**QUIZZLER – seminarski rad iz Objektno orijentiranog programiranja (120)**

Ivan Tomić, 2020./2021.

**DIZAJN**

**Botuni za odgovore**



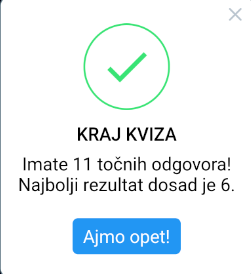
Slika 1 – početni ekran

**Traka s popisom točnih odgovora**

**Tekst pitanja**

Glavni ekran sastoji se od tri dijela – tekst pitanja, botuna za odgovore i trakom s popisom točnih odgovora (ovisno o odgovorima korisnika).

Korisnik odgovara na pitanja (15 ukupno), a nakon odgovora pokazuje se sljedeće i, ovisno o odgovoru, na dnu se ekrana prikaže oznaka za točan, tj. netočan odgovor. Po završetku kviza prikazuje se pop-up ekran i obavještava korisnika o broju točnih odgovora, kao i najvećem broju („highscore“) otkad se aplikacija instalirala. To znači da će se najveći rezultat zapamtiti, čak i ako se aplikacija ugasi.

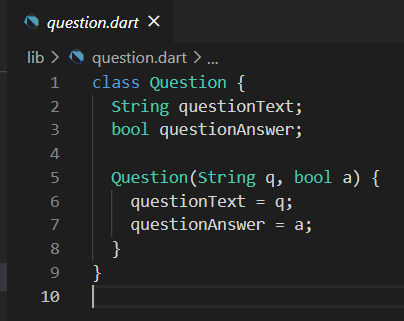


Slika 2 – završni ekran

**KOD**

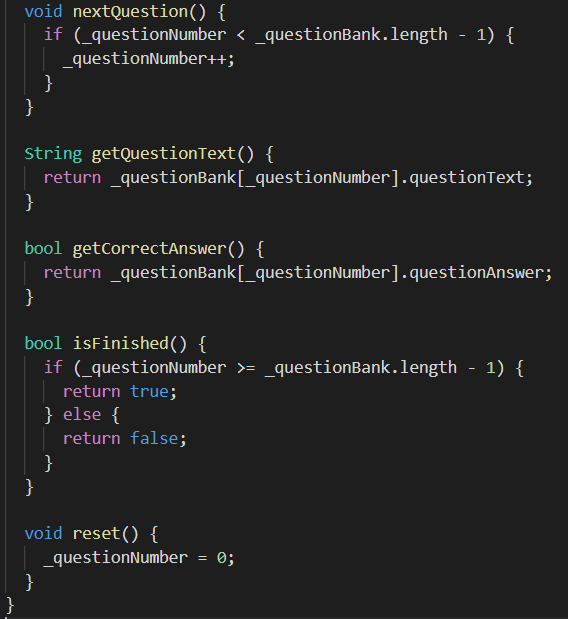
Source code sadrži tri datoteke:

1. question.dart – definicija klase Question
2. quiz\_brain.dart – popis (lista) svih pitanja i funkcije pomoću kojih baratamo s pitanjima
3. main.dart – glavna datoteka, sadrži cijeli dizajn i logiku programa

