Izrada aplikacije za Taboo igru

Kaučić, Matija

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

Sveučilišni studij

IZRADA APLIKACIJE ZA TABOO IGRU

Završni rad

Matija Kaučić

Osijek, 2020. godina

SADRŽAJ

1. UVOD

Početkom 20. stoljeća, mobilni telefoni i njihovo korištenje postali su dio ljudske svakodnevice. U svrhu veće korisnosti mobilnih telefona, razvijeni su pametni mobilni telefoni. Pametni mobilni telefoni imaju sve više funkcionalnosti koje doprinose sve široj populaciji korisnika. Uzimajući u obzir navedeno, počele su se razvijati mobilne aplikacije. Njihova je zadaća olakšati i poboljšati život ljudima. Namjene aplikacija su različite; od edukativnih do aplikacija za zabavu. Aplikacija razvijena za potrebe ovoga radu ima dvostruku namjenu zbog čega je prikladna za šire ciljano tržište.

U ovom će se radu objasniti izrada mobilne aplikacije za društvenu igru *Taboo* koja se može igrati u dvije ekipe s najviše 15 igrača u svakoj. Na početku će se odabrati koliko će igrača svake ekipe sudjelovati u igri. Korisnik prvo unosi željena imena ekipa koje se natječu, odabire broj rundi i vremensko razdoblje trajanja igre te započinje igru. Pravila igre, kao i dodatne informacije o njoj, predstavljena su unutar aplikacije. Na tržištu već postoji veliki broj implementacija društvene igre *Taboo*, a ova će se verzija razlikovati po opciji odabira težine taboo kartica pa će biti jednako prikladna i za početnike i za naprednije korisnike.

Aplikacija je napravljena za operacijski sustav Android. Razvijena je u Java programskom jeziku, u razvojnom okruženju Android Studio. Za izradu aplikacije i razumijevanje kôda potrebno je dobro znanje Java programskog jezika i poznavanje koncepata i rada unutar okruženja Android Studija.

Rad se sastoji od pet poglavlja. Nakon uvoda o mobilnim aplikacijama, u drugom će se poglavlju detaljno opisati pravila i upute za igranje igre *Taboo*. U trećem će poglavlju biti objašnjeni alati korišteni za izradu mobilne aplikacije, odnosno operacijski sustav Android, razvojno okruženje Android Studio, programski jezik Java i jezik za označavanje podataka XML. Programsko rješenje mobilne aplikacije koje uključuje izradu dizajna i funkcionalnost, tijek i način rada funkcionalnosti aplikacije bit će opisani u četvrtom poglavlju. Na kraju, zaključit će se o rezultatima uspješnosti same aplikacije i njezinog razvoja te će se opisati mogući problemi i potencijalne dorade.

1.1. Zadatak završnog rada

Taboo je društvena igra riječima koju može igrati više natjecatelja u skupinama. U igri se dobije zadana riječ koju natjecatelj treba objasniti članovima svog tima tako da ne smije koristiti zadane mu taboo riječi. Igra je prikladna za usvajanje novoga i uvježbavanje usvojenoga vokabulara prilikom učenja stranoga jezika. Zadatak je ovog završnog rada izraditi aplikaciju za navedenu igru tako da se integriraju baze podataka rječnika sinonima i antonima koji će se generirati i stvoriti bazu taboo riječi za zadane polazne riječi.

2. PRAVILA IGRE TABOO

Taboo je igra u kojoj sudjeluju dvije ekipe. Odvojeni od ostatka svog tima, stoje dva igrača - onaj koji objašnjava pojam svom timu i igrač suprotnog time koji kontrolira taboo riječi. Svaki tim ima po jednu minutu vremena za pogađanje riječi. Igrač koji objašnjava mora što brže objasniti svom timu pojam, ali pri tomu ne smije koristiti pet taboo riječi niti prekršiti pravilo nagovještaja. Zvučni efekti koji opisuju traženu riječ, npr. lajanje, mijaukanje i sl., bilo kakve gestikulacije, crtanje i pjevanje također nisu dopušteni. Suigrači izvikuju riječi za koje misle da su traženi pojam. Nema kaznenih bodova za netočne odgovore, ali se oduzima po jedan bod ako igrač koji objašnjava traženi pojam spomene taboo riječ. Kada suigrači pogode traženi pojam, igrač uzima novu riječ i objašnjava im sljedeći pojam i tako sve dok ne istekne vrijeme. Nakon isteka vremena, računaju se bodovi i zatim na isti način igra drugi tim te se na kraju uspoređuju bodovi.

Igrač suparničkog tima stoji pored igrača koji objašnjava riječi i prati popis taboo riječi. Ukoliko se upotrijebi taboo riječ ili druga nedopuštena tehnika, zviždaljkom se upozoravaju timovi i dobivaju se kazneni bodovi. Igrač koji objašnjava pojmove može preskočiti neku riječ ako to želi, nema kazne za preskakanje kartice.

Bodovanje: za svaku točno pogođenu riječ tim dobiva jedan bod, a po jedan se kazneni bod dobije svaki put kada igrač koji opisuje traženu riječ spomene taboo riječ ili upotrijebi drugu nedopuštenu tehniku. Tim koji ima više bodova pobjeđuje.

3. OPIS KORIŠTENIH TEHNOLOGIJA

Tehnologije korištene za razvoj aplikaciju su operacijski sustav Android, razvojno okruženje Android Studio, programski jezik Java i XML opisni jezik. Navedene tehnologije su unutar ovoga poglavlja.

