

TEHNIČKA ŠKOLA RUĐERA BOŠKOVIĆA
GETALDIĆEVA 4, ZAGREB

ZAVRŠNI STRUČNI RAD:
Music Player za Android

MENTOR: Ivan Šaban

UČENIK: Fran Bosančić
RAZRED: 4.B

Zagreb, travanj 2023.

Sadržaj

Sadržaj	ii
1. Uvod	1
2. Teorijski dio.....	2
2.1. Android Studio	2
2.1.1. Gradle	4
2.2. Java	6
2.2.1. Korištenje Jave za izradu Android aplikacija.....	7
2.3. XML	8
3. Tehničko-tehnološki dio	9
3.1. Važne datoteke	11
3.2. Korisničko sučelje	12
3.2.1. Constraint layout i dp	13
3.2.2. Layout moje aplikacije	14
3.2.2.1. Activity_main	15
3.2.2.2. Pjesma_u_listi	16
3.2.2.3. Pustanje_pjesama	19
3.3. Kod za funkcionalnost aplikacije	21
3.3.1. Dozvola za čitanje podataka.....	22
3.3.2. Objašnjenje klasa.....	23
3.3.2.1. Main Activity	23
3.3.2.2. Adapter za pjesme.....	25
3.3.2.3. Pjesma.....	27
3.3.2.4. Aktivnost pjesama	28
3.3.3. Animacija	35
3.3.4. Ikone	35
3.3.5. Prikaz aplikacije na Android uređaju	37
4. Zaključak	39
5. Literatura	40
6. Popis slika.....	41
7. Popis tablica.....	42

1. Uvod

U današnje vrijeme, kada mobilni uređaji poput pametnih telefona postaju sveprisutni, glazbeni playeri su postali neizostavni dio svakodnevnog života milijuna ljudi diljem svijeta. Ovaj završni rad se bavi izradom Music Player-a za Android operativni sustav, koji bi omogućio korisnicima pristup njihovoj glazbi na jednostavan i intuitivan način. Cilj ovog rada je pružiti korisnicima izvrsno iskustvo korištenja aplikacije koja bi mogla pratiti i upravljati korisničkom glazbenom kolekcijom, pružajući im ugodno korisničko sučelje i jednostavan način za pronalaženje i preslušavanje pjesama. Kroz ovaj rad, bit će istraženi različiti aspekti razvoja glazbenog playera za Android, uključujući upotrebu različitih tehnologija i alata, dizajn korisničkog sučelja i funkcionalnosti, te sigurnost i stabilnost aplikacije. Ovaj rad je motiviran potrebom za izradom kvalitetnog Music Player-a za Android koji se ne bi koristio nikakvom vrstom monetizacije. popularnim glazbenim playerima na tržištu, pružajući korisnicima visoku razinu funkcionalnosti. Cilj ovog rada je prikazati proces razvoja Music Player-a za Android, od početne ideje do konačnog proizvoda. Reklame su postale neizostavan dio moderne online industrije i mobilnih aplikacija. Mnoge aplikacije su besplatne za preuzimanje i korištenje, no kako bi se isplatile i zaradile, koriste se reklame koje se prikazuju korisnicima. Međutim, mnogi korisnici smatraju reklame dosadnim i iritantnim, posebice kada su prečeste ili neprimjerene. Upravo zato, Music Player koji je razvijen bez reklama može biti veliki plus za korisnike. Ova aplikacija omogućuje korisnicima da uživaju u svojoj glazbi bez prekida i bez smetnji od dosadnih reklama. To znači da korisnici mogu pristupiti svojoj glazbenoj kolekciji i uživati u slušanju svojih omiljenih pjesama bez ikakvih ometanja, što može biti veliki plus za korisničko iskustvo. Ovakav pristup izbjegavanja reklama također može pomoći u poboljšanju percepcije brenda Music Player-a među korisnicima, koji će prepoznati kvalitetu i brigu koju je tim uložio u razvoj aplikacije. To može dovesti do veće privrženosti i zadovoljenosti korisnika, što može biti ključno za dugoročni uspjeh aplikacije na tržištu.

Konačni cilj ovog rada je izraditi Music Player za Android koji će biti jednostavan za korištenje, brz, stabilan i siguran, te koji će korisnicima omogućiti pristup njihovoj glazbenoj kolekciji bez reklama. U izradi ovog rada, korišteni su različiti alati i tehnologije, kao što su Android Studio, Java programski jezik, te različiti dodaci i biblioteke. Ovaj rad se sastoji od nekoliko poglavlja, koji će obrađivati različite aspekte razvoja Music Player-a, od početnih ideja i zahtjeva, preko dizajna korisničkog sučelja i funkcionalnosti, do implementacije i testiranja aplikacije.

2. Teorijski dio

Za izradu Music Player aplikacije koristio sam se Android Studijom, integriranim razvojnim okruženjem za razvoj mobilnih aplikacija za Android operativni sustav, te Java programskim jezikom koji je jedan od najpopularnijih programskih jezika koji se koriste za razvoj mobilnih aplikacija. Mogao sam se koristiti programskim jezikom Kotlin, ali sam se odlučio za korištenje Java, pošto je Java vrlo slična programskom jeziku C# kojim se koristimo u predmetu „Napredno i objektno programiranje“. Kombinacija ova dva alata omogućila mi je izradu visoko kvalitetne i funkcionalne aplikacije koja pruža korisnicima bogatstvo opcija i funkcija koje se očekuju od modernih mobilnih aplikacija. Korištenje Android Studija i Java programskog jezika omogućuje sigurnost, brzinu i stabilnost aplikacije te lako održavanje i skalabilnost aplikacije.

2.1. Android Studio

Android Studio je vrlo moćan alat za programiranje mobilnih aplikacija za Android operativni sustav. Razvijen je od strane Google-a, a besplatan je alat dostupan za preuzimanje i instalaciju na računala sa sustavima Windows, Mac ili Linux. Jedna od glavnih prednosti Android Studija je integracija s Android SDK-om (Software Development Kit). Android SDK je kolekcija različitih alata, biblioteka i drugih komponenti potrebnih za razvoj aplikacija za Android platformu. Android Studio dolazi s integriranim SDK-om i čini proces instalacije, konfiguracije i upravljanja SDK-om vrlo jednostavnim. Još jedna velika prednost Android Studija je integracija s Android Virtual Device Manager-om (AVD Manager), što omogućuje programerima da testiraju svoje aplikacije na virtualnim uređajima s različitim verzijama operativnog sustava, veličinama zaslona i drugim karakteristikama. Ovo omogućuje programerima da vide kako će aplikacija izgledati i funkcionirati na različitim uređajima prije nego što ju objave na tržište. Jedna od ključnih značajki Android Studija je integrirani Gradle Build sustav. Gradle je automatizacijski alat za izgradnju i održavanje projekata, a integracija s Android Studijem omogućuje programerima da jednostavno konfiguriraju i upravljaju procesom izgradnje aplikacija za Android. Ovo uključuje automatsko generiranje APK datoteka, što je datoteka koja se instalira na uređaje s Androidom. Android Studio također ima integriranu potporu za različite verzije Jave, uključujući Java 8, što olakšava pisanje boljeg i optimiziranog koda. Osim toga, Android Studio također podržava različite jezike programiranja, uključujući Kotlin, C++, Python i drugi. Android Studio nudi mnoge alate koji programerima olakšavaju razvoj aplikacija. Na primjer, Layout Editor je vizualni alat za dizajniranje

korisničkog sučelja, a omogućuje programerima da stvore složena korisnička sučelja bez potrebe za kodiranjem. Debugger je drugi koristan alat koji omogućuje programerima da pronalaze i otklanjaju pogreške u kodu. Android Studio također nudi mnoge druge značajke i alate, uključujući integraciju s Git-om, alat za analizu performansi i alat za testiranje performansi. Uz sve ove značajke i alate, Android Studio je vrlo fleksibilan alat koji podržava različite vrste aplikacija, uključujući aplikacije za mobilne uređaje, nosive uređaje, Android TV i druge platforme. Uz to, Android Studio podržava različite arhitekture aplikacija, uključujući arhitekturu zasnovanu na MVVM (Model-View-ViewModel) i MVP (Model-View-Presenter), što programerima omogućuje da razviju aplikacije koje su skalabilne i održive. Android Studio također ima integriranu mogućnost za objavljivanje aplikacija na Google Play Store, što omogućuje programerima da jednostavno objave svoje aplikacije i dosegnu milijune korisnika diljem svijeta. Android Studio je alat koji je izgrađen na temelju IntelliJ IDEA platforme, što ga čini vrlo stabilnim i pouzdanim alatom. Također, razvojni tim Google-a neprestano radi na poboljšanju Android Studija i dodavanju novih značajki kako bi se olakšao razvoj aplikacija. Uz sve ove značajke, Android Studio je vrlo popularan alat među programerima za razvoj aplikacija za Android. Uz to, Android platforma se široko koristi diljem svijeta, što programerima pruža veliku priliku za razvoj aplikacija koje mogu dosegnuti milijune korisnika.

Tablica 1: Povijest verzija Android Studia

VERZIJA	DATUM IZLASKA
1.0	Prosinac 2014.
1.1	Veljača 2015.
1.2	Travanj 2015.
1.3	Srpanj 2015.
1.4	Rujan 2015.
1.5	Studen 2015.
2.0	Travanj 2016.
2.1	Travanj 2016.
2.2	Rujan 2016.
2.3	Ožujak 2017.
3.0	Listopad 2017.
3.1	Ožujak 2018.
3.2	Rujan 2018.
3.3	Siječanj 2019.
3.4	Travanj 2019.
3.5	Kolovoz 2019.
3.6	Veljača 2020.
4.0	Svibanj 2020.
4.1	Listopad 2020.
4.2	Svibanj 2021.
Arctic Fox (2020.3.1)	Srpanj 2021.
Bumblebee (2021.1.1)	Siječanj 2022.
Chipmunk (2021.2.1)	Svibanj 2022.
Dolphin (2021.3.1)	Rujan 2022.
Electric Eel (2022.1.1)	Siječanj 2023.
Flamingo (2022.2.1)	Travanj 2023.

(Izvor: Android Developers i Wikipedia 2023)

2.1.1. Gradle

Gradle je alat za automatizaciju izgradnje projekata koji se razvio kako bi se olakšao i ubrzao proces izgradnje aplikacija. Uobičajeno, proces izgradnje aplikacije uključuje niz koraka, od kompiliranja koda do pakiranja aplikacije u izvršnu datoteku, i ovi koraci mogu biti vrlo složeni i zahtjevni za izvođenje ručno. Gradle rješava ovaj problem automatizirajući ovaj proces. Glavna prednost Gradlea je njegova fleksibilnost i skalabilnost, što ga čini savršenim alatom za projekte svih veličina. Može se prilagoditi različitim projektima i konfiguracijama, a također podržava velik broj dodataka (plugins) koji olakšavaju rad na određenim vrstama projekata. To znači da se može lako prilagoditi različitim potrebama različitih timova i projekata.

Gradle koristi Groovy kao jezik za pisanje skripti, što znači da programeri mogu pisati skripte koje su jednostavne za razumijevanje i koje se lako mogu prilagoditi različitim projektima. Također, konfiguracija se može pisati i u programskom jeziku Kotlin, što je česta praksa u Android razvoju.

Kako bi koristili Gradle u Android Studiju, programeri moraju definirati Gradle skriptu koja opisuje kako će se njihov projekt izgraditi. Ta skripta se obično naziva "build.gradle" i nalazi se u korijenskom direktoriju projekta. U toj skripti programeri definiraju različite faze izgradnje, ovisno o potrebama projekta, te definiraju ovisnosti između različitih dijelova projekta. Na primjer, u Gradle skripti se mogu definirati zadaci za čišćenje (clean), kompilaciju koda (compile), testiranje (test), pakiranje aplikacije (assemble) i sl. Također se može definirati kako se ovisnosti (dependencies) između različitih modula projekta upravljaju i kako se mogu preuzimati i integrirati različiti dodaci (plugins). Jedna od ključnih prednosti Gradlea je njegova sposobnost da se integrira s drugim alatima i sustavima za izgradnju. Na primjer, može se lako integrirati s Jenkinsom, Bamboo, TeamCityjem i drugim alatima za kontinuiranu integraciju (CI) i kontinuirano isporučivanje (CD), što olakšava automatizaciju cjelokupnog procesa razvoja i isporuke aplikacija. Gradle također podržava paralelno izvođenje zadatka (task), što znači da se zadaci mogu izvršavati istovremeno, što može znatno ubrzati proces izgradnje za velike projekte. Osim toga, Gradle koristi inkrementalnu izgradnju (incremental builds), što znači da će samo dijelovi koda koji su se promijenili od posljednje izgradnje biti ponovno izgrađeni, što dodatno ubrzava proces izgradnje. Uz podršku za različite programsko okruženja, Gradle također nudi široku podršku za upravljanje ovisnostima (dependency management). Programeri mogu lako dodati ovisnosti u svoj projekt i upravljati njima korištenjem Maven ili Ivy repozitorija. Gradle također nudi fleksibilan sustav za konfiguriranje varijabli okoline, što znači da programeri mogu jednostavno prilagoditi svoj projekt različitim okruženjima. Na primjer, mogu se definirati različite postavke za razvojno okruženje, testno okruženje i produkcijsko okruženje. Uz sve to, Gradle je vrlo dobro dokumentiran, a programeri mogu pronaći mnogo resursa na internetu koji će im pomoći u radu s ovim alatom. Također, Gradle ima aktivnu zajednicu korisnika koji stalno rade na poboljšanju ovog alata i razvijaju nove dodatke za olakšavanje rada na projektima. Gradle je vrlo moćan i fleksibilan alat koji može biti od velike koristi za razvoj Android aplikacija. Uz podršku za različite programsko okruženja, fleksibilan sustav za konfiguriranje varijabli okoline i veliku količinu dodataka, Gradle je idealan alat za automatizaciju procesa izgradnje aplikacija za različite projekte i timove.

