Академија струковних студија Шумадије

Пројектни задатак из предмета

Развој мобилних апликација

Игра скочко

-Тема-

Студент:
Дарко Словић, 017/2019

Предметни наставник:
проф. др Хрвоје Пушкарић.

Крагујевац, школске 2021/22.

САДРЖАЈ

1. Увод

Мобилне апликације су постале чешћи појам код информатичара, разлог томе доприноси то што су мобилни телефони скорије постали неопходна алатка човеку за рад, комуникацију и забаву. Разлог зашто су мобилни телефони постали толико заступљенији да су постали неопходни је функција апликација које се налазе на мобилним телефонима. Апликације чине мобилни телефон корисним на специфичан начин. Постоји мноштво различитих апликација који могу допринаети човеку на различите начине. И тако, да би човек био ефикаснији у послу и да би имао удобнији живот потребан му је мобилни телефон, да би мобилни телефон имао могућност да удовољи потребама човека потребне су му апликације, а мобилне апликације треба да направи информатичар.

Како би информатичар могао ефикасније да развија квалитетне мобилне апликације по брзини која прати захтеве потрошача и трендове, информатичару су пружени алати који му помажу да испуни та очекивања, један од тих алата је Андроид студио. Андроид студио је развојно окружење специјано направњен да производи апликације гуглевог оперативног система за мобилне телефоне Андроид. Изграђен је на ЈетБраинс-овом ИнтеллиЈ ИДЕА софтверу тако да подржава Јава програмски језик, а на Андроид студиу се могу развијати апликације са Котлин програмским језиком.

Апликација Скочко која је направљена на развојном окружењу Андроид студио омогућава кориснику да се забави игром која је заснована на правилу игре Скочко, на којој је име апликације такође инспирисано. Циљ игре скочко јесте да се пронађе комбинација одређених елемената уз малу помоћ која даје инструкције када је играч погодио одређени елемент који се налази у комбинациј и у зависности да ли је тај погођен елемент на месту или не, даје одређну индикацију. Играчу се не даје до знања који елемент је погодио и који елемент није на месту већ само број елемената који су на месту и број елемената који нису.

2. Пројектни задатак и његова реализација

Апликација Скочко има циљ да кориснику апликације пружи забаву уз помоћ игре Скочко на мобилном телефону. Пошто је ово мобилна апликација корисник ће моћи да се забави било где да оде у било којем окружењу, на досадној прослави, на реду за зубара, на путу до школе или посла, докле год има мобилни телефон са батеријом и инсталинраном апликацијом моћи ће да пружи изазов свом мозгу, притом да избегне досаду. Апликацију могу користити особе свих узраста и навењена је за мобилне телефоне са андроид оперативним системом, зато што је апликација развејена уз помоћ Андроид студио развојног окружења.

Андроид студио развојно окружење, поред развијања мобилних апликација за андроид оперативни систем такође може развити апликације за смарт сатове, смарт телевизоре, аутомобилске интерфејсе и таблет уређаје које поседују андроид оперативни систем. Што се тиче андроид оперативног система, Андроид студио подржава старије и нај новије андроид оперативне системе. Тестирање апликација на андроид студиу се врши уз помоћ виртуалних машина, то јест виртуелних мобилних телефона са андроид оперативним системом. Разој интерфејса у андроид студиу је базиран на XML програмском језику, али није потребно да се добро познаје језик да би се развио интерфрјс јер андроид студио поједностављаа развој интерфејса уз помоћ своје радне површине која аутомацки пише XML код на основу атрибута који се мењају и опција које се изаберу на радној површини. Поред XML за развој интерфејса, функционалност апликације се обавља уз помоћ Јаве или Котлин програмског језика.

Апликација Скочко је развијена уз помоћ Јава и XML програмског језика. За интерфејс је коришћен XML, а све остале функционалности и логика саме игре је уређена на Јави програмском језику. За тестирање апликације је коришћан виртуелни мобилни телефон Nexus 5X API 26 са оперативним системом Opeo 30 преко андроид студио виртуелног окружења.