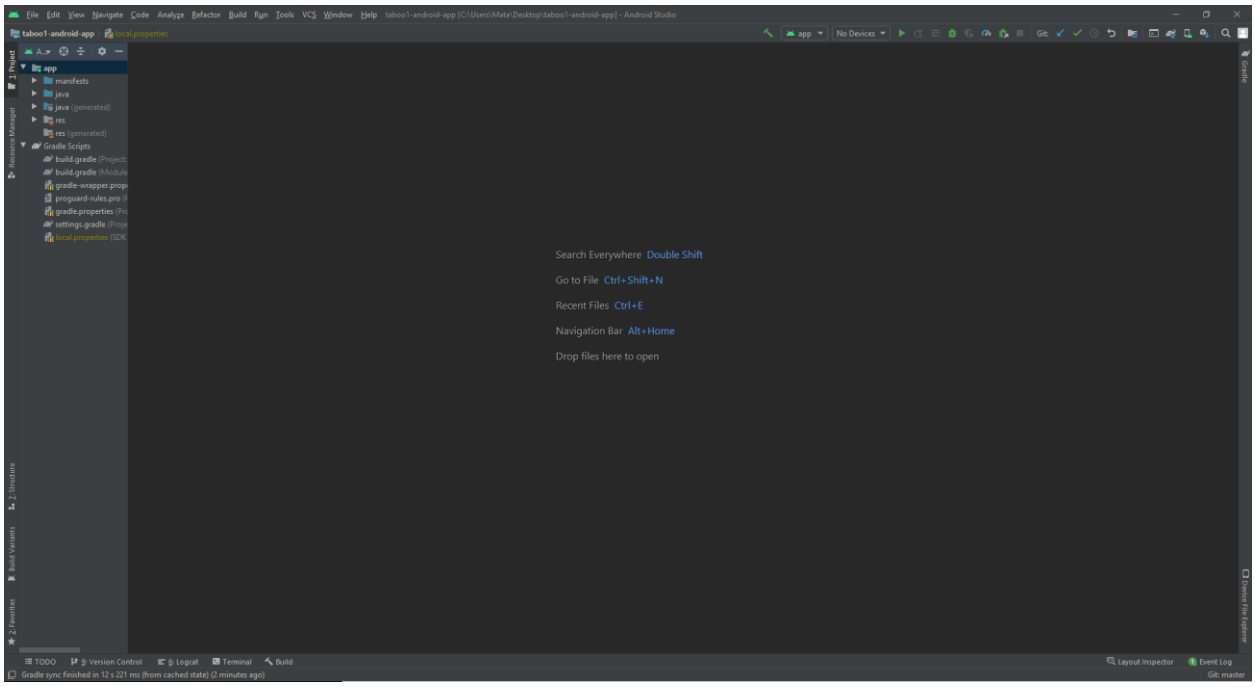
3.1. Operacijski sustav Android

Krajem 20. stoljeća na tržištu su se pojavili prvi pametni mobiteli. Prvi pametan mobitel službeno je napravljen 1992. godine, a već 1994. godine bio je dostupan široj javnosti. U pitanju je bio IBM Simon Personal Communicator ili IBM Simon [1]. Potražnja za pametnim mobitelima postajala je sve veća, a kupci su očekivali sve bolje uređaje; uređaje koji bi imali više funkcija te bili što jednostavniji i pristupačniji. U svega dvadesetak godina, tehnologija je napredovala s IBM Simona na pametne mobitele koji su uobičajeni u svakodnevnom životu poput iPhone-a, Samsunga, Huawei-a i slično. Veliki utjecaj u tom procesu imala je i tvrtka Android Inc. koja je osnovana u listopadu 2003. i čija je namjena prvotno trebala biti razvijanje operacijskog sustava za digitalne kamere, ali s obzirom da je tržište premalo, prebacili su se na razvijanje operacijskog sustava za mobitele. Open Handset Alliance (OHA) 2007. godine otkriva Android - mobilnu platformu otvorenog kôda baziranu na Linux jezgri. Android se tada na tržištu natjecao s mobilnim platformama svih većih tehnoloških kompanija, a jedine koje su se uspjele održati, a i dan danas se koriste kao operacijski sustavi svojih pametnih mobitela, su iOS tvrtke Apple i Windows Phone tvrtke Microsoft. Android je danas najprodavaniji mobilni operacijski sustav za pametne telefone i tu titulu drži od 2011., a za tablete od 2013. godine. Prema podacima iz svibnja 2019. godine [2], Android ima više od 2,5 milijarde mjesečno aktivnih korisnika, što je puno više od bilo kojeg drugog operacijskog sustava, a trgovina Play sadrži više od 3,5 milijuna aplikacija. Zanimljiv podatak kod ovog operacijskog sustava jest i da zasad postoji 18 verzija Androida i svi su, osim prvih i zadnjih dviju, nazvani po nekoj slastici. Verzija 1.5 je bila nazvana *Cupcake*, verzija 1.6 *Donout* i tako abecednim redom do Androida 9.0 koji je bio poznat i kao *Pie* [3].

3.2. Razvojno okruženje Android Studio

Android Studio je službeno integrirano razvojno okruženje za Googleov operacijski sustav Android, a temelji se na IntelliJ IDEA softveru. Prvi je put bio najavljen 2013. na Googleovoj konferenciji razvojnih inženjera, a prva stabilna verzija puštena za javnost bila je krajem 2014. godine. Android Studio je tako zamijenio *Eclipse Android Development Tools* (E-ADT) kao

primarno okruženje za razvoj android aplikacija. Osim svoje jednostavnosti, kompaktnosti i izravnog i čitkog grafičkog sučelja (Slika 3.1.), dostupan je na svim popularnijim operacijskim sustavima kao što su Windows, Linux i macOS i potpuno je besplatan te zato broji rastući broj svakodnevnih korisnika.



Slika 3.1. Sučelje razvojnog okruženja Android Studio

Još su neke od funkcionalnosti Android Studija brzo otklanjanje pogrešaka (engl. *debugging*), Google Cloud kompatibilnost, integrirana sučelja za komunikaciju s *online* bazama podataka poput Firebase-a, predlošci čestih funkcionalnosti aplikacija, prethodni pregled trenutnog izgleda zaslona aplikacije te mogućnost testiranja aplikacije bez fizičkog pametnog mobitela, već pomoću virtualnog uređaja i na svim operacijskim sustavima.

Do svibnja 2019. preferirani Googleov programski jezik za razvoj Android aplikacija bila je Java kada ju je zamijenio Kotlin [4], ali Android Studio još uvijek nudi podršku za Javu, kao i za C++.

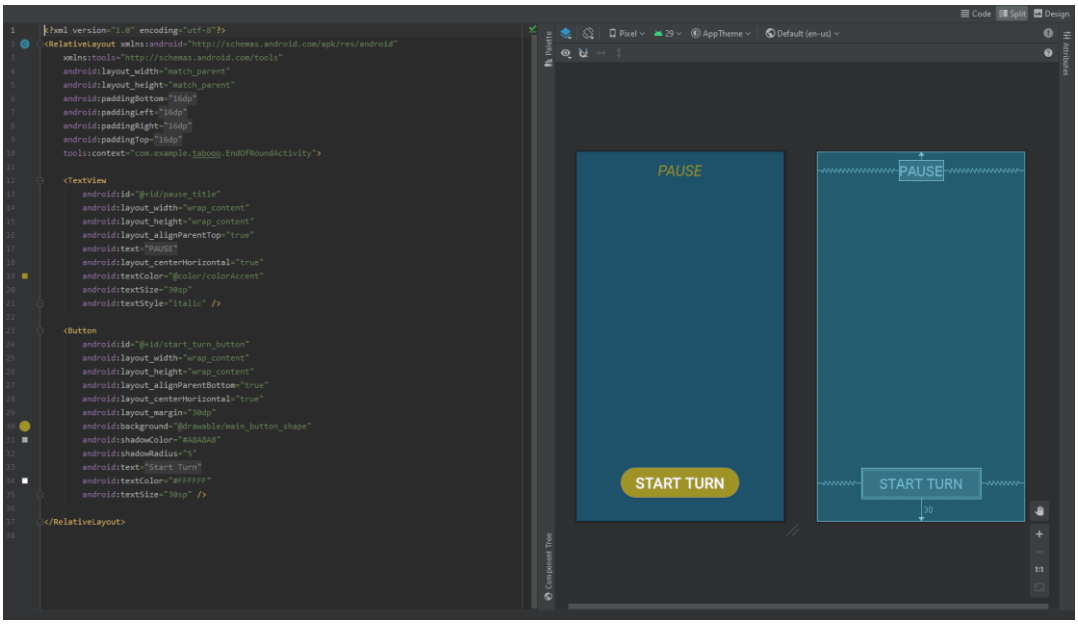
3.3. Programski jezik Java

Inženjeri u tvrtci Sun Microsystems 1991. su godine započeli, a 1995. i dovršili razvoj objektno orijentiranog programskog jezika koji se temelji na klasama. Važno je napomenuti kako Java nije u potpunosti objektno orijentirani jezik jer sadrži neke tipove podataka koji nisu objekti

(int, float, bool i sl.). Glavna ideja ovog programskog jezika bila je da se kôd koji programer jednom napiše može pokrenuti na svim platformama koje podržavaju Javu bez potrebe za ponovnim kompiliranjem. To je moguće na svim računalima za koje postoji Java Virtual Machine (JVM) koji omogućuje prikaz aplikacija bez pretvaranja kôda u strojni jezik. Sintaksa se temelji na C i C++ jeziku da bi programerima bila poznatija i lakše shvatljiva [5]. Najvažnije karakteristike ovog programskog jezika su višenitnost [6], kompatibilnost rada s bazama podataka, web aplikacije s vezom klijent–server itd. U današnje doba, Java je vrlo popularan programski jezik zbog svoje višenamjenske karakteristike te jednostavnosti, lakšeg pisanja i čitljivosti kôda. Iako je višenamjenski jezik, uglavnom se koristi za izradu aplikacija.

3.4. XML

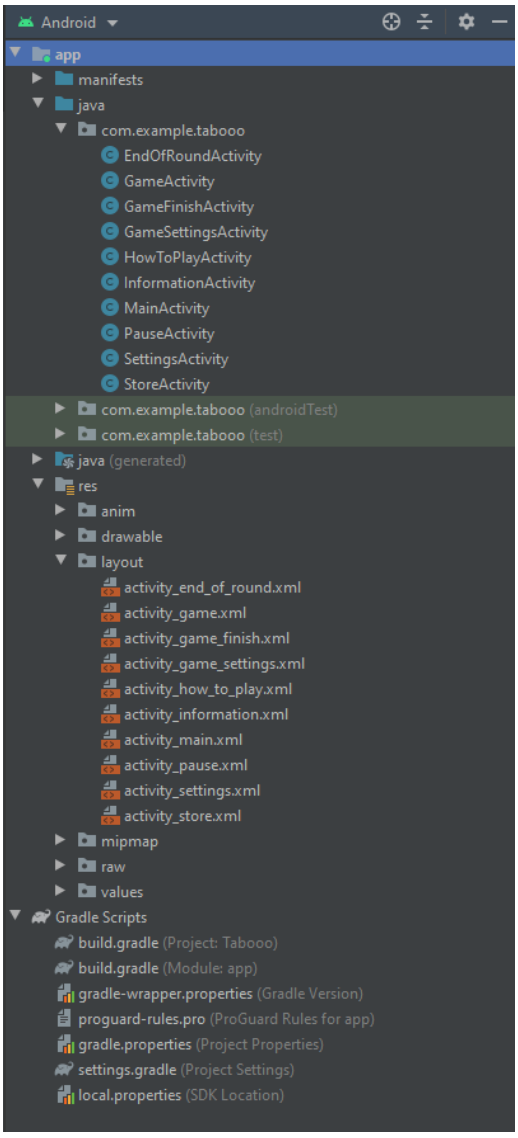
XML (engl. *EXtensible Markup Language*) je jezik za označavanje podataka. Za razliku od drugih programskih jezika, on služi samo za opisivanje podataka, a sam po sebi nema neku funkciju [7]. Glavna misao vodilja u stvaranju ovakvog jezika bila je stvaranje jezika koji će biti jednostavan i čitljiv i ljudima i računalu. Format i sintaksa su vrlo slični HTML jeziku gdje sadržaj elemenata mora biti omeđen s lijeve i s desne strane početnom i završnom oznakom. Unutar gore spomenutih oznaka pišu se sve specifikacije određenog elementa - od boje, veličine teksta, fonta pa sve do visine i širine samog elementa. U ovoj je aplikaciji XML korišten za dizajniranje grafičkog sučelja vidljivog korisniku kao što je prikazano na Slici 3.2.



