Izrada alarm Android aplikacije s kvizom

Sabo, Dominik

SVEUČILIŠTE JOSIPA JURJA STROSSMAYERA U OSIJEKU
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA

Sveučilišni studij

Izrada alarm Android aplikacije s kvizom

Završni rad

Dominik Sabo

Osijek, 2019.

Sadržaj

1. UVOD

Ovaj završni rad namijenjen je ljudima kojima običan alarm nije dovoljan da ih u potpunosti probudi. Mnogo ljudi koji koriste uobičajeni alarm na svome mobilnom uređaju ugase ga bez da otvore oči te se vrate spavanju ili samo pritisnu „snooze" kako bi mogli spavati još pet minuta. Nakon što prođe vrijeme predviđeno za „snooze", samo si opet daju još pet minuta; što nije zdravo za ljudsko tijelo. U ovoj aplikaciji korisnik može podesiti alarm isto kao i u aplikaciji za postavljanje alarma koja je već instalirana kada je mobitel pokrenut po prvi puta, no ovdje, zajedno s uobičajenim postavkama, korisnik može odabrati temu pitanja koja će aplikacija postaviti kada alarm zazvoni, te težinu kviza. Ovisno o težini kviza, kviz ima veći broj ponuđenih odgovora, kako bi bilo teže nagađati, te je kazna za krivo odgovoreno pitanje stroža (na način da korisnik mora odgovoriti na više pitanja kako bi se alarm ugasio). Korisnik u aplikaciji može zadati proizvoljan broj budilica, namjestiti da zvoni samo jednom ili da se ponavlja određenim danima u tjednu, te si namjestiti vrstu zvuka/melodije koji želi. Ova aplikacija trebala bi razbuditi korisnika poticanjem na razmišljanje čim se probudi.

Ova aplikacija razvijena je za operacijski sustav Android u Java programskom jeziku. Potrebno je dobro znanje Java jezika te koncepata koji se koriste u Androidu zajedno s korištenjem razvojnog okruženja Android Studio u kojem je cijela aplikacija napravljena.

Strukturu ovog rada tvore poglavlje u kojem se nalazi opis korištenih tehnologija, zatim poglavlje u kojem je opisan razvoj aplikacije. Nakon toga slijedi poglavlje s prikazom strukture aplikacije. Na kraju rada nalazi se zaključak.

1.1. Zadatak završnog rada

Zadatak ovog rada je izraditi Andorid aplikaciju čija će svrha biti korisniku omogućiti buđenje s kvizom kako bi odmah aktivirao mozak kada se probudi te kako bi se odmah razbudio. Aplikacija treba imati mogućnost podešavanja više alarma s različitim vremenima, melodijama alarma, te različitim temama i težinama kvizova koji se aktiviraju kada alarm zazvoni. Naravno, aplikacija ima i mogućnost uređivanja te brisanja postojećih alarma.

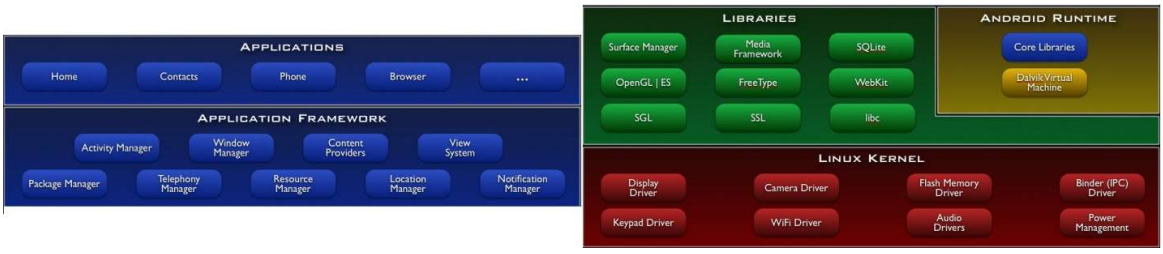
2. OPIS KORIŠTENIH TEHNOLOGIJA

Tehnologije korištene za razvoj ove aplikacije opisane su u ovom poglavlju. To su operacijski sustav Android, Android Studio, programski jezik Java i opisni jezik XML.

2.1. Operacijski sustav Android

Krajem 20. stoljeća počinje razvoj *smartphonea* (pametnih telefona). S vremenom su zahtjevi korisnika postajali sve veći, a proizvođači su ih tako i udovoljavali razvijanjem sve jačih i svestranijih uređaja. Android Inc., tvrtka koja se takvim uređajima bavila 2003. dobila je ideju za sustav Android. Razvoj takvog sustava bio je poprilično skup, te je ideja opstala tako što ju je Google 2005. godine vidio kao priliku izlaza na tržište mobilnih telefona. Godine 2007. Google osniva konzorcij *Open Handset Alliance* (OHA) koji je za svrhu uzeo normiranje svih mobilnih uređaja. Nedugo poslije konzorciju se pridružuju tvrtke koje su dio mobilne industrije. Neposredno nakon toga, OHA izdaje Android, mobilnu platformu otvorenog koda utemeljenu na Linux operativnom sustavu. Taj događaj smatra se prvom pojavom Android operacijskog sustava. Android je ovih dana najrašireniji operacijski sustav korišten u pametnim telefonima. Zasniva se na Linux 2.6 jezgri i napisan je u programskom jeziku C/C++, ali unatoč tome, većina aplikacija na Androidu napisana je u Javi koristeći pritom Android razvojne programske komponente (engl. *Android Software Development Kit*, SDK). Prema [1] arhitektura sustava Android sastoji se od tri razine (Slika 2.1.):