 Slika 3 – klasa Question

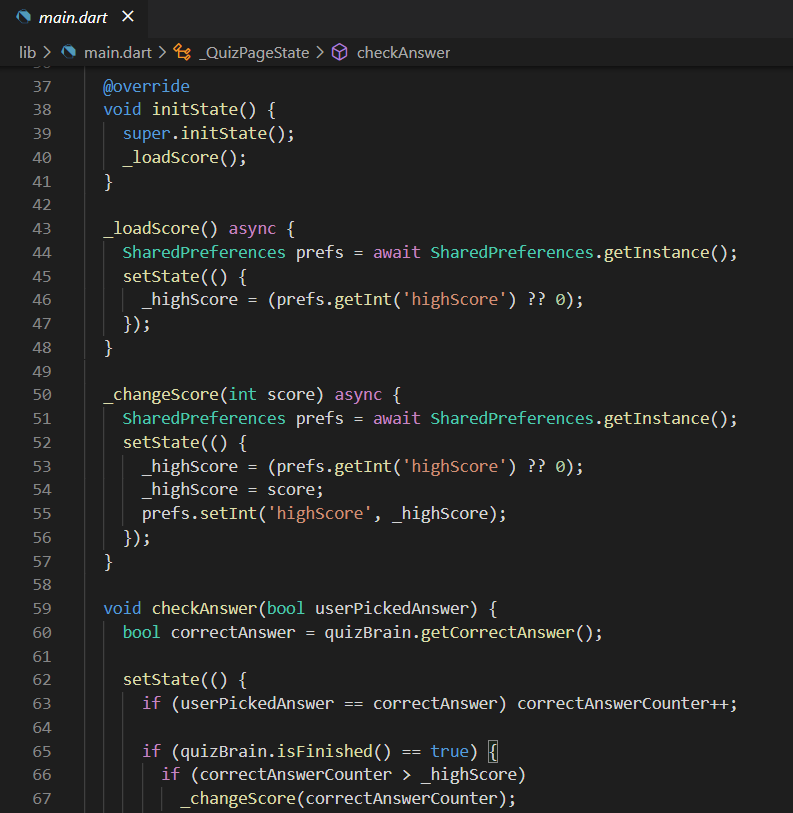
Da bi aplikacija funkcionirala, klasu Question potrebno je bilo definirati na način da se lako učitava tekst pitanja i točnost izjave.

Datoteka quiz\_brain.dart, osim liste s pitanjima, sadrži dvije get() funkcije (za tekst i odgovor), dvije funkcije za baratanje s brojem pitanja (inkrement i resetiranje), kao i funkciju koja provjerava je li korisnik došao do kraja kviza. Osnova funkcioniranja je objekt QuizBrain u kojem je lista imena \_questionBank, koja u sebi sadrži popis svih pitanja.



Slika 4 – funkcije klase QuizBrain

Da bi aplikacija funkcionirala, potrebno je bilo dodati dva eksterna paketa, koja su jedna od velikih prednosti razvoja aplikacija pomoću Fluttera. Jedan od njih je rFlutter Alert, pomoću kojeg dobivamo gotovi dizajn za pop up ekran, a drugi je Shared Preferences, s kojim možemo implementirati čitanje i zapisivanje podataka lokalno, u našem slučaju najveći broj točnih odgovora.



Slika 5 – čitanje i pisanje podataka na disk

Funkcije initState() i \_loadScore() događaju se pri učitavanju aplikacije, a funkciju \_changeScore(int) pozivamo ako je korisnik upravo postavio novi rekord (što provjeravamo u retku 66). Da bi funkcije radile što se traži od njih, trebaju raditi asinkorono.

Botuni za odgovore, zajedno s tekstom pitanja, čine jedan stupac glavnog ekrana. S obzirom na ogromne mogućnosti Fluttera za uređivanje tzv. „widgeta“, moguća je lako i uredno prilagoditi sve željene elemente svojim potrebama. Kao primjer, uzet ćemo kod za botun „Točno“:

 Slika 6 – botun „Točno“

Promatrajmo npr. widget FlatButton, koji je dijete od Paddinga i unuk od widgeta Expanded, koji obuhvaća sve widgete za ovaj botun, a kao svoje dijete ima widget Text. Ima 4 definirana svojstva:

1. boja teksta (textColor) – bijela
2. boja ispune (color) – zelena
3. dijete (child) – Text widget
4. funkciju ako je pritisnut (onPressed()) – checkAnswer(true)

Text widget dalje ima svoja uređena svojstva (dijete TextStyle, a ono ima svoja itd.). I više je nego očito da je osnova programiranja u Flutteru koncept widgeta i njihova međusobna interakcija. Budući da je dosta widgeta unaprijed već definirano, ovaj botun automatski zna što to znači da ga je korisnik dodirnuo i koju animaciju da prikaže prilikom dodira (ako nije promijenjena od strane programera). To uvelike olakšava postupak izrade programa, jer je potrebno fokusirati se samo na način kako već gotove stvari djeluju međusobno, ako je potreban uobičajena funkcija. Naravno, moguće je prilagoditi cijeli program sebi, ali to bi oduzelo previše vremena nepotrebno.