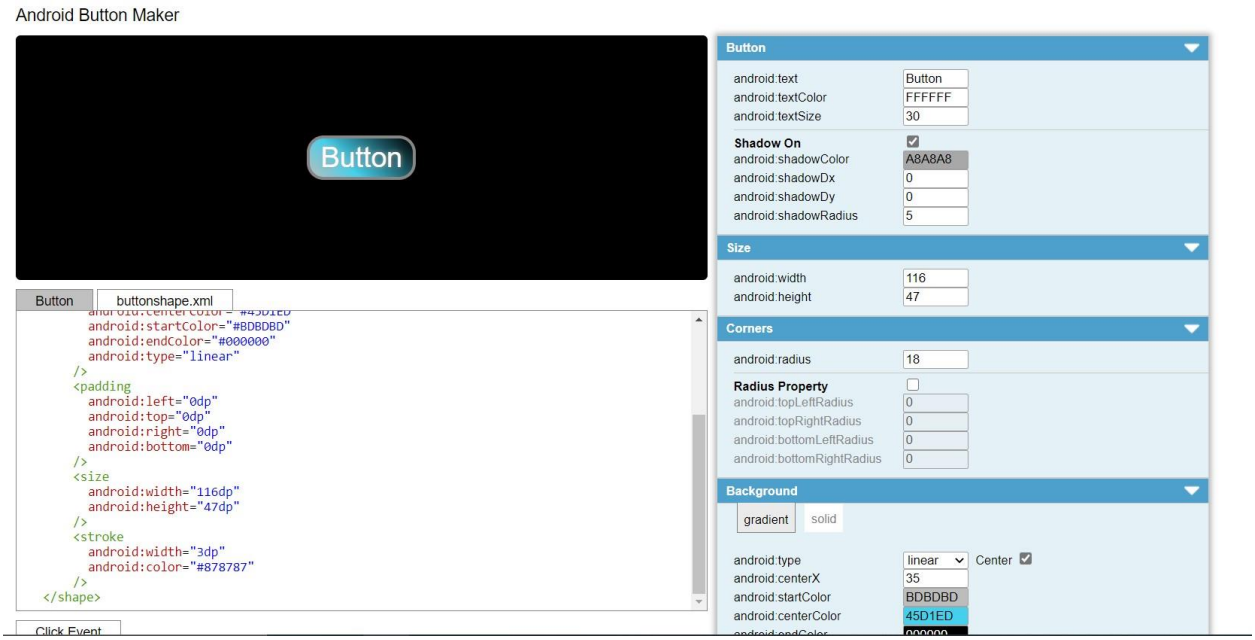
Početni zaslon je također moguće grafički sastaviti putem pisanja koda (slika 17), no puno je jednostavnije koristiti metodu „drag and drop“ s palete gumbića, edittextova, widgeta…

```
<Button
    android:id="@+id/buttonAdd"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginStart="53dp"
    android:layout_marginLeft="53dp"
    android:layout_marginTop="341dp"
    android:text="Upiši"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toTopOf="parent" />
```

Slika 17: Izgled koda za gumb "Upiši" - grafičko sučelje

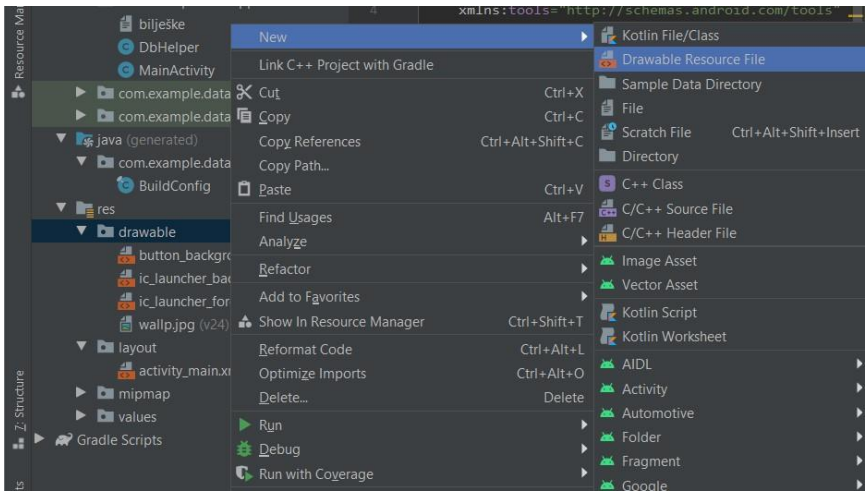
Također za sve što se nalazi na zaslonu potrebno je dodati restrikcije (marginu visine i širine) jer bi inače svi elementi prilikom pokretanja aplikacije „skočili“ u gornji lijevi kut.

Da bi promijenili izgled gumbića moramo sami izraditi svoj dizajn. U mom slučaju pronašao sam web stranicu (slika 18) na kojoj je to vrlo jednostavno promijeniti. Dakle otvorimo stranicu http://angrytools.com/android/button/ i mijenjamo veličinu gumbića, sjenu, boju pozadine, boju slova… Kad nešto promijenimo automatski se generira i mijenja kod za taj gumbić.