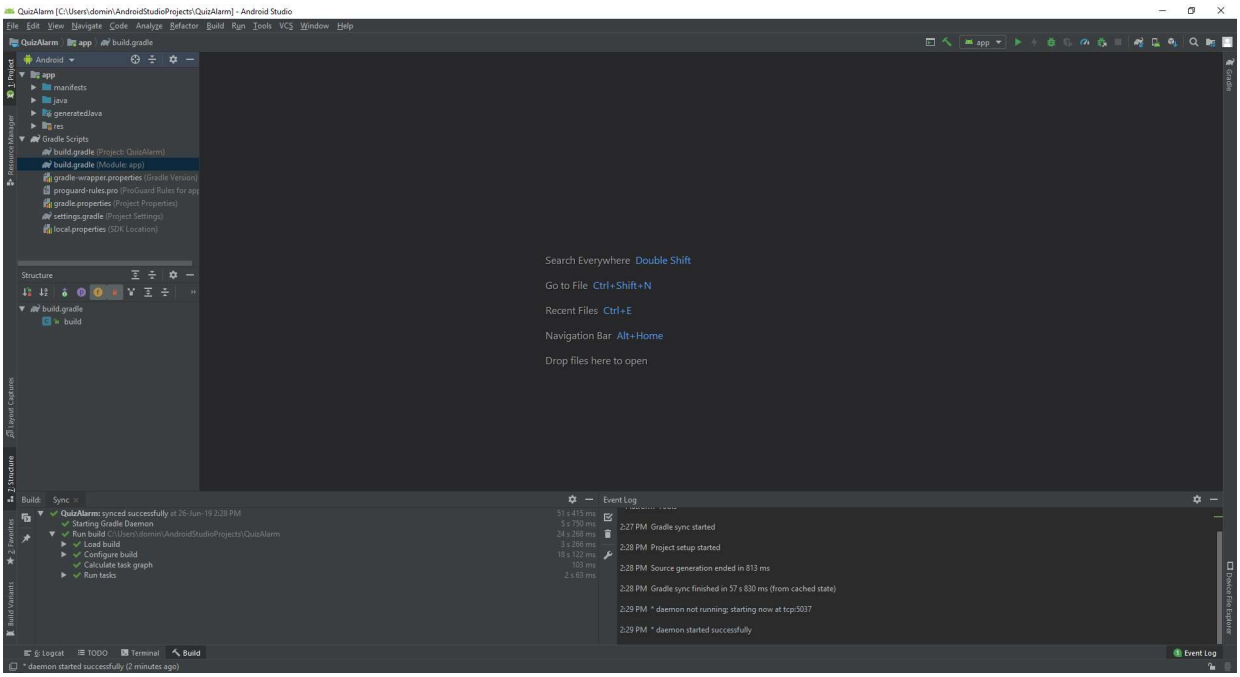
- Linux jezgra u kojoj se nalaze upravljački programi,
- C/C++ knjižnice iznad jezgre,
- te sloj vidljiv korisniku – Android Runtime



Slika 2.1. Razine Android arhitekture [5]

2.2. Android Studio

Android Studio temelji se na IntelliJ IDEA softveru. Googleova konferencija razvojnih inženjera objavljuje ga u svibnju 2013. godine. Prema [2] bio je u stupnju ranog pristupa, izdanog u svibnju 2013. godine u inačici 0.1, nakon čega je ušao u beta status krenuvši u lipnju 2014. godine kada je izdana inačica 0.8. Inačica 1.0, prva stabilna verzija, izdana je u prosincu 2014. godine. Android Studio napravljen je posebno za izrađivanje novih Android aplikacija i sada je službeni razvojni alat za Android operacijski sustav Moguće ga je preuzeti besplatno i dostupan je za sve najraširenije računalne operacijske sustave. Android Studio pruža mnoge značajke koje omogućuju efikasniju izradu aplikacija za Android. Neke od tih značajki su: brz i bogati emulator mobilnog uređaja radi testiranja aplikacija bez fizičkog pametnog telefona, podrška za C++, Javu i Kotlin, Instant Run, opsežni alati za testiranje i uklanjanje grešaka itd. Ovaj završni rad je gotovo u potpunosti napravljen upravo u Android Studio okruženju. (Slika 2.2.)



Slika 2.2. Sučelje Android Studia

2.3. Java

Patrick Naughton, James Gosling i drugi inženjeri u tvrtki Sun Microsystems razvili su Javu, programski jezik objektno orijentiranog dizajna, u kojoj se kod nalazi unutar klasa. Godine 1991. u projektu Green započinje razvoj Jave, a objavljuje se 1995. godine. Tada je to za programere bilo nešto potpuno novo. U odnosu na tadašnje programe, programi u Javi mogli su se bez problema pokretati na svim operativni sustavima, jer svako računalo na kojemu je instaliran *Java Virtual Machine* (JVM) može izvoditi programe napisane u Javi i prevedene u Java bajt kod. C Programi su se posebno prilagođavali operacijskim sustavima na kojima su se trebali izvoditi. Java je među glavnim programskim jezicima koji se koriste za razvoj mobilnih aplikacija na Androidu. Java se ubraja među više programske jezike kod kojih je nedostatak da se prije izvođenja moraju prevesti. No viši programski jezici imaju više prednosti nego nedostataka. Među prednostima su mnogo jednostavnije programiranje, kraće vrijeme pisanja i kod im je lakše čitljiv. Ako na računalu Java nije instalirana, Internet stranice i aplikacije kodirane u Javi neće raditi. Java se može besplatno preuzeti i gotovo je standard da ju svako računalo ima instaliranu. [3]