Slika 3.2. Primjer XML kôda

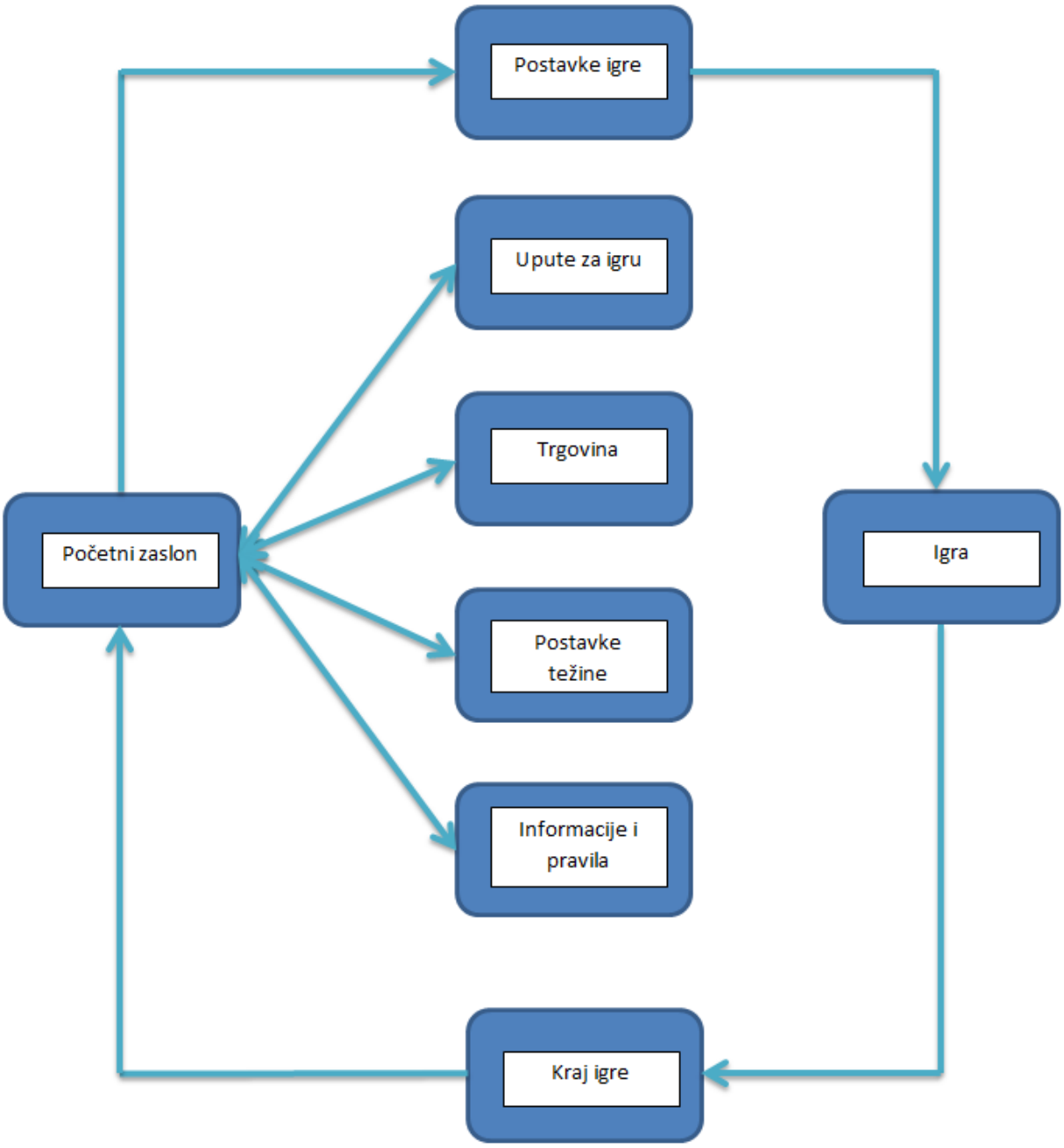
4. PROGRAMSKO RJEŠENJE MOBILNE APLIKACIJE

U nastavku je opisano programsko rješenje mobilne aplikacije koje je podijeljeno na programski kôd dizajna i kôd funkcionalnosti aplikacije. Dizajn kôda se sastoji od XML jezika, a funkcionalnosti i logika aplikacije pisani su u Javi. Struktura projekta prikazana je na Slici 4.1. na kojoj se vidi da je napravljeno po deset aktivnosti za oba dijela – dizajn i logiku kôda. Aktivnost (engl. *activity*) predstavlja jedan mogući zaslon aplikacije, a aplikacija ih ima deset.



Slika 4.1. Struktura mobilne aplikacije

Svaka aktivnost zasebno je rađena i povezana je pritiskom na određenu tipku, odnosno događaj u aplikaciji. Tijek aktivnosti aplikacije vidljiv je na dijagramu na Slici 4.2.



Slika 4.2. Dijagram tijeka aktivnosti aplikacije

4.1. Dizajn mobilne aplikacije

Dizajn aplikacije rađen je pomoću jezika za označavanje podataka - XML. Podijeljen je u deset XML datoteka, a u strukturi projekta one se nalaze pod datotekom resursa pod nazivom *layout*. Osim integriranih oblika i komponenta dizajna koje posjeduje Android Studio, nekada je potrebno implementirati vanjske komponente poput *CardView*-a [8]. *CardView* je mala aplikacija ili sučeljni program (engl. *widget*) koja je predstavljena dolaskom Android-a 7.0. U osnovi, *CardView* je vrlo sličan *FrameLayout*-u [9] uz razliku što ima zaobljene rubove. *CardView* obavija *layout* pa je često korišten kao kontejner za različite elemente pogleda (engl. view).

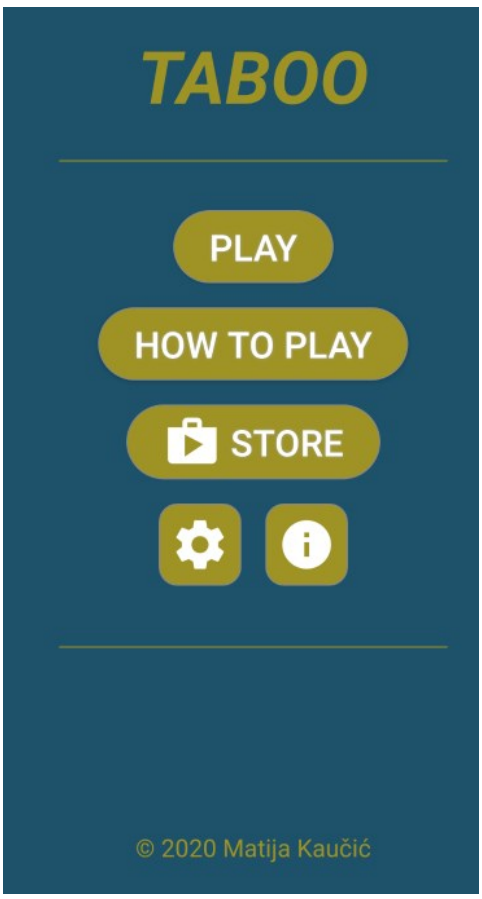
Implementacija vanjskih komponenti omogućena je dodavanjem implementacija u Gradle skriptu (Slika 4.3.) koja govori koje se sve biblioteke moraju uključiti u projekt da bi on ispravno radio.

```
dependencies {
    implementation fileTree(dir: "libs", include: ["*.jar"])
    implementation 'androidx.appcompat:appcompat:1.1.0'
    implementation 'androidx.constraintlayout:constraintlayout:1.1.3'
    testImplementation 'junit:junit:4.12'
    androidTestImplementation 'androidx.test.ext:junit:1.1.1'
    androidTestImplementation 'androidx.test.espresso:espresso-core:3.2.0'
    implementation 'androidx.cardview:cardview:1.0.0'
```

