Academy of Vocational Studies Šumadija

Академија струковних студија Шумадија

Одсек Крагујевац

Project assignment from the course

Development of mobile applications

Wordle game

-Topic-

Kragujevac, academic year 2021/22.

THE CONTENT

1. introduction

Mobile applications have become a more common term among computer scientists, the reason contributing to this is that mobile phones have recently become a necessary tool for man for work, communication and entertainment. The reason why cell phones have become so prevalent that they have become indispensable is the function of applications found on cell phones. Applications make a mobile phone useful in a specific way. There are many different applications that can contribute to a person in different ways. And so, in order for a person to be more efficient at work and to have a more comfortable life, he needs a mobile phone, in order for a mobile phone to be able to meet the needs of a person, he needs applications, and mobile applications should be made by an IT specialist.

In order for the IT professional to more efficiently develop quality mobile applications at a speed that follows consumer demands and trends, the IT professional is provided with tools that help him meet those expectations, one of those tools is Android studio. Android studio is a development environment specially designed to produce applications of Google's operating system for Android mobile phones. It was built on JetBrains' IntelliJ IDEA software so that it supports the Java programming language, and Android Studio can develop applications with the Kotlin programming language.

The Wordle app, which is built on the Android studio development environment, allows the user to have fun with a game that is based on the rules of the Wordle game, from which the name of the app is also inspired. The aim of the Wordle game is to find a combination of certain elements with a little help that gives instructions when the player has hit a certain element that is in the combination and depending on whether that hit element is in place or not, it gives a certain indication. The player is not told which element hit and which element is out of place, but only the number of elements that are in place and the number of elements that are not.

2. Project assignment and its implementation

The Wordle application aims to provide entertainment to the application user with the help of the Wordle game on the mobile phone. Since this is a mobile application the user will be able to have fun anywhere he goes in any environment, at a boring party, in line at the dentist, on the way to school or work, as long as he has a mobile phone with a battery and an installed application he will be able to provide challenge your brain while avoiding boredom. The application can be used by people of all ages and is intended for mobile phones with the Android operating system, because the application was developed with the help of the Android studio development environment.

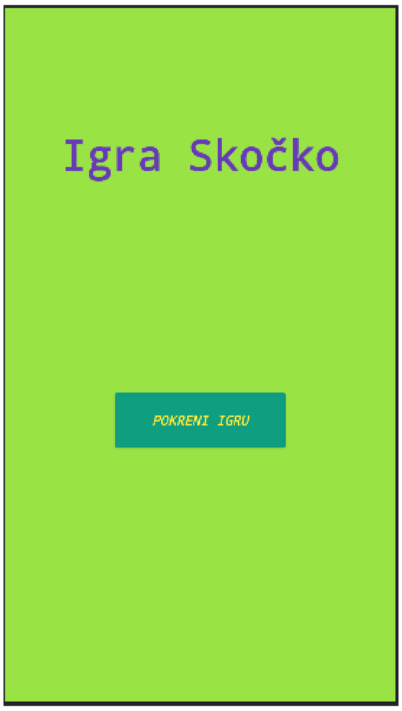
The Android studio development environment, in addition to developing mobile applications for the Android operating system, can also develop applications for smart watches, smart TVs, car interfaces and tablet devices that have the Android operating system. Regarding android operating system, android studio supports older and latest android operating systems. Testing applications on Android Studio is done with the help of virtual machines, that is, virtual mobile phones with the Android operating system. The interface development in android studio is based on the XML programming language, but it is not necessary to have a good knowledge of the language to develop the interface because android studio simplifies the development of the interface with the help of its desktop that automatically writes XML code based on the attributes that change and the options that are selected on the desktop. In addition to XML for interface development, the functionality of the application is performed with the help of Java or Kotlin programming language.