2.4 XML

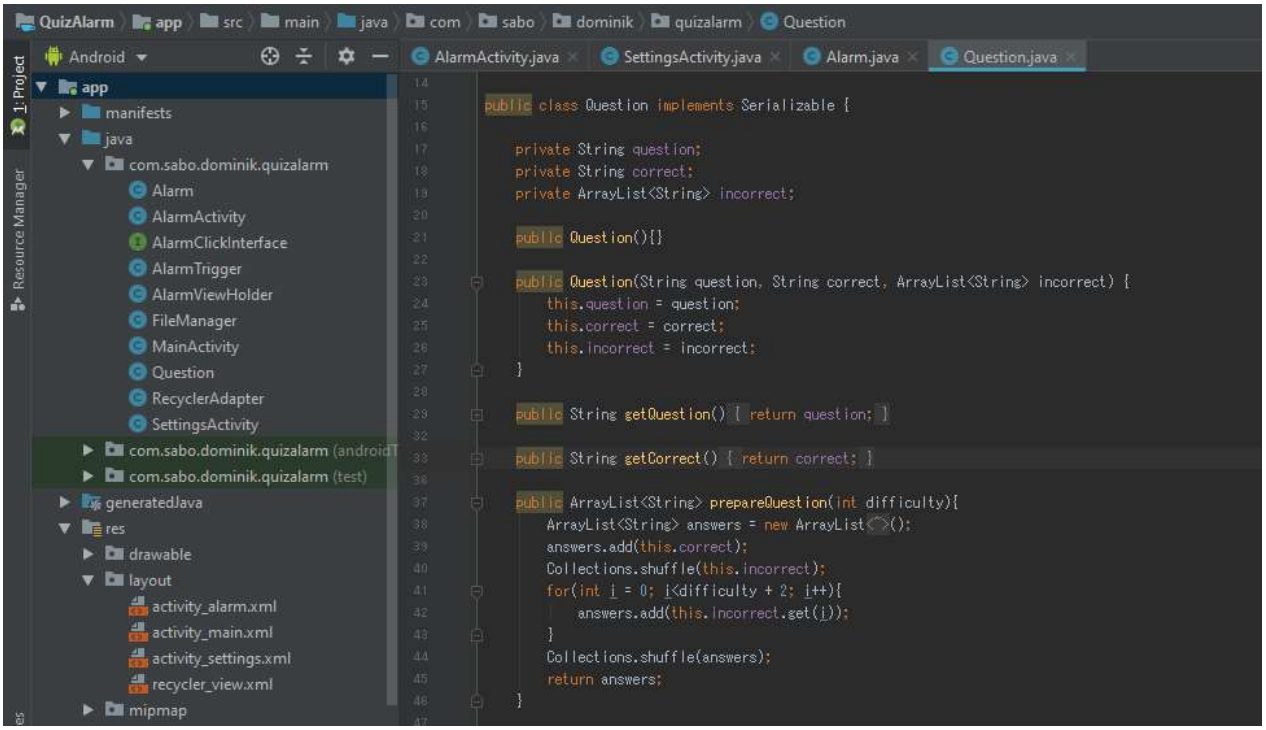
XML (engl. *Extensible Markup Language*) raširen je jezik sa svrhom označavanja podataka. Tekstualnim formatom prikazuju se informacije kao što su dokumenti, razni podaci i slično. XML dokumenti Čitljivi su i ljudima i aplikacijama. Budući da obrada XML dokumenata ne započinje dokle god postoje greške, XML dokumenti se mogu veoma pouzdano obrađivati pomoću računalnih aplikacija. XML poput HTML-a koristi oznake. Kod za jedan element započinje s oznakama < >, zatvara se s oznakom </ >, a između tih oznaka nalaze se sve specifikacije koje taj element ima, npr. veličina fonta, udaljenost od ruba, boja elementa, veličina samog elementa i druge. Primjer jednog XML *TextView* elementa vidi se na odsječku koda 2.1. Prema [4] oznake u XML-u slične su kao u HTML-u, ali su dizajnirane za drukčije potrebe. XML se koristi u svrhe razmjene, pohrane i povećanja dostupnosti podataka. [4]

```
<TextView
    android:id="@+id/tvNew"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="New Alarm"
    android:layout_centerHorizontal="true"
    android:layout_marginTop="15dp"
    android:textSize="30dp"/>
```

Odsječak koda 2.1. Primjer XML koda

3. RAZVOJ APLIKACIJE

U ovom poglavlju opisan je razvoj aplikacije za alarm s kvizom s naglaskom na najznačajnije korištene koncepte Android programiranja u ovom radu. U ovoj aplikaciji ti koncepti su: knjižnica Recycler View za omogućavanje prikaza zadanih alarma u listi, te njihovim uređivanjem i brisanjem, prenošenje podataka između različitih aktivnosti u aplikaciji pomoću klase Intent, klasa Alarm Manager zajedno sa svime što ona koristi, te na kraju implementacija kviza u alarm.



Slika 3.1. Prikaz svih klasa i XML dokumenata od kojih se sastoji ovaj projekt zajedno s dijelom koda iz klase *Question*

3.1. RecyclerView

RecyclerView je pogled u XML-u koji služi za kreiranje liste kutija koje sadrže jednake poglede (u ovom slučaju svaki podatak u listi je jedan alarm). Ako se ta lista napuni preko cijelog ekrana mobitela, *RecyclerView* omogućava da se kroz listu može prolaziti povlačenje prsta preko zaslona. Također, *RecyclerView* čuva od ekstenzivnog korištenja memorije tako da podatke koji nisu

vidljivi na ekranu makne iz memorije te ih ponovno učita kada se vrate na ekran. *RecyclerView* nije uobičajena knjižnica u Java programskom jeziku pa se mora uvesti u program tako da se u Gradle od aplikacije doda linija koda koja ga implementira (Odsječak koda 3.1.)

```
dependencies {
    implementation fileTree(dir: 'libs', include: ['*.jar'])
    implementation 'com.android.support:appcompat-v7:28.0.0'
    implementation 'com.android.support.constraint:constraint-layout:1.1.3'
    implementation 'com.android.support:recyclerview-v7:28.0.0'
    testImplementation 'junit:junit:4.12'
    androidTestImplementation 'com.android.support.test:runner:1.0.2'
    androidTestImplementation 'com.android.support.test.espresso:espresso-core:3.0.2'
}
```

Odsječak koda 3.1. Implementacija *RecyclerView*-a u aplikaciju

U ovoj aplikaciji *RecyclerView* se koristi kako bi se za svaki pojedinačni alarm koji korisnik napravi na glavnoj aktivnosti aplikacije prikazalo ime alarma, kada alarm zvoni, koja tema je kviza zadana za alarm te na kraju je li alarm aktiviran ili deaktiviran. Kada se aplikacija prvi puta pokrene, u *RecyclerView* se postavljaju dva neaktivna alarma uobičajena za svaku alarm aplikaciju.

3.2. Intent

Klasa *Intent* u Androidu koristi se za pokretanje novih aktivnosti u aplikaciji, te za prenošenje podataka između njih. Za prijenos podataka, *Intent* ima metodu *putExtra()* koja kao atribute prima varijablu ili konstantu koju prenosi, te tekstualni ključ pomoću kojega se taj isti podatak može izvući metodom *getExtra()* u aktivnosti koju je taj *Intent* pokrenuo. Također, *Intenti* se koriste za postavljanje *BroadcastManagera* koji služe za okidanje različitih servisa ili aktivnosti dok aplikacija radi u pozadini.