Slika 4.3. Gradle skripta

4.1.1. Početni zaslon

Početni zaslon (activity_main) označava prvi vidljivi zaslon kada se pokrene mobilna aplikacija. Konkretno, na ovom zaslonu (Slika 4.4.), postoji *TextView* s imenom aplikacije te pet gumba od kojih svaki ima svoju funkcionalnost. Prikazan je i primjer XML kôda za gumb *Store* (Slika 4.5.). Njegova je posebnost što u lijevom uglu samog gumba ima sliku koja se mora dodati u datoteku resursa pod *drawable* i od tamo se referencira naredbom *@drawable/.*



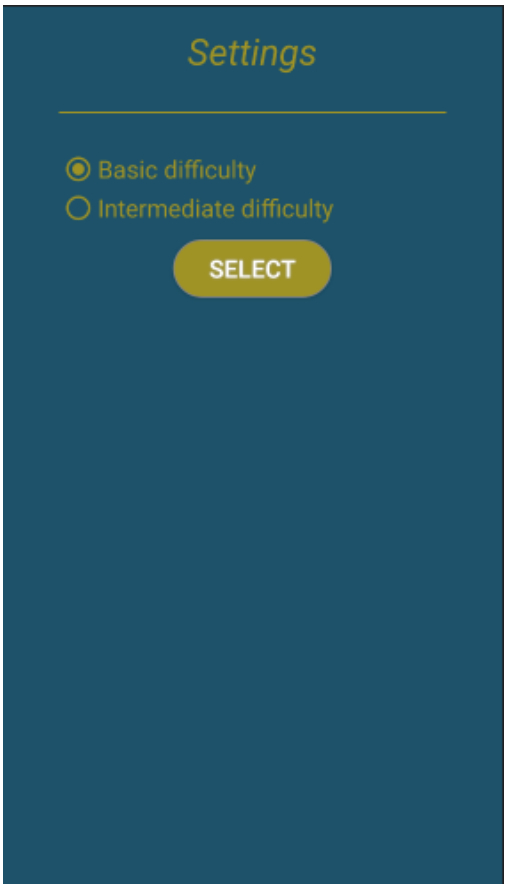
Slika 4.4. Izgled početnog zaslona

```
<Button
    android:id="@+id/store_button"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_below="@+id/how_to_play_button"
    android:layout_centerHorizontal="true"
    android:layout_margin="10dp"
    android:background="@drawable/main_button_shape"
    android:contentDescription="Store"
    android:drawableLeft="@drawable/ic_shop_white_48dp"
    android:paddingBottom="7dp"
    android:paddingLeft="30dp"
    android:paddingRight="30dp"
    android:paddingTop="7dp"
    android:shadowColor="#A8A8A8"
    android:shadowRadius="5"
    android:text=" Store"
    android:textColor="#FFFFFF"
    android:textSize="30sp" />
```

Slika 4.5. XML kôd za *Store* gumb

4.1.2. Postavke težine

Postavke težine odabiru se na posebnom zaslonu koji je prikazan na slici 4.6. Zadana je osnovna težina (*basic difficulty*). U postavkama su stavljena dva radio gumba koja su uključena u jednu radio grupu (Slika 4.7.), što omogućuje da u svakom trenutku samo jedan od tih dvaju gumba može biti odabran. Također, postoji i tipka koja potvrđuje odabir trenutne težine nakon što se označi željena težina putem radio gumba.



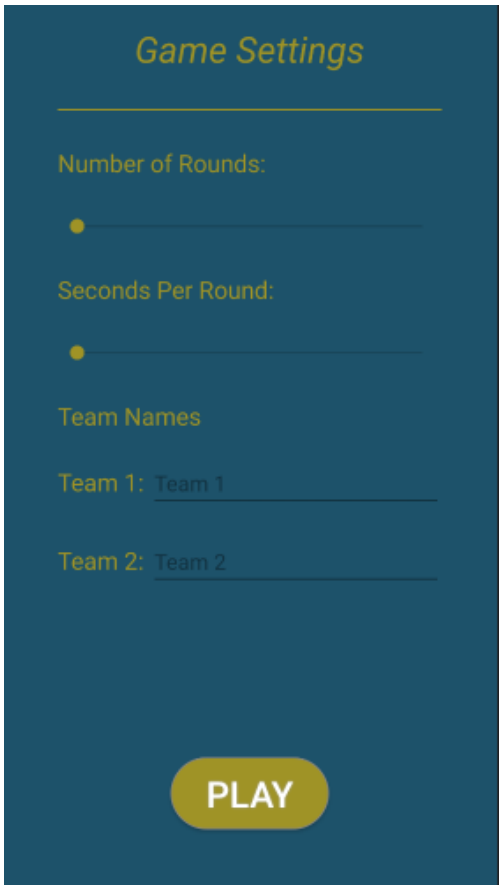
Slika 4.6. Zaslon postavke težine

```
<RadioGroup xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/settings_radio_group"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:orientation="vertical"
    android:layout_below="@+id/settings_view">

    <RadioButton android:id="@+id/radio_button_basic"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginLeft="30dp"
        android:layout_marginRight="30dp"
        android:text="Basic difficulty"
        android:textColor="@color/colorAccent"
        android:textSize="20dp"
        android:checked="true"
        android:onClick="onRadioButtonClicked"
        android:buttonTint="@color/colorAccent"/>

    <RadioButton android:id="@+id/radio_button_intermediate"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginLeft="30dp"
        android:layout_marginRight="30dp"
        android:text="Intermediate difficulty"
        android:textColor="@color/colorAccent"
        android:textSize="20dp"
        android:onClick="onRadioButtonClicked"
        android:buttonTint="@color/colorAccent"/>
</RadioGroup>
```

Slika 4.7. XML kôd radio grupe

4.1.3. Postavke igre

Zaslon postavaka igre (Slika 4.8.) pokreće se nakon što korisnik odabere gumb *Play* s početnog zaslona aplikacije. Zaslon se sastoji od nekoliko TextView-ova koji tekstualno opisuju elemente samog zaslona. Koriste se dva SeekBar-a (Slika 4.9.) pomoću kojih korisnik može odabrati željeni broj rundi i vremensko trajanje svake runde u sekundama. Zadnji korišteni element je EditText (Slika 4.10.) koji od korisnika zahtijeva unos teksta za nazive ekipa. Također, ako korisnik ne upiše imena ekipa, po *defaultu* je zadano da su imena ekipa *Team 1* i *Team 2*.



Slika 4.8. Zaslon postavki igre

```
<SeekBar
    android:id="@+id/num_rounds_seek_bar"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_below="@+id/num_rounds_choose_text_view"
    android:layout_margin="30dp"
    android:max="14" />
```

Slika 4.9. XML kôd SeekBar elementa

```
<EditText
    android:id="@+id/team_name1_editText"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:hint="Team 1"
    android:singleLine="true"
    android:maxLength="20"
    android:maxLines="1" />
```

Slika 4.10. XML kôd EditText elementa

4.1.4. Aktivnost igre

Zaslon koji prikazuje aktivnost igre (Slika 4.11.) dizajnerski je najkompleksniji u cijeloj aplikaciji. Zaslon aktivnosti igre sastoji se od nekoliko TextView-ova koji predstavljaju tajmer, ime ekipe, broj trenutne runde, ukupan rezultat, rezultat trenutne runde itd. Koriste se četiri ImageButton-a koji služe za označavanje točnog odgovora, preskakanje trenutne kartice, dodjeljivanje negativnog boda i stanku. CardView je bijele boje i smješten je u sredini zaslona (Slika 4.12.). Služi kao kontejner za TextView elemente koji označuju jednu traženu riječ i pet taboo riječi.

Slika 4.11. Zaslon aktivnosti igre

```
<TextView
    android:id="@+id/round_timer_text_view"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_alignParentLeft="true"
    android:layout_alignParentTop="true"
    android:layout_marginLeft="28dp"
    android:layout_marginRight="15dp"
    android:layout_marginTop="3dp"
    android:text="time_left"
    android:textSize="35sp"
    android:textStyle="bold"
    android:textColor="@color/colorAccent"/>
```

Slika 4.12. XML kôd tajmera

4.2. Funkcionalnosti mobilne aplikacije

Funkcionalni dio mobilne aplikacije pisan je u programskom jeziku Javi. Funkcionalnosti koje aplikacija omogućuje su:

- mogućnost odabira broja rundi;
- mogućnost odabira vremenskog trajanja jedne runde;
- postavljanje imena timova;
- promjena težine igre;
- upute za igranje;
- pravila igre i način bodovanja;
- informacije o aplikaciji.