2.2. Java

Java je visokonaponski programski jezik koji se izvorno razvio u Sun Microsystemsu, a kasnije je stekao popularnost kao jezik za razvoj softvera u mnogim industrijama, uključujući poslovne aplikacije, mobilne aplikacije, web aplikacije, desktop aplikacije i igre. Java je objektno orijentirani jezik, što znači da programeri koriste objekte i klase da bi organizirali svoj kod na logičan način. Objekti su instance klasa koje sadrže podatke i metode za obradu tih podataka. Java također podržava apstraktnu klasu i sučelja, što omogućuje programerima da razvijaju modularne aplikacije koje su jednostavne za održavanje i proširivanje. Java ima bogatu bazu biblioteka i okvira (frameworks), koji se koriste za razvoj aplikacija u Javi. Ovi alati omogućuju programerima da brzo i jednostavno razvijaju aplikacije visoke kvalitete, bez potrebe za pisanjem puno koda. Java također ima ugrađene biblioteke za rad s grafičkim sučeljima, mrežnim protokolima, bazama podataka i drugim ključnim tehnologijama koje su potrebne za razvoj modernih aplikacija. Java se koristi za razvoj aplikacija u različitim vrstama industrija, uključujući financije, telekomunikacije, medicinu, obrazovanje i mnoge druge. Java se također često koristi za razvoj aplikacija za Android platformu, jer Android koristi Java programski jezik. Java ima ugrađene mehanizme za upravljanje iznimkama (exceptions), što omogućuje programerima da lakše rade s greškama i iznimkama koje mogu nastati tijekom izvršavanja programa. Java također ima mehanizme za upravljanje memorijom koji automatski upravljaju memorijom koju koristi aplikacija, što čini Javu pouzdanom i sigurnom platformom za razvoj velikih, složenih aplikacija. Java također ima ugrađene mehanizme za višenitno programiranje, što omogućuje programerima da razviju aplikacije koje mogu istovremeno obrađivati više zadataka i izvršavati ih u različitim nitima (threads). Ova značajka omogućuje Javi da bude učinkovita u radu s velikim količinama podataka i u rješavanju problema koji zahtijevaju brzu obradu podataka. Java je također poznata po svojoj sigurnosti i pouzdanosti, što je ključno za mnoge vrste aplikacija koje se koriste u poslovnim okruženjima. Java ima ugrađene mehanizme za zaštitu od sigurnosnih prijetnji, poput autentikacije i autorizacije, koji pomažu u zaštiti podataka od neovlaštenog pristupa. Java također ima ugrađene mehanizme za enkripciju podataka i digitalni potpis, što omogućuje programerima da stvore sigurne aplikacije koje su otporne na napade hakera. Jedna od velikih prednosti Jave je njezina platformna neovisnost. Java aplikacije se mogu izvršavati na različitim operacijskim sustavima, bez potrebe za prilagođavanjem koda za svaku platformu zasebno. To je moguće zahvaljujući Java Virtual Machine (JVM), koja interpretira Java kod i izvršava ga na različitim operacijskim sustavima. Java se također koristi u razvoju web aplikacija, uključujući JavaServer Pages (JSP) i JavaServer Faces (JSF). JSP omogućuje programerima da stvore dinamičke web stranice koje sadrže Java kod, dok JSF omogućuje razvoj kompleksnih web aplikacija koje se sastoje od više stranica. Uz sve ove značajke, Java ima veliku zajednicu programera koja

redovito doprinosi razvoju novih alata, biblioteka i okvira za Javu. Ovo otvara mnoge mogućnosti za programere da razviju inovativne aplikacije i učinkovitije procese razvoja. Ukratko, Java je objektno orijentirani, visokonaponski programski jezik s bogatim skupom biblioteka i okvira koji se koristi u mnogim vrstama industrija. Java je poznata po svojoj platformnoj neovisnosti, sigurnosti, pouzdanosti i mogućnosti višenitnog programiranja. Java je popularna za razvoj desktop, web i mobilnih aplikacija, kao i za razvoj aplikacija za Android platformu.

Tablica 2: Povijest verzija Jave

JDK Beta	1995.
JDK 1.0	Siječanj, 1996.
JDK 1.1	Veljača, 1997.
J2SE 1.2	Prosinac, 1998.
J2SE 1.3	Svibanj, 2000.
J2SE 1.4	Veljača, 2002.
J2SE 5.0	Rujan, 2004.
Java SE 6	Prosinac, 2006.
Java SE 7	Srpanj, 2011.
Java SE 8 (LTS)	Ožujak, 2014.
Java SE 9	Rujan, 2017.
Java SE 10	Ožujak, 2018.
Java SE 11 (LTS)	Rujan, 2018.
Java SE 12	Ožujak, 2019.
Java SE 13	Rujan, 2019.
Java SE 14	Ožujak, 2020.
Java SE 15	Rujan, 2020.
Java SE 16	Ožujak, 2021
Java SE 17 (LTS)	Rujan, 2021.
Java SE 18	Ožujak, 2022.
Java SE 19	Rujan, 2022.
Java SE 20	Ožujak, 2023.

(Izvor: Android Developers i Wikipedia 2023)

2.2.1. Korištenje Jave za izradu Android aplikacija

U korištenju Jave za izradu Android aplikacija, programeri mogu koristiti različite koncepte i tehnologije kako bi izgradili visoko kvalitetne i funkcionalne aplikacije. Jedan od najvažnijih koncepta za izradu Android aplikacija je razvoj na Android platformi, što znači da se aplikacije grade za specifičnu platformu i njezine zahtjeve. Jedan od najvažnijih alata za

izradu Android aplikacija je Android SDK, što uključuje set alata, biblioteka i drugih resursa koji se koriste za razvoj aplikacija za Android. Ovaj kit za razvoj softvera uključuje Android Studio, koji je integrirano razvojno okruženje koje omogućuje programerima da lako stvore, testiraju i implementiraju aplikacije na Android platformi. Kada koriste Javu za izradu Android aplikacija, programeri koriste Android SDK i Android Studio za razvoj aplikacija. Android SDK uključuje biblioteke koje omogućuju programerima da komuniciraju s različitim hardverskim i softverskim komponentama na Android uređaju, uključujući zaslone osjetljive na dodir, kameru, Wi-Fi, Bluetooth i GPS. Korištenje Jave za izradu Android aplikacija omogućuje programerima da koriste razne biblioteke i okvire za stvaranje aplikacija. Jedan od najpopularnijih je Android Jetpack, koji je set biblioteka i alata koji omogućuju programerima da lakše razvijaju aplikacije za Android. Jetpack sadrži komponente za upravljanje životnim ciklusom aktivnosti i fragmenata, navigaciju između zaslona, upravljanje podacima, izradu korisničkog sučelja, testiranje i mnogo više. Osim Jetpack-a, programeri mogu koristiti i druge biblioteke i okvire poput Retrofit za upravljanje mrežnim pozivima, Picasso za učitavanje slika, ButterKnife za brži razvoj korisničkog sučelja, i mnoge druge. Korištenje Jave za izradu Android aplikacija omogućuje programerima da razvijaju aplikacije koje su brze, pouzdane i sigurne. Programeri također mogu koristiti različite tehnike poput materijalnog dizajna za stvaranje privlačnih i modernih korisničkih sučelja. Uz to, Java pruža snažne značajke programskog jezika poput objektno orijentirane paradigme, lambda izraza, generičkih tipova i mnogih drugih značajki koje olakšavaju razvoj aplikacija.

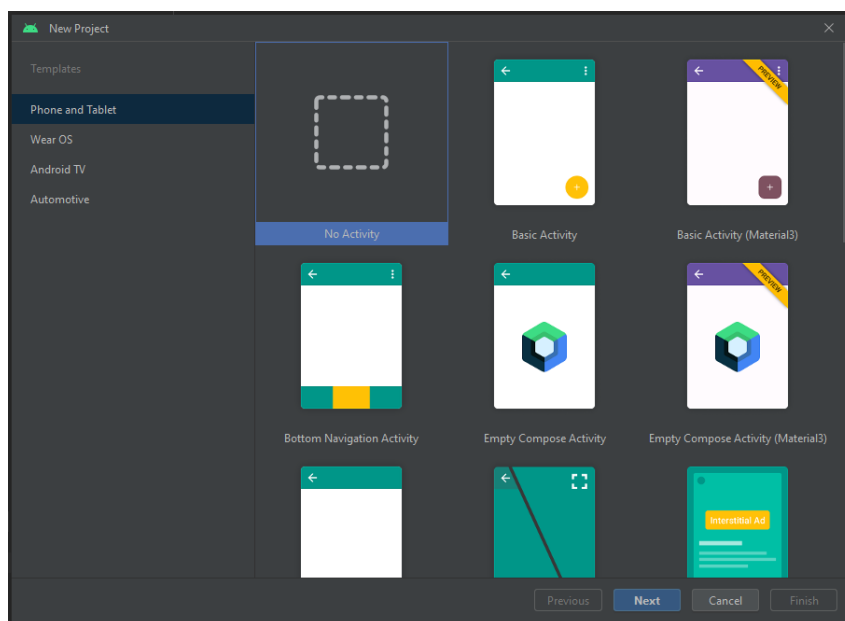
2.3. XML

XML (Extensible Markup Language) se često koristi za definiranje korisničkog sučelja u Android aplikacijama. Osim toga, XML se također koristi za definiranje različitih drugih elemenata u Android aplikacijama, kao što su datoteke manifesta, resursi, itd. Korištenje XML-a u Android aplikacijama omogućava razdvajanje koda i izgleda aplikacije. XML datoteke se koriste za definiranje izgleda korisničkog sučelja, dok se Java kod koristi za logiku aplikacije. To znači da dizajneri mogu raditi na izgledu korisničkog sučelja bez utjecaja na kod developera. XML se koristi za definiranje različitih elemenata korisničkog sučelja, kao što su gumbi, tekstualna polja, slike, itd. Svaki element ima svoje atribute koji se mogu definirati u XML-u. Primjerice, atributi gumba mogu uključivati boju pozadine, veličinu fonta, itd. XML se također koristi za definiranje različitih resursa aplikacije, kao što su slike, nizovi, stilovi, itd. Na taj način, resursi se mogu lako ponovno koristiti u različitim dijelovima aplikacije i pojednostavljuje se održavanje aplikacije. Jedna od ključnih prednosti korištenja XML-a u Android aplikacijama je mogućnost podrške za više jezika. Android omogućuje stvaranje različitih verzija datoteka resursa za različite jezike, što omogućuje lokalizaciju aplikacije za

različite tržišta i korisnike. Kao što smo već spomenuli, XML se također koristi za definiranje datoteke manifesta, koja opisuje aplikaciju Android sustavu. Ova datoteka sadrži informacije o aplikaciji, kao što su naziv, ikona, dozvole, itd. Android Studio automatski generira manifest datoteku tijekom stvaranja nove aplikacije, ali programeri mogu ručno uređivati datoteku kako bi prilagodili određene značajke aplikacije. Kao što smo vidjeli, korištenje XML-a u Android aplikacijama je ključno za definiranje korisničkog sučelja i resursa aplikacije. XML omogućuje razdvajanje koda i izgleda aplikacije, što pojednostavljuje razvoj aplikacije i održavanje aplikacije.

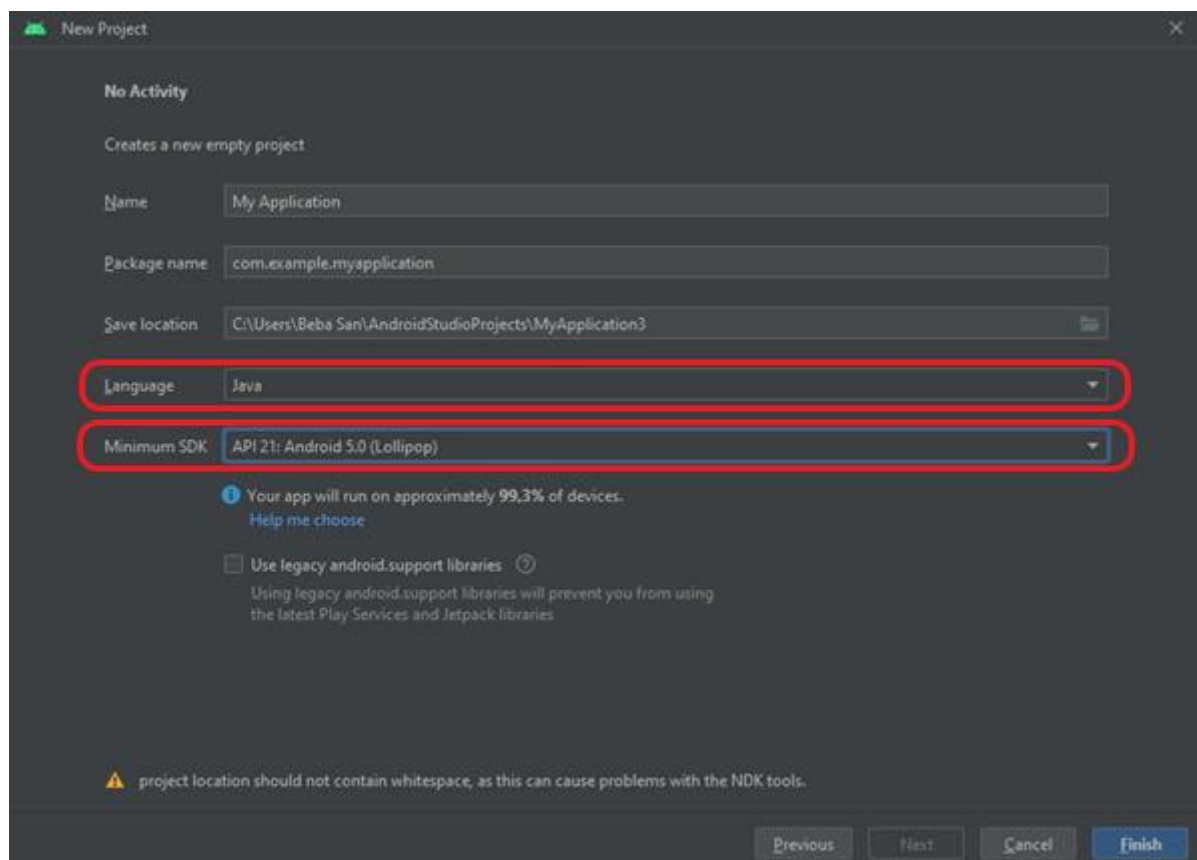
3. Tehničko-tehnološki dio

Za izradu bilo kakve aplikacije u Android Studiu prvobitno je potrebno napraviti novi projekt te odabrati koju vrstu Activity-a ćemo koristiti. Za izradu svoje Music Player aplikacije odabrao sam "No Activity" te sam naknadno sam kreirao korisnička sučelja.