Kako je postavljanje alarma u aplikaciji dosta složeno jer ima puno varijabli na koje treba obratiti pozornost (vrijeme, naziv, melodija, tema i težina kviza), kako bi se te postavke mogle postaviti na elegantan način, potrebno je napraviti novu aktivnost u kojoj se nalaze sve te postavke. U glavnoj aktivnosti aplikacije nalazi se gumb za dodavanje novog alarma. Pritiskom na taj gumb pravi

se novi *Intent* koji pokreće aktivnost metodom *startActivityForResult()* za postavke te nakon što se u toj aktivnosti sve postavi i pritisne se gumb za dalje, stvara se opet novi *Intent* u koji se prema odsječku koda 3.2. spremaju podaci dobiveni iz popunjene aktivnosti za postavke te se odmah završava ista aktivnost. Po završetku aktivnosti za postavke se zbog *startActivityForResult()* u glavnoj aktivnosti izvršava metoda *onActivityResult()* koja tada preuzme sve podatke iz *Intenta* iz aktivnosti za postavke te ih spremi u listu alarma i napravi novi podatak u *RecyclerViewu*.

```
private void setIntentData(Intent data) {
    if(etAlarmName.getText().toString().equals("")) data.putExtra( name: "name",  value: "Alarm");
    else data.putExtra( name: "name", etAlarmName.getText().toString());
    data.putExtra( name: "ringtone", ringtonePath.toString());
    data.putExtra( name: "quizTheme", tvTheme.getText().toString());
    data.putExtra( name: "quizDifficulty", quizDifficulty);
    data.putIntegerArrayListExtra( name: "weekDays", weekDays);
    data.putExtra( name: "hour", timePicker.getHour());
    data.putExtra( name: "minute", timePicker.getMinute());
}
```

Odsječak koda 3.2. Postavljanje varijabli iz aktivnosti za postavke u *Intent*

Za uređivanje već postojećeg alarma, kada se pritisne na alarm u *RecyclerViewu*, u novi *Intent* se spremaju podaci tog alarma koji se tada šalju u aktivnost za postavke kako bi u toj aktivnosti sve već bilo na mjestu. Ako korisnik pritisne na gumb za dalje, nikakvi podaci u tom alarmu se ne bi izmijenili, već bi ostali takvi kakvi su bili prije.

Pri postavljanju melodije alarma, potrebno je pitati korisnika za dopuštenje iščitavanja podataka s vanjskog medija kako bi korisnik mogao odabrati bilo koji zvuk koji želi sa svoga mobilnog uređaja.

3.3. AlarmManager

AlarmManager je klasa koja daje aplikaciji pristup za korištenje servisa za alarm koji je ugniježđen u sustavu Androida. *AlarmManager* omogućuje da se postavi okidač koji će pokrenuti klasu kojoj je dodijeljen *BroadcastReceiver*. U *BroadcastRecieveru* je moguće napraviti bilo što, od promjene nekih varijabli, spremanja ili učitavanja podataka, okidanja notifikacija na mobitelu, pa i pokretanje aktivnosti kao što je slučaj u ovome radu (Odsječak koda 3.3.)

```java
public class AlarmTrigger extends BroadcastReceiver {
    @Override
    public void onReceive(Context context, Intent intent) {
        Alarm alarm = new Alarm(intent);

        boolean flag = false;
        for(int i = 0; i<7; i++) {
            if (alarm.getWeekDay().get(i) == 1){
                flag = true;
                break;
            }
        }
        if(flag){
            Intent alarmIntent = new Intent(context.getApplicationContext(), AlarmTrigger.class);
            alarm.setIntentData(alarmIntent);

            PendingIntent pendingIntent = PendingIntent.getBroadcast(context, intent.getExtras().getInt( key: "position"), alarmIntent, FLAG_UPDATE_CURRENT);
            AlarmManager alarmManager = (AlarmManager) context.getSystemService(ALARM_SERVICE);
            alarmManager.setExactAndAllowWhileIdle(AlarmManager.RTC_WAKEUP,  triggerAtMillis: System.currentTimeMillis() + alarm.calculateRingingTime(), pendingIntent);
        }

        Intent ringIntent = new Intent(context.getApplicationContext(), AlarmActivity.class);
        alarm.setIntentData(ringIntent);
        ringIntent.addFlags(Intent.FLAG_ACTIVITY_MULTIPLE_TASK | Intent.FLAG_ACTIVITY_RESET_TASK_IF_NEEDED);
        if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.O_MR1) {
            ringIntent.addFlags(Intent.FLAG_ACTIVITY_NEW_TASK);
        }

        context.startActivity(ringIntent);
    }
}
```

Odsječak koda 3.3. *BroadcastReceiver* i njegove funkcije

AlarmManager ima nekoliko raznih metoda za postavljanje okidača, no u ovome radu je bitna *setExactAndAllowWhileIdle()* koja postavi okidač koji će se pokrenuti u točno vrijeme kada je zadan i da se može pokrenuti dok je uređaj neaktivan, iako to ne funkcionira dok je uređaj ugašen. Toj metodi se treba definirati koji sat prati. Ovdje je to sat u stvarnom vremenu koji dopušta očitavanje i kada je uređaj neaktivan (*RTC_WAKEUP*) te kada se okidač treba okinuti u milisekundama. Budući da Android koristi sat u milisekundama koji broji od ponoći prvog siječnja 1970. godine, vrijeme za zvono alarma se izračunava tako što se uzme sadašnje vrijeme te se na njega doda broj milisekundi do postavljenog vremena alarma. Na kraju, *AlarmManageru* treba se predati *PendingIntent* koji će pokrenuti aktivnost *BroadcastReceivera*. *PendingIntent* se instancira davajući mu kontekst aplikacije, njegov ID koji govori je li taj *PendingIntent* isti kao i neki drugi (tako se postiže više alarma paralelno), neke opcije ako je potrebno i običan *Intent*. U taj *Intent* umeću se svi podaci alarma koji su potrebni za vrijeme kada će alarm zvoniti, a to su tema i težina kviza te melodija alarma. Taj postupak vidljiv je u odječku koda 3.4. Kada se pokrene *BroadcastReceiver*, on preko *Intenta* koji primi od *AlarmManagera* pokrene aktivnost alarma u kojoj alarm započinje zvoniti te pokreće kviz.

```
private void setAlarm(int position){
    Intent intent = new Intent( packageContext: MainActivity.this, AlarmTrigger.class);
    alarms.get(position).setIntentData(intent);
    intent.putExtra( name: "position", position);
    PendingIntent pendingIntent = PendingIntent.getBroadcast(getApplicationContext(), position, intent, FLAG_UPDATE_CURRENT);
    AlarmManager alarmManager = (AlarmManager)getSystemService(ALARM_SERVICE);
    alarmManager.setExactAndAllowWhileIdle(AlarmManager.RTC_WAKEUP,  triggerAtMillis: System.currentTimeMillis() + alarms.get(position).calculateRingingTime(), pendingIntent);
}
```