4.2.1. Početni zaslon

Početni se zaslon sastoji od tipki *Play, How to play, Store, Settings* i *Information* i klikom na bilo koji od njih otvorit će se novi zaslon. Klikom na *Play* otvara se novi zaslon - postavke igre. Klikom na *How to play* otvaraju se upute za korištenje mobilne aplikacije; klikom na *Settings* postavke težine, klikom na *Store* novi zaslon u kojemu je ispisano kako aplikacija, nažalost, još nije dostupna na *Trgovini Play*, a klikom na *Information* korisnik dobije prijepis pravila igre i sl. To se postiže postavljanjem prisluškivača klika (engl. *OnClickListener*) na elementima zaslona, odnosno uglavnom nekog gumba. Za primjer uzmimo aktivnost informacija. U kôdu na Slici 4.13. vidljiva je implementaciju prisluškivača ako korisnik klikne na gumb *Information*. Unutar same metode potrebno je prepisati preko (engl. *override*) metode onClick koja kao argument prima element tipa View. Potrebno je inicijalizirati instancu klase Intent, njoj u konstruktoru predati kontekst (engl. *context*) i aktivnost koju želimo pokrenuti pri događaju pritiska na gumb *Information*. Zatim, koristimo metodu za pokretanje aktivnosti te njoj predamo gore inicijaliziranu instancu klase Intent. Na isti način postavljaju se prisluškivači i na ostale gumbe gdje svaki otvara svoju odgovarajuću aktivnost.

```
    information.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View view) {
            Intent intent = new Intent( packageContext: MainActivity.this, InformationActivity.class);
            startActivity(intent);
        }
    });
}
```

Slika 4.13. Postavljanje prisluškivača na tipku *Information*

4.2.2. Postavke težine

Početno zadana težina taboo kartica je osnovna razina, a korisnik može ručno birati između osnovne i napredne razine, što ostvaruje klikom na željeni gumb (RadioButton). U aktivnosti je definirana kontejnerska klasa (Slika 4.14.) koja u sebi sprema trenutnu vrijednost odabrane težine kartica. U njoj se nalazi statični element tipa *Difficulty* koji za početnu vrijednost ima zadanu lakšu razinu. Tip podataka *Difficulty* definiran je kao element tipa enum. Enum je posebna vrsta klase koja predstavlja grupu konstanti. Željene konstante se pišu unutar uglate zagrade, odvajaju zarezom i praksa je da se pišu tiskanim slovima. Vrijednosti enum elementima mogu se dodavati korištenjem točke kao što je prikazano u slučaju varijable *currentDifficulty*. Unutar klase *DifficultySettings*, definirana je i javna metoda *setDifficulty* za postavljanje željene težine kartica. Metoda *setDifficulty* kao argument prima element tipa *Difficulty* i zatim zamjenjuje trenutnu težinu kartica danom težinom.

```
public static class DifficultySettings{

    public static Difficulty currentDifficulty = Difficulty.EASY;

    public static void setDifficulty (Difficulty newDifficulty){
        currentDifficulty = newDifficulty;
    }
}

public enum Difficulty{

    EASY, HARD, MEDIUM
}
```

Slika 4.14. Kontejnerska klasa za pohranu trenutne vrijednosti težine

Na Slici 4.15. prikazana je javna metoda za odabir težine koja za argument prima instancu klase *View*. Na temelju referenci gumba (RadioButton), u *if* petlji prvo se provjerava je li pritisnut gumb za lakšu težinu te ako je, poziva se metoda *setDifficulty*, postavlja se vrijednost trenutne težine u osnovnu i izlazi se iz petlje. Ako nije pritisnuta tipka za lakšu težinu, u drugoj *if* petlji provjerava se je li stisnuta tipka za naprednu težinu. Ako je, poziva se metoda *setDifficulty*, postavlja se vrijednost trenutne težine u naprednu te se zatim izlazi iz petlje.

```
public void onRadioButtonClicked(View view) {
    // Is the button now checked?
    boolean checked = ((RadioButton) view).isChecked();

    // Check which radio button was clicked
    switch(view.getId()) {
        case R.id.radio_button_basic:
            if (checked)
                // basic difficulty is selected
                DifficultySettings.setDifficulty(Difficulty.EASY);
                break;
        case R.id.radio_button_intermediate:
            if (checked)
                // intermediate difficulty is selected
                DifficultySettings.setDifficulty(Difficulty.MEDIUM);
                break;
    }
}
```

Slika 4.15. Metoda za odabir trenutne težine

Sljedeća metoda ima za zadaću u obliku *toast* obavijesti [10] informirati korisnika koju je težinu odabrao nakon što stisne tipku *Select* (Slika 4.16.). *Toast* obavijesti su kratke poruke koje se obično nalaze na dnu ekrana mobitela. Koriste se kako bi korisnicima pružili kratke i brze informacije o onome što se trenutno događa u mobilnoj aplikaciji. Prvo se definira radio grupa koja predstavlja dva gumba koji se nalaze u zaslonu postavka težine, a zatim i varijabla tipa *Button* koja predstavlja odabrani gumb. Postavlja se prisluškivač na odabrani gumb i unutar te metode mora se prepisati preko (engl. *override*) *onClick* metode. Unutar *onClick* metode, varijabli *selectedID* koja je tipa *int*, dodjeljuje se vrijednost identifikacijskog broja odabranog gumba. Tim se načinom u svakom trenutku može naći trenutno odabrani gumb, a kako bi korisnik znao da je odabrao željenu težinu, također se ispisuje i *toast* obavijest. Ispis *toast* obavijesti se ostvaruje metodom *makeTex*t i predavanjem željenih argumenata - konteksta, teksta obavijesti te vremenske duljine *toast* obavijesti koja je ili *LENGTH_LONG* ili *LENGTH_SHORT*. Za prikaz na korisničkom sučelju koristi se metoda *show*().

```
public void addListenerOnButton() {

    radioSelectGroup = (RadioGroup) findViewById(R.id.settings_radio_group);
    selectButton = (Button) findViewById(R.id.settings_select_button);

    selectButton.setOnClickListener(new View.OnClickListener() {

        @Override
        public void onClick(View v) {

            // get selected radio button from radioGroup
            int selectedId = radioSelectGroup.getCheckedRadioButtonId();

            // find the radiobutton by returned id
            radioSelectButton = (RadioButton) findViewById(selectedId);

            Toast.makeText( context: SettingsActivity.this,
                    radioSelectButton.getText(), Toast.LENGTH_SHORT).show();

        }

    });

}
```

Slika 4.16. Metoda za ispis toast obavijesti

4.2.3. Postavke igre

U aktivnosti postavki igre postavlja se prisluškivač na tipku *Play* (Slika 4.17.) pomoću koje se pokreće aktivnost igre. Unutar same metode prisluškivača, prvo se inicijaliziraju dvije varijable tipa *EditText* koje su nazvane *name1* i *name2*, a označavaju imena ekipa. Zatim se inicijaliziraju dvije varijable tipa *SeekBar* koje označavaju broj rundi i količinu vremena po rundi. Sva četiri elementa tada se referenciraju preko odgovarajućih XML grafičkih elemenata pomoću naredbe *findViewById*. Nakon toga, ovisno o korisničkom odabiru, dodjeljuju se odabrane vrijednosti trajanja runde i broja runde gore navedenim varijablama pomoću metode *getProgress* koja vraća vrijednost tipa *int*, a označava pomak prema desno na *SeekBar-u*. U *if* petlji (name1.getText().equals(null)), osigurava se da postoji neko ime ekipe čak i ako korisnik ne upiše nijedno. Ako korisnik upiše ime ekipe, onda se pozivanjem metode getText().toString(), preko varijable *name1*, dodjeljuje ime ekipe koje je vidljivo na korisničkom sučelju tijekom aktivnosti igre. Isti princip je i za drugu *if* petlju i naziv druge ekipe. Naredbama *putExtra* na kraju metode osigurava se da podaci odabrani u ovoj aktivnosti budu vidljivi i u aktivnosti igre.

```
play_button.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        Intent intent = new Intent( packageContext: GameSettingsActivity.this, GameActivity.class);

        EditText name1 = (EditText)findViewById(R.id.team_name1_editText);
        EditText name2 = (EditText)findViewById(R.id.team_name2_editText);
        SeekBar numRoundsSeekBar = (SeekBar)findViewById(R.id.num_rounds_seek_bar);
        SeekBar secondPerRoundSeekBar = (SeekBar)findViewById(R.id.time_in_round_seek_bar);

        int numRoundsProgress = numRoundsSeekBar.getProgress();
        int secondPerRoundProgress = secondPerRoundSeekBar.getProgress();

        String message1 = "";
        String message2 = "";
        if (name1.getText().equals(null)) {
            message1 = "Team 1";
        }
        else {
            message1 = name1.getText().toString();
        }
        if (name2.getText().equals(null)) {
            message2 = "Team 2";
        }
        else {
            message2 = name2.getText().toString();
        }

        intent.putExtra(TEAM_ONE_NAME, message1);
        intent.putExtra(TEAM_TWO_NAME, message2);
        intent.putExtra(NUM_ROUNDS, Integer.toString( i: numRoundsProgress+1));
        intent.putExtra(SECONDS_PER_ROUND, Integer.toString( i: secondPerRoundProgress+15));

        startActivity(intent);
    }
});
```

Slika 4.17. Prisluškivač na tipku *Play*

4.2.4. Aktivnost igre

Kako je bilo i po pitanju dizajna, tako je i u logičkom smislu ovo najsloženija aktivnost cijele aplikacije, a to je zato što ima najviše funkcionalnosti. Zadaća ove aktivnosti je istovremeno generirati nove taboo kartice, upravljati tajmerom, u stvarnom vremenu ažurirati broj bodova ekipe koja je trenutno na redu te na kraju kruga promijeniti ekipu koja igra.