Slika 1: Izrada novog projekta u Android Studiu

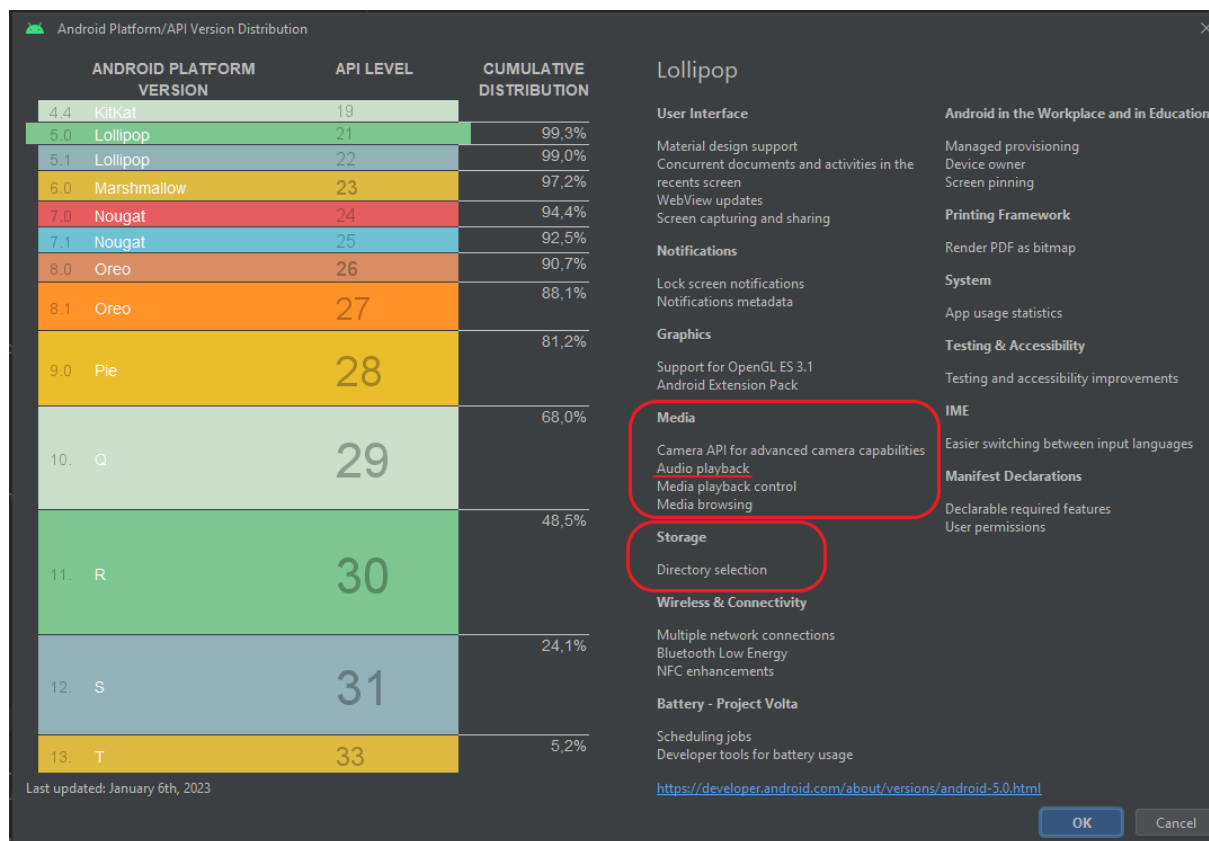
Nakon odabira vrste Activity-a koju ćemo koristiti potrebno je ispuniti kao što je navedeno na slici 2.



Slika 2: Odabir postavki novog projekta

Najvažnije kod ovog koraka je odabira programskog jezika te Minimum SDK. Za svoj projekt sam odabrao za jezik Javu te za Minimum SDK: API 21: Android 5.0 (Lollipop). Odabir prave verzije Minimum SDK-a ovisi o funkcionalnostima koje korisnik planira koristiti u aplikaciji, kao i o ciljanoj publici za koju je aplikacija namijenjena. Ako se korisnik odluči za stariju verziju Minimum SDK-a, to će mu omogućiti da aplikacija bude kompatibilna sa širim rasponom uređaja, uključujući starije modele telefona i tableta. Međutim, korisnik mora biti svjestan da starija verzija Minimum SDK-a može ograničiti funkcionalnosti koje može koristiti u aplikaciji, budući da neke od najnovijih značajki Android platforme možda neće biti podržane na starijim verzijama operativnog sustava. Ako se korisnik odluči za noviju verziju Minimum SDK-a, to će mu omogućiti da iskoristi najnovije značajke i funkcionalnosti Android platforme, ali će aplikacija biti kompatibilna samo s uređajima koji podržavaju tu verziju operativnog sustava ili novije verzije. Stoga, kada korisnik odlučuje o Minimum SDK verziji, treba razmotriti svoje potrebe i ciljeve, kao i željenu publiku za aplikaciju.

Ja sam se odlučio za odabir Minimum SDK: API 21: Android 5.0 (Lollipop) pošto ga podržava većina mobilnih uređaja te ta verzija podržava upravljanjem memorijom uređaja i kontroliranjem medija tj. u mom slučaju zvuka.



Slika 3: Distribucija API verzija

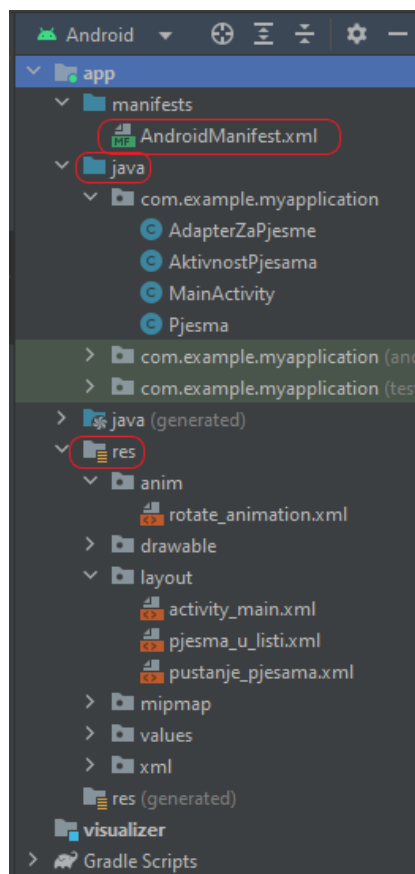
3.1. Važne datoteke

Datoteke u Android Studiu koje imaju specifične te vrlo važne funkcije u razvoju Android aplikacija:

1. AndroidManifest.xml - ova datoteka se koristi za definiranje različitih informacija o aplikaciji. To uključuje informacije o autoru, verziji aplikacije, dozvolama koje aplikacija zahtijeva i komponentama aplikacije kao što su aktivnosti, usluge, primatelji i davatelji sadržaja. Ova datoteka je obavezna i ne može biti izostavljena jer Android sistem koristi ove informacije kako bi mogao instalirati i upravljati aplikacijom.
2. java/ - ovo je direktorij koji sadrži izvorni kod aplikacije napisan u programskom jeziku Java ili Kotlin. Ovdje se nalaze sve klase, metode i funkcije koje se koriste u aplikaciji. Ovo je glavni direktorij u kojem razvijate svoju aplikaciju.

3. res/ - ovo je direktorij koji sadrži sve resurse koji se koriste u aplikaciji. To uključuje različite vrste datoteka kao što su slike, ikone, stringovi, layout datoteke, stilovi, boje, animacije i drugo. Ovo je važan direktorij jer se u njemu nalaze svi vizualni i zvučni elementi koje aplikacija koristi za svoje sučelje i interakciju s korisnikom.

Ukratko, AndroidManifest.xml se koristi za definiranje osnovnih informacija o aplikaciji, java/ direktorij za pisanje izvornog koda aplikacije, a res/ direktorij za spremanje svih resursa koji se koriste u aplikaciji.

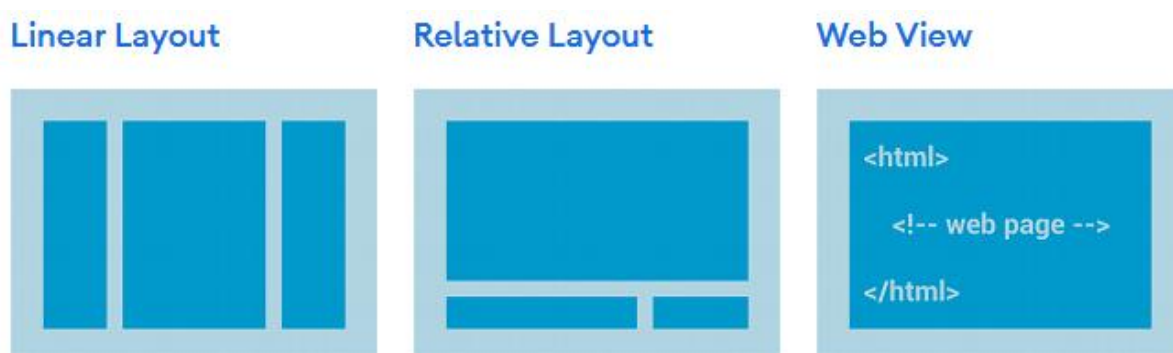


Slika 4: Prikaz strukture datoteka u Android Studiu

3.2. Korisničko sučelje

Layout u Android Studiju se koristi za definiranje izgleda korisničkog sučelja aplikacije. To uključuje postavljanje različitih elemenata kao što su gumbi, tekstualna polja, slike, liste i drugi vizualni elementi na ekranu aplikacije. Layout se obično definira pomoću XML koda, koji se može uređivati u Android Studiju putem vizualnog uređivača ili ručno pisanjem koda. Vizualni uređivač omogućuje programerima da postave i mijenjaju elemente koristeći povlačenje i

ispuštanje, kao i podešavanje svojstava elemenata putem grafičkog korisničkog sučelja. Postoji nekoliko različitih vrsta layout-a koje se mogu koristiti u Android Studiju, uključujući ConstraintLayout, LinearLayout, RelativeLayout i GridLayout. Svaki od ovih layout-a ima svoje prednosti i nedostatke, ovisno o zahtjevima aplikacije i složenosti dizajna korisničkog sučelja. Programeri također mogu koristiti različite dimenzije i mjernu jedinicu za postavljanje elemenata u layout-u, kao što su pikseli, dip-ovi (density independent pixels) ili postotci. To omogućuje aplikacijama da budu skalabilne i prilagodljive različitim veličinama zaslona i gustoćama piksela. Ukratko, layout u Android Studiju omogućuje programerima da definiraju izgled korisničkog sučelja svoje aplikacije putem XML koda ili vizualnog uređivača, koristeći različite vrste layout-a i mjernih jedinica za postavljanje elemenata na zaslonu.

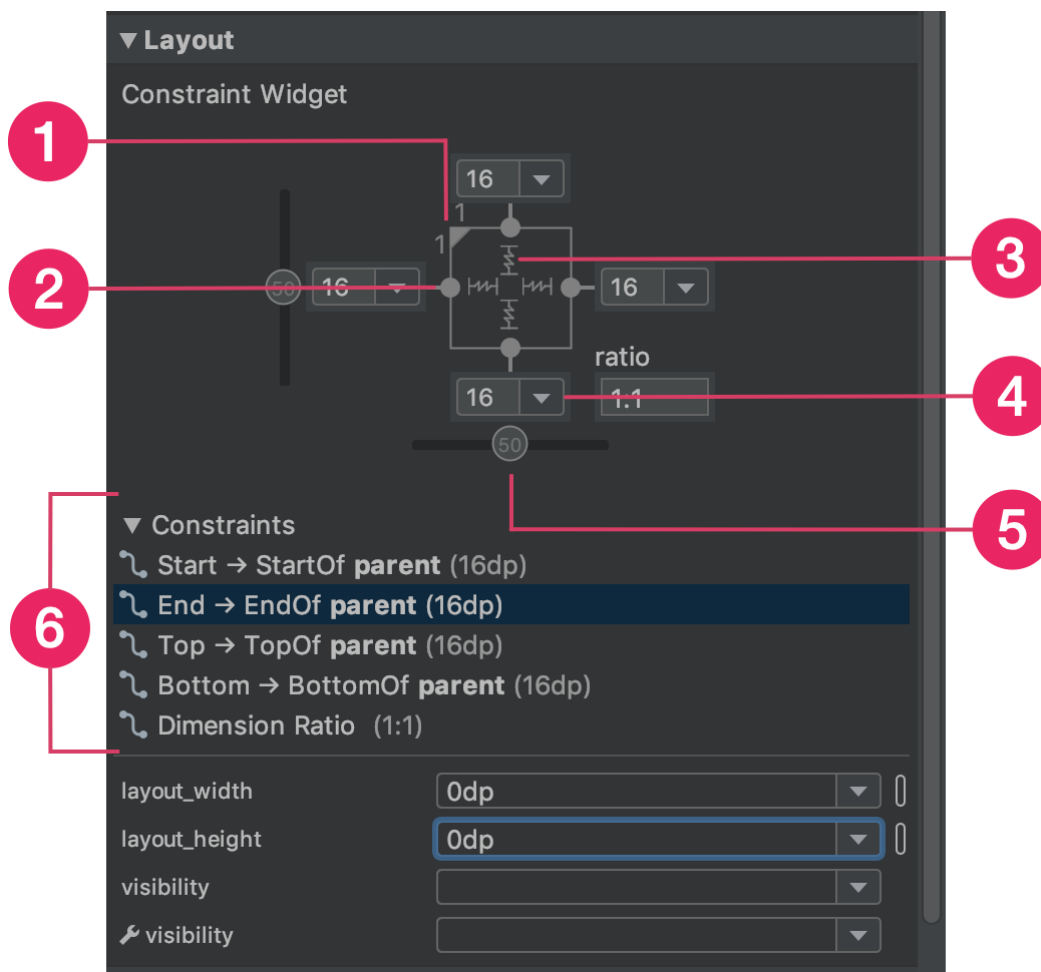


Slika 5: Različite vrste layout-a (Izvor: Android Developers)

3.2.1. Constraint layout i dp

Za svoju aplikaciju sam se koristio Constraint Layout-om i dp-om. Constraint Layout je vrsta layout-a u Android Studiju koja omogućuje programerima da postave elemente korisničkog sučelja na zaslon aplikacije pomoću ograničenja (constraints). To znači da se elementi postavljaju u odnosu na druge elemente ili granice zaslona, što pruža veću fleksibilnost u postavljanju elemenata u odnosu na druge vrste layout-a. Constraint Layout također omogućuje programerima da koriste različite dimenzije i mjernu jedinicu za postavljanje elemenata. Jedna od tih mjernih jedinica je "dp" (density-independent pixel), koja se koristi za postavljanje veličine elemenata na zaslonu neovisno o gustoći piksela. To znači da se veličina elemenata neće mijenjati kada se aplikacija prikazuje na različitim uređajima s različitim gustoćama piksela. Korištenje dp-a umjesto fiksnih piksela (px) osigurava da se elementi korisničkog sučelja prilagođavaju veličini zaslona i gustoći piksela, što poboljšava korisničko iskustvo i olakšava skaliranje aplikacije na različite uređaje. Ukratko, Constraint Layout omogućuje programerima da postave elemente korisničkog sučelja aplikacije pomoću

ograničenja, što pruža veću fleksibilnost u postavljanju elemenata. Također, korištenje dp-a umjesto fiksnih piksela osigurava da se veličina elemenata prilagođava veličini zaslona i gustoći piksela, poboljšavajući tako korisničko iskustvo i skalabilnost aplikacije.