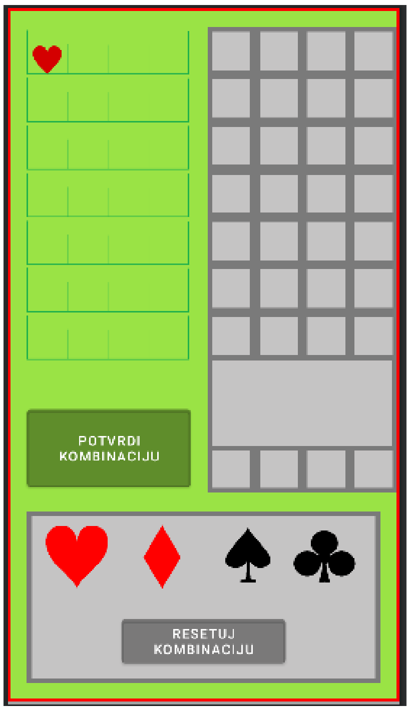
The Wordle application was developed with the help of Java and XML programming language. XML was used for the interface, and all other functionalities and the logic of the game itself were arranged in the Java programming language. For testing the application, a virtual mobile phone Nexus 5X API 26 with the operating system Oreo 30 was used through the android studio virtual environment.

3. Application description

3.1 User Interface

The interface shown to the user consists of two parts. The first part is a simple display that contains the name of the game as a title, the logo of the game in the middle and below the logo there is a button with the help of which the user can start the game when he is ready. The background of the prog and the other interface is colored light green while the other part has a red frame.





The second part of the interface is where the user will play the game. This interface is divided into three parts. The first part is located on the left side of the interface, in this part there are seven rows with four columns, on each of these columns the user can see the elements he selected in the attempt. Below the seven rows there is a button with the help of which the user can submit his combination if he has filled all four columns in a row. When a combination is entered, the combination remains in that line, and the new combination is entered in the line below, so the user can see which elements he used in the previous combination.

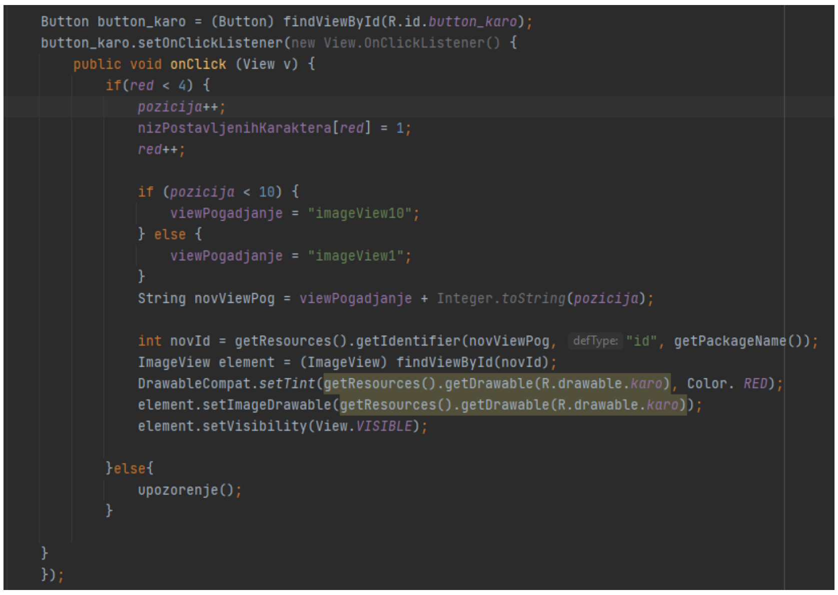
On the right side there is a table where each row and each column is separated by dark gray lines, the background of the table is colored light gray. The table consists of seven rows and four columns, the rows are arranged parallel to the rows on the left. These lines are used to display the result of the combination entered by the user. The results are displayed with a green tick if an element in the combination is in the affected and correct place. Another variant of the result is displayed with a yellow question mark, which means that the user has entered the correct element, but that element is not in the correct place. In the queue, the selected elements are displayed first, followed by question marks. The result should only show the user how many elements he hit and in what state they are, not in what position those elements are. Below the seven rows is a row without columns, this row is used for communication with the user via text messages. Messages are displayed when the user wants to enter more than four elements in one combination, when he presses the button to submit the combination and did not enter four elements, when he is on his last attempt, when he ran out of attempts and when he guessed the correct combination. The text is red for all notifications except for the notification when the user enters the correct combination. Below the notification row is another row with four columns. Columns are separated by lines in the same style as the first seven rows. This row displays the correct combination when the user runs out of tries and when the user guesses the correct combination. The last two rows are parallel to the button for submitting the combination on the left side.