Prve od navedenih metoda su metode za ažuriranje bodova. Nakon što korisnik dobije odgovarajuću povratnu informaciju (točan odgovor) od svoje ekipe, pritišće tipku s kvačicom te time ekipa dobiva bod. To se ostvaruje dodavanjem prisluškivača na tipku kvačice (Slika 4.18.). Unutar *onClick* metode, prvo se u *if* petlji provjerava koja je ekipa na redu, a to nam govori vrijednost varijable *turn*. Varijabla *turn* je tipa *bool*, a početno joj je postavljena vrijednost istinita (engl. *true*), što označava da je na redu prva ekipa. U metodi za kraj kruga nalazi se linija kôda *turn=!turn* koja mijenja vrijednost varijable *turn* u lažnu te označava da je na redu iduća ekipa. Ako je varijabla *turn* istinita, dodjeljuje se jedan bod *unarnim* operatorom inkrementa (++) prvoj ekipi, a ako varijabla *turn* nije istinita, bod se dodjeljuje drugoj ekipi. Metoda *updateCard* učitava novu karticu, a metoda u*pdateScores* pohranjuje trenutni rezultat.

```
correct_button.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        // update score
        if (turn) {
            team_one_score++;
            team_one_round_score++;
        }
        else {
            team_two_score++;
            team_two_round_score++;
        }
        updateCard();
        updateScores();
    }
});
```

Slika 4.18. Metoda za dodavanje bodova

Isto tako, ako korisnik kaže taboo riječ, pritišće se tipka s križićem i tada njegova ekipa gubi jedan bod. Sličnom logikom kao i u metodi za dodjeljivanje boda, negativni bod dodjeljuje se postavljajući prisluškivač na tipku križića (Slika 4.19.). U ovom slučaju, jedina je razlika što se koristi operator dekrement (--).

```
taboo_button.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        // update score
        if (turn) {
            team_one_score--;
            team_one_round_score--;
        }
        else {
            team_two_score--;
            team_two_round_score--;
        }
        updateScores();
        updateCard();
    }
});
```

Slika 4.19. Metoda za dodavanje negativnih bodova

Također, korisnik može preskočiti trenutnu karticu i tada se ta kartica vraća u fond kartica (Slika 4.20.). U ovoj metodi ne događa se nikakvo dodjeljivanje bodova; samo se metodom *updateCard* otvara nova kartica.

```
pass_button.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        updateCard();
    }
});
```

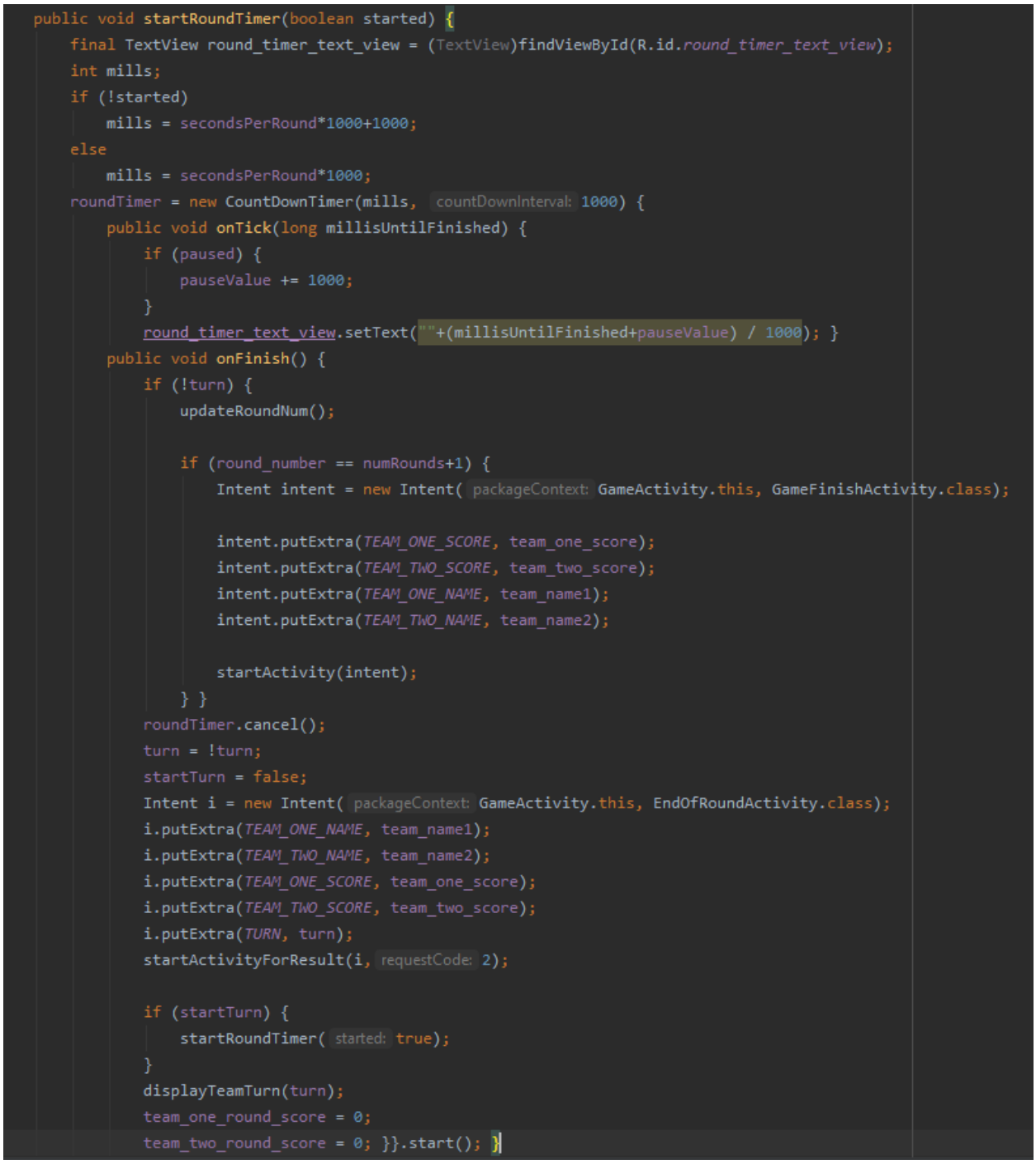
Slika 4.20. Metoda za preskakanje trenutne kartice

Nadalje, metoda za ažuriranje rednog broja runde radi na principu inkrementa kada obje ekipe odigraju svoj potez i u formatu je *Trenutna runda/Korisnički odabran broj rundi*. Na početku aktivnosti, zadana je početna vrijednost koja je jednaka jedan jer broj rundi kreće od prve runde pa nadalje (Slika 4.21.). Ova metoda se jedino poziva u metodi upravljanja tajmerom jer se tada jedino mijenja broj trenutne runde.

```
public void updateRoundNum () {
    TextView roundNumTextView = (TextView)findViewById(R.id.round_text_view);
    roundNumTextView.setText("Round " + round_number + "/" + numRounds);
    round_number++;
}
```