Slika 6: Postavljanje atributa Constraint layout-a kroz grafičko sučelje Android Studia (Izvor: Android Developers)

3.2.2. Layout moje aplikacije

Layout moje aplikacije se sastoji od tri dijela: activity_main, pjesma_u_listi i pustanje_pjesama.

1. activity_main.xml - ovaj layout predstavlja glavnu aktivnost aplikacije i sastoji se od ConstraintLayout-a i RecyclerView-a, što omogućuje prikaz liste pjesama. Također, postoje i gumb "OSVJEŽI", koji korisniku omogućuje ponovno učitavanje liste pjesama.

2. pjesma_u_listi.xml - ovaj layout predstavlja pojedinačnu stavku u listi pjesama, a sastoji se od ConstraintLayout-a i tri elementa: slike ikone, naziva pjesme i gumba za brisanje. Ovaj se layout koristi za svaku stavku u RecyclerView-u.
3. pustanje_pjesama.xml - ovaj layout predstavlja zasebnu aktivnost u aplikaciji i koristi se za prikaz informacija o trenutnoj pjesmi koja se reproducira. Sastoji se od ConstraintLayout-a i jednog TextView-a za prikaz naziva pjesme. Ovaj se layout poziva kada korisnik odabere određenu pjesmu iz liste u RecyclerView-u.

3.2.2.1. Activity_main

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    tools:context=".MainActivity">

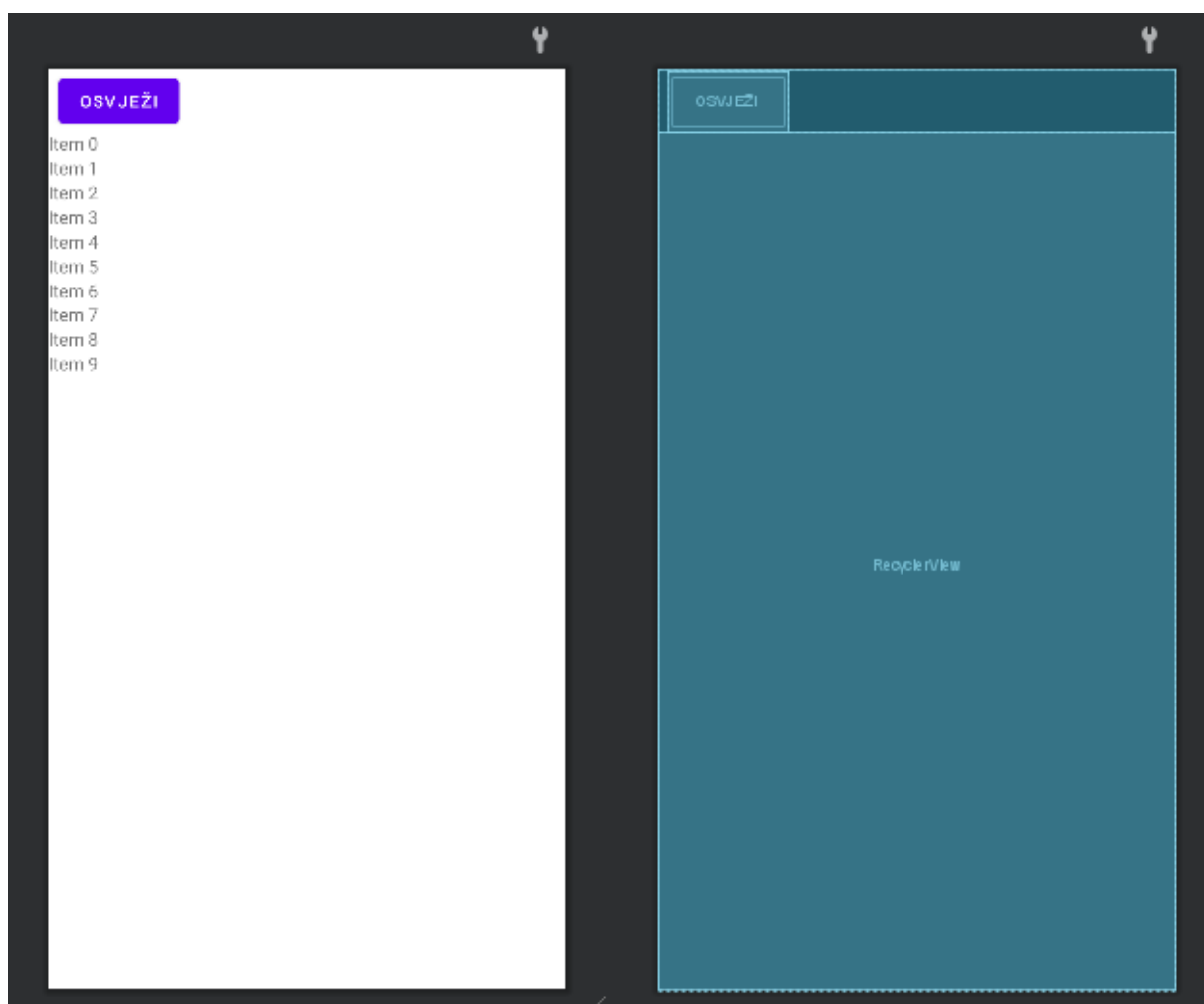
    <androidx.recyclerview.widget.RecyclerView
        android:id="@+id/RecyclerView"
        android:layout_width="match_parent"
        android:layout_height="0dp"
        android:layout_marginTop="50dp"
        android:layout_weight="1"
        android:background="@android:color/white"
        android:scrollbars="vertical"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.0"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintVertical_bias="0.0" />

    <Button
        android:id="@+id/OsvjeziGumb"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginStart="8dp"
        android:text="OSVJEŽI"
        app:layout_constraintBottom_toTopOf="@+id/RecyclerView"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.0"
        app:layout_constraintStart_toStartOf="parent" />

</androidx.constraintlayout.widget.ConstraintLayout>
```

Ovaj kod je XML kod koji opisuje layout za Android aplikaciju. Prvi redak određuje verziju XML formata i karakternu enkodiranje koje se koristi. Sljedeće tri linije definiraju imenski prostor za Android, ConstraintLayout i tools. Nakon toga slijede atributi za ConstraintLayout. android:layout_width i android:layout_height postavljaju dimenzije layouta na "match_parent",

što znači da će se layout proširiti preko cijelog ekrana. Ovaj layout ima orijentaciju "vertical", što znači da će se sve komponente layouta slagati jedna ispod druge. Nakon toga dolazi RecyclerView, koji je popis koji se prikazuje u aplikaciji. Njegov id je RecyclerView, a dimenzije su postavljene tako da se proteže cijelom širinom ekrana i visinom koja će se automatski prilagoditi veličini ekrana. Margin top je postavljen na 50dp kako bi se izbjeglo preklapanje s gumbom. Background je bijeli, a scrollbars je postavljen na "vertical" kako bi se prikazala vertikalna traka za pomicanje ako je popis predugačak. Konačno, postoji gumb s id-em OsvjeziGumb koji je pričvršćen na dno RecyclerView-ja. Dimenzije su postavljene na "wrap_content", što znači da će se dimenzije prilagoditi veličini



Slika 7: activity_main izgled

3.2.2.2. Pjesma_u_listi

Kod:

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/IDRelativeLayoutPjesme"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_marginLeft="5dp"
    android:layout_marginTop="5dp"
    android:layout_marginRight="5dp"
    android:background="@drawable/zaobljenirubovi"
    android:padding="5dp"
    android:paddingTop="5dp"
    android:paddingBottom="5dp">

    <ImageView
        android:id="@+id/IDZaIkonu"
        android:layout_width="40dp"
        android:layout_height="40dp"
        android:layout_centerVertical="true"
        android:src="@drawable/ikonamuzike"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

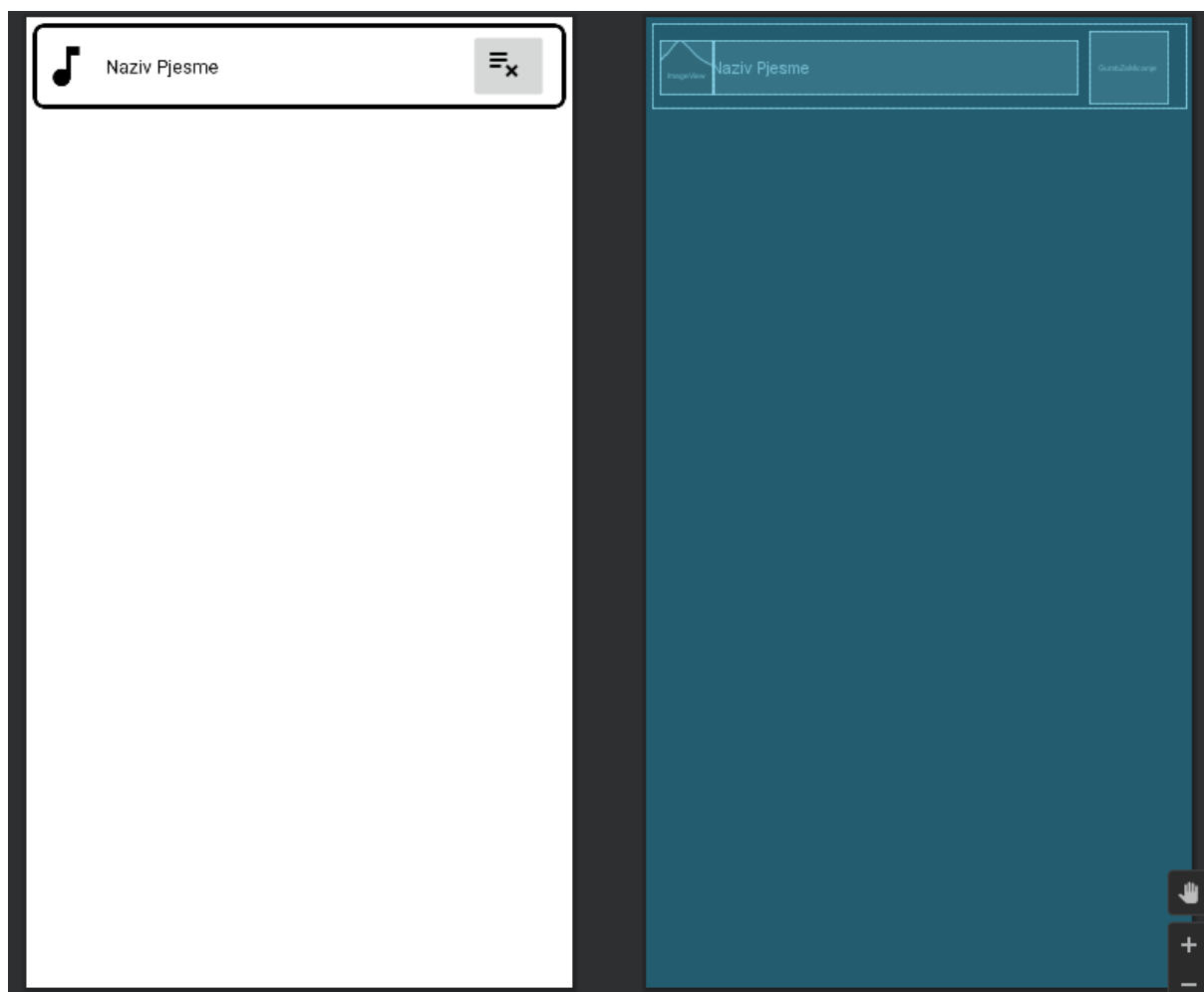
    <TextView
        android:id="@+id/ImePjesme"
        android:layout_width="275dp"
        android:layout_height="40dp"
        android:layout_centerVertical="true"
        android:layout_toEndOf="@id/IDZaIkonu"
        android:ellipsize="marquee"
        android:marqueeRepeatLimit="marquee_forever"
        android:padding="10dp"

        android:scrollHorizontally="true"
        android:singleLine="true"
        android:text="Naziv Pjesme"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintStart_toEndOf="@+id/IDZaIkonu"
        app:layout_constraintTop_toTopOf="parent" />

    <ImageButton
        android:id="@+id/GumbZaMicanje"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_toEndOf="@id/ImePjesme"
        android:src="@drawable/ukloni"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toEndOf="@+id/ImePjesme"
        app:layout_constraintTop_toTopOf="parent" />

</androidx.constraintlayout.widget.ConstraintLayout>
```

Ovaj kod je XML kod za prikaz jednog retka na popisu pjesama u Android aplikaciji. Koristi se ConstraintLayout kao root view, koji omogućuje lako pozicioniranje elemenata unutar njega. Unutar ConstraintLayout-a, postoje tri elementa: ImageView, TextView i ImageButton. ImageView prikazuje ikonu glazbe, a prikazana ikona dolazi iz drawable resursa aplikacije. TextView prikazuje naziv pjesme koji može biti duži od širine TextView-a, pa se koristi marquee animacija kako bi se naziv pjesme kontinuirano prikazivao. ImageButton prikazuje ikonu za brisanje i koristi se za brisanje određene pjesme s popisa. Svaki element koristi različite parametre kako bi se pravilno pozicionirao unutar ConstraintLayout-a. Primjerice, ImageView je pozicioniran na lijevoj strani, a TextView na sredini, uz malo razmaka između njih. ImageButton se pozicionira na kraju redka s desne strane. Svaki element također koristi app namespace i layout_constraint parametre kako bi se pravilno pozicionirao.



