Dokumentacja projektu

Aplikacja mobilna – „Milionerzy”

Autorzy:

Damian Samek

Dariusz Niemczycki

Informatyka III I st.

gr. Lab III

1. Cele projektu

Celem naszego projektu było zaprojektowanie aplikacji mobilnej – gry opartej o rozgrywkę z popularnego teleturnieju Milionerzy

2. Wykorzystane technologie

Grę „Milionerzy” napisaliśmy w programie Android Studio. Warstwę logiczną zaprogramowaliśmy w języku Java, natomiast graficzny interfejs został zaimplementowany w plikach XML. Aplikacja łączy się z zewnętrzną bazą danych Firebase, z której pobiera pytania, a także pobiera i zapisuje wyniki.

3. Opis projektu

Gra składa się z czetrech głównych Aktywności (MainActivity, PlayActivity, ScoreActivity oraz HighscoreActivity).

MainActivity, czyli menu główne naszej aplikacji

Rozgrywka (PlayActivity):
Na początku naszej rozgrywki do lokalnej listy pytań pobierana jest lista 36 pytań z bazy danych. Na każdy z 12 poziomów przypada po 3 pytania, które są losowo dobierane podczas jego rozegrania. Następnie wczytywane jest wylosowane pytanie.

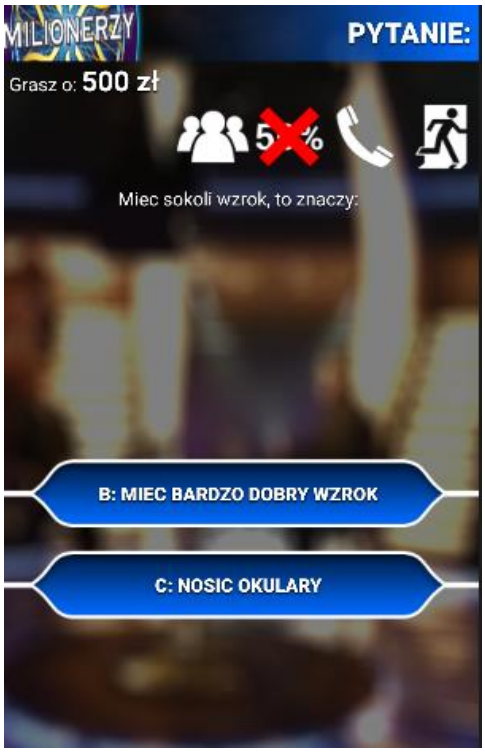
Widok wczytanego pytania wraz z odpowiedziami

Gracz poprzez kliknięcie przycisku wybiera prawidłową odpowiedź.

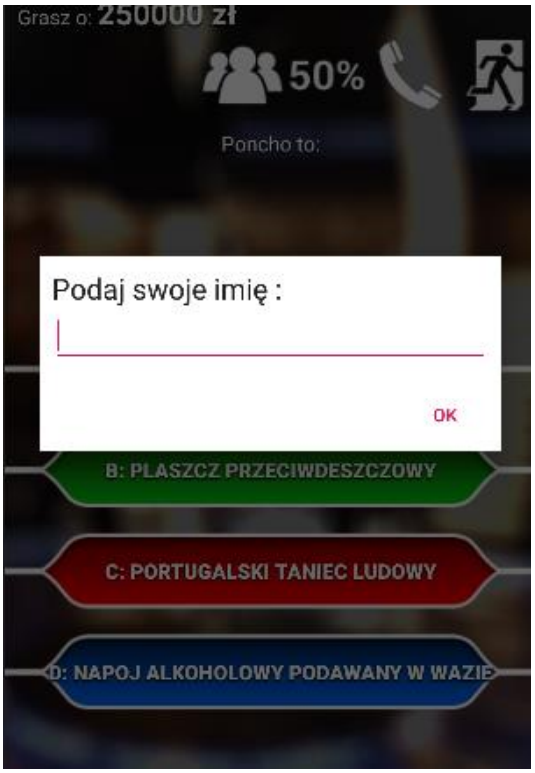
W przypadku, gdy odpowiedź jest poprawna, przycisk zmienia kolor na zielony.

W przypadku błędnej odpowiedzi, wybrana przez gracza odpowiedź zostaje podświetlona czerwonym kolorem, natomiast zielonym kolorem podkreślona jest odpowiedź prawidłowa.

Zostały również zaimplementowane koła ratunkowe, które możemy wykorzystać klikając na poszczególne przyciski na panelu kół ratunkowych znajdującym się pod kwotą do wygrania.