Slika 4.21. Metoda za ažuriranje rednog broja runde

Jedna od zanimljivijih metoda jest metoda zadužena za upravljanje tajmerom (Slika 4.22.). Ona ima implementirane metode koje su zadužene za ponašanje aplikacije pri predugoj stanci i nakon što prođe odabrano vrijeme trajanja jedne runde. Metoda, za argument, prima *bool* varijablu koja označava je li počeo tajmer ili nije. U metodi se pronalazi korisnički odabran podatak o vremenu trajanja svakog kruga i ako je krug započeo, pokreće se tajmer. Metoda *onFinish* je zadužena za ponašanje aplikacije prilikom završetka runde, odnosno igre. Petlja *if* (!turn) događa se svaki put pri promjeni poteza i ona je zadužena za promjenu rednog broja trenutne runde pomoću metode *updateRoundNum*. Unutar te petlje nalazi se još jedna petlja čija je zadaća prekinuti izvođenje aktivnosti igre ako je trenutan redni broj runde veći od korisnički zadanog broja rundi. Po završetku odbrojavanja tajmera, tajmer se isključuje i preko *intent*-a šalju se imena ekipa i brojevi bodova u aktivnost koja se sljedeća poziva, a to je *GameFinishActivity*. Na kraju metode provjerava se je li *startTurn* varijabla istinita te ako je, nastavlja se igra i ponovno se postavlja tajmer. Također, mijenja se ekipa koja je trenutno na redu za igru te se ponovno postavljaju rezultati trenutnog kruga za obje ekipe na nulu. Ukupan rezultat koji su ekipe ostvarile ostat će pohranjen i ažuriran nakon svake runde dodavanjem bodova trenutne runde na ukupan zbroj bodova.



```
public void startRoundTimer(boolean started) {
    final TextView round_timer_text_view = (TextView)findViewById(R.id.round_timer_text_view);
    int mills;
    if (!started)
        mills = secondsPerRound*1000+1000;
    else
        mills = secondsPerRound*1000;
    roundTimer = new CountDownTimer(mills,  countDownInterval: 1000) {
        public void onTick(long millisUntilFinished) {
            if (paused) {
                pauseValue += 1000;
            }
            round_timer_text_view.setText(""+(millisUntilFinished+pauseValue) / 1000); }
        public void onFinish() {
            if (!turn) {
                updateRoundNum();

                if (round_number == numRounds+1) {
                    Intent intent = new Intent( packageContext: GameActivity.this, GameFinishActivity.class);

                    intent.putExtra(TEAM_ONE_SCORE, team_one_score);
                    intent.putExtra(TEAM_TWO_SCORE, team_two_score);
                    intent.putExtra(TEAM_ONE_NAME, team_name1);
                    intent.putExtra(TEAM_TWO_NAME, team_name2);

                    startActivity(intent);
                } }
            roundTimer.cancel();
            turn = !turn;
            startTurn = false;
            Intent i = new Intent( packageContext: GameActivity.this, EndOfRoundActivity.class);
            i.putExtra(TEAM_ONE_NAME, team_name1);
            i.putExtra(TEAM_TWO_NAME, team_name2);
            i.putExtra(TEAM_ONE_SCORE, team_one_score);
            i.putExtra(TEAM_TWO_SCORE, team_two_score);
            i.putExtra(TURN, turn);
            startActivityForResult(i, requestCode: 2);

            if (startTurn) {
                startRoundTimer( started: true);
            }
            displayTeamTurn(turn);
            team_one_round_score = 0;
            team_two_round_score = 0; }}.start(); }
```

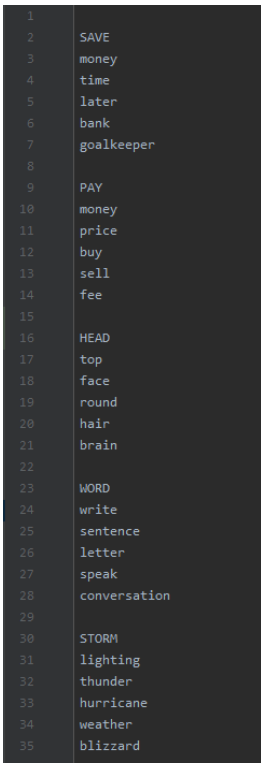
Slika 4.22. Metoda za upravljanje tajmerom i događajima nakon završetka tajmera

Posljednja je metoda za učitavanje taboo kartica (Slika 4.23.). Ona radi na principu pribavljanja vrijednosti trenutno odabrane težine iz referenci radio gumba. Metoda *getCards* vraća dinamičku matricu koja sadrži podatke tipa *String*. U njoj se prvo definira varijabla tipa *bool* kojoj se pridodaje osnovna vrijednost težine, odnosno *easy*. Nakon toga, inicijalizira se varijabla tipa

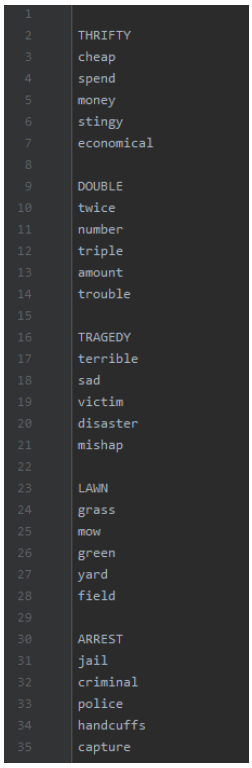
InputStream i pomoću metode *openRawResource* čita se trenutno odabrana postavka težine. Linija kôda (isEasy ? R.raw.*basic_taboo_cards* : R.raw.*intermediate_taboo_cards*) je kraći oblik *if* petlje. Ona za vrijednost varijable *is* vraća R.raw.*basic_taboo_cards* ako je tvrdnja u petlji istinita, odnosno ako je odabran gumb za osnovnu težinu. Ako tvrdnja u petlji nije istinita, odnosno ako je odabrana tipka za naprednu težinu, za vrijednost varijable *is* vraća R.raw.*intemediate_taboo_cards*. Zatim se definira varijabla *s* tipa *BufferedReader* koja se koristi za čitanje teksta iz *InputStream* varijable. Definira se nova dinamička matrica koja sadrži podatke tipa *Sting*, a nazvana je *out*. Također, definira se varijabla *index* tipa *int* i postavlja se na vrijednost nula. Varijabla *index* će označavati poziciju trenutno pročitane kartice. U *while* petlji prvo se provjerava je li varijabla *s* spremna biti pročitana. Ako je, poziva se metoda *readLine* kako bi se pročitala iduća linija iz *s* varijable. Inicijalizira se prazan dinamični niz *textList* koji će sadržati podatke tipa *String* nakon čega se taj niz dodaje u dinamičku matricu *out*. Zatim je napisana *for* petlja koja će se ponavljati šest puta i koja će pročitati šest linija jednu po jednu. Prva linija koju će pročitati bit će tražena riječ, a sljedećih pet će biti taboo riječi. To je zato što se taboo kartice u datoteci zapisuju kao šest riječi gdje prva riječ predstavlja traženu riječ, a ostalih pet su taboo riječi koje su na kraju odvojene znakom za novi red (Slike 4.24. i 4.25.). Po završetku *while* for petlje, inkrementalno se povećava varijabla *index* i označava da je gotovo čitanje taboo kartice i da se može prijeći na drugu. Ovaj proces se ponavlja sve dok je varijabla *s* spremna biti pročitana, odnosno dok se ne pročitaju sve linije iz datoteke koja sadrži taboo kartice. Po završetku metode, zatvaraju se varijable *InputStream-a* i *BufferedReader-a* te se vraća dinamička matrica *out*.

```
public ArrayList<ArrayList<String>> getCards() throws IOException {

    boolean isEasy = SettingsActivity.DifficultySettings.currentDifficulty == SettingsActivity.Difficulty.EASY;
    InputStream is = this.getResources().openRawResource(isEasy ? R.raw.basic_taboo_cards : R.raw.intermediate_taboo_cards);
    BufferedReader s = new BufferedReader(new InputStreamReader(is));

    ArrayList<ArrayList<String>> out = new ArrayList<ArrayList<String>>();

    int index = 0;
    while (s.ready()) {
        s.readLine();
        ArrayList<String> textList = new ArrayList<String>();
        out.add(textList);
        for (int i = 0; i < 6; i++) {
            out.get(index).add(s.readLine());
        }
        index++;
    }
    is.close();
    s.close();
    return out;
}
```

Slika 4.23. Metoda za učitavanje taboo kartica



Slika 4.24. Taboo kartice lakše težine



Slika 4.25. Taboo kartice napredne težine

4.3. Prikaz mobilne aplikacije na fizičkom uređaju

U ovom poglavlju biti će prikazana razvijena mobilna aplikacija u okolišu pravog fizičkog Android uređaja, izgled sučelja mobilne aplikacije i interakcija s korisnikom aplikacije. U nekim slučajevima (nedovoljno sklopovski jako stolno ili prijenosno računalo), priključivanje fizičkog uređaja jedini je način da se obavi simulacija razvijene aplikacije.

4.3.1. Instalacija aplikacije na fizički uređaj

Prvo, treba se USB kabelom priključiti fizički uređaj s računalom. Zatim, na mobilnom uređaju trebaju se odabrati *Postavke* i unutar njih odabrati *O telefonu* te pritiskati broj verzije Androida dok ne iskoči obavijest o načinu rada za razvojne programere. Nakon toga, korisnik se mora vratiti u *Postavke*, odabrati *Sustav i ažuriranja* i u tom zaslonu odabrati *Razvojne opcije*. U njima je potrebno omogućiti uklanjanje programske pogreške na USB-u. Tada bi se mobilni uređaj morao pojaviti na popisu povezanih uređaja. Kad je to napravljeno, u Android Studiju se odabere *Run > Run 'app'* ili se jednostavno pritisne kombinacija tipki *Shift + F10* nakon čega će započeti instalacija mobilne aplikacije na korisnikovom pametnom mobitelu.