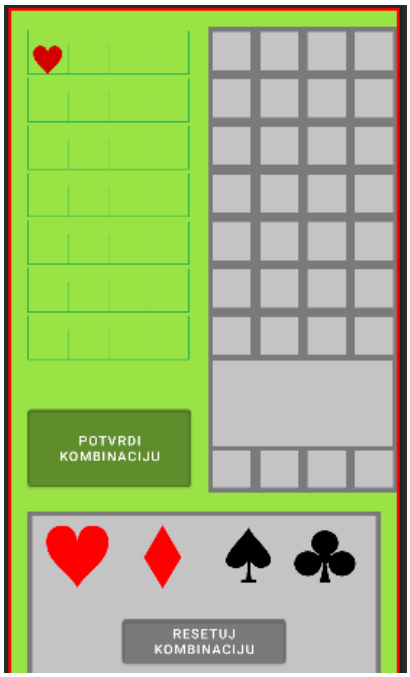
3. Опис апликације

3.1 Кориснички интерфејс

Интерфејс који се приказује кориснику се састоји из два дела. Први део је једноставни приказ који саджи име игре као наслов, лого игре у средини и испод логоа се налази дугме уз помоћ којег корисник може да покрене игру када је спреман(слика 1). Позадина прог и другог интерфејса је обојен светло зеленом бојом док други део има црвени оквир.

Други део интерфејса је где ће корисник играти игру (Слика 2). Овај интерфејс

Слика 1: Први део интерфејса



Слика 2: Други део интерфејса

је подељен на три дела. Први део се налази на левој страни интерфејса, у овом делу се налазе седам редова по четири колоне, на свакој од тих колона корисник може видети елементе који је селектовао у покушају. Испод седам редова се налази дугме уз помоћ којег корисник може да преда своју комбинацију ако је попунио све четири колоне у реду. Када је комбинација унета комбинација остаје у том реду, а нова комбинација се уписује у реду испод, тако да корисник може видети које елементе је искористио у прошлој комбинацији.

На децној страни се налази табела где је сваки ред и свака колона одвојена тамно сивим линијама, позадина табеле је обојена светло сивом бојом. Табела се састоји од седам редова и четири колона, редови су паралелно поређани упоред са редовима са леве стране. Ови редови служе да приказују резултат комбинације коју је корисник унео. Резултати се приказују зеленом штиклом ако је неки елемент у комбинацији на погођеном и исправном месту. Друга варијанта резултата се приказује жутим упитником који значи да је корисник унео исправан елемент али тај елемент није на исправном месту. У реду се најпре приказују погођени елементи, а за њима се приказују упитници. Резултат треба кориснику само да приказује колико

елемената је погогио и у којем су стању, а не у којој су позицији ти елементи. Испод седам редова се налази ред без колона, овај ред служи за комуникацију са корисником преко текстуалних порука. Поруке се приказују када корисник жели да унесе више од четри елемената у једној комбинацији, када притисне дугме за предају комбинације, а није унео четири елемената, када је на последњем покушају, када је остао без покушаја и када је погодио тачну комбинацију. Текст је црвене боје за сва обавештења осим за обавештење када корисник унесе правилну комбинацију. Испод реда за обавештења се налази још један ред са четири колоне. Колоне су одвојене линијама истим стилом као и прва седам реда. У овом реду се приказује правакомбинација када корисник остане без покушаја и када корисник погоди тачну комбинацију. Задња два реда су паралелно поређана упоред са дугметом за предају комбинације на левј страни.

Трећи део се налази на дну интерфејса испод дугмета за предају комбинације и табеле са резултатима.Трећи део се састоји из палете светло сиве позадине и тамно сиве границе. У палети се налазе пет дугмета, прва четири дугмета се налазе на горњем делу палете и имају облик елмената у комбинацији. Елементи су херц, каро, пик и треф притиском на једно од тих дугмета апликација убацује елемент у ред табеле са леве стране, елемент убачен у ред је елемент чији облик дугме носи облик. Испод тих дугмића се налази једно веће дугме која кориснику омогућава да избрише тренутну комбинацију коју је унео у текућем реду, дако да може да исправи грешку пре него што је испроба и унесе нову комбинацију.

Када корисник изгуби или победи игру изаћи ће мали приказ који даје кориснику опцију да ресетује игру и покуша да погоди нову комбинацију, ако притисне дугме ДА, а ако притисне НЕ апликација ће се затворити.

