

**Технически университет**

**София**

**Курсова работа по**

**„Програмиране на мобилни устройства”**

**Изпълнена от:**

**Студент: Красимир Етов**

**Факултет: ФКСТ**

**Специалност: КСИ**

**Факултетен № 121216142**

**Група: 51**

**Дата: 26.05.2019 Проверил:**

**Град: София**

**Същност**

Да се реализира тестова система с предварително зададени въпроси, като в края на теста да се покаже броя верни отговори на потребителя и съответно да може да се вземе теста на ново.

Приложението представлява тестова система с предварително зададени въпроси, на които потребителя трябва да отговори с „True” или “False”. След края на въпросите се изписва резултат показващ броя на правилно отговорените въпроси от потребителя.



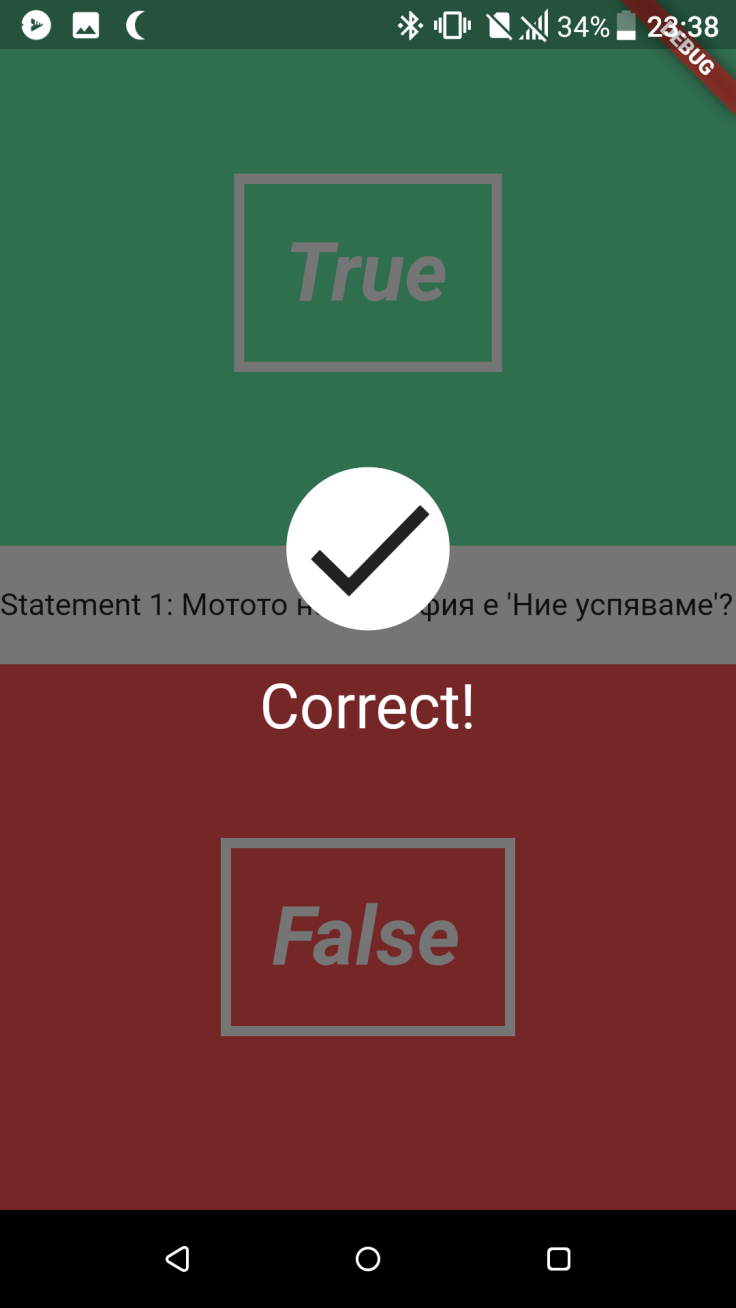
**Инструкции за ползване**

Когато стартираме приложението ни се отваря началният екран, който представлява бутон и при натискането му стартираме куиза.