Slika 8: pjesma_u_listi izgled

3.2.2.3. Pustanje_pjesama

Kod:

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:orientation="vertical"
    android:layout_width="match_parent"
    android:layout_height="match_parent">

    <TextView
        android:id="@+id/naziv_pjesme"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginTop="16dp"
        android:layout_marginBottom="482dp"
        android:ellipsize="marquee"
        android:marqueeRepeatLimit="marquee_forever"
        android:scrollHorizontally="true"
        android:singleLine="true"
        android:text="ImePjesme"
        android:textAlignment="center"
        android:paddingLeft="25dp"
        android:paddingRight="25dp"
        android:textSize="24sp"
        android:textStyle="bold"
        app:layout_constraintBottom_toTopOf="@+id/seek_bar"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent"
    />

    <ImageButton
        android:id="@+id/Zaustavi_Pokreni"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginBottom="64dp"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toStartOf="@+id/Sljedeca"
        app:layout_constraintStart_toEndOf="@+id/Prijasnja"
        app:srcCompat="@drawable/pauziraj" />

    <ImageButton
        android:id="@+id/Sljedeca"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginEnd="64dp"
        android:layout_marginBottom="64dp"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:srcCompat="@drawable/sljedeca" />

    <ImageButton
        android:id="@+id/Prijasnja"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginStart="64dp"
```

```

        android:layout_marginBottom="64dp"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:srcCompat="@drawable/prijasnja" />

```

<SeekBar

```

        android:id="@+id/seek_bar"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginStart="56dp"
        android:layout_marginEnd="56dp"
        android:layout_marginBottom="36dp"
        app:layout_constraintBottom_toTopOf="@+id/Zaustavi_Pokreni"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.0"
        app:layout_constraintStart_toStartOf="parent" />

```

<TextView

```

        android:id="@+id/trenutno_vr"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginBottom="12dp"
        android:text="Trenutno vrijeme"
        app:layout_constraintBottom_toTopOf="@+id/seek_bar"
        app:layout_constraintEnd_toStartOf="@+id/ukupno_vr"
        app:layout_constraintHorizontal_bias="0.216"
        app:layout_constraintStart_toStartOf="parent" />

```

<TextView

```

        android:id="@+id/ukupno_vr"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginEnd="56dp"
        android:layout_marginBottom="12dp"
        android:text="Ukupno vrijeme"
        app:layout_constraintBottom_toTopOf="@+id/seek_bar"
        app:layout_constraintEnd_toEndOf="parent" />

```

<ImageView

```

        android:id="@+id/Animacija"
        android:layout_width="200dp"
        android:layout_height="200dp"
        app:layout_constraintBottom_toTopOf="@+id/seek_bar"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/naziv_pjesme"
        app:srcCompat="@drawable/ikonamuzike" />

```

<ImageButton

```

        android:id="@+id/LoopGumb"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/Zaustavi_Pokreni"
        app:srcCompat="@drawable/loop_ugasen" />

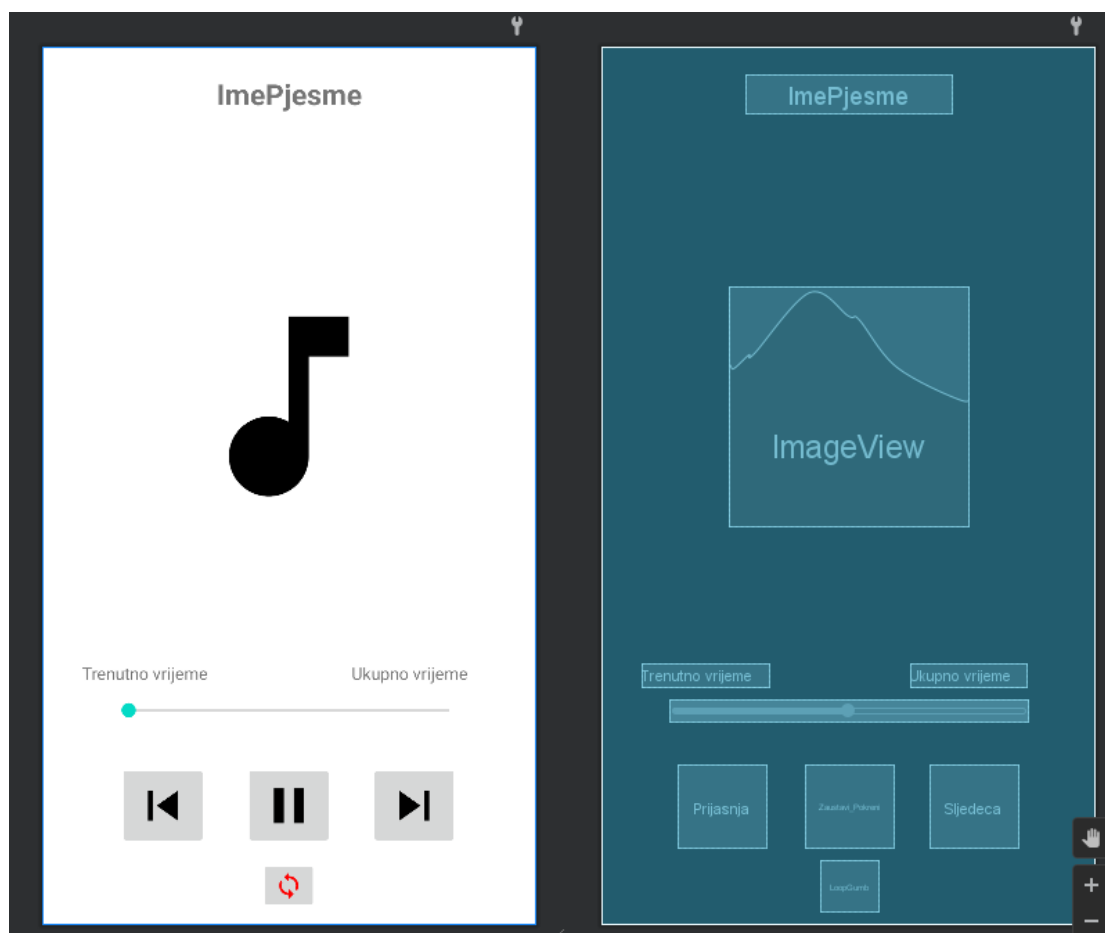
```

```

</androidx.constraintlayout.widget.ConstraintLayout>

```

Ovaj kod predstavlja layout za Android aplikaciju koji se sastoji od nekoliko elemenata raspoređenih unutar ConstraintLayouta. Elementi koji se koriste u ovom layoutu su TextView, ImageButton, SeekBar, ImageView i ImageButton. Svi elementi imaju definirane attribute koji se koriste za raspoređivanje elemenata na ekranu i stiliziranje njihovog izgleda. TextView prikazuje naziv pjesme koja se trenutno reproducira, a ImageButton elementi omogućuju kontrolu reprodukcije pjesme (pokretanje, pauziranje, prethodna i sljedeća pjesma). SeekBar se koristi za prikaz trenutnog vremena pjesme i za mogućnost pomicanja na određeni trenutak u pjesmi. ImageView prikazuje ikonu glazbe, a ImageButton element se koristi za omogućavanje ponavljanja pjesme. Svi elementi su raspoređeni pomoću ConstraintLayouta koji im omogućuje da se dinamički prilagode veličini zaslona uređaja na kojem se aplikacija izvršava. Također, atributi koji počinju sa app: i tools: korišteni su za prilagodbu elementa tijekom razvoja aplikacije i ne utječu na izgled u produkciji.



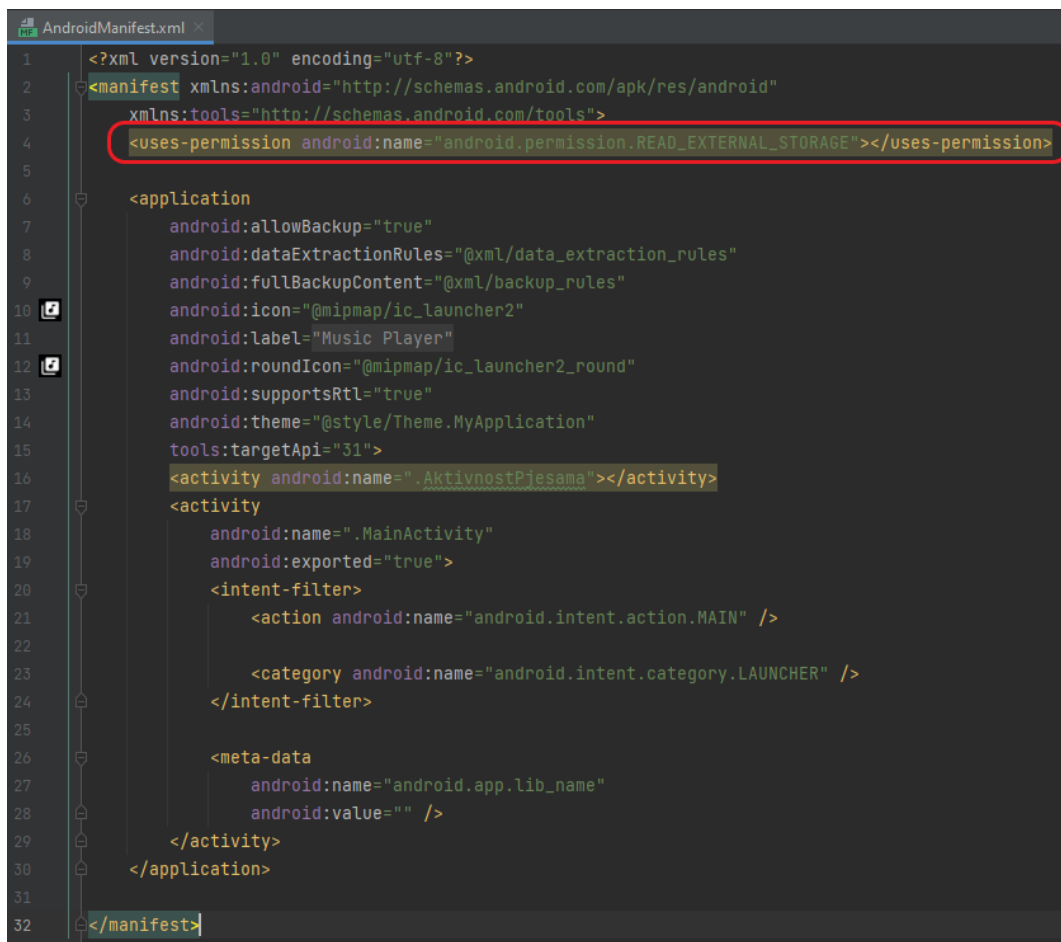
Slika 9: pustanje_pjesama izgled

3.3. Kod za funkcionalnost aplikacije

3.3.1. Dozvola za čitanje podataka

<uses-permission

android:name="android.permission.READ_EXTERNAL_STORAGE"></uses-permission> je dozvola koja omogućava aplikaciji da čita podatke sa vanjskog pohranjivanja uređaja, kao što je SD kartica ili interna pohrana. Za music player aplikaciju, ova dozvola je važna jer će aplikacija koristiti ovu dozvolu za pristupanje glazbenim datotekama koje se nalaze na vanjskom pohranjivanju uređaja, tako da korisnik može reproducirati svoju glazbu. Ako aplikacija nema ovu dozvolu, ona neće moći pristupiti datotekama s glazbom na vanjskom pohranjivanju uređaja i neće moći reproducirati glazbu. Stoga je važno da aplikacija koja ima funkcionalnost reprodukcije glazbe ima ovu dozvolu kako bi omogućila korisniku da reproducira svoju glazbu koja se nalazi na vanjskom pohranjivanju uređaja. Ovaj dio koda potrebno je dodati u AndroidManifest.xml.



```
1 <?xml version="1.0" encoding="utf-8"?>
2 <manifest xmlns:android="http://schemas.android.com/apk/res/android"
3     xmlns:tools="http://schemas.android.com/tools">
4     <uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE"></uses-permission>
5
6     <application
7         android:allowBackup="true"
8         android:dataExtractionRules="@xml/data_extraction_rules"
9         android:fullBackupContent="@xml/backup_rules"
10        android:icon="@mipmap/ic_launcher2"
11        android:label="Music Player"
12        android:roundIcon="@mipmap/ic_launcher2_round"
13        android:supportsRtl="true"
14        android:theme="@style/Theme.MyApplication"
15        tools:targetApi="31">
16        <activity android:name=".AktivnostPjesama"></activity>
17        <activity
18            android:name=".MainActivity"
19            android:exported="true">
20            <intent-filter>
21                <action android:name="android.intent.action.MAIN" />
22
23                <category android:name="android.intent.category.LAUNCHER" />
24            </intent-filter>
25
26            <meta-data
27                android:name="android.app.lib_name"
28                android:value="" />
29            </activity>
30        </application>
31
32 </manifest>
```

Slika 10: Dozvola za čitanje podataka

3.3.2. Objašnjenje klasa

Ova aplikacija se sastoji od četiri Java klase MainActivity, AdapterZaPjesme,, Pjesma i AktivnostPjesama koje su ključne za rad aplikacije.

3.3.2.1. Main Activity

MainActivity.java je glavna aktivnost aplikacije. U ovoj aktivnosti se prikazuje RecyclerView sa svim pjesmama, a također i gumb za osvježavanje liste. Ova aktivnost također provjerava dozvole za čitanje vanjskog spremišta uređaja. Ako aplikacija nema dozvolu, prikazuje se dijalog za traženje dozvole. U suprotnom se dohvaćaju pjesme iz vanjskog spremišta, AdapterZaPjesme se kreira i postavlja na RecyclerView. Klikom na gumb za osvježavanje, maknut će se sve pjesme iz SharedPreferences-a te će se dohvatiti nove pjesme. Ako se dodaju nove pjesme u vanjsko spremište, aplikacija se mora osvježiti da bi se prikazale nove pjesme.

Kod:

```
package com.example.myapplication;

public class MainActivity extends AppCompatActivity {

    private static final int MY_PERMISSIONS_REQUEST_READ_EXTERNAL_STORAGE =
0;
    RecyclerView Lista;
    AdapterZaPjesme AdapterPjesama;
    ArrayList<String> ListaPjesama;
    SharedPreferences sharedPreferences;

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

        Lista = findViewById(R.id.RecyclerView);
        ListaPjesama = new ArrayList<String>();
        sharedPreferences = getSharedPreferences("maknute_pjesme",
MODE_PRIVATE);
        AdapterPjesama = new AdapterZaPjesme(ListaPjesama, this,
sharedPreferences);