3.2 Изворни код апликације

Код апликације се извршава притиском дугмића (Слика 3). Свако дугме има своју функционалност, дугме за убацивање елемената има наредбу да открије „Image View“ (објекат који приказује слике и слично) на табели у десној стррани и да

```
Button button_karo = (Button) findViewById(R.id.button_karo);
button_karo.setOnClickListener(new View.OnClickListener() {
    public void onClick (View v) {
        if(red < 4) {
            pozicija++;
            nizPostavljenihKaraktera[red] = 1;
            red++;

            if (pozicija < 10) {
                viewPogadjanje = "imageView10";
            } else {
                viewPogadjanje = "imageView1";
            }
            String novViewPog = viewPogadjanje + Integer.toString(pozicija);

            int novId = getResources().getIdentifier(novViewPog,  defType: "id", getPackageName());
            ImageView element = (ImageView) findViewById(novId);
            DrawableCompat.setTint(getResources().getDrawable(R.drawable.karo), Color. RED);
            element.setImageDrawable(getResources().getDrawable(R.drawable.karo));
            element.setVisibility(View.VISIBLE);

        }else{
            upozorenje();
        }

}
});
```

Слика 3: Код дугмета каро

промени „Drawable“ (слика која се ставља на елементе) на елемент који одговара дугмету „херц, каро, ...“. При сваком притиску дугмета за убацивање елемента повећавају се два посебна променљива интиџера „row“ и „pozicija“. „row“ мери колоне у реду тако да може максимално да иде до четири зато што табела има четири колоне, променљива „row“ онемогућује корисника да убаци више од четири елемента пре притиска дугмета за предају комбинације изјавом “if“. „row“ ће се вратити на нулу када корисник притисне дугме за предају комбинације или дугме за ресетовање комбинације.

Променљива „pozicija“ се повећава без обзира на ред. Само у случају притиска дугмета за ресетовање комбинације променљива „pozicija“ ће се уманити са вредности променљиве „row“. Променљива „pozicija“ омогућује проналазак тачног односно следећег „Image View“ који се треба открити и променити. „ID“ од „Image View“ садржи основни део који гласи „imageView1“, реч на почетку означава да се ради о елементу „Image View“, а јединица означава да се ради о табели са леве стране. Уз помоћ променљиве „String“ типа који носи врдност основног дела и променљиве „pozicija“ може се добити исправан „ID“. Основа се бира у зависности да ли је „pozicija“ једноцифрена или двоцифрена, ако је једноцифрена променљива ће да глаци „imageView10“ тако када се прво дугме притисне „pozicija“ ће имати вредност један и ако претворимо вредност интиџер променљиве „pozicija“ у стринг вредност, па притом додамо ту вредност на већ дефинисану променњиву која носи основу „ID“, добиће се исправан „ID“ следећг „Image View“(imageView101, imageView102, imageView103...).

```
Button button_reset = (Button) findViewById(R.id.button_resetuj);
button_reset.setOnClickListener(new View.OnClickListener() {
    public void onClick (View v) {
        if (red != 0) {
            for (int i = 1; i < 5; i++){
                if (pozicija < 13) {
                    viewPogadjanje = "imageView10";
                } else {
                    viewPogadjanje = "imageView1";
                }
                System.out.println(viewPogadjanje);
                if(pozicija >= 9){
                    if(pozicija <=12){
                        if(i>= 2){
                            viewPogadjanje = "imageView1";
                        }
                    }
                }
                String novViewPog = viewPogadjanje + Integer.toString( (pozicija - red) + i);
                int novId = getResources().getIdentifier(novViewPog,  defType: "id", getPackageName());
                ImageView element = (ImageView) findViewById(novId);
                if(element.getVisibility() == View.VISIBLE){
                    element.setVisibility(View.INVISIBLE);
                }
            }
            pozicija = pozicija - red;
            red = 0;
        }
    }
});
```

Слика 4: Дугме за ресетовање

Дугме за ресетовање(Слика 4) комбинације смањује променљиву „pozicija“ са променљивом „row“, а вредност „row“ враћа на нулу. Пре тога уз помоћ ових вредности слично дугмету за убацивање новог елемента, дугме за ресетовање комбинације може да нађе „ID“ од испраног „Image View“ којег треба да претвори

невидљивим тиме симулирати процес брисања реда. Уз помоћ разлике променљиве „pozicija“ и „row“ можемо добити број мање од почетне колоне додавањем јединице и и на основу „ID“ налепимо тај број добићемо „ID“ од „Image View“ прве колоне. Уместо јединице ставимо променљиву петље која не см ић више од четри и без обзира на то колико елемената се налази у реду сви ће бити сакривени.