The third part is located at the bottom of the interface below the submit button and the results table. The third part consists of a light gray background palette and a dark gray border. There are five buttons in the palette, the first four buttons are located on the upper part of the palette and have the shape of elements in combination. The elements are hertz, diamond, spade and club. By pressing one of those buttons, the application inserts an element into the row of the table on the left, the element inserted into the row is the element whose shape the button carries. Below those buttons is a larger button that allows the user to delete the current combination they have entered in the current row, although they can correct a mistake before trying it and entering a new combination.

When the user loses or wins the game a small display will pop up giving the user the option to reset the game and try to guess a new combination, if they press the YES button and if they press NO the app will close.

3.2 Application Source Code

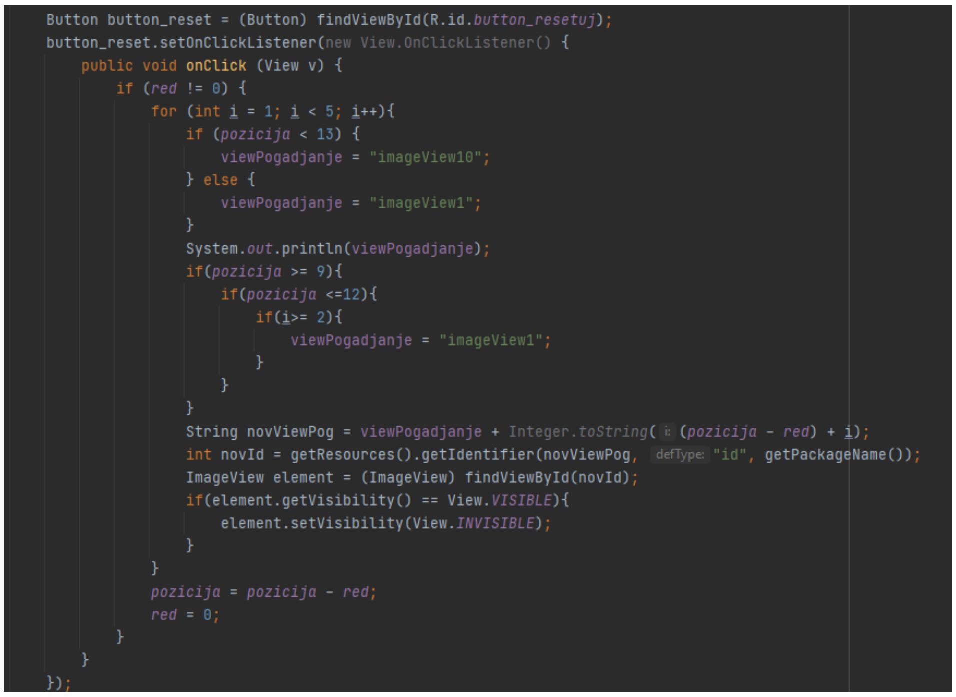
The application code is executed by pressing a button. Each button has its own functionality, the element insert button has a command to detect the "Image View" (an object that displays images and the like) on the table on the right side and change the



```
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();
        }

}
});
```

"Drawable" (an image to be placed on the elements) to the element corresponding to the button "hertz, karo, ...". Each time the insert element button is pressed, two separate integer variables "row" and "position" are incremented. "row" measures the columns in a row so it can go up to four because the table has four columns, the "row" variable prevents the user from inserting more than four elements before pressing the button to submit the combination with an "if" statement. "row" will reset to zero when the user presses the submit combo button or reset combo button.