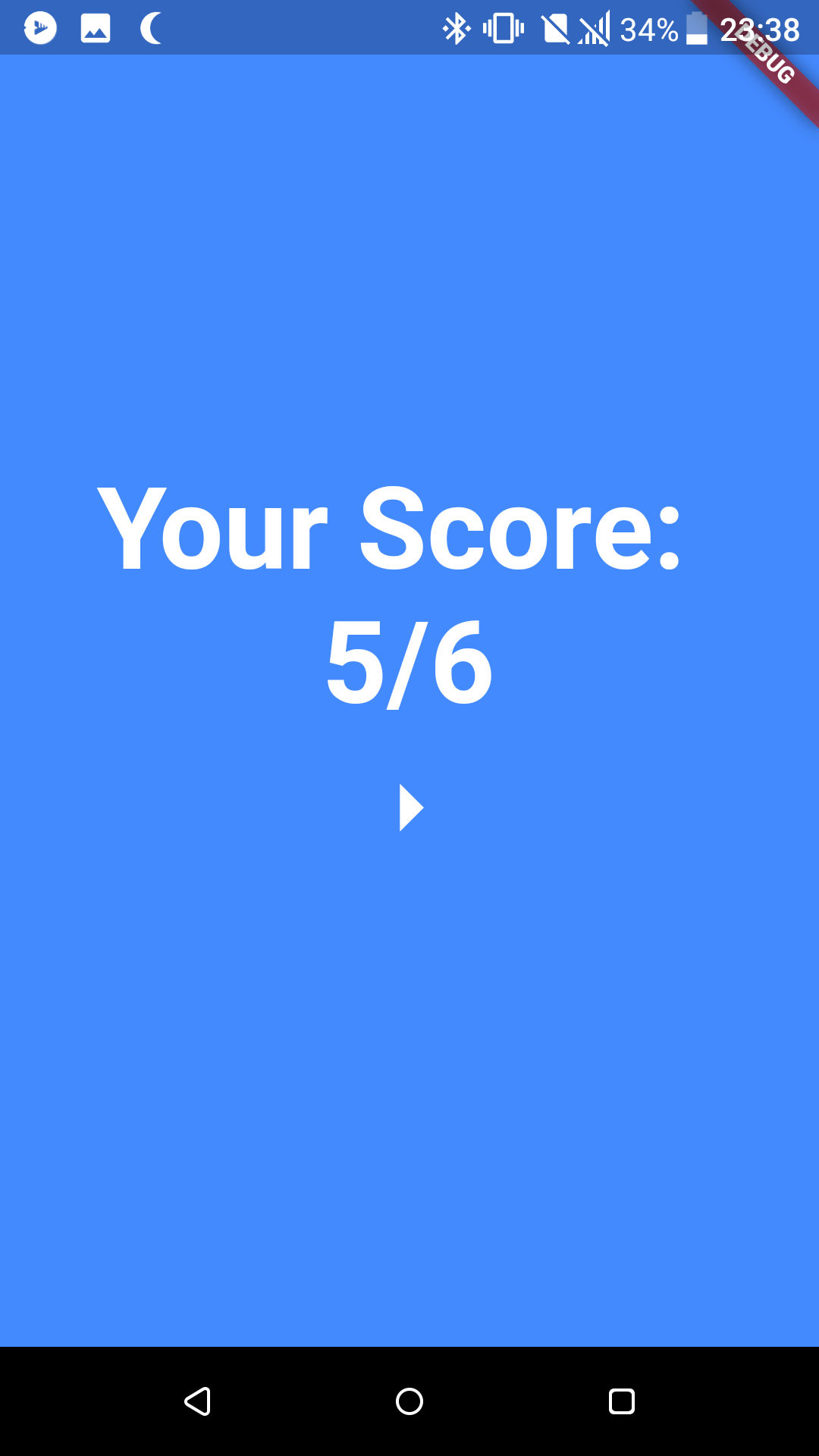
След стартиране на теста се показва първият въпрос и при неговото отговаряне се зарежда следващия докато не свършат заредените въпроси.



Всеки един от въпросите има 2 отговора като само един от тях е верен.

След отговарянето на всички въпроси се отваря нов прозорец показващ броя на вярно отговорените въпроси заедно с бутон, който при натискане ни връща към началния екран



**Реализация**

Приложението е реализирано с framework на име Flutter, който работи с така наречените Widget-и за визуализация и използва програмния език Dart.