        Lista.setLayoutManager(new LinearLayoutManager(this));
        Lista.setAdapter(AdapterPjesama);

        if (ContextCompat.checkSelfPermission(this,
Manifest.permission.READ_EXTERNAL_STORAGE)
!= PackageManager.PERMISSION_GRANTED) {

            if (ActivityCompat.shouldShowRequestPermissionRationale(this,
Manifest.permission.READ_EXTERNAL_STORAGE)) {
                // Prikaži objašnjenje korisniku *asinkrono* -- ne blokiraj
                // ova nit čeka odgovor korisnika! Nakon što korisnik
                // vidi objašnjenje, pokušava se ponovno zatražiti
```

```

dopuštenje.
    } else {
        // traži se dopuštenje
        ActivityCompat.requestPermissions(this,
            new
String[] {Manifest.permission.READ_EXTERNAL_STORAGE},
            MY_PERMISSIONS_REQUEST_READ_EXTERNAL_STORAGE);
    }
    } else {
        DobivanjePjesama();
        AdapterPjesama = new AdapterZaPjesme(ListaPjesama, this,
sharedPreferences);
        Lista.setLayoutManager(new LinearLayoutManager(this));
        Lista.setAdapter(AdapterPjesama);
    }

    Button OsvjeziGumb = findViewById(R.id.OsvjeziGumb);
    OsvjeziGumb.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View view) {
            sharedPreferences.edit().clear().apply();
            DobivanjePjesama();
        }
    });
}

@Override
public void onRequestPermissionsResult(int requestCode, @NonNull
String[] permissions, @NonNull int[] grantResults) {
    super.onRequestPermissionsResult(requestCode, permissions,
grantResults);
    if (requestCode == MY_PERMISSIONS_REQUEST_READ_EXTERNAL_STORAGE) {
        if (grantResults.length > 0 && grantResults[0] ==
PackageManager.PERMISSION_GRANTED) {
            DobivanjePjesama();
        } else {
            // dopuštenje odbijeno
        }
        return;
    }
}

private void DobivanjePjesama() {
    ListaPjesama.clear();
    ContentResolver contentResolver = getContentResolver();
    Uri songUri = MediaStore.Audio.Media.EXTERNAL_CONTENT_URI;
    String selection = MediaStore.Audio.Media.IS_MUSIC + " != 0";
    Cursor CursorPjesama = contentResolver.query(songUri, null,
selection, null, MediaStore.Audio.Media.DATE_ADDED + " DESC");
    if (CursorPjesama != null && CursorPjesama.moveToFirst()) {
        int songTitle =
CursorPjesama.getColumnIndex(MediaStore.Audio.Media.TITLE);
        do {
            String TrenutnaPjesma = CursorPjesama.getString(songTitle);
            String MaknutaPjesma =
sharedPreferences.getString(TrenutnaPjesma, null);
            if (MaknutaPjesma == null) {
                ListaPjesama.add(TrenutnaPjesma);
            }
        }
    }
}

```

```

        } while (CursorPjesama.moveToNext());
    }
    CursorPjesama.close();
    AdapterPjesama.notifyDataSetChanged();
}
}

```

3.3.2.2. Adapter za pjesme

AdapterZaPjesme klasa se koristi za prikazivanje liste pjesama koje su dobivene iz uređaja koristeći RecyclerView. U MainActivity klasi, kreiraju se RecyclerView, AdapterZaPjesme, ArrayList za pohranu naziva pjesama i SharedPreferences za pohranu maknutih pjesama. Ako aplikacija nema dozvolu za čitanje pohrane, tada se koristi metoda ActivityCompat.requestPermissions() da bi se zatražila dozvola. Ako aplikacija ima dozvolu, tada se poziva metoda DobivanjePjesama() koja pronalazi sve pjesme u uređaju i sprema njihove nazive u ArrayList ListaPjesama. Kada je AdapterZaPjesme inicijaliziran, RecyclerView se postavlja na korištenje LinearLayoutManager. Klikom na gumb za osvježavanje, sve maknute pjesme se uklanjaju iz SharedPreferences-a i dobivaju se pjesme iz uređaja pozivom metode DobivanjePjesama(). U AdapterZaPjesme klasi, u metodi onCreateViewHolder() se inflatira view layouta R.layout.pjesma_u_listi za svaki redak u RecyclerView-u. U metodi onBindViewHolder(), postavljaju se TextView-ovi za prikaz naziva pjesama i ImageButton za brisanje pjesama. Klikom na ImageButton, pojavljuje se dijalog za potvrdu brisanja, a klikom na TextView, preusmjerava se na AktivnostPjesama. Kada se klikne TextView u AdapterZaPjesme klasi, stvara se novi Intent koji se šalje u AktivnostPjesama s ArrayListom svih pjesama u uređaju i imenom pjesme koja je kliknuta.

Kod:

```

package com.example.myapplication;

public class AdapterZaPjesme extends
RecyclerView.Adapter<AdapterZaPjesme.MusicViewHolder> {
    private ArrayList<String> ListaPjesama;
    private Context context;
    private SharedPreferences sharedPreferences;

    public AdapterZaPjesme(ArrayList<String> ListaPjesama, Context context,
SharedPreferences sharedPreferences) {
        this.ListaPjesama = ListaPjesama;
        this.context = context;
        this.sharedPreferences = sharedPreferences;
    }

    @NonNull
    @Override
    public MusicViewHolder onCreateViewHolder(@NonNull ViewGroup parent,

```

```

int viewType) {
    View view =
LayoutInflater.from(parent.getContext()).inflate(R.layout.pjesma_u_listi,
parent, false);
    return new MusicViewHolder(view);
}

@Override
public void onBindViewHolder(@NonNull MusicViewHolder holder,
@SuppressLint("RecyclerView") int position) {
    final String ImePjesme = ListaPjesama.get(position);
    holder.ImePjesmeTextView.setText(ImePjesme);
    holder.GumbZaMicanjePjesme.setOnClickListener(new
View.OnClickListener() {
        @Override
        public void onClick(View view) {
            AlertDialog.Builder builder = new
AlertDialog.Builder(context);
            builder.setTitle("Jeste li sigurni da želite maknuti pjesmu
iz liste?");
            builder.setPositiveButton("Da", new
DialogInterface.OnClickListener() {
                @Override
                public void onClick(DialogInterface dialog, int which)
{
                    ListaPjesama.remove(ImePjesme);
                    SharedPreferences.Editor editor =
sharedPreferences.edit();
                    editor.putString(ImePjesme, ImePjesme);
                    editor.apply();
                    notifyDataSetChanged();
                }
            });
            builder.setNegativeButton("Ne", new
DialogInterface.OnClickListener() {
                @Override
                public void onClick(DialogInterface dialog, int which)
{
                    dialog.cancel();
                }
            });
            builder.show();
        }
    });
    holder.itemView.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View view) {
            Intent intent = new Intent(context,
AktivnostPjesama.class);
            intent.putStringArrayListExtra("lista_pjesama",
ListaPjesama);
            intent.putExtra("ime_pjesme", ImePjesme);
            context.startActivity(intent);
        }
    });
}

@Override
public int getItemCount() {
    return ListaPjesama.size();
}

```

```

public class MusicViewHolder extends RecyclerView.ViewHolder {
    TextView ImePjesmeTextView;
    ImageButton GumbZaMicanjePjesme;

    public MusicViewHolder(View itemView) {
        super(itemView);
        ImePjesmeTextView = itemView.findViewById(R.id.ImePjesme);
        ImePjesmeTextView.setSelected(true);
        //OVO SVE JE POTREBNO ZA ANIMACIJU PREDUGOG TEKSTA
        (UKLJUCUJUCI: ImePjesmeTextView.setSelected(true))
        //          android:ellipsize="marquee"
        //          android:singleLine="true"
        //          android:marqueeRepeatLimit="marquee_forever"
        //          android:scrollHorizontally="true"
        GumbZaMicanjePjesme =
        itemView.findViewById(R.id.GumbZaMicanje);
    }
}

```

3.3.2.3. Pjesma

Klasa "Pjesma" je definirana kao Java razred koji implementira sučelje "Serializable". Razred "Pjesma" ima tri svojstva: "putanja", "naslov" i "trajanje", svi su tipa String. Klasa također sadrži tri metode: "getPath()" vraća putanju pjesme, "getTitle()" vraća naslov pjesme i "getDuration()" vraća trajanje pjesme. Osim toga, postoje i tri set metode koje omogućuju postavljanje vrijednosti za ove tri svojstva: "setPath()", "setTitle()" i "setDuration()".

Kod:

```

package com.example.myapplication;

import java.io.Serializable;

public class Pjesma implements Serializable {
    String putanja;
    String naslov;
    String trajanje;

    public Pjesma(String putanja, String naslov, String trajanje) {
        this.putanja = putanja;
        this.naslov = naslov;
        this.trajanje = trajanje;
    }

    public String getPath() {
        return putanja;
    }

    public void setPath(String putanja) {
        this.putanja = putanja;
    }
}

```

```

public String getTitle() {
    return naslov;
}

public void setTitle(String naslov) {
    this.naslov = naslov;
}

public String getDuration() {
    return trajanje;
}

public void setDuration(String trajanje) {
    this.trajanje = trajanje;
}
}

```

3.3.2.4. Aktivnost pjesama

Klasa "AktivnostPjesama" je definirana kao aktivnost Android aplikacije koja nasljeđuje klasu "AppCompatActivity". Klasa implementira dva sučelja: "View.OnClickListener" i "SeekBar.OnSeekBarChangeListener". Klasa sadrži mnoge varijable i objekte, kao što su: "ImageView" objekt za prikazivanje animirane slike, "MediaPlayer" objekt za reprodukciju glazbe, "TextView" objekte za prikazivanje naziva pjesme, trenutnog vremena i ukupnog vremena, "ImageButton" objekte za kontrolu reprodukcije, "SeekBar" objekt za prikazivanje napretka reprodukcije i "Handler" objekt za ažuriranje vremena reprodukcije i pozicije "SeekBar". Također, postoje metode za pokretanje i zaustavljanje animacije slike ("StartRotacije()" i "StopRotacije()"), ažuriranje pozicije "SeekBar" ("AzuriranjePozicijeSeekbara()") i dobivanje putanje pjesme na temelju naziva pjesme ("DobivanjePutanjePjesme()"). Metoda "onCreate()" se poziva kada se aktivnost prvi put stvori i koristi se za postavljanje sučelja korisničkog sučelja i povezivanje svih gumba i drugih elemenata sučelja s odgovarajućim metodama koje se izvršavaju kada se klikne na njih. Također, postavlja se "MediaPlayer" objekt da reproducira pjesmu i ažurira se "SeekBar" da prikazuje trenutni napredak reprodukcije. Ako se odabere opcija "loop" ("LoopGumb"), pjesma će se ponavljati nakon što se završi. Nakon završetka reprodukcije, ažuriraju se elementi korisničkog sučelja da bi se prikazale informacije o sljedećoj pjesmi. Ukratko, ovaj kod predstavlja aktivnost koja se koristi za reprodukciju glazbe. Kod koristi "MediaPlayer" objekt za reprodukciju i "SeekBar" objekt za prikazivanje napretka reprodukcije. Klasa "AktivnostPjesama" implementira sučelja koja omogućuju praćenje događaja klika na gumb i promjena napretka reprodukcije pomoću "SeekBar". Također, klasa ima metode za ažuriranje korisničkog sučelja i manipuliranje reprodukcijom glazbe.

Kod:

```
package com.example.myapplication;

public class AktivnostPjesama extends AppCompatActivity implements
View.OnClickListener, SeekBar.OnSeekBarChangeListener {

    ImageView animacija;
    Animation AnimacijaRotiranja;
    MediaPlayer mediaPlayer;
    TextView ImePjesmeTextView, TrenutnoVrijemeTextView,
    UkupnoVrijemeTextView;
    String ImePjesme;
    ImageButton PlayStopGumb, PrijasnjaPjesmaGumb, SljedecaPjesmaGumb,
    LoopGumb;
    SeekBar seekBar;
    Handler handler;
    Runnable runnable;

    int UkupnoVrijeme;
    boolean loopaLiSe = false;

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

        ImePjesmeTextView = findViewById(R.id.naziv_pjesme);
        ImePjesme = getIntent().getStringExtra("ime_pjesme");
        ImePjesmeTextView.setText(ImePjesme);
        ImePjesmeTextView.setSelected(true);
        TrenutnoVrijemeTextView = findViewById(R.id.trenutno_vr);
        UkupnoVrijemeTextView = findViewById(R.id.ukupno_vr);

        PlayStopGumb = findViewById(R.id.Zaustavi_Pokreni);
        PrijasnjaPjesmaGumb = findViewById(R.id.Prijasnja);
        SljedecaPjesmaGumb = findViewById(R.id.Sljedeca);
        seekBar = findViewById(R.id.seek_bar);
        animacija = findViewById(R.id.Animacija);
        AnimacijaRotiranja = AnimationUtils.loadAnimation(this,
R.anim.rotate_animation);

        PlayStopGumb.setOnClickListener(this);
        PrijasnjaPjesmaGumb.setOnClickListener(this);
        SljedecaPjesmaGumb.setOnClickListener(this);
        seekBar.setOnSeekBarChangeListener(this);
        LoopGumb = findViewById(R.id.LoopGumb);
        LoopGumb.setOnClickListener(this);

        handler = new Handler();

        mediaPlayer = new MediaPlayer();
        try {
            mediaPlayer.setDataSource(DobivanjePutanjePjesme(ImePjesme));
            mediaPlayer.prepare();
            mediaPlayer.setOnCompletionListener(new
MediaPlayer.OnCompletionListener() {
                @Override
                public void onCompletion(MediaPlayer mp) {
                    if (loopaLiSe == true) {
```

```

        mediaPlayer.seekTo(0);
        mediaPlayer.start();
        StartRotacije();
    } else {
        PustanjeSljedecePjesme();
        handler.removeCallbacks(runnable);
        StopRotacije();
        StartRotacije();
    }
    // Ažuriranje elemenata korisničkog sučelja odmah nakon
    što media player završi reprodukciju pjesme
    AzuriranjeTextViewova();
    AzuriranjePozicijeSeekbara();
}
});

UkupnoVrijeme = mediaPlayer.getDuration();

seekBar.setMax(UkupnoVrijeme);
seekBar.setClickable(false);