Odsječak koda 3.4. Postavljanje alarma

Naknadno, ako se u aktivnosti za postavke postavi ponavljanje za bilo koji dan, *BroadcastReciever* će odmah pomoću podataka iz poslanog mu *Intenta* postaviti novi alarm koji će zvoniti na sljedeći postavljeni dan.

3.4. Kviz

Glavni dio ove aplikacije je kviz u kojemu korisnik mora odgovoriti na pet pitanja točno kako bi se alarm ugasio. Kako bi gašenje alarma samo na taj način bilo omogućeno na uređaju, alarm se također može isključiti tako da se aplikacija ugasi iz liste nedavnih aplikacija na mobitelu. Kada se pokrene aktivnost alarma, poziva se *WakeLock* koji probudi mobilni uređaj i prikaže aktivnost alarma na zaslonu, te se pali *Media Player* koji počinje svirati zadanu melodiju alarma. Naposlijetku, u aktivnosti se pali kviz koji ovisno o temi koja je zadana na alarmu, učita pitanja iz datoteka koje su napravljene kada se aplikacija po prvi puta pokrene na uređaju te ih promiješa kako bi bila u drukčijem redoslijedu svaki puta. Pitanja se očitaju u obliku liste objekata klase *Question* u kojoj su definirana polja za pitanje, točan odgovor i lista netočnih odgovora. Klasa *Question* ima metodu pr*epareQuestion()* koja vrati pomiješanu listu tekstualnih varijabli od kojih je jedna točan odgovor, a ostale su netočni odgovori (Odsječak koda 3.5.) Pritiskom na bilo koji od odgovora poziva u aktivnosti metodu *checkAnswer()* koja uspoređuje tekst pritisnutog odgovora s točnim odgovorom objekta *Question*, te ako je jednak, daje korisniku jedan bod, a ako nije jednak, dodaje se brojaču koji ovisno o težini kviza kazni korisnika negativnim bodom. Kada se odgovori točno na pet pitanja, aktivnost gasi *WakeLock* i *Media Player* te vrati korisnika na zaključani zaslon uređaja. Za sada, aplikacija ima pet kategorija po 20 pitanja, no daljnjim razvijanjem aplikacije taj broj može se lako povećati.

```
private void setUpNextQuestion() {
    questionNumber++;
    if (questionNumber == questions.size()) {
        questionNumber = 0;
        Collections.shuffle(questions);
    }

    ArrayList<String> question = questions.get(questionNumber).prepareQuestion(alarm.getQuizDifficulty());
    tvQuestion.setText(questions.get(questionNumber).getQuestion());
    for (int i = 0; i < question.size(); i++) {
        answers.get(i).setText(question.get(i));
    }
}
```

```
public ArrayList<String> prepareQuestion(int difficulty){
    ArrayList<String> answers = new ArrayList<>();
    answers.add(this.correct);
    Collections.shuffle(this.incorrect);
    for(int i = 0; i<difficulty + 2; i++){
        answers.add(this.incorrect.get(i));
    }
    Collections.shuffle(answers);
    return answers;
}
```

Odsječak koda 3.5. Način na koji aplikacija priprema pitanja za kviz

4. STRUKTURA APLIKACIJE

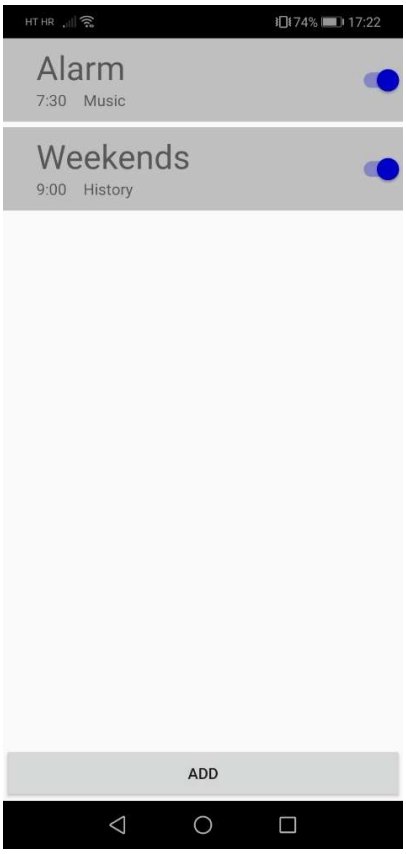
U ovom poglavlju pokazati će se izgled aplikacije na uređaju i opisati svi pogledi koji se nalaze u aktivnostima te njihove funkcije.

4.1. Glavna aktivnost

U glavnoj aktivnosti aplikacije nalazi se *RecyclerView* u kojem svaka kutija sadrži ime alarma, vrijeme kada alarm zvoni, temu kviza i status alarma.

Pritiskom na alarm u *RecyclerViewu* otvara se aktivnost postavki koja u sebi ima postavljene podatke alarma iz te kutije *RecyclerViewa*.

Gumb *Add* otvara novu aktivnost postavki u kojoj se postavlja novi alarm.