За саму логику игре су коришћени бројеви, херц означава нула, каро јединица, пик двојка и каро тројка. Направљена је листа која држи четири броја. При покретању игре ова листа је попуњена случајним бројевима од нуле до три који представљају елементе(Слика 5). Направљена је још једна листа која има четири науле као вредност. Уз помоћ те листе се мери колико јединица, нула и слично комбинација има. На пример ако комбинација има две нуле једну двојку и тројку, нула на првој(односно нултој, зато што се позиције листе почињу од нуле) позицији ће се повећати за два броја, а нуле на трећој и четвртој позицији ће се повећати за један број.

Нај комплекснији део изворног кода се извршава притиском дугмета за предају комбинације(Слика 6). Код се извршава само ако су нуешена четири елемента односна ако променњива „row“ има вредност четири, у супротн се обавља функција која ће приказати и изменити вредност сакривеног „Text View“ у реду за приказивање порука. Дефинисана је нова листа исто као и листа која служи за бројење вредности насумичне комбинације. Ова листа служи да преузме вредности листе која броји вредности насумичне комбинације, јер уз помоћ тих вредности се може утврдити да ли је елемент проверен у циклусу смањивањем вредности на позицији. У суштини овај део кода има улогу да изброји колико унетих елемената погођени су на месту и колико елемената су погођени, а нису на месту. Па ће уз помоћ тог броја одредити којим „Image View“ колонама да додели упитник или штиклу. Одређивање „ID“ вредности се обавља на сличан начин као и дугмићи за унос елемената и

```
private void generisiKaraktere() {
    Random rand = new Random();
    for (int i = 0; i < nizGenerisanihKaraktera.length; i++) {
        int slucajanBroj = rand.nextInt( bound: 4);
        nizGenerisanihKaraktera[i] = nizMogucihKaraktera[slucajanBroj];
        if(slucajanBroj == 0){brojKaraktera[0]++;}
        else if (slucajanBroj == 1){brojKaraktera[1]++;}
        else if (slucajanBroj == 2){brojKaraktera[2]++;}
        else {brojKaraktera[3]++;}
    }
```

Слика 5: Метода која генерише и броји карактере

ресетовање комбинације. Разлика је у томе што основа „ID“ вредности за „Image View“ са десне табеле је „imageView2“, двојка означава да се ради о пољима са десне табеле. Још једна разлика су променљиве које попуњавају „ID“ вредност. Прва цифра која се налепљује на променљиву са основом је „brojPokusaja“ који мери колико пута је притиснуто дугме за проверу комбинације уз испуњен први услов. Ова цифра означава ред у табели. Друга променљива је „uneverzalniBrojač“. Ова променљива је укупан број погођених елемената, што на месту, што ван места. На

основу броја који је добијен предходним кодом где се упоређују бројеви генерисане и унешене комбинације, се одређује колико ће се ставити штикал, а колико упитника. На крају када се изгуби или победи игра на методама који приказују порке је уграђена још једна метода под притиском дугмета опције да, која враћа све интиџер промењљиве на нулу и генерише нову листу случајних комбинација. Такође сакрива све елементе и обавештења. Опција не, затвара апликацију.