// Stvaranje novog Runnable objekta koji ažurira trenutno
vrijeme i položaj trake za traženje svake sekunde
runnable = new Runnable() {
    @Override
    public void run() {
        AzuriranjeTextViewova();
        AzuriranjePozicijeSeekbara();
        handler.postDelayed(this, 100);
    }
};
handler.postDelayed(runnable, 100);
mediaPlayer.start();
StartRotacije();
} catch (IOException e) {
    e.printStackTrace();
}

AzuriranjeTextViewova();
}

private void StartRotacije() {
    animacija.startAnimation(AnimacijaRotiranja);
}

private void StopRotacije() {
    animacija.clearAnimation();
}

private void AzuriranjePozicijeSeekbara() {
    // Ažuriranje pozicije SeekBara s trenutnom pozicijom MediaPlayera
    int TrenutnaPozicija = mediaPlayer.getCurrentPosition();
    seekBar.setProgress(TrenutnaPozicija);
}

private String DobivanjePutanjePjesme(String ImePjesme) {
    String putanja = null;
    ContentResolver contentResolver = getContentResolver();
    Uri songUri = MediaStore.Audio.Media.EXTERNAL_CONTENT_URI;

```



```

        String selection = MediaStore.Audio.Media.TITLE + "=?";
        String[] selectionArgs = {ImePjesme};
        Cursor CursorPjesama = contentResolver.query(songUri, null,
selection, selectionArgs, MediaStore.Audio.Media.DATE_ADDED + " DESC");
        if (CursorPjesama != null && CursorPjesama.moveToFirst()) {
            int songPath =
CursorPjesama.getColumnIndex(MediaStore.Audio.Media.DATA);
            putanja = CursorPjesama.getString(songPath);
        }
        CursorPjesama.close();
        return putanja;
    }

    private void PustanjeSljedecePjesme() {
        if (mediaPlayer != null) {
            List<String> songList =
getIntent().getStringArrayListExtra("lista_pjesama");
            int TrenutniIndex = songList.indexOf(ImePjesme);
            if (TrenutniIndex != -1) {
                if (TrenutniIndex < songList.size() - 1) {
                    // pustanje sljedeće pjesme
                    String nextSongName = songList.get(TrenutniIndex + 1);
                    mediaPlayer.stop();
                    mediaPlayer.reset();
                    try {
mediaPlayer.setDataSource(DobivanjePutanjePjesme(nextSongName));
                        mediaPlayer.prepare();
                        mediaPlayer.start();
                        PlayStopGumb.setImageResource(R.drawable.pauziraj);
                        StartRotacije(); //počinje animacija
                        AzuriranjeSeekBar();
                        AzuriranjeTextViewova();
                        ImePjesme = nextSongName;
                        ImePjesmeTextView.setText(ImePjesme);
                        UkupnoVrijeme = mediaPlayer.getDuration();
                        seekBar.setMax(UkupnoVrijeme);

                    } catch (IOException e) {
                        e.printStackTrace();
                    }
                } else {
                    // pustanje prve pjesme u listi
                    String firstSongName = songList.get(0);
                    mediaPlayer.stop();
                    mediaPlayer.reset();
                    try {
mediaPlayer.setDataSource(DobivanjePutanjePjesme(firstSongName));
                        mediaPlayer.prepare();
                        mediaPlayer.start();
                        StartRotacije(); // počinje se animacija
                        AzuriranjeSeekBar();
                        AzuriranjeTextViewova();
                        ImePjesme = firstSongName;
                        ImePjesmeTextView.setText(ImePjesme);
                        UkupnoVrijeme = mediaPlayer.getDuration();
                        seekBar.setMax(UkupnoVrijeme);

                    } catch (IOException e) {
                        e.printStackTrace();
                    }
                }
            }
        }
    }

```

```

    }
    }
}

private void PustanjePrijašnjePjesme() {
    if (mediaPlayer != null) {
        List<String> songList =
getIntent().getStringArrayListExtra("lista_pjesama");
        int TrenutniIndex = songList.indexOf(ImePjesme);
        if (TrenutniIndex > 0) {
            // puštanje Prijašnje pjesme
            String prevSongName = songList.get(TrenutniIndex - 1);
            mediaPlayer.stop();
            mediaPlayer.reset();
            try {
mediaPlayer.setDataSource(DobivanjePutanjePjesme(prevSongName));
                mediaPlayer.prepare();
                mediaPlayer.start();
                StartRotacije();
                AzuriranjeSeekBar();
                AzuriranjeTextViewova();
                ImePjesme = prevSongName;
                ImePjesmeTextView.setText(ImePjesme);
                UkupnoVrijeme = mediaPlayer.getDuration();
                seekBar.setMax(UkupnoVrijeme);
                PlayStopGumb.setImageResource(R.drawable.pauziraj);

            } catch (IOException e) {
                e.printStackTrace();
            }
        } else {
            // reproduciranje posljednje pjesme na popisu
            String lastSongName = songList.get(songList.size() - 1);
            mediaPlayer.stop();
            mediaPlayer.reset();
            try {
mediaPlayer.setDataSource(DobivanjePutanjePjesme(lastSongName));
                mediaPlayer.prepare();
                mediaPlayer.start();
                StartRotacije();
                AzuriranjeSeekBar();
                AzuriranjeTextViewova();
                ImePjesme = lastSongName;
                ImePjesmeTextView.setText(ImePjesme);
                UkupnoVrijeme = mediaPlayer.getDuration();
                seekBar.setMax(UkupnoVrijeme);
                PlayStopGumb.setImageResource(R.drawable.pauziraj);

            } catch (IOException e) {
                e.printStackTrace();
            }
        }
    }
}

```

```

        private void AzuriranjeSeekBar() {
            if (mediaPlayer != null) {
                int TrenutnaPozicija = mediaPlayer.getCurrentPosition();
                seekBar.setProgress(TrenutnaPozicija);

TrenutnoVrijemeTextView.setText(FormatiranjeVremena(TrenutnaPozicija));
                UkupnoVrijemeTextView.setText(FormatiranjeVremena(UkupnoVrijeme
- TrenutnaPozicija));
                handler.postDelayed(runnable, 100);
            }
        }
        private String FormatiranjeVremena(int vrijemeuMilis) {
            int sekunde = vrijemeuMilis / 1000;
            int minute = sekunde / 60;
            sekunde %= 60;
            return String.format("%02d:%02d", minute, sekunde);
        }

        private void AzuriranjeTextViewova() {
            int currentTime = mediaPlayer.getCurrentPosition();
            int minute = (currentTime / 1000) / 60;
            int sekunde = (currentTime / 1000) % 60;
            String TrenutnoFormatiranoVrijeme = String.format("%02d:%02d",
minute, sekunde);
            TrenutnoVrijemeTextView.setText(TrenutnoFormatiranoVrijeme);

            minute = (UkupnoVrijeme / 1000) / 60;
            sekunde = (UkupnoVrijeme / 1000) % 60;
            String UkupnoFormatiranoVrijeme = String.format("%02d:%02d",
minute, sekunde);
            UkupnoVrijemeTextView.setText(UkupnoFormatiranoVrijeme);
        }
        @Override
        public void onClick(View v) {
            switch (v.getId()) {
                case R.id.Zaustavi_Pokreni:
                    if (mediaPlayer.isPlaying()) {
                        mediaPlayer.pause();
                        PlayStopGumb.setImageResource(R.drawable.pokreni);
                        StopRotacije();
                    } else {
                        mediaPlayer.start();
                        PlayStopGumb.setImageResource(R.drawable.pauziraj);
                        StartRotacije();
                    }
                    // Ažuriranje elemenata korisničkog sučelja odmah nakon što
korisnik klikne gumb za reprodukciju/pauzu
                    AzuriranjeTextViewova();
                    AzuriranjePozicijeSeekbara();
                    break;
                case R.id.Prijasnja:
                    PustanjePrijasnePjesme();
                    // Ažuriranje elemenata korisničkog sučelja odmah nakon što
korisnik klikne gumb Prijašnja
                    AzuriranjeTextViewova();
                    AzuriranjePozicijeSeekbara();
                    break;
                case R.id.Sljedeca:
                    PustanjeSljedecePjesme();
                    // Ažuriranje elemenata korisničkog sučelja odmah nakon što

```

```

korisnik klikne gumb Prijašnja
        AzuriranjeTextViewova();
        AzuriranjePozicijeSeekBara();
        break;
    case R.id.LoopGumb:
        // Loop gumb je pritisnut
        loopaLiSe = !loopaLiSe; // mijenjanje loop stanja
        if (loopaLiSe) {
            LoopGumb.setImageResource(R.drawable.loop_upaljen); //
mijenja se slika
            mediaPlayer.setLooping(true);
        } else {
            LoopGumb.setImageResource(R.drawable.loop_ugasen); //
mijenja se slika
            mediaPlayer.setLooping(false);
        }
        break;
    }
}
@Override
public void onProgressChanged(SeekBar seekBar, int progress, boolean
fromUser) {
    if (fromUser) {
        mediaPlayer.seekTo(progress);
        AzuriranjeTextViewova();
    }
    if (fromUser && progress == seekBar.getMax()) {
        //Ako je SeekBar korisnik postavio na max, pokreće se rotacija
i pušta sljedeća pjesma
        PustanjeSljedecePjesme();
    }
}

public void onStartTrackingTouch(SeekBar seekBar) {
    // Uklonjaju se sva ažuriranja pozicije SeekBar dok korisnik s njom
komunicira
    handler.removeCallbacks(runnable);
}

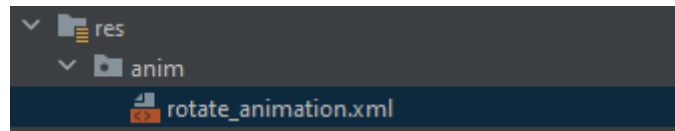
@Override
public void onStopTrackingTouch(SeekBar seekBar) {
    // Kada korisnik pusti seekbar, nastavlja se s ažuriranjem položaja
seekbara
    int TrenutnaPozicija = seekBar.getProgress();
    mediaPlayer.seekTo(TrenutnaPozicija);
    AzuriranjeTextViewova();
    handler.postDelayed(runnable, 100);
}

@Override
protected void onDestroy() {
    super.onDestroy();
    if (mediaPlayer != null) {
        mediaPlayer.release();
        mediaPlayer = null;
    }
    handler.removeCallbacks(runnable);
    StopRotacije();
}
}

```

3.3.3. Animacija

Animacija za rotiranje ikone glazbe dok se pušta pjesma se nalazi u res/anim pod imenom rotate_animation.xml.



Slika 11: Prikaz datoteke za animaciju

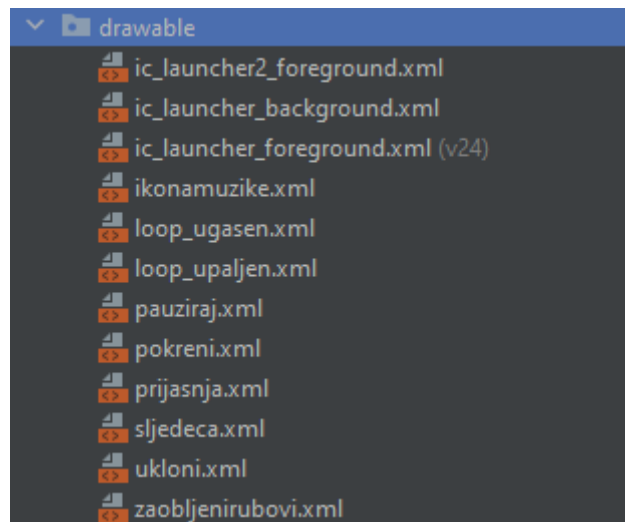
Kod animacije koriste se različiti atributi za definiranje parametara rotacije, kao što su "android:fromDegrees" i "android:toDegrees" koji definiraju kut rotacije od i do kojih će se izvoditi animacija. Atributi "android:pivotX" i "android:pivotY" se koriste za definiranje položaja oko kojeg će se izvoditi rotacija. Također, definirani su i atributi za trajanje animacije "android:duration", broj ponavljanja animacije "android:repeatCount" i opcije za zadržavanje animacije nakon završetka "android:fillAfter" i "android:fillEnabled".

Kod:

```
<?xml version="1.0" encoding="utf-8"?>
<set xmlns:android="http://schemas.android.com/apk/res/android">
  <rotate
    android:fromDegrees="0"
    android:toDegrees="360"
    android:pivotX="50%"
    android:pivotY="50%"
    android:duration="5000"
    android:repeatCount="infinite"
    android:fillAfter="true"
    android:fillEnabled="true"
  />
</set>
```

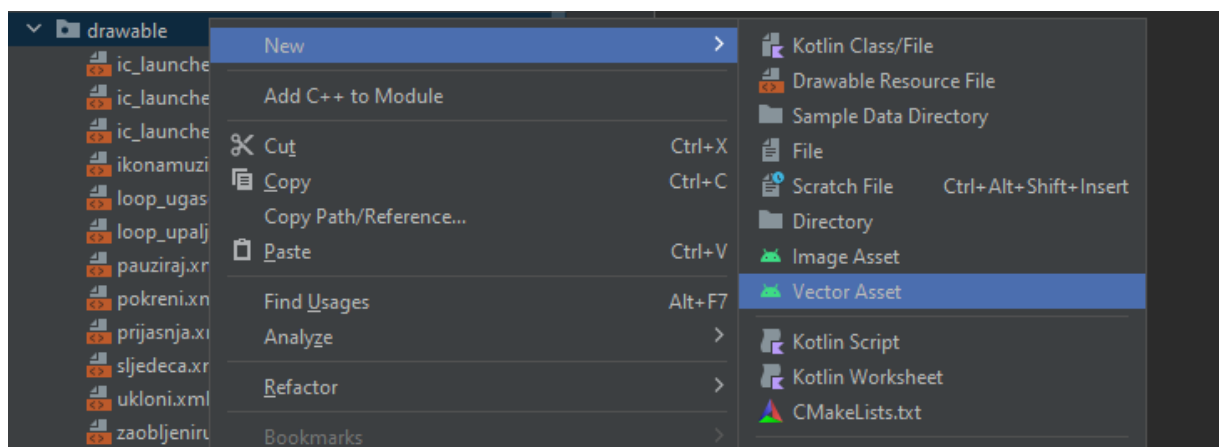
3.3.4. Ikone

Drawable folder u Android Studiu je jedna od osnovnih struktura direktorija za spremanje slika i ostalih grafika koje se koriste u aplikaciji. To je standardni direktorij koji se nalazi unutar "res" direktorija, a koristi se za pohranu različitih vrsta slika, kao što su ikone, pozadine, slike botuna itd. U Drawable folderu se obično nalaze različite vrste datoteka slika, uključujući PNG, JPEG, SVG i druge formate. Te slike se kasnije mogu koristiti u kodu aplikacije za prikazivanje u korisničkom sučelju, kao i u drugim dijelovima aplikacije koji zahtijevaju prikazivanje slika.



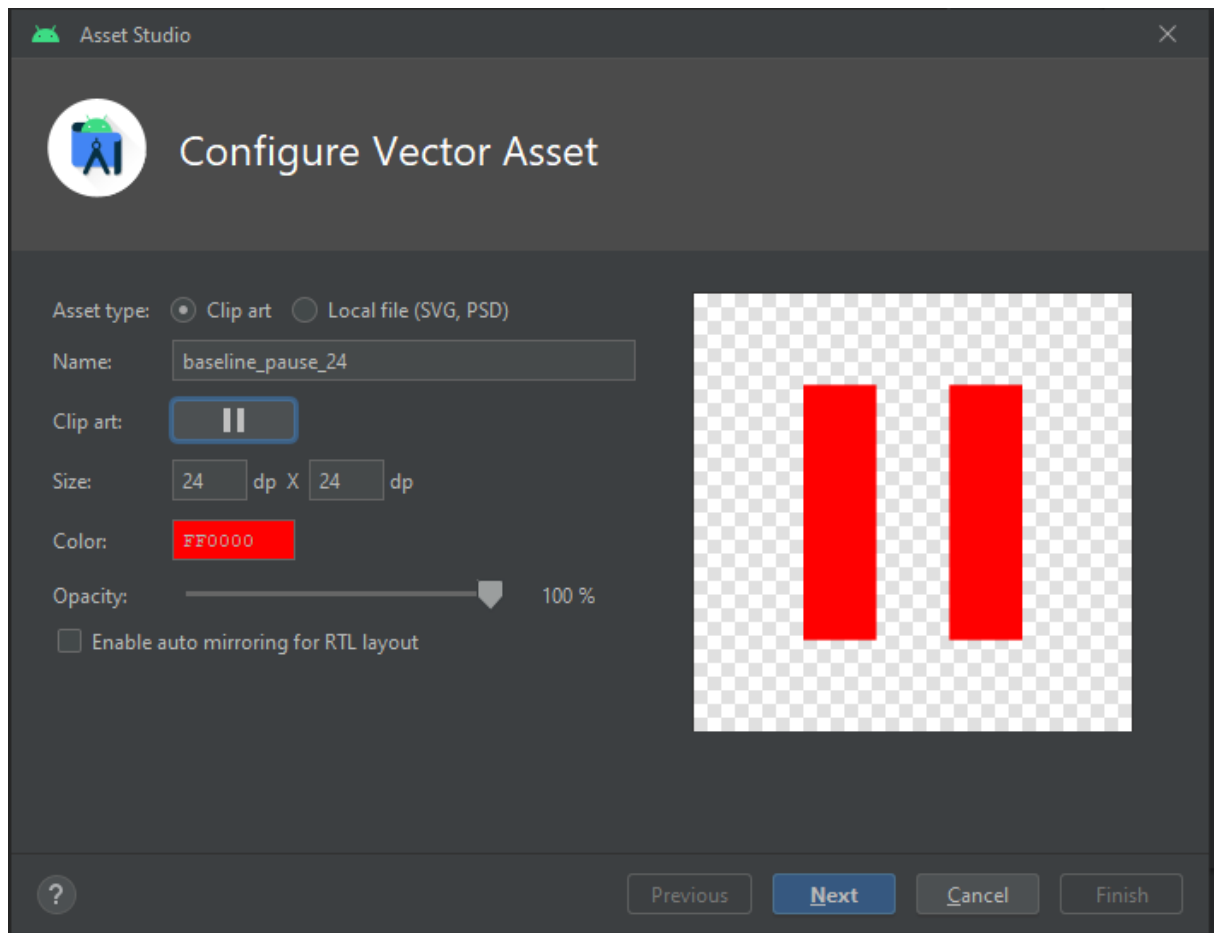
Slika 12: Datoteke u drawable direktoriju

Za stvaranje novih ikona koje sam koristio u aplikaciji radio sam nove Vector asset-e kao što je prikazano na slikama:



Slika 13: Stvaranje ikona

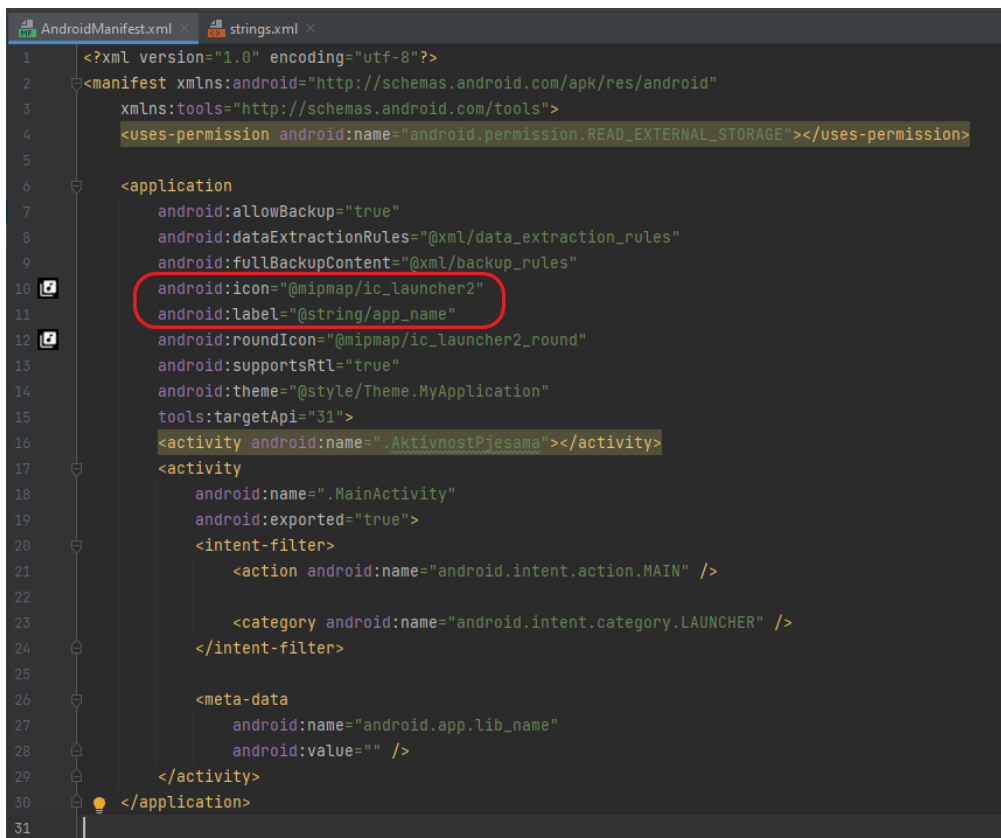
Na primjer za stvaranje crvenog pause gumba možemo se koristiti opcijama koja su na slici. Možemo mijenjati ime ikone, možemo birati mnoštvo ikona koje već postoje u Android Studiu i nalaze se u "Clip art:", možemo mijenjati veličinu ikone, boju i prozirnost.



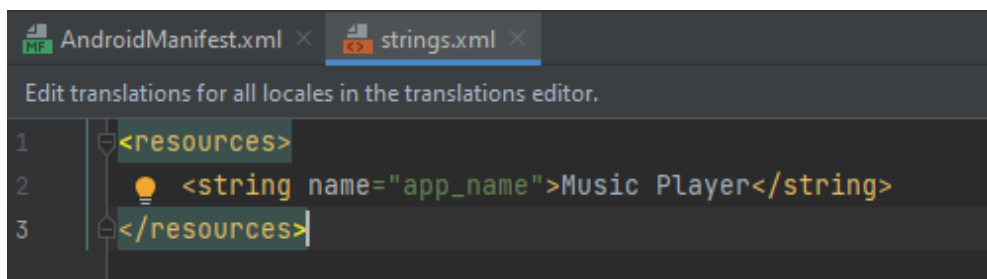
Slika 14: Stvaranje nove ikone

3.3.5. Prikaz aplikacije na Android uređaju

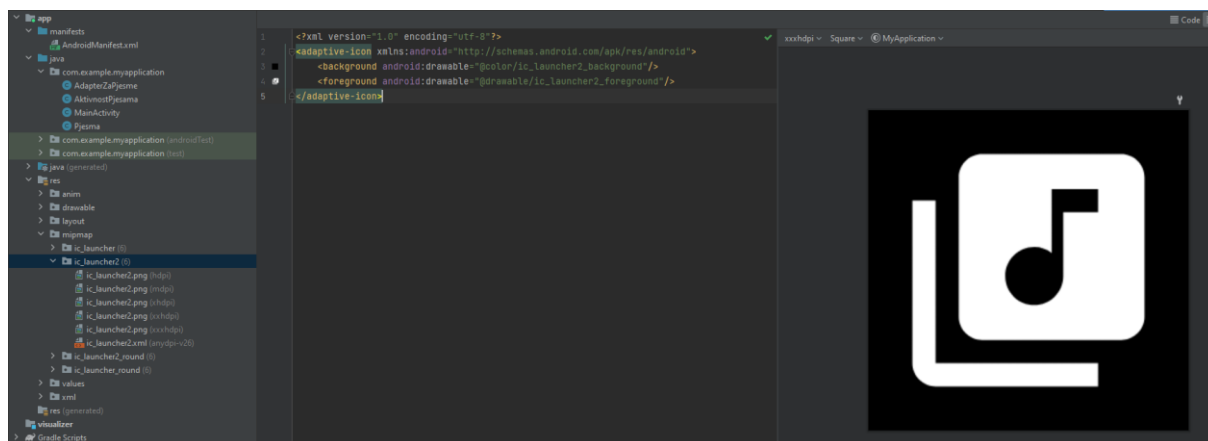
Kako bi promijenio ime i ikonu aplikacije koja je instalirana na Android uređaju moramo sam napraviti novi background i foreground te promijeniti dio koda. Novu ikonu sam nazvao "ic_launcher2" te naziv datoteke koja se nalazi u strings.xml sam promijenio u "Music Player".



Slika 15: Promjena ikona i imena



Slika 16: Novi naziv aplikacije kada se prikazuje na Android uređaju



Slika 17: Izgled nove ikone, gdje se nalazi i od čega se sastoji

4. Zaključak

U ovom radu istraživao sam proces razvoja Music Player-a za Android operativni sustav s ciljem omogućavanja korisnicima pristupa njihovoj glazbenoj kolekciji na jednostavan i intuitivan način. U fokusu rada bila je izrada brze, stabilne i sigurne aplikacije bez reklama. Za razradu rada koristio sam različite tehnologije i alate poput Android Studija, Java programskog jezika, te različitih dodataka i biblioteka. Kroz rad su obrađeni različiti aspekti razvoja aplikacije, uključujući zahtjeve, dizajn korisničkog sučelja i funkcionalnosti, te implementaciju i testiranje aplikacije. Moj konačni cilj bio je pružiti korisnicima izvrsno iskustvo korištenja Music Player-a za Android, koji će im omogućiti praćenje i upravljanje glazbenom kolekcijom, pružajući im ugodno korisničko sučelje i jednostavan način za pronalaženje i preslušavanje pjesama. Tijekom rada naišao sam na nekoliko problema i situacija koje sam morao pomno proučiti i istražiti na internetu, no motivacija i puno uloženog rada omogućili su mi da ostvarim cilj rada. Korištene metode i tehnike u razradi rada uključivale su analizu zahtjeva korisnika, planiranje funkcionalnosti, programiranje u programskom jeziku Java i testiranje performansi aplikacije. Za razradu rada koristio sam alate poput Android Studija, GitHub-a i alate za dizajn korisničkog sučelja kako bih izradio prototip aplikacije te testirao njegove pretpostavke. Prototip je potvrdio ispravnost koncepta, ali neke pretpostavke su zahtijevale daljnje istraživanje i poboljšanje. Rezultati rada obuhvatili su opis ključnih koraka i izazova u procesu izrade Music Player aplikacije za Android. Sve u svemu, proces razvoja Music Player-a za Android operativni sustav bio je izazovan, ali i vrlo zanimljiv, te sam uspješno ostvario svoj cilj u izradi kvalitetne aplikacije za korisnike Android platforme.

5. Literatura

- [1] "Android Studio." [Online]. URL: <https://developer.android.com/studio/>. [Pristupljeno: 10. travnja 2023.]
- [2] "MediaPlayer." [Online]. URL: <https://developer.android.com/guide/topics/media/mediaplayer>. [Pristupljeno: 10. travnja 2023.]
- [3] "Stack Overflow." [Online]. URL: <https://stackoverflow.com/questions/tagged/android+music-player>. [Pristupljeno: 12. travnja 2023.]
- [4] "Android Studio." [Online]. URL: https://android.fandom.com/wiki/Android_Studio. [Pristupljeno: 13. travnja 2023.]
- [5] "Java (programski jezik)." [Online]. URL: [https://en.wikipedia.org/wiki/Java_\(programming_language\)](https://en.wikipedia.org/wiki/Java_(programming_language)). [Pristupljeno: 10. travnja 2023.]

6. Popis slika

Slika 1: Izrada novog projekta u Android Studiu	9
Slika 2: Odabir postavki novog projekta	10
Slika 3: Distribucija API verzija	11
Slika 4: Prikaz strukture datoteka u Android Studiu	12
Slika 5: Različite vrste layout-a (Izvor: Android Developers)	13
Slika 6: Postavljanje atributa Constraint layout-a kroz grafičko sučelje Android Studia (Izvor: Android Developers)	14
Slika 7: activity_main izgled	16
Slika 8: pjesma_u_listi izgled	18
Slika 9: pustanje_pjesama izgled	21
Slika 11: Prikaz datoteke za animaciju	35
Slika 12: Datoteke u drawable direktoriju	36
Slika 13: Stvaranje ikona	36
Slika 14: Stvaranje nove ikone	37
Slika 15: Promjena ikona i imena	38
Slika 16: Novi naziv aplikacije kada se prikazuje na Android uređaju	38
Slika 17: Izgled nove ikone, gdje se nalazi i od čega se sastoji	39

7. Popis tablica

Tablica 1: Povijest verzija Android Studia	4
Tablica 2: Povijest verzija Jave	7