Slika 4.1. Glavna aktivnost aplikacije

4.2. Aktivnost postavki

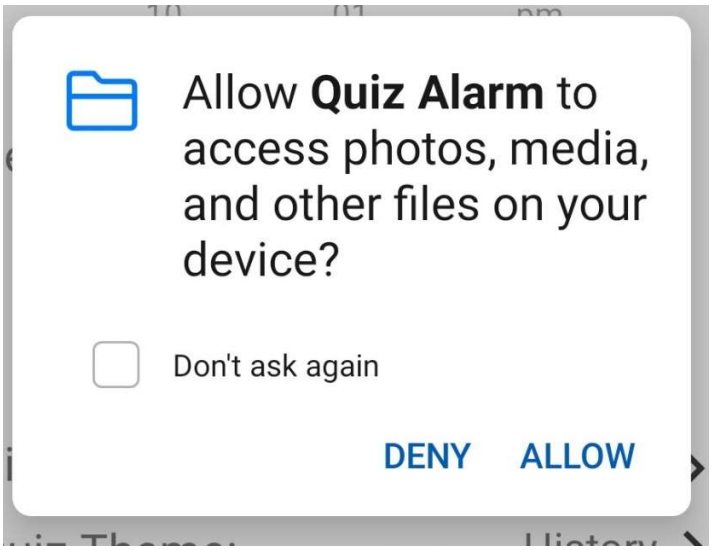
U aktivnosti postavki nalazi se naslov koji predstavlja ime alarma, gumbovi za natrag i za završetak. Pritiskom gumba natrag, aplikacija se vraća u glavnu aktivnost te se ne sprema novi alarm, pritiskom na gumb završetak spremaju se podaci i u *RecyclerView* se dodaje novi aktivan alarm.

Time Picker omogućuje biranje vremena kada će alarm zvoniti. Kada se pravi novi alarm, *Time Picker* je postavljen na trenutno vrijeme sata kada je pritisnuta tipka, a kada se uređuje postavljeni alarm, *Time Picker* se postavi na vrijeme tog zadanog alarma.

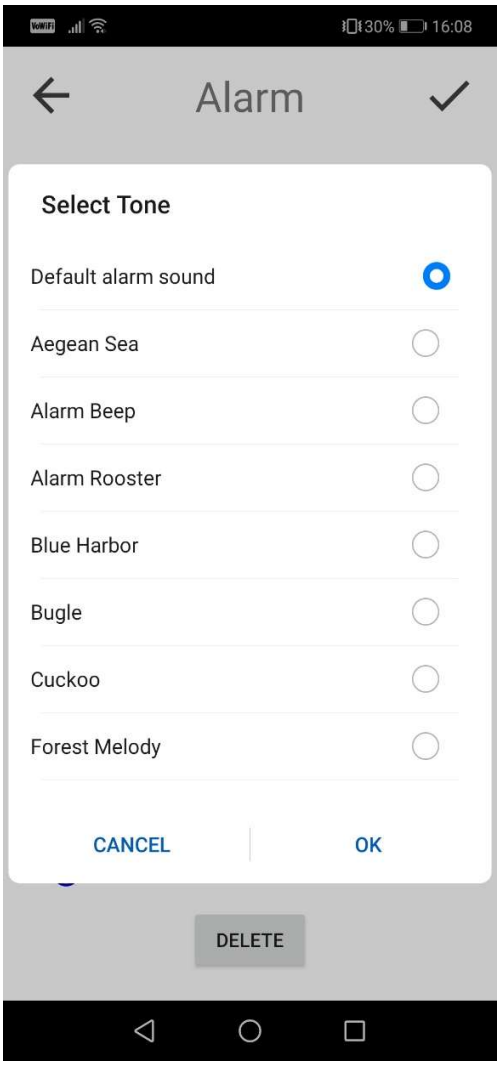
Ispod *Time Pickera* nalazi se nekoliko *TextViewova* koji predstavljaju dane u tjednu. Ako alarm treba zvoniti na određeni dan, taj dan će biti označen plavom bojom. Uobičajeno je da alarm treba zvoniti na radne dane kada se postavlja novi alarm.

Ispod toga nalazi se *EditText* za postavljanje imena alarma. Ako je to polje ostavljeno prazno, alarm će samo poprimiti ime „Alarm“.

Nakon toga nalazi se izbornik melodije alarma. Pritiskom na ime melodije ili na strjelicu pokraj, aplikacija odlazi u novu podaktivnost u kojoj se pomoću *RadioButtona* bira melodija. Za biranje melodije korisnik mora dozvoliti aplikaciji da čita medijske datoteke. (slika 4.1.). Ako korisnik označi „*Don't ask again*“ i pritisne „*Deny*“, aplikacija neće dozvoliti biranje melodije, osim ako korisnik poslije toga ručno ne promijeni dopuštenje u postavkama mobilnog uređaja.