The "position" variable is incremented regardless of order. Only in the case of pressing the combination reset button will the variable "position" be decremented from the value of the variable "row". The variable "position" makes it possible to find the correct or next "Image View" that needs to be detected and changed. The "ID" of "Image View" contains a body that reads "imageView1", the word at the beginning indicates that it is an "Image View" element, and the unit indicates that it is a table on the left. With the help of a "String" variable of type that carries the value of the base part and a "position" variable, the correct "ID" can be obtained. The basis is chosen depending on whether the "position" is a single digit or a double digit, if it is a single digit the variable will be "imageView10" so when the first button is pressed the "position" will have the value one and if we convert the integer value of the "position" variable to a string value , while adding that value to the already defined variable that carries the base "ID", the correct "ID" of the next "Image View" (imageView101, imageView102, imageView103...) will be obtained.

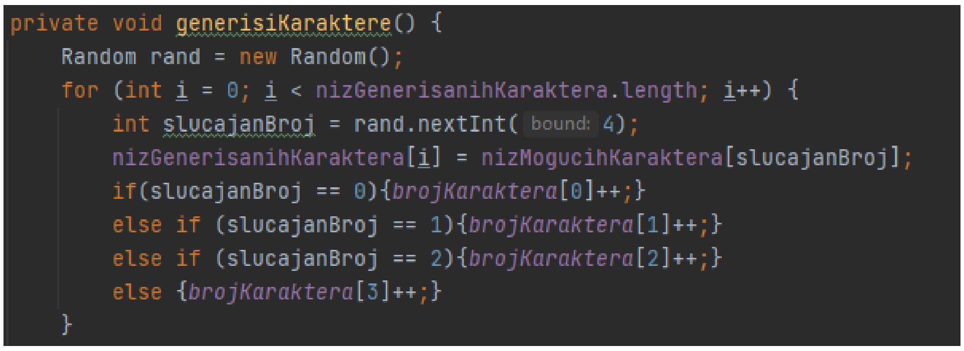


```java
Button button_reset = (Button) findViewById(R.id.button_resetuj);
button_reset.setOnClickListener(new View.OnClickListener() {
    public void onClick (View v) {
        if (red != 0) {
            for (int i = 1; i < 5; i++){
                if (pozicija < 13) {
                    viewPogadjanje = "imageView10";
                } else {
                    viewPogadjanje = "imageView1";
                }
                System.out.println(viewPogadjanje);
                if(pozicija >= 9){
                    if(pozicija <=12){
                        if(i>= 2){
                            viewPogadjanje = "imageView1";
                        }
                    }
                }
                String novViewPog = viewPogadjanje + Integer.toString( i: (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;
        }
    }
});
```

The reset button of the combination decrements the "position" variable with the "row" variable, and returns the "row" value to zero. Before that with the help of these values, similar to the button for inserting a new element, the reset button of the combination can find the "ID" from the washed "Image View" that it needs to make invisible, thereby simulating the row deletion process. With the help of the difference of the variable "position" and "row" we can get a number less than the initial column by adding a unit and and based on the "ID" paste that number we will get the "ID" of the "Image View" of the first column. Instead of unit, we put a loop variable that cannot contain more than four, and regardless of how many elements are in the row, they will all be hidden.

Numbers are used for the logic of the game, hertz means zero, checkered one, spade two and checkered three. A list is created that holds four numbers. When starting the game, this list is filled with random numbers from zero to three that represent the elements. Another list is created which has four naulas as value. With the help of that list, it is measured how many units, zeros and similar combinations there are. For example, if the combination has two zeros, one two and three, the zero in the first (or zero, because the list positions start from zero) position will be increased by two numbers, and the zeros in the third and fourth positions will be increased by one number.