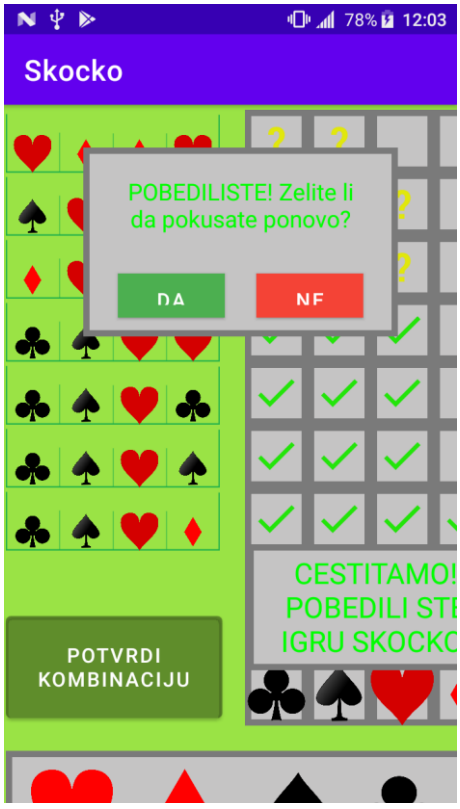
```
Button button_potvrdi = (Button) findViewById(R.id.button_potvrdi);
button_potvrdi.setOnClickListener(new View.OnClickListener() {

    public void onClick (View v) {
        if (red == 4) {
            int brojPogodjenihNaMestu = 0;
            int brojPogodjenih = 0;
            int univerzalniBrojac = 0;
            int[] brojKaraktera2 = {0, 0, 0, 0};
            brojPokusaja++;
            for(int j =0; j<4; j++){
                brojKaraktera2[j] = brojKaraktera[j];
            }
            for (int i = 0; i < 4; i++) {
                if (nizGenerisanihKaraktera[i] == nizPostavljenihKaraktera[i]) {
                    brojPogodjenihNaMestu++;
                    brojKaraktera2[nizPostavljenihKaraktera[i]]--;
                }
            }
            for (int i = 0; i < 4; i++) {
                if(nizGenerisanihKaraktera[i] != nizPostavljenihKaraktera[i]) {
                    for(int x = 0; x<4; x++){
                        if(nizGenerisanihKaraktera[x] == nizPostavljenihKaraktera[i]){
                            if(brojKaraktera2[nizPostavljenihKaraktera[i]] > 0){
                                brojPogodjenih++;
                                brojKaraktera2[nizPostavljenihKaraktera[i]]--;
                            }
                        }
                    }
                }
            }

            if(brojPogodjenihNaMestu != 0){
                for(int j = 0; j < brojPogodjenihNaMestu; j++){
                    String novViewRez = viewRezultat + Integer.toString(brojPokusaja) + Integer.toString( i: univerzalniBrojac + 1);
                    int novId = getResources().getIdentifier(novViewRez,  defType: "id", getPackageName());
                    ImageView element = (ImageView) findViewById(novId);
                    element.setImageDrawable(getResources().getDrawable(R.drawable.pogodjen_u_mestu_foreground));
                    univerzalniBrojac++;
                }
            }
            if(brojPogodjenih != 0){
                for(int j = 0; j < brojPogodjenih; j++){
                    String novViewRez = viewRezultat + Integer.toString(brojPokusaja) + Integer.toString( i: univerzalniBrojac + 1);
                    int novId = getResources().getIdentifier(novViewRez,  defType: "id", getPackageName());
                    ImageView element = (ImageView) findViewById(novId);
                    element.setImageDrawable(getResources().getDrawable(R.drawable.pogodjen_u_foreground));
                    univerzalniBrojac++;
                }
            }
            red = 0;
            if (brojPokusaja == 6){
                upozorenje3();
            }else{
                TextView upoz = (TextView) findViewById(R.id.textView_info);
                if (upoz.getVisibility() == View.VISIBLE){
                    upoz.setVisibility(View.INVISIBLE);
                }
            }

            if (brojPogodjenihNaMestu == 4){
                cestitka();
            }
            if(brojPokusaja == 7){
                if(brojPogodjenihNaMestu != 4){
                    obavstenje();
                }
            }
        }else{
            upozorenje2();
        }
    }
});
```

Слика 6: Код дугмета за проверу комбинације

4. Закључак

Андроид студио отвара разне могућности и пружа корисне алате који добро долазе при развијању мобилних апликација. Уз помоћ овог развојног окружења сам могао илустровати своју идеју за апликацију коју сам желео да направим. Што се тиче овог пројекта нисам имао довољно времена да имплементирам све што сам имао на уму.

Игри недостаје тежина, изазв и награда за успех. Уз више времена и боље опреме би дефинитивно донео ове ствари апликацији. Што се тиче тежини и изазова унео би временско ограничење које би корисник могао да изабере на почетном екрану. Такође би направио различите тежине. Лака тежина би била оваква каква је апликација сада, нормална тежина би садржала више елемената, а тешка би имала више елеманата и веће комбинације. За награду би унео систем поена и на основу тога коју тежине и времена коју је корисник изабрао, корисник би добио већи или мањи број поена. Број поена би се чувао у бази података заједно са именом корисника који се уноси када се победи игра. Такође би измислио лого за апликацију да буде као иконица и да се приказује на почетној страни.



Слика 7: Побеђена игра