4.3.2. Simulacija aplikacije na fizičkom uređaju

Nakon što se aplikacija instalira na korisnikov mobilni uređaj, mora ju se pokrenuti. Pri pokretanju će se pojaviti početni zaslon te će na njemu biti vidljive tipke *Play, How to play, Store, Settings* i *Information*.

Ovisno koja se od navedenih tipki stisne pokreće se odgovarajući zaslon:

- *Information* - zaslon informacija u kojemu pišu podaci o namijeni aplikacije te pravilima i načinu bodovanja.
- *How to play* - zaslon uputa za korištenje aplikacije. U njemu pišu koraci pravilnog pokretanja aplikacije u slučaju novih korisnika.
- *Store* - zaslon trgovine u kojemu piše poruka kako aplikacija još nije dostupna u *Trgovini Play*.
- *Settings* - zaslon postavki težine u kojemu korisnik može izabrati želi li da mu taboo riječi budu osnovne ili napredne težine. Ako korisnik izričito ne odabere nijednu težinu, početno je zadana osnovna težina. Korisnik označava koju težinu želi tako što odabere jedan od dvaju gumba (RadioButton) i zatim pritisne tipku *Select*. Nakon pritiska tipke *Select*, korisniku se ispisuje toast obavijest o njegovom odabiru.
- *Play* - zaslon postavki igre u kojoj korisnik može odrediti željene postavke poput broja rundi, količine vremena u svakoj rundi te imena ekipa. Ako korisnik u zaslonu postavki igre opet odabere tipku *Play*, započinje se aktivnost igre prikazana na slici 4.26.

Slika 4.26. Zaslon aktivnosti igre

5. ZAKLJUČAK

Mobilna aplikacija za društvenu igru *Taboo* uspješno je izrađena. Zadani su ciljevi ispunjeni i sve su funkcionalnosti ostvarene. Projekt je u potpunosti razvijen u razvojnom okruženju Android Studio jer Android ima najviše aktivnih mjesečnih korisnika, a cilj aplikacije je bio da se dopre do što većeg broja korisnika. Pisan je XML opisnim jezikom i programskim jezikom Java.

Dana je teoretska podloga razvojnog okruženja za izradu aplikacija, korištenog programskog jezik i korištenog jezika za označavanje podataka. Izgled aplikacije dodatno je prikazan slikama, a objašnjenje korištene logike opisno te dijelovima kôda i dijagramom toka. Izradom ove aplikacije stečena su znanja o gore spomenutim tehnologijama i dobra su podloga za daljnje usavršavanje u području izrade mobilnih aplikacija.

Aplikacija pomaže pri boljem usvajanju i uvježbavanju stranoga jezika te proširenju korisnikova vokabulara. Aplikacija je poučnog i zabavnog karaktera. Korisničko sučelje dovoljno je jednostavno da bi se i manje informatički pismeni korisnici u njemu lagano snašli. Igra je namijenjena svim uzrastima, neovisno o razini znanja stranog jezika jer je ona prije svega obrazovnog karaktera.

Prilikom izrade aplikacije došlo je do pojava problema, ponajprije zbog neiskustva rada u tom području. Problemi su bili otklonjeni dodatnim istraživanjem literature i proučavanjem sličnih rješenja. Aplikacija sadrži manje greške. Najočitija greška je nastavljanje igre iako je prijeđen broj korisnički određenih rundi. Taj je problem moguće riješiti pažljivom provjerom logike iza kôda aktivnosti igre ili čak ponovnim pisanjem kôda same aktivnosti.

Daljnje razvijanje aplikacije moguće je dodavanjem taboo kartica, razina težine, uređivanjem aktivnosti stanke i završetka tajmera, implementiranjem API tehnologije za povlačenje riječi, dodavanjem opcije spremanja najboljeg rezultata, opcije arhiviranja dvoboja dviju ekipa te promjene u grafičkom izgledu aplikacije.

LITERATURA

[1] Simon Personal Communicator, History-Computer, dostupno na: https://www.history-computer.com/ModernComputer/Personal/Simon_of_IBM.html [15.9.2020.]

[2] E. Protalinski, Android passes 2.5 billion monthly active devices, VentureBeat, svibanj 2019., dostupno na: https://www.venturebeat.com/2019/05/07/android-passes-2-5-billion-monthly-active-devices/ [15.9.2020.]

[3] J.R. Raphael, Android versions: A living history from 1.0 to 11, Computerworld, rujan 2020., dostupno na: https://www.computerworld.com/article/3235946/android-versions-a-living-history-from-1-0-to-today.html [15.9.2020.]

[4] F. Lardinois, Kotlin is now Google's preferred language for Android app development, TechCrunch, svibanj 2019., dostupno na: https://www.techcrunch.com/2019/05/07/kotlin-is-now-googles-preferred-language-for-android-app-development/ [15.9.2020.]

[5] H. Kabutz, Doing Things with Java that Should Not Be Possible: Once Upon an Oak ..., srpanj 2003., dostupno na https://www.artima.com/weblogs/viewpost.jsp?thread=7555 [16.9.2020.]

[6] J. Šribar, B. Motik, Demistificirani C++, Element, Zagreb, 2001

[7] Wikipedia, Markup language, dostupno na: https://en.wikipedia.org/wiki/Markup_language [16.9.2020.]

[8] CardView, Android Developers, dostupno na: https://developer.android.com/guide/topics/ui/layout/cardview [16.9.2020.]

[9] FrameLayout, Android Developers, dostupno na: https://developer.android.com/reference/android/widget/FrameLayout [23.9.2020.]

[10] Toast, Android Developers, dostupno na: https://developer.android.com/reference/android/widget/Toast [16.9.2020.]

SAŽETAK

U ovom završnom radu razvijena je Android mobilna aplikacija za *Taboo* igru. Aplikacija ima mogućnost odabira između dviju težina, broja rundi i njihova vremenskog trajanja. Bodovanje se radi vlastoručno. U aplikaciji se nalaze upute i pravila igre. Struktura aplikacije sastoji se od deset zaslona od kojih svaki ima svoju funkcionalnost. Struktura i tijek aktivnosti aplikacije dodatno su pojašnjeni dijagramom. Aplikacija je u potpunosti razvijena u okruženju Android Studio, a pisana je u programskom jeziku Javi.

Teorijski dio rada služi za opisivanje korištenih tehnologija. Veći su dio rada prikazi kôda i pripadajuća objašnjenja koja služe za dodatno pojašnjenje logike iza funkcionalnosti aplikacije. Izgled aplikacije prikazan je slikama. Tekstualni dio rada dodatno je popraćen slikama radi lakšeg snalaženja. Pojašnjena je instalacija aplikacije i njena upotreba na fizičkom uređaju.

Ključne riječi: Android, mobilna aplikacija, obrazovno-zabavni sadržaj, strani jezik, Taboo, učenje

ABSTRACT

Taboo game application

In this Bachelor's thesis, a task of developing an Android application for a game called *Taboo* was carried out. In the application, the user can choose between two difficulty levels, a number of rounds and the duration time of each round. Scoring is done manually by pressing a button in the application. In the application itself, there are both instructions on how to play the game and the game rules. The structure of application consists of ten activities. Each activity has its functionality. The application structure and flowchart are further clarified using charts. The application was fully developed in the development environment called Android Studio and was written in Java programming language.

The theoretical part of this thesis is used for describing the used technologies. A large segment of the thesis includes code examples used for better explaining of the logic behind the application functionalities. The textual parts are accompanied by pictures for better understanding. The layout is shown on a physical device. Installation and usage of the application on a physical device are also explained.

Key words: Android, edutainment, foreign language, learning, mobile application, Taboo

ŽIVOTOPIS

Matija Kaučić rođen je 4. siječnja 1998. godine u Zagrebu. Svoje osnovnoškolsko obrazovanje započinje 2004. godine u Osnovnoj školi Davorin Trstenjak u Čađavici. 2006. godine, zajedno s obitelji, seli u Slatinu i tamo nastavlja svoje osnovnoškolsko obrazovanje do 2012. godine u Osnovnoj školi Josip Kozarac. 2012. upisuje Opću gimnaziju u Srednjoj školi Marka Marulića u Slatini koju završava 2016. godine. Iste godine, upisuje se na Fakultet elektrotehnike, računarstva i informacijskih tehnologija u Osijeku na sveučilišni preddiplomski studij računarstva.