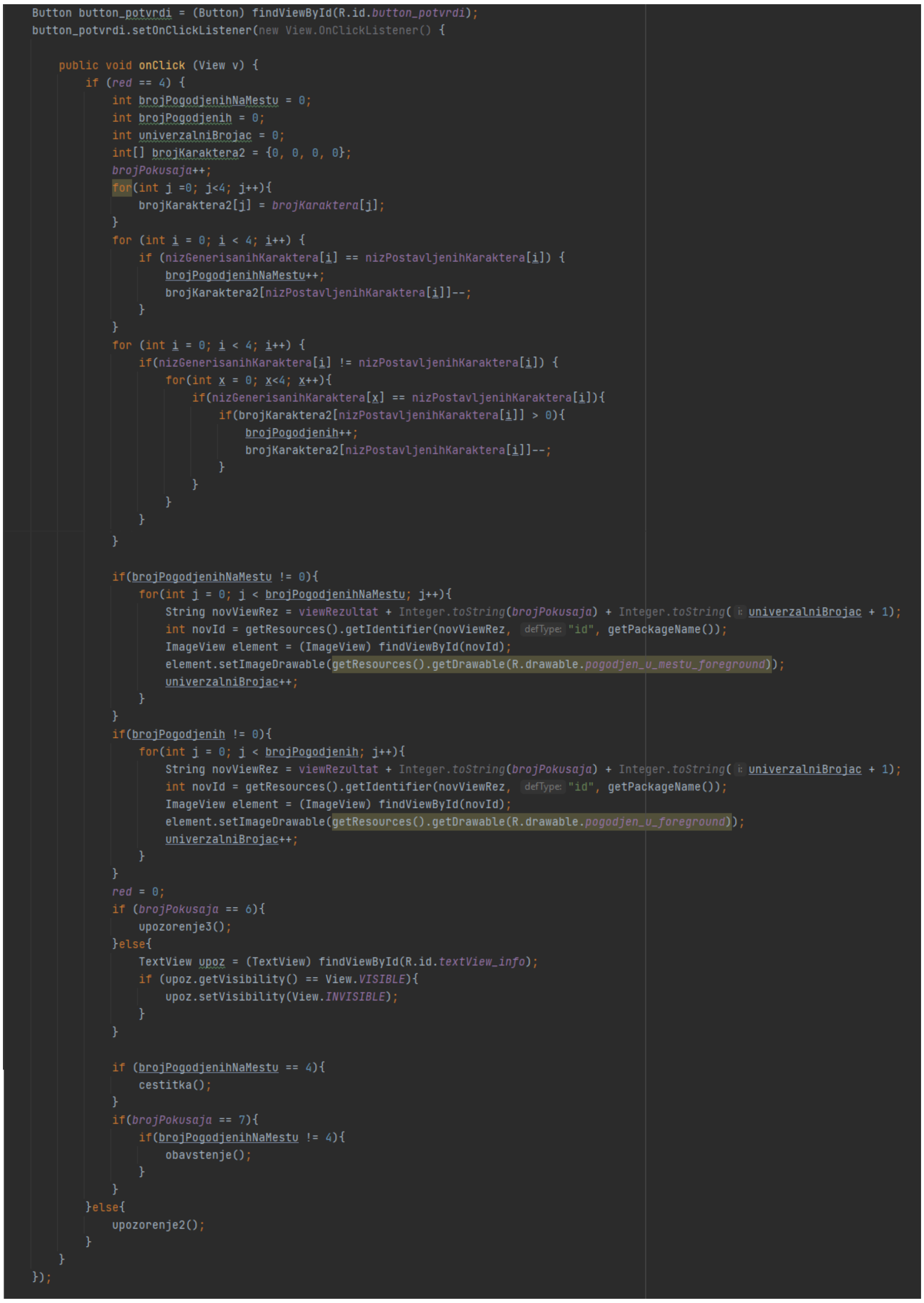
The most complex part of the source code is executed by pressing the button to submit the combination. The code is executed only if four elements are entered, i.e. if the variable "row" has a value of four, otherwise a function is executed that will display and change the value of the hidden "Text View" in the row for displaying messages. A new list is defined, just like the list used to count the values of the random combination. This list is used to retrieve the values of the list that counts the values of the random combination, because with the help of these values it can be determined whether the element is checked in the cycle by decreasing the value at the position. Basically this piece of code has the role of counting how many input elements are hit in place and how many elements are hit and not in place. So, with the help of that number, it will determine which "Image View" columns to assign a question mark or a check mark to. Determining the "ID" value is



```
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]++;}
    }
```