Slika 4.1. Dijalog za dopuštenje

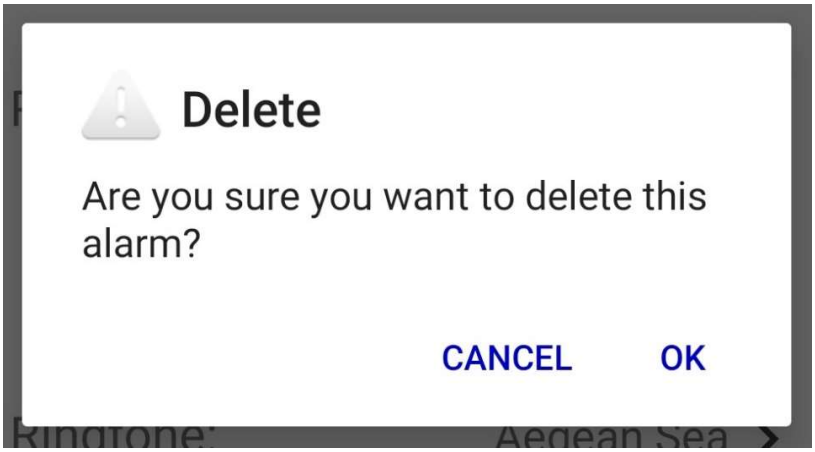


Slika 4.2. Izbor melodije alarma

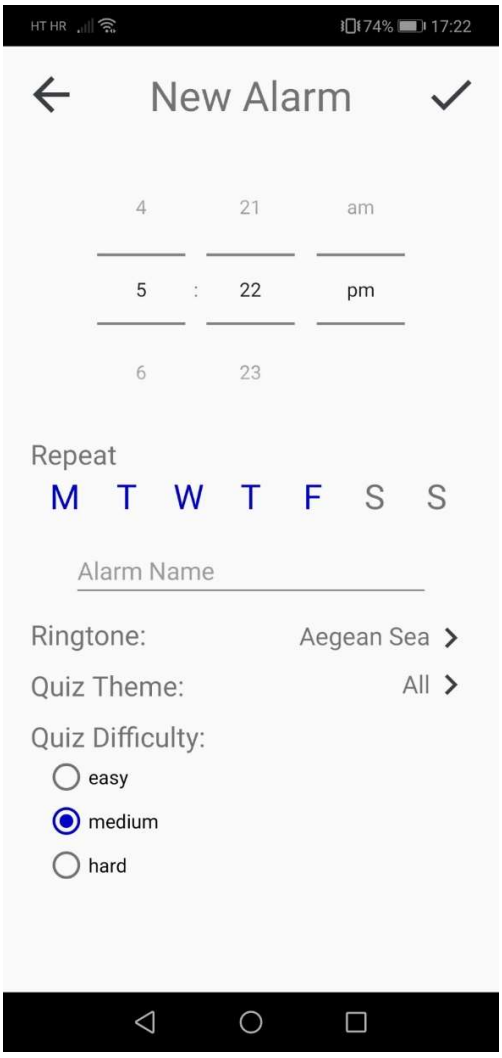
Ispod je izbornik teme kviza. Uobičajene su sve teme zajedno. Pritiskom na naziv teme otvara se mali izbornik u kojemu su sve teme prikazane. Tada se pritiskom na željenu temu odabire tema pitanja u kvizu.

Na kraju se nalazi izbornik za težinu kviza. Ako je postavljeno na „*easy*“, na svakom pitanju su ponuđena tri odgovora, i bod se oduzme tek za tri pogrešna odgovora. Ako je „*medium*“ ponuđena su četiri odgovora i kazna se dobiva za dva pogrešna odgovora. Pod „*hard*“ ima pet ponuđenih odgovora, i kazna je za svaki pogrešan odgovor

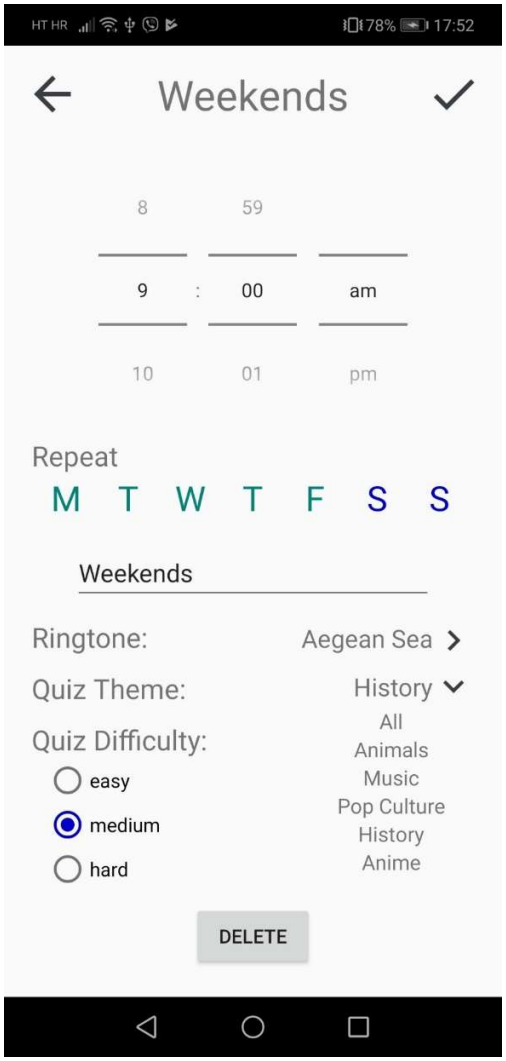
Ako se ne pravi novi alarm, već se uređuje prethodno postavljeni, na dnu aktivnosti (slika 4.5.) nalazi se gumb *Delete*, koji kada se pritisne izbaci dijalog za brisanje alarma (slika 4.3.). Kada korisnik pritisne „OK“ alarm se prvo deaktivira *AlarmManagerom*, nakon čega se briše iz liste alarma i iz *RecyclerViewa*.



Slika 4.3. Dijalog za brisanje alarma



Slika 4.4. Pritisak na gumb *Add*

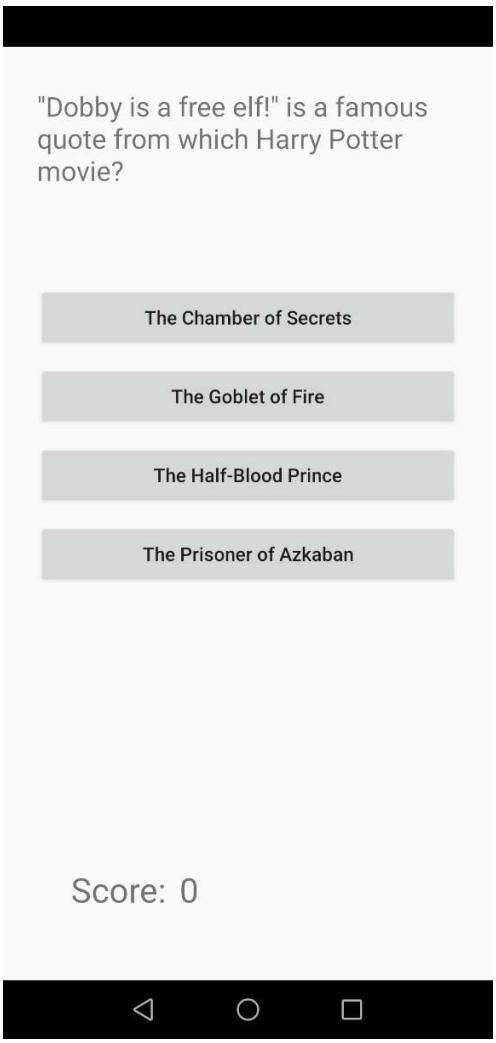


Slika 4.5. Pritisak na alarm u *RecyclerViewu* (s pritisnutom temom kviza)