Налични са няколко на брой widget-и разпределени по папки:

* Pages:
  + landing\_page.dart – Тук се визуализира началната страница на приложението
  + quiz\_page.dart – Тук се намира логиката за въпросите – как и колко да продължи куиза, както и самите въпроси
  + score\_page.dart – Тук се визуализира финалната страница на приложението, когато потребителя отговори на всички въпроси
* UI:
  + answer\_button.dart – Този widget се грижи за визуализацията на бутоните “True” и “False”, които се използват за отговор на въпросите.
  + correct\_wrong\_overlay.dart – Този widget се грижи за анимациите появяващи се при отговор на въпрос и смяна на екраните.
  + question\_text.dart
* Utils:
  + question.dart – Клас модел, който се грижи за визуализацията на въпроса, който потребителя трябва да отговори
  + quiz.dart – Клас модел, който брои правилните отговори, съдържа логиката за преминаване към следващ въпрос и за разбъркване на въпросите при стартиране на приложението.

Main.dart – Стартовата точка на приложението. За начален екран извикваме LandingPage widget-a.

**Код**

**Main.dart**

void main() {

runApp(new MaterialApp(

home: new LandingPage(),

));

}

**LandingPage.dart**

class LandingPage extends StatelessWidget {

@override

Widget build(BuildContext context) {

return new Material(

color: Colors.greenAccent,

child: new InkWell(

onTap: () => Navigator.of(context).push(new MaterialPageRoute(builder: (BuildContext context) => new QuizPage())),

child: new Column(

mainAxisAlignment: MainAxisAlignment.center,

children: <Widget>[

new Text("Lets Quizzz", style: new TextStyle(color: Colors.white, fontSize: 50.0, fontWeight: FontWeight.bold),),

new Text("Tap to start!", style: new TextStyle(color: Colors.white, fontSize: 20.0, fontWeight: FontWeight.bold),)

],

),

),

);

}

}

**QuizPage.dart**

class QuizPage extends StatefulWidget {

@override

State createState() => new QuizPageState();

}

class QuizPageState extends State<QuizPage> {

Question currentQuestion;

Quiz quiz = new Quiz([

new Question("Мусала е най-високият връх в България?.", true),

new Question("Най-източната част на България е нос Шабла", false),

new Question("България е създадена през 681 г.", true),

new Question("Хан Омуртаг е написал първите писани закони?", false),

new Question("Границите на България са били на четири морета?", false),

new Question("Мотото на ТУ-София е 'Ние успяваме'?", true)

]);

String questionText;

int questionNumber;

bool isCorrect;

bool overlayShouldBeVisible = false;

@override

void initState() {

super.initState();

currentQuestion = quiz.nextQuestion;

questionText = currentQuestion.question;

questionNumber = quiz.questionNumber;

}

void handleAnswer(bool answer) {

isCorrect = (currentQuestion.answer == answer);

quiz.answer(isCorrect);

this.setState(() {

overlayShouldBeVisible = true;

});

}

@override

Widget build(BuildContext context) {

return new Stack(

fit: StackFit.expand,

children: <Widget>[

new Column( // This is our main page

children: <Widget>[

new AnswerButton(true, () => handleAnswer(true)), //true button

new QuestionText(questionText, questionNumber),

new AnswerButton(false, () => handleAnswer(false)), // false button

],

),

overlayShouldBeVisible == true ? new CorrectWrongOverlay(

isCorrect,

() {

if (quiz.length == questionNumber) {

Navigator.of(context).pushAndRemoveUntil(new MaterialPageRoute(builder: (BuildContext context) => new ScorePage(quiz.score, quiz.length)), (Route route) => route == null);

return;

}

currentQuestion = quiz.nextQuestion;

this.setState(() {

overlayShouldBeVisible = false;

questionText = currentQuestion.question;

questionNumber = quiz.questionNumber;

});

}

) : new Container()

],

);

}

}

**ScorePage.dart**

class ScorePage extends StatelessWidget {

final int score;

final int totalQuestions;

ScorePage(this.score, this.totalQuestions);