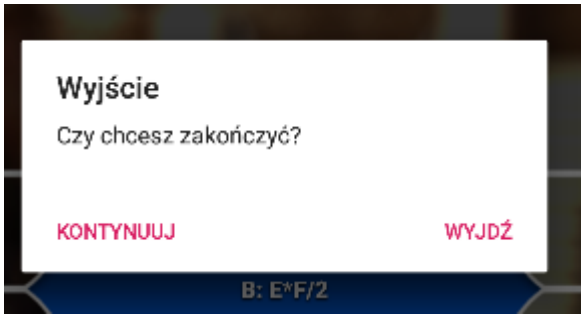
Działanie koła ratunkowego „pół na pół”



Jeżeli gracz wycofa się klikając na przycisk ucieczki z panelu kół ratunkowych lub odpowie błędnie na pytanie, zostaje poproszony o imię

Po wpisaniu imienia aplikacja przechodzi do ScoreActivity, czyli podsumowania gry, gdzie pokazana jest wygrana oraz czas, w jakim została zdobyta

Po kilku sekundach gracz zostaje przeniesiony do HighscoreActivity, czyli listy najlepszych graczy. Gracze posortowani są według wygranej i czasu, w którym ją zdobyli.



W przypadku wciśnięcia przycisku „wstecz” na panelu dolnym, aplikacja wyświetla okno dialogowe:

Aplikacja działa również w orientacji poziomej.

4. Wyjaśnienie fragmentów kodu

```
private void calculateTime(){
    tEnd = System.currentTimeMillis();
    long tDelta = tEnd - tStart;
    elapsedSeconds = (int)(tDelta / 1000.0);
    elapsedMinutes = elapsedSeconds/60;
    secondsAfterConversion = elapsedSeconds-elapsedMinutes*60;
}
```

Czas zdobycia nagrody to różnica uzyskanego poprzez odczyt aktualnego czasu systemowego przy stworzeniu PlayActivity czyli komponentu rozgrywki oraz przy zakończeniu rozgrywki. elapsedMinutes oraz secondsAfterConversion to zmienne służące do stworzenia łańcucha znaków z czasem, a elapsedSeconds to zmienna, po której sortowane były wyniki zawodników zaraz po sortowaniu według zdobytej kwoty.

```
public void retrieveHighscoreList() {
    final DatabaseReference databaseReference = FirebaseDatabase.getInstance().getReference().child("highscore");
    final Query myTopScoresQuery = databaseReference.orderByChild("score").limitToLast(10);
    myTopScoresQuery.addValueEventListener(new ValueEventListener() {
        @Override
        public void onDataChange(DataSnapshot dataSnapshot) {
            for (DataSnapshot childDataSnapshot : dataSnapshot.getChildren()) {

                Result result = new Result();
                result.setName(childDataSnapshot.child("name").getValue().toString());
                result.setScore(Integer.valueOf(childDataSnapshot.child("score").getValue().toString()));
                result.setMinutes(Integer.valueOf(childDataSnapshot.child("minutes").getValue().toString()));
                result.setSeconds(Integer.valueOf(childDataSnapshot.child("seconds").getValue().toString()));
                result.setSecondsAfterConversion(Integer.valueOf(childDataSnapshot.child("secondsAfterConversion").getValue().toString()));
                highscores.add(result);
                }
```

Firebase nie udostępnia możliwości sortowania wyników w kolejności malejącej, więc jako referencję do bazy wzięliśmy bazę wyników posortowanych w kolejności rosnącej, ograniczoną do ostatnich 10 wyników. Każdy z tych wyników zapisaliśmy do obiektów typu Result i zamieściliśmy w lokalnej liście.

```
highscores.sort((Comparator) (result1, result2) → {
        if(result1.getScore()==result2.getScore()){
        if(result1.getSeconds()==result2.getSeconds())
            return 0;
        else if(result1.getSeconds()> result2.getSeconds())
            return -1;
        else
            return 1;
    }
        else return 0;
});

    for(int i=9; i>=0; i--)
    {
        highscoresStrings.add(highscoresStrings.size()+1+". "+highscores.get(i).toString());
    }
```

Zaimplementowaliśmy własny mechanizm porównujący obiekty, który w przypadku wykrycia dwóch graczy o tej samej wygranej kwocie stawiał wyżej tego, który zrobił to w krótszym czasie. Na koniec musieliśmy użyć pętli z dekrementacją, w której wkładaliśmy zwycięzców do ostatecznej listy o porządku malejącym. W ten sposób udało się nam posortować w sposób malejący rekordy z bazy danych po stronie klienta. Niestety ta funkcjonalność działa tylko w najnowszych wersjach systemu Android, ponieważ funkcja Collections.sort() z języka Java nie jest obsługiwana przez starsze API.

Problem został rozwiązany poprzez podwójne użycie funkcji OrderByChild() klasy Query, najpierw po potomku „score”, następnie po potomku „seconds”.