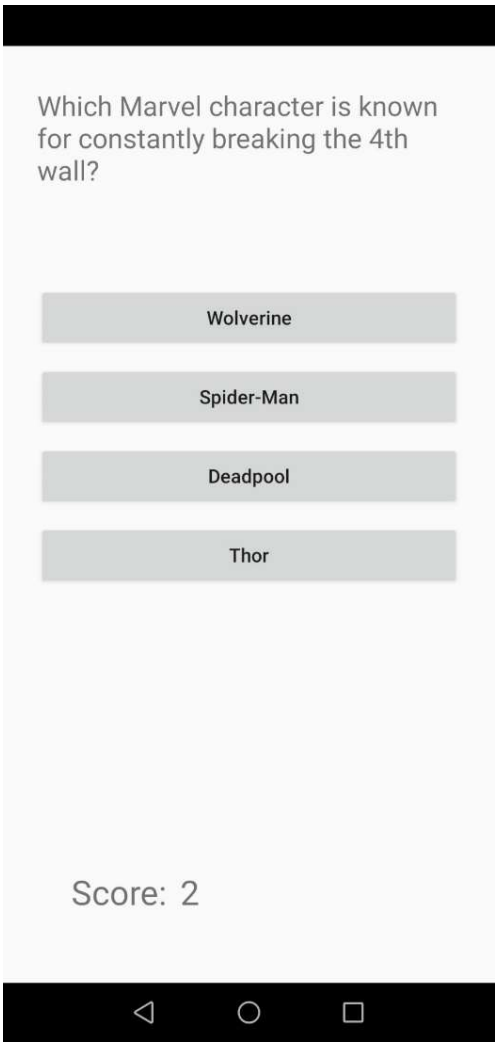
4.3. Aktivnost alarma

U aktivnosti alarma nalazi se jedan *TextView* na vrhu koji postavlja pitanje te tri do pet gumbova od kojih jedan ima točan odgovor na pitanje, a ostali netočne odgovore.

Na dnu aktivnosti nalazi se *TextView* koji predstavlja bodove koje je korisnik prikupio. Kada vrijednost bodova dođe do pet, alarm se gasi te se uređaj vraća na prijašnji zaslon (zaključani zaslon ili aplikaciju koju je korisnik koristio u vrijeme paljenja alarma.)



Slika 4.6. Tek upaljena aktivnost alarma



Slika 4.7. Aktivnost alarma s dva točna odgovora

ZAKLJUČAK

Mobilne aplikacije postale su neizostavna svakodnevica u gotovo svim dijelovima života, uključujući i u zamjenjivanju budilica. Ovaj rad zahtijevao je podosta istraživanja i proučavanja tehnologija za programiranje androida. Aplikacija je gotovo u cijelosti razvijena u Android Studiu, čija je razvojna okolina dosta intuitivna i poprilično je lako raditi s njime. Bilo je veoma poučno izrađivati ovu aplikaciju jer se susrelo s nekim manje poznatim konceptima programiranja u Android Studiu, a i s nekim dosad potpuno nepoznatima. Aplikacija je zamišljena da razbudi ljude koji se teško ustaju iz kreveta korištenjem uobičajene budilice. Ovom aplikacijom korisnik može podesiti budilicu za proizvoljno vrijeme i mogućnost ponavljanja proizvoljnim danima. Također si može namjestiti vlastito zvono te odabrati između nekolicine tema pitanja za kviz. Trenutno je broj pitanja u aplikaciji malen, no kroz daljnji razvoj, može se dodati koliko god je potrebno. Tijekom izrade aplikacije postavljeno se na stajalište krajnjeg korisnika i zaključeno je da je jako bitno da sučelje aplikacije bude jednostavno i razumljivo.

LITERATURA

[1] Wikipedija, Android, dostupno na:
https://hr.wikipedia.org/wiki/Android_(operacijski_sustav),
(lipanj 2019.)

[2] Meet android studio, dostupno na:
https://developer.android.com/studio/intro,
(lipanj 2019.)

[3] Java, dostupno na:
https://www.java.com/,
(lipanj 2019.)

[4] Wikipedija, XML, dostupno na:
https://hr.wikipedia.org/wiki/XML,
(lipanj 2019.)

[5] Android Pit Blogspot,dostupno na
http://android-pit.blogspot.com/2012/01/arhitektura-androida.html
(kolovoz 2019.)

SAŽETAK

U ovom završnom radu razvijena je Android mobilna aplikacija budilice s kvizom nazvana Quiz Alarm. Korisnici mogu podesiti više alarma s različitim temama i težinama kviza koji se rješava kada alarm zazvoni. Kada se točno odgovori na pet pitanja alarm se gasi. Aplikacija je izrađena u Java programskom jeziku unutar programskog okruženja Android Studio. Teorijski dio objašnjava tehnologije koje se koriste za izradu aplikacije. Također je objašnjen i izgled te funkcija aplikacije popraćen slikama kako bi se točno dočaralo ono što je objašnjeno u tekstu

Ključne riječi: Android, aplikacija, budilica

ABSTRACT

Android quiz alarm application

In this bachelor's thesis, an alarm Android mobile application with a quiz called Quiz Alarm was developed. Users can set multiple quiz alarms with different themes and difficulties. The quiz is played when the alarm rings. The alarm shuts off when the user answers five quiz questions correctly. The application was made in the Java programming language in the Android Studio IDE. The theoretical part of the thesis explains the technologies used to create the application. Using pictures as well as text, the design and function of the application is also accurately described.

Keywords: Android, apps, alarm

ŽIVOTOPIS

Dominik Sabo rođen je 15. travnja 1998. godine u Našicama. Pohađao je Osnovnu školu Antuna Gustava Matoša, Čačinci. Nakon odličnim uspjehom završenog osnovnoškolskog obrazovanja upisuje Tehničku Školu u Požegi, smjer Tehničara za računarstvo. Srednju školu završava kao najučenik generacije odličnim uspjehom te dobiva direktan upis na sveučilišni studij. 2016. godine upisuje preddiplomski sveučilišni studij, smjer računarstvo na Fakultetu elektrotehnike, računarstva i informacijskih tehnologija, Osijek. Ima izvrsno znanje engleskog jezika te razgovorno zna japanski jezik. Poznaje C, C++, C#, Javu, SQL, HTML, CSS i XML.