@override

Widget build(BuildContext context) {

return new Material(

color: Colors.blueAccent,

child: new Column(

mainAxisAlignment: MainAxisAlignment.center,

children: <Widget>[

new Text("Your Score: ", style: new TextStyle(color: Colors.white, fontWeight: FontWeight.bold, fontSize: 50.0),),

new Text(score.toString() + "/" + totalQuestions.toString(), style: new TextStyle(color: Colors.white, fontWeight: FontWeight.bold, fontSize: 50.0)),

new IconButton(

icon: new Icon(Icons.arrow\_right),

color: Colors.white,

iconSize: 50.0,

onPressed: () => Navigator.of(context).pushAndRemoveUntil(new MaterialPageRoute(builder: (BuildContext context) => new LandingPage()), (Route route) => route == null)

)

],

)

);

}

}

**AnswerButton.dart**

class AnswerButton extends StatelessWidget {

final bool \_answer;

final VoidCallback \_onTap;

AnswerButton(this.\_answer, this.\_onTap);

@override

Widget build(BuildContext context) {

return new Expanded( // true button

child: new Material(

color: \_answer == true ? Colors.greenAccent : Colors.redAccent,

child: new InkWell(

onTap: () => \_onTap(),

child: new Center(

child: new Container(

decoration: new BoxDecoration(

border: new Border.all(color: Colors.white, width: 5.0)

),

padding: new EdgeInsets.all(20.0),

child: new Text(\_answer == true ? "True" : "False",

style: new TextStyle(color: Colors.white, fontSize: 40.0, fontWeight: FontWeight.bold, fontStyle: FontStyle.italic)

),

)

),

),

),

);

}

}

**CorrectWrongOverlay.dart**

class CorrectWrongOverlay extends StatefulWidget {

final bool \_isCorrect;

final VoidCallback \_onTap;

CorrectWrongOverlay(this.\_isCorrect, this.\_onTap);

@override

State createState() => new CorrectWrongOverlayState();

}

class CorrectWrongOverlayState extends State<CorrectWrongOverlay> with SingleTickerProviderStateMixin {

Animation<double> \_iconAnimation;

AnimationController \_iconAnimationController;

@override

void initState() {

super.initState();

\_iconAnimationController = new AnimationController(duration: new Duration(seconds: 2), vsync: this);

\_iconAnimation = new CurvedAnimation(parent: \_iconAnimationController, curve: Curves.elasticOut);

\_iconAnimation.addListener(() => this.setState(() {}));

\_iconAnimationController.forward();

}

@override

void dispose() {

\_iconAnimationController.dispose();

super.dispose();

}

@override

Widget build(BuildContext context) {

return new Material(

color: Colors.black54,

child: new InkWell(

onTap: () => widget.\_onTap(),

child: new Column(

mainAxisAlignment: MainAxisAlignment.center,

children: <Widget>[

new Container(

decoration: new BoxDecoration(

color: Colors.white,

shape: BoxShape.circle

),

child: new Transform.rotate(

angle: \_iconAnimation.value \* 2 \* PI,

child: new Icon(widget.\_isCorrect == true ? Icons.done : Icons.clear, size: \_iconAnimation.value \* 80.0,),

),

),

new Padding(

padding: new EdgeInsets.only(bottom: 20.0),

),

new Text(widget.\_isCorrect == true ? "Correct!" : "Wrong!", style: new TextStyle(color: Colors.white, fontSize: 30.0),)

],

),

),

);

}

}

**QuestionText.dart**

class QuestionText extends StatefulWidget {

final String \_question;

final int \_questionNumber;

QuestionText(this.\_question, this.\_questionNumber);

@override

State createState() => new QuestionTextState();

}

class QuestionTextState extends State<QuestionText> with SingleTickerProviderStateMixin {

Animation<double> \_fontSizeAnimation;

AnimationController \_fontSizeAnimationController;

@override

void initState() {

super.initState();

\_fontSizeAnimationController = new AnimationController(duration: new Duration(milliseconds: 500), vsync: this);

\_fontSizeAnimation = new CurvedAnimation(parent: \_fontSizeAnimationController, curve: Curves.bounceOut);

\_fontSizeAnimation.addListener(() => this.setState(() {}));

\_fontSizeAnimationController.forward();

}

@override

void dispose() {

\_fontSizeAnimationController.