done in a similar way to the buttons for entering elements and resetting the combination. The difference is that the base "ID" value for the "Image View" from the right table is "imageView2", the two indicates that it is the fields from the right table. Another difference is the variables that populate the "ID" value. The first number that is pasted to the variable with the base is "countTries" which measures the number of times the combination check button has been pressed with the first condition met. This figure indicates the row in the table. The second variable is "universalCounter". This variable is the total number of hit elements, some in place, some out of place. Based on the number obtained by the previous code, where the numbers of the generated and entered combinations are compared, it is determined how many checkmarks will be placed, and how many question marks will be placed.

At the end, when the game is lost or won, another method is built into the methods that display the dice, by pressing the yes option button, which resets all integer variables to zero and generates a new list of random combinations. It also hides all elements and notifications. The no option closes the application.

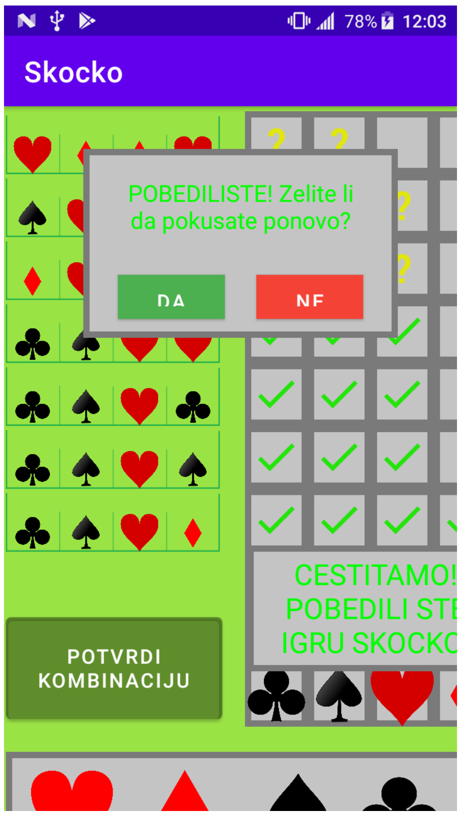


```java
Button button_potvrdi = (Button) findViewById(R.id.button_potvrdi);
button_potvrdi.setOnClickListener(new View.OnClickListener() {

    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( univerzalniBrojac + 1);
                    int novId = getResources().getIdentifier(novViewRez,  "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( univerzalniBrojac + 1);
                    int novId = getResources().getIdentifier(novViewRez,  "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();
        }
    }
});
```

4. Conclusion

Android studio opens up various possibilities and provides useful tools that come in handy when developing mobile applications. With the help of this development environment I was able to illustrate my idea for the application I wanted to create. Regarding this project, I didn't have enough time to implement everything I had in mind.

The game lacks difficulty, challenge and reward for success. With more time and better equipment it would definitely bring these things to the app. As for the difficulty and challenge, he would enter a time limit that the user could select on the home screen. He would also make different weights. Easy difficulty would be the way the app is now, normal difficulty would have more elements, and difficult would have more elements and bigger combinations. For the reward, he would introduce a point system and based on what weight and time the user chose, the user would get a higher or lower number of points. The number of points would be stored in the database along with the username entered when the game is won. He would also invent a logo for the app to be like an icon and displayed on the home page.


POBEDILISTE! Zelite li da pokusate ponovo?

CESTITAMO! POBEDILI STE IGRU SKOCKO