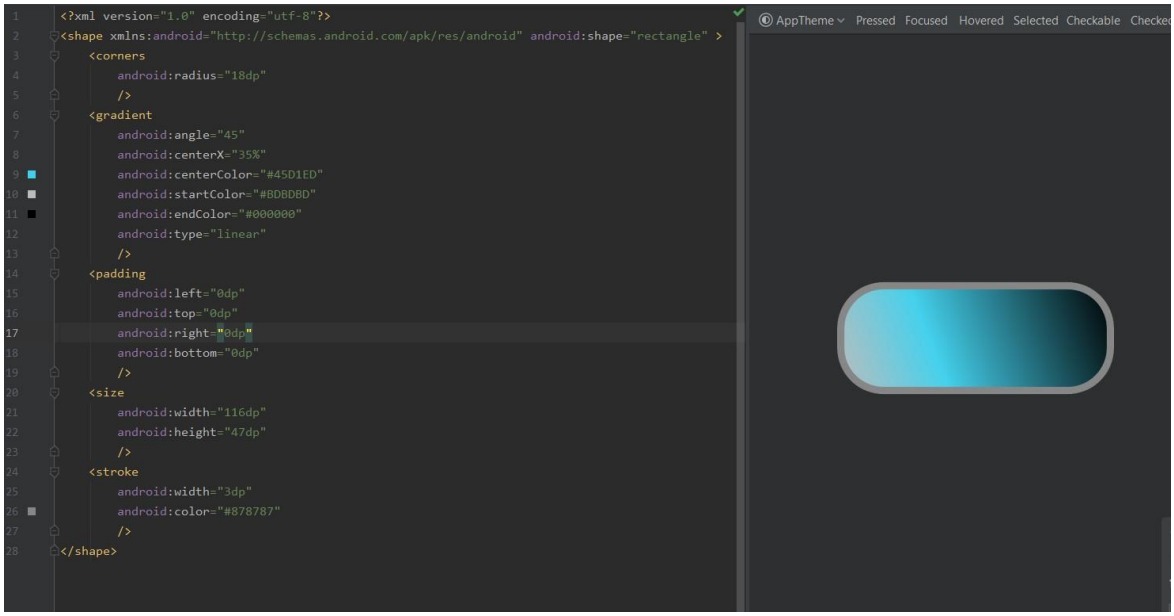
Slika 18: Promjena dizajna gumbića

Što se tiče implementiranja istog (slika 19) u Android studio potrebno je: **desni klik na drawable > new > Drawable Resource File**



Slika 19: Stvaranja file-a za dizajn gumbića

Zatim samo treba kopirati kod koji se generirao na web stranicu u taj file (slika 20), a sa desne strane nam se prikaže naš gumbić.



Slika 20: Kod za izgled gumbića

5.3. Kodiranje aplikacije

Aplikacija se sastoji od 2 „java class“ datoteke odnosno 2 dijela koda. Prvi dio koda napisan je u *MainActivity* gdje pozivamo naredbe koje smo definirali u *DbHelper* klasi.

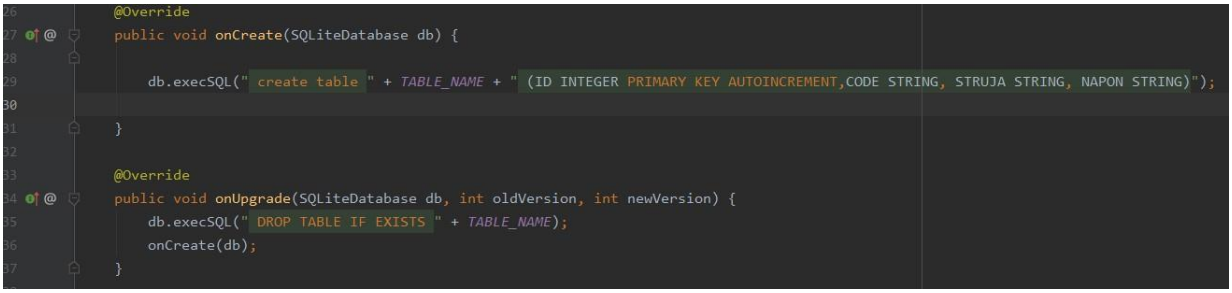
Nakon što smo definirali grafičko sučelje i znamo koji gumbi su nam potrebni, tada krećemo na pisanje koda koji rezultira onime što smo si zamislili.

U prvom koraku stvaramo *DbHelper* klasu i deklariramo ime naše baze podataka te ime tablice (relacije) i stupaca u njoj. Zatim pozivamo funkciju tj. konstruktora (slika 21) koji zahtjeva 4 argumenta *super(context, DATABASE_NAME, NULL, 1)* i stvara bazu podatka kad god je pozvan.

Slika 21: Deklariranje baze i pozivanje njezinog konstruktora u DbHelper

Da bi se stvorila tablica u našoj bazi moramo pozvati *onCreate* metodu (slika 22) čiji je jedini argument *SQLiteDatabase*. Naredba *exec* izvršava upit koji joj postavimo, a u ovom slučaju je to stvaranje tablice. Argumenti naredbe *exec* su ime tablice te njezini stupci i pripadajući tipovi podataka (string). „db“ je varijabla od SQLiteDatabase pa se može direktno koristiti.

onUpgrade metoda nam služi da se prilikom ažuriranja tablice stara izbriše, a stvara se nova i ažurirana.



Slika 22: onCreate i onUpgrade metode u DbHelper

5.3.1. Metoda za upis podataka

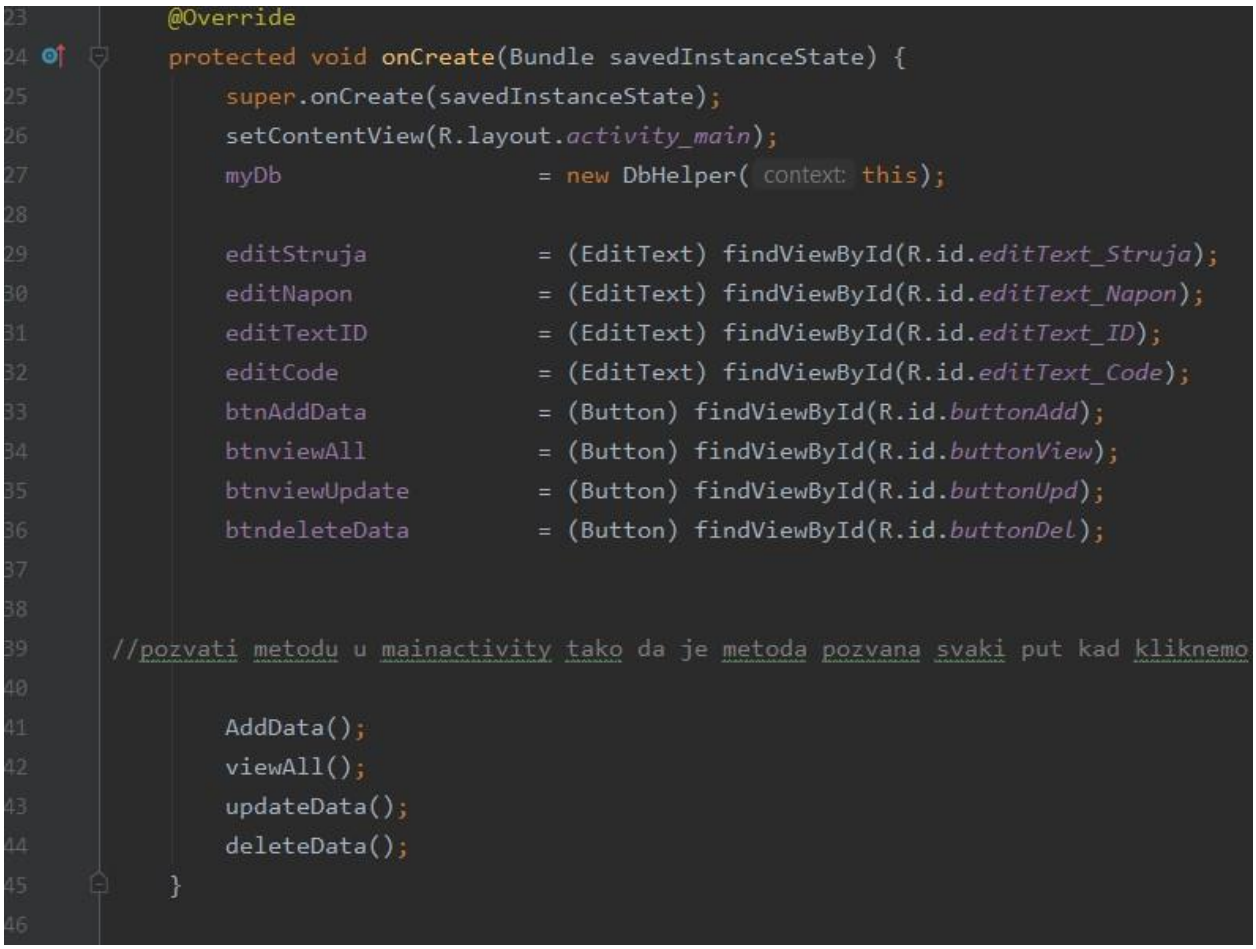
Stvaramo novu boolean metodu *insertData* (slika 23) da bismo unosili podatke u našu bazu podataka. Boolean zbog toga što nam metoda vraća „true“ ili „false“ odnosno jesu li podaci upisani ili nisu. Naredba *SQLiteDatabase db = this.getWritableDatabse();* nam služi kako bismo

vidjeli našu bazu kad je ona stvorena. Stvaramo klasu *ContentValues* u koju upisujemo podatke. Budući da se radi o boolean metodi onda moramo imati neki rezultat. Zato dodajemo naredbu *long result = db.insert(TABLE_NAME, null, contentValues);* koju provjeravamo naredbom if.

```
39      public boolean insertData(String code, String struja, String napon) {
40          // služi samo za provjeru tj. da vidimo našu stvorenu bazu podataka
41          SQLiteDatabase db = this.getWritableDatabase();
42          ContentValues contentValues = new ContentValues();
43          // stupac u koji želimo unijeti podatke, zatim vrijednost
44          contentValues.put(COL_1, code);
45          contentValues.put(COL_2, struja);
46          contentValues.put(COL_3, napon);
47          long result = db.insert(TABLE_NAME, nullColumnHack: null,contentValues);
48          if(result == -1)
49              return false;
50          else
51              return true;
52
53      }
```

Slika 23: Kreiranje metode za upis podataka u DbHelper

Sve varijable i metode koje koristimo u DbHelper.java moramo definirati i pozvati u MainActivity.java (slika 24) kako bi metoda bila pozvana svaki puta kada kliknemo gumbić. Naredba *myDb = new DbHelper(this);* poziva konstruktora od DbHelper klase. Naredba *findViewById* traži izgled aktivnosti prema ID-u koji smo definirali u XML datoteci (svakom elementu koji se nalazi na zaslonu se dodjeljuje ID) postavlja izgled sučelja.



```
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        myDb                = new DbHelper( context: this);

        editStruja          = (EditText) findViewById(R.id.editText_Struja);
        editNapon           = (EditText) findViewById(R.id.editText_Napon);
        editTextID          = (EditText) findViewById(R.id.editText_ID);
        editCode            = (EditText) findViewById(R.id.editText_Code);
        btnAddData          = (Button) findViewById(R.id.buttonAdd);
        btnviewAll          = (Button) findViewById(R.id.buttonView);
        btnviewUpdate       = (Button) findViewById(R.id.buttonUpd);
        btndeleteData       = (Button) findViewById(R.id.buttonDel);


    //pozvati metodu u mainactivity tako da je metoda pozvana svaki put kad kliknemo

        AddData();
        viewAll();
        updateData();
        deleteData();
    }
```

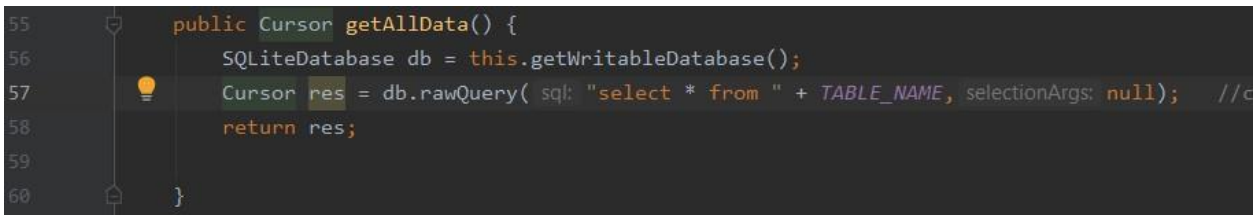
Slika 24: Korištene varijable i metode

Tada u MainActivity.java stvaramo metodu *AddData()* (slika 25). Naredbom *setOnClickListener* tzv. osluškivač događaja definirano je da gumbić bude osjetljiv na dodir. Dakle osluškivač se povezuje na događaj nekog od prethodno navedenih elemenata. Npr. pritiskom na gumb „Upiši“ pokreće se funkcija *insertData*. Naredbom *getText().toString()* konvertiramo naše editTextove u string. Također kako bismo dobili povratnu poruku na zaslonu o tome jesu li naši podaci upisani koristimo naredbu *Toast.makeText(this, Podaci upisani“, Toast.LENGTH_LONG).show();*

Slika 25: AddData metoda u MainActivity

5.3.2. Metoda za pristup i spremanje podataka

Stvaramo novu metodu *getAllData()* pomoću Cursos klase (slika 26). Slijedno tome stvara se instanca Cursos klasi i uzima se instanca od SQLiteDataBase „db“, da bi stvorili naredbu pomoću koje spremamo rezultat i dobivamo pristup istom. Kod upita za prikaz podataka *Cursor res = db.rawQuery("select * from " + TABLE_NAME,null);* koristi se SQL naredba „select *“ koja prikazuje sve dostupne podatke iz tablice koju odaberemo. Na kraju vraćamo instancu Cursora odnosno *return res;* gdje je spremljen naš rezultat.



Slika 26: Metoda za pristup podacima u DbHelper

Slijedno tome potrebno je napraviti viewAll() metodu u MainActivity gdje isto tako dodajemo osluškivač kao i kod prethodnog gumbića (slika 27). Koristi se tip Void jer metoda nema return vrijednost. Spremamo rezultat u Cursor jer vraća njegov objekt *Cursor res = myDb.getAllData();* . Zatim pišemo if funkciju *if (res.getCount() == 0) {*

showMessage("Error","Nema upisanih podataka");

return;

}

koja, ukoliko nema upisanih podataka, pokazuje error.

Ukoliko ima rezultata spremamo ih pomoću buffera koristeći while petlju i moveToNext() metodu koja pomiče Cursor na sljedeći rezultat. Dakle naredba *buffer.append("Id :" + res.getString(0) + "\n");* dohvaća naš string (podatke) s brojem stupca koji smo zadali (u ovom slučaju 0). Podatke prikazujemo konvertirajući buffer u string *showMessage("Data",buffer.toString());* .
Jednostavnom naredbom */n/n* postižemo prazan red kod sljedećeg ispisa.

```
100     public void viewAll() {
101         btnviewAll.setOnClickListener(
102                 new View.OnClickListener() {
103                     @Override
104                     public void onClick(View v) {
105                         Cursor res = myDb.getAllData();
106                         if (res.getCount() == 0) {
107     //prikazi poruku
108                             showMessage( title: "Error", Message: "Nema upisanih podataka");
109                             return;
110                         }
111     //ako ima rezultata (podataka)
112                         StringBuffer buffer = new StringBuffer();
113                         while (res.moveToNext()) {
114                             buffer.append("Id :" + res.getString( columnIndex: 0) + "\n");
115                             buffer.append("Code :" + res.getString( columnIndex: 1) + "\n");
116                             buffer.append("Struja :" + res.getString( columnIndex: 2) + "\n");
117                             buffer.append("Napon :" + res.getString( columnIndex: 3) + "\n\n");
118                             //sljedeceg ispisa imamo red prazno
119                         }
120                         showMessage( title: "Data",buffer.toString());
121
122                     }
123                 }
124         );
125     }
```

Slika 27: Metoda za pristup podatcima u MainActivity

5.3.3. Metoda za prikaz podataka

Metoda treba dva argumenta (slika 28), a to su naslov (title) i sama poruka odnosno podatak (Message). Koristeći Builder *AlertDialog.Builder builder = new AlertDialog.Builder(this);*

postavljamo naslov i podatke u novom prozorčiću. Naredbom *builder.setCancelable(true);* omogućujemo prekidanje prozorčića nakon što smo ga upotrijebili.

```
127     public void showMessage(String title,String Message){
128         AlertDialog.Builder builder = new AlertDialog.Builder( context: this);
129         builder.setCancelable(true);
130         builder.setTitle(title);
131         builder.setMessage(Message);
132         builder.show();
133     }
```

Slika 28: Prikaz podataka

5.3.4. Metoda za ažuriranje podataka

Metoda za ažuriranje podataka je tipa boolean jer nam vraća jesu li podaci ažurirani ili nisu. Ima onoliko argumenata koliko imamo stupaca za ažurirati te argument Id kojim biramo stupac za ažuriranje. Ponovo koristimo klasu contentValues kao i kod upisa podataka no ovdje moramo dodati i contentValues za Id. Jedina razlika između funkcije za ažuriranje i funkcije za upis je metoda *db.update(TABLE_NAME, contentValues,"Id = ?", new String[] { Id });* koja za argumente ima ime naše tablice, contentValues, Id koji je ujedno i uvjet pod kojim želimo ažurirati i četvrti argument je new String[] (slika 29).

```
63      public boolean updateData (String Id, String code, String struja, String napon) {
64          SQLiteDatabase db = this.getWritableDatabase();
65          ContentValues contentValues = new ContentValues();
66          contentValues.put(COL_0, Id);
67          contentValues.put(COL_1, code);
68          contentValues.put(COL_2, struja);
69          contentValues.put(COL_3, napon);
70          db.update(TABLE_NAME, contentValues, whereClause: "Id = ?", new String[] { Id });
71          return true;
72
73      }
```

Slika 29: Metoda za ažuriranje podataka u DbHelper

Koristimo boolean naredbu da bi provjerili je su li podaci ažurirani (slika 30):

boolean *isUpdate = myDb.updateData(editTextID.getText().toString(),*
editCode.getText().toString(), editStruja.getText().toString(),
editNapon.getText().toString());

te kao što je moguće uočiti, getText-ove je potrebno konvertirati u string. Na kraju pomoću if naredbe slijedi provjera i povratna informacija pomoću *Toast.makeText* .

```
public void updateData() {
    btnviewUpdate.setOnClickListener(
            (v) -> {
                    boolean isUpdate = myDb.updateData(editTextID.getText().toString(),
                            editCode.getText().toString(), editStruja.getText().toString(),
                            editNapon.getText().toString() );
                    if(isUpdate == true)
                        Toast.makeText( context: MainActivity.this, text: "Podaci ažurirani", Toast.LENGTH_LONG).show();
                    else
                        Toast.makeText( context: MainActivity.this, text: "Podaci nisu ažurirani", Toast.LENGTH_LONG).show();
            }
    );
}
```

Slika 30: updateData metoda u MainActivity

5.3.5. Metoda za brisanje podataka

Na kraju nam dolazi brisanje podataka iz naše baze. Ta metoda vraća integer i za argument je potreban samo String id jer je to uvjet prema kojem će se podatci brisati. Naredba *db.delete* uzima tri argumenta (slika 31): ime tablice, klazula prema kojoj želimo brisati podatke što je u našem slučaju Id, te vrijednost new String[] koja će zamijeniti „?“ . Odmah možemo postaviti return jer je u pitanju integer.

```
public Integer deleteData (String id) {
    SQLiteDatabase db = this.getWritableDatabase();
    return db.delete(TABLE_NAME,  whereClause: "Id = ?", new String[] { id } );
}

}
```

Slika 31: Metoda za brisanje podataka u DbHelper

Deklariramo integer varijablu pod imenom deletedRows te uzimamo objekt myDb i funkciju koju smo stvorili deleteData (slika 32). Funkcija uzima jedan argument Id kojeg je potrebno konvertirati u string. Na posljetku pomoću if i *Toast.makeText* naredbe provjeravamo i stvaramo povratnu informaciju jesu li podaci izbrisani ili ne.

```
public void deleteData() {
    btndeleteData.setOnClickListener(
            new View.OnClickListener() {
                @Override
                public void onClick(View v) {
                    Integer deletedRows = myDb.deleteData(editTextID.getText().toString() );
                    if (deletedRows > 0)
                        Toast.makeText( context: MainActivity.this, text: "Podaci izbrisani", Toast.LENGTH_LONG).show();
                    else
                        Toast.makeText( context: MainActivity.this, text: "Podaci nisu izbrisani", Toast.LENGTH_LONG).show();
                }
            }
    );
}
```

Slika 32: Metoda za brisanje podataka u MainActivity

5.3.6. Stvaranje instanci

Za svaki gumbić i EditText potrebno je stvoriti njihove instance što je vrlo jednostavno. To nam omogućuje da ih kasnije u kodu možemo pozvati (slika 33).

```
public class MainActivity extends AppCompatActivity {
    DbHelper myDb;
    EditText editStruja, editNapon, editTextID, editCode;
    Button btnAddData;
    Button btnviewAll;
    Button btnviewUpdate;
    Button btndeleteData;

```

Slika 33: Stvaranje instanci za gumbiće i edit textove

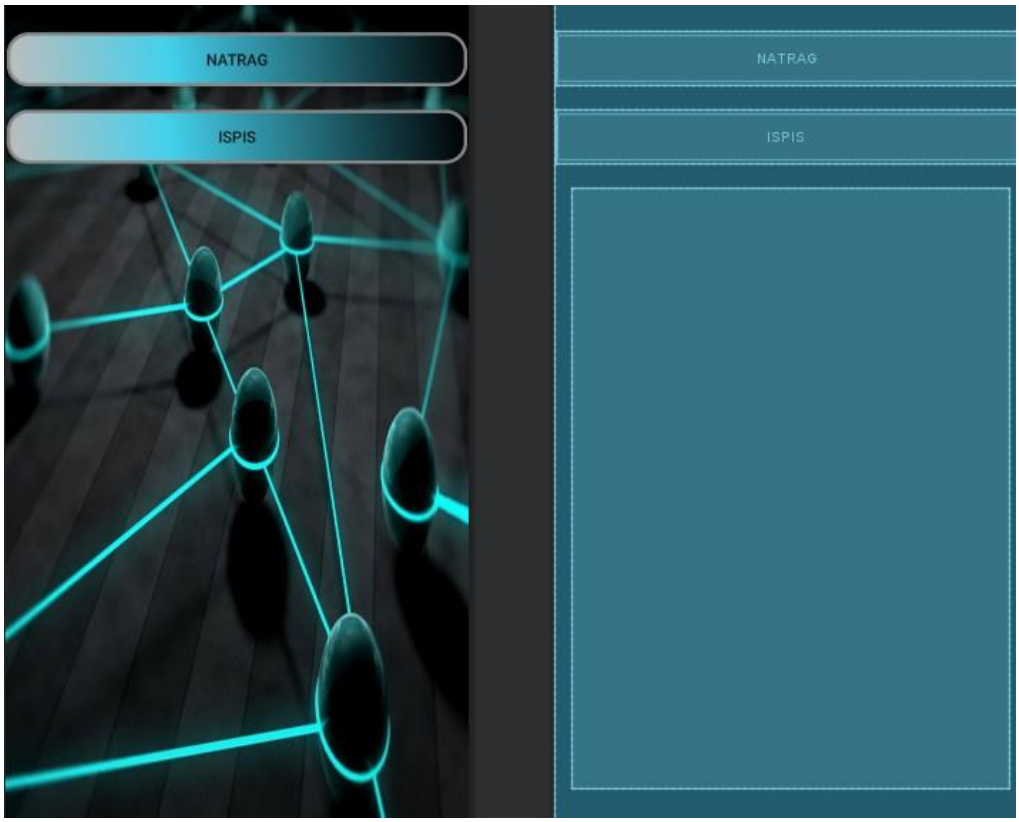
5.3.7. Metoda za čitanje CSV datoteka

U projektu otvorimo direktorij **Java > desni klik na com.example.dataapp20 > New > Activity > Empy Activity**. Tada se stvori nova .java i .xml datoteka. To nam služi kako bismo kod čitanja CSV datoteke otvorili novi prozor koji se otvori klikom gumbića „CSV“ (slika 34). Stvaramo novi Intent koji ima dva argumenta, a to su: klasa u kojoj se trenutno nalazimo (MainActivity) i klasa koju želimo otvoriti (CSVActivity).

```
48          buttonCSV.setOnClickListener(new View.OnClickListener() {
49              @Override
50              public void onClick(View v) {
51
52                  Intent intent = new Intent( packageContext: MainActivity.this, CSVActivity.class);
53                  startActivity(intent);
54              }
55          });
```

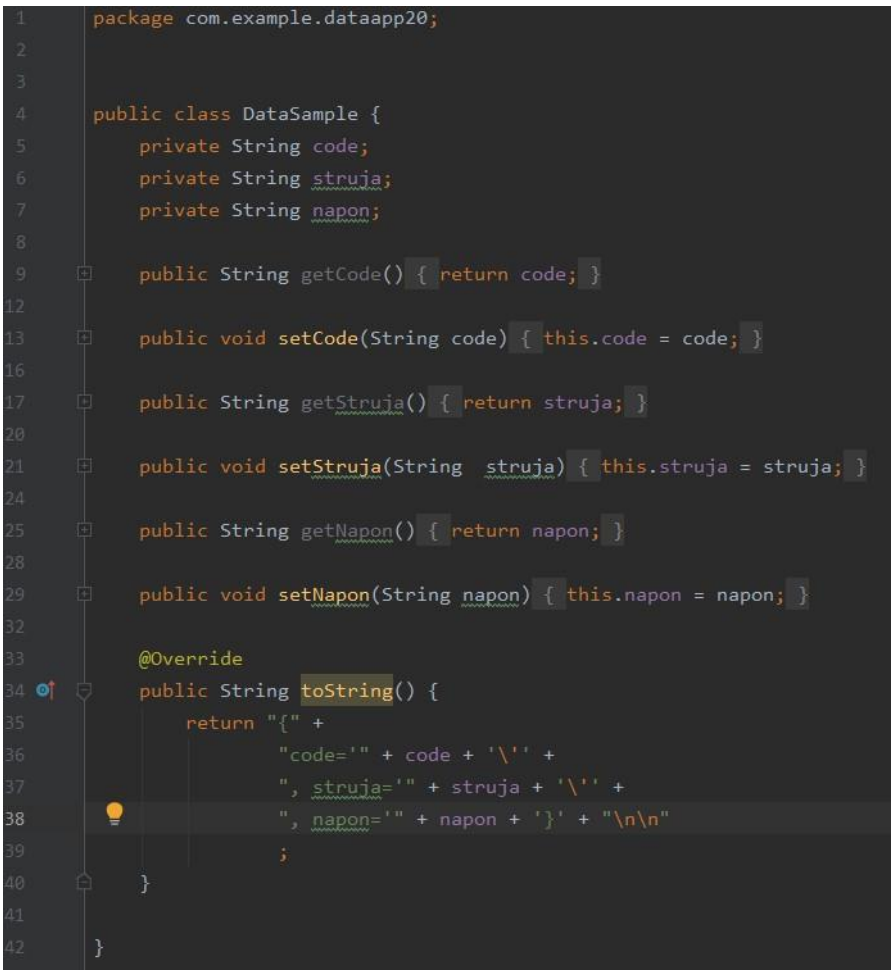
Slika 34: Metoda gumbića za otvaranje novog prozora u CSVActivity

Kad se otvori novi prozor (slika 35) na njemu se nalazi gumbić za povratak na početni zaslon i gumbić za čitanje CSV datoteke.



Slika 35: Izgled drugog prozora

Za čitanje CSV datoteke stvaramo novu klasu pod imenom DataSample (slika 36). U toj klasi definiramo vrstu podataka koje želimo. Na kraju stvaramo metodu *String* koja nam kod ispisa vraća podatke u string.



```
package com.example.dataapp20;


public class DataSample {
    private String code;
    private String struja;
    private String napon;

    public String getCode() { return code; }

    public void setCode(String code) { this.code = code; }

    public String getStruja() { return struja; }

    public void setStruja(String  struja) { this.struja = struja; }

    public String getNapon() { return napon; }

    public void setNapon(String napon) { this.napon = napon; }

    @Override
    public String toString() {
        return "{" +
                "code='" + code + '\'' +
                ", struja='" + struja + '\'' +
                ", napon='" + napon + '}' + "\n\n"
                ;
    }

}
```

Slika 36: Klasa za definiranje oblika podataka

Zatim se vraćamo u glavnu aktivnost gdje stvaramo metodu *readData()* za čitanje datoteke (slika 37). Naredbom *InputStream is = getResources().openRawResource(R.raw.data);* stvaramo pristup datoteci „data“ koja se nalazi u direktoriju „raw“. Budući da je CSV datoteke najbolje obrađivati redak po redak onda koristimo *BufferedReader reader = new BufferedReader(*

new InputStreamReader(is, Charset.forName("UTF-8"))

);

i koristimo UTF-8 način zapisa.

```
private List<DataSample> dataSamples = new ArrayList<>();
private void readData () {
    InputStream is = getResources().openRawResource(R.raw.data);
    BufferedReader reader = new BufferedReader(
            new InputStreamReader(is, Charset.forName("UTF-8"))
    );
```

Slika 37: Stvaranje pristupa CSV datoteci

Nakon toga stvaramo petlju koja čita retke CSV datoteke (slika38). Java indexi započinju s 0, pa zbog toga u naredbu *sample.setCode(tokens[0]);* upisujemo indeks 0 jer se Code nalazi u prvom stupcu.

```
66      try {
67                  reader.readLine();
68
69                  while ((line = reader.readLine()) != null) {
70
71                      String[] tokens = line.split( regex: ",");
72
73                      DataSample sample = new DataSample();
74                      sample.setCode(tokens[0]);
75                      sample.setStruja(tokens[1]);
76                      sample.setNapon(tokens[2]);
77                      dataSamples.add(sample);
78
79              Log.d( tag: "CSV datoteka",  msg: " " + sample);
80
81          }
```

Slika 38: Petlja za čitanje CSV datoteke

Da bi prikazali sadržaj CSV datoteke potrebno je napraviti *display()* metodu (slika 39) koja treba jedan argument, a to je u ovom slučaju lista dataSamples.

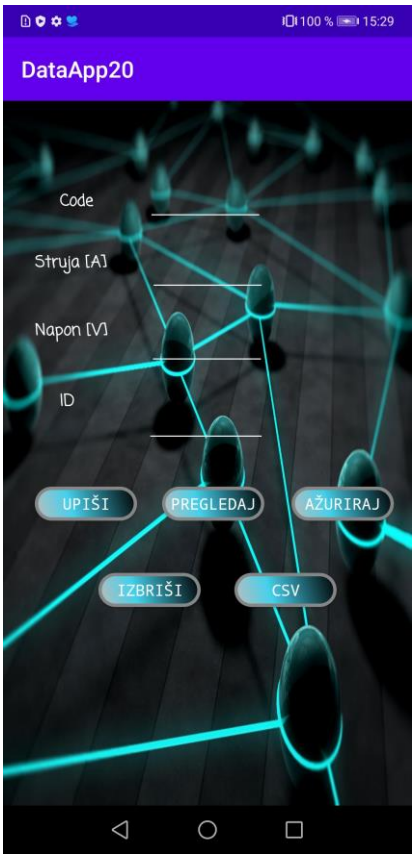
```
90      public void display(String dataSamples) {
91          TextView txt = (TextView) findViewById(R.id.txt);
92          txt.setText(dataSamples);
93      }
94
95  }
```

Slika 39: Metoda za prikaz sadržaja CSV datoteke

Tu metodu pozivamo u *readData()* metodi i konvertiramo u string da bi prikaz bio moguć *display(dataSamples.toString());* .

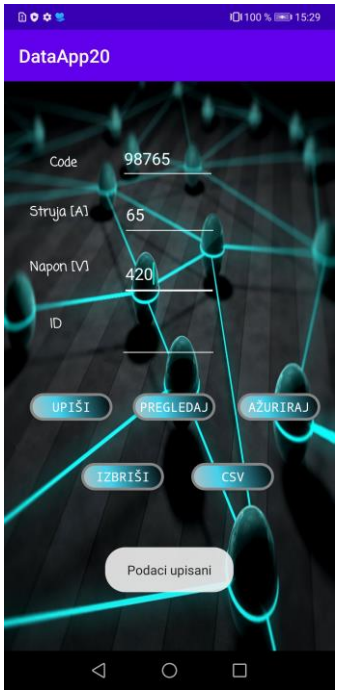
6. Primjer rada aplikacije

Mobilni uređaj je povezan putem USB kabela sa računalom kako bi se uspostavila veza između Android Studia i uređaja. Pokretanje se vrši kao i kod emulatora samo što na padajućoj listi odaberemo novi uređaj (u ovom slučaju Huawei P20) na kojem želimo pokrenuti aplikaciju. Na navedenom mobilnom uređaju nalazi se Android verzija 10.



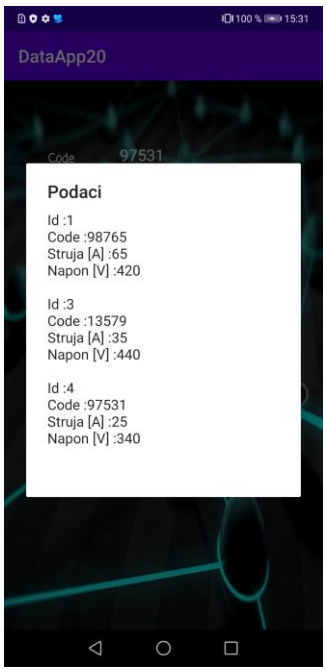
Slika 40: Izgled pokrenute aplikacije na uređaju P20

Kad je aplikacija pokrenuta upisujemo potrebne vrijednosti. Nakon unosa podataka na dnu ekrana dobivamo povratnu informaciju „Podaci upisani“. (slika 41).



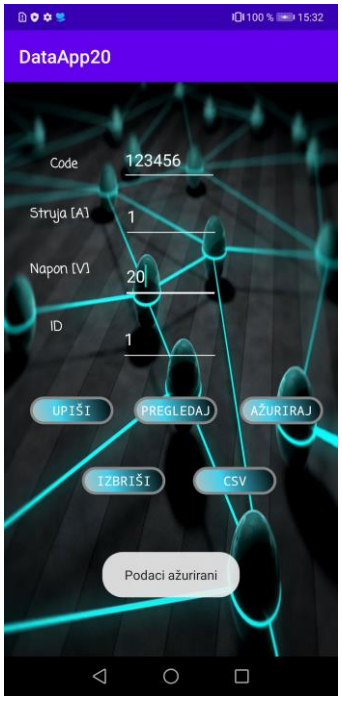
Slika 41: Primjer upisa vrijednosti u aplikaciju

Kada kliknemo gumbić „Pregledaj“ (slika 42) pojavljuje se prikaz svih podataka koje smo upisivali.



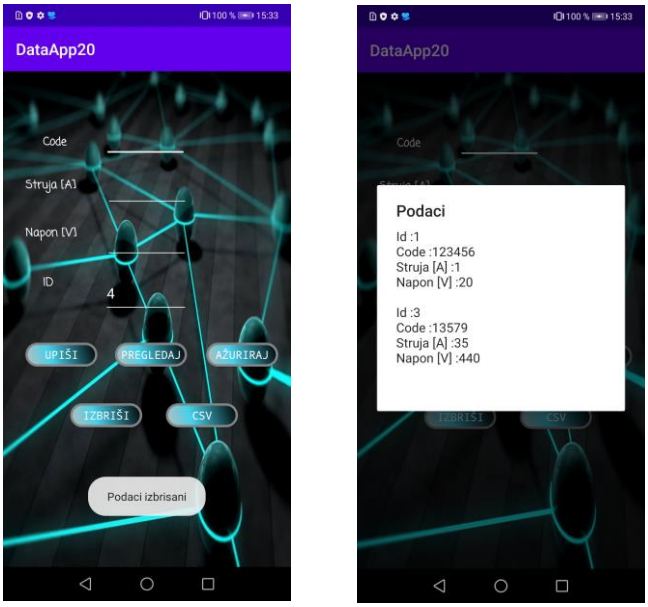
Slika 42: Prikazani podaci u aplikaciji

Da bi ažurirali podatke potrebno je prvo unijeti ID koji pripada određenim podacima (tablici), a nakon toga nove, ažurne podatke (slika 43). Tako su u ovom primjeru ažurirani podaci (serijski broj, struja i napon) čiji ID iznosi 1.



Slika 43: Primjer ažuriranja podataka i povratna informacija

Da bi izbrisali podatke (relaciju) potrebno je upisati ID i kliknuti gumbić „Izbriši“ (slika 44). U ovom primjeru izbrisani su podaci čiji ID iznosi 4.



Slika 44: Primjer brisanja podataka

Klikom na gumbić „CSV“ otvara nam se novi prozor na kojem imamo opciju povratka i opciju čitanja CSV datoteke (slika 45).

Slika 45: Izgled novog prozora klikom na gumbić CSV

Klikom gumbića „Ispis“ na zaslon se ispisuje sadržaj CSV datoteke (slika 46).

Slika 46: Ispis sadržaja CSV datoteke

7. Zaključak

Svi prijenosni uređaji kao što su pametni telefoni, tablet i sl. postali su svakodnevno prisutno sredstvo. Budući da do određene mjere mogu zamijeniti stolna računala, pronalaze svoju primjenu u gotovo svim područjima rada. Tako Android čini prvi izbor u odabiru tih uređaja prema podacima od Global Stats koji navode da 75.25% čini korisnike Androida, a preostalih 25.15% čine korisnici iOS-a. Jednostavan je za koristiti, korisnik vrlo lako može prilagoditi sučelje prema svojoj volji i ono vrlo važno – to je operativni sustav kod kojeg možemo mijenjati kod prema svojim potrebama. Olakšanje u radu pruža bogata baza biblioteka te funkcionalnosti za izradu aplikacija.

Kod razvoja aplikacije „DataApp20“ prikazana je upotreba nekih metoda i funkcija koje su korištene za izradu Android aplikacije. Pojašnjeni su pojmovi vezani za Android, relacijsku bazu podataka, Android Studio, te je pojašnjen razvoj aplikacije.

Potrebno je spomenuti da prilikom izrade novih aplikacija nije potrebno veliko predznanje nekog specifičnog programskog jezika. Naravno, potrebno je uložiti truda, rada, vremena, volje i istraživanja i moguće je izraditi aplikaciju za privatne, ali i poslovne procese. Ukoliko prilikom razvoja aplikacije dolazi do potrebe za dodatnim saznanjima, moguće je koristiti se dostupnim bibliotekama sa Interneta, ali i forumima na kojima su Android programeri, čineći neku vrstu zajednice, aktivni i spremni pomoći. Daljnji razvitak projekata poželjan je koristeći GitLab platformu gdje je moguće razvijati projekte unutar zajednice, a može se i provoditi integracija dostupnih open source projekata.

Sveučilište Sjever

IZJAVA O AUTORSTVU
I
SUGLASNOST ZA JAVNU OBJAVU

Završni/diplomski rad isključivo je autorsko djelo studenta koji je isti izradio te student odgovara za istinitost, izvornost i ispravnost teksta rada. U radu se ne smiju koristiti dijelovi tuđih radova (knjiga, članaka, doktorskih disertacija, magistarskih radova, izvora s interneta, i drugih izvora) bez navođenja izvora i autora navedenih radova. Svi dijelovi tuđih radova moraju biti pravilno navedeni i citirani. Dijelovi tuđih radova koji nisu pravilno citirani, smatraju se plagijatom, odnosno nezakonitim prisvajanjem tuđeg znanstvenog ili stručnoga rada. Sukladno navedenom studenti su dužni potpisati izjavu o autorstvu rada.

Ja, Lovro Šalamon *(ime i prezime)* pod punom moralnom, materijalnom i kaznenom odgovornošću, izjavljujem da sam isključivi autor/~~ica~~ završnog/~~diplomskog~~ *(obrisati nepotrebno)* rada pod naslovom Razvoj mobilne BDBMS aplikacije za praćenje u industriji *(upisati naslov)* te da u navedenom radu nisu na nedozvoljeni način (bez pravilnog citiranja) korišteni dijelovi tuđih radova.

Student/ica:
(upisati ime i prezime)

(vlastoručni potpis)

Sukladno Zakonu o znanstvenoj djelatnost i visokom obrazovanju završne/diplomske radove sveučilišta su dužna trajno objaviti na javnoj internetskoj bazi sveučilišne knjižnice u sastavu sveučilišta te kopirati u javnu internetsku bazu završnih/diplomskih radova Nacionalne i sveučilišne knjižnice. Završni radovi istovrsnih umjetničkih studija koji se realiziraju kroz umjetnička ostvarenja objavljuju se na odgovarajući način.

Ja, Lovro Šalamon *(ime i prezime)* neopozivo izjavljujem da sam suglasan/~~na~~ s javnom objavom završnog/~~diplomskog~~ *(obrisati nepotrebno)* rada pod naslovom Razvoj mobilne BDBMS aplikacije za praćenje u industriji *(upisati naslov)* čiji sam autor/ica.

Student/ica:
(upisati ime i prezime)

(vlastoručni potpis)

8. Literatura

[1] K. K. Mookhey, Nilesh Burghate: Linux: Security, Audit and Control Features, 2005.

[2] https://www.androidcentral.com/ [datum pristupa: 25.7.2020.]

[3] https://expandedramblings.com/index.php/android-statistics/ [datum pristupa: 9.9.2020.]

[4]https://statista.com/statistics/266210/number-of-available-applications-in-the-google-play-store/ [datum pristupa: 9.9.2020.]

[5] P.K. Dixit: Android, 2014.

[6] IntroBooks Team: History of Google Android, 2020.

[7] https://www.androidcentral.com/ [datum pristupa: 25.7.2020.]

[8] Mladen Varga: Baze podataka: Konceptualno, logičko i fizičko modeliranje podataka, 2016.

[9] Mike Owens: The Defenitive Guide to SQLite, 2006.

[10] https://www.sqlite.org/index.html [datum pristupa: 28.7.2020.]

[11] Z. Skočir, I. Matasić, B. Vrdoljak: Organizacija obrade podataka, 2007.

[12] https://developer.android.com/studio/intro [datum pristupa: 28.7.2020.]

[13] https://commons.wikimedia.org/wiki/File:Android_Studio_icon.svg [9.9.2020.]

Popis slika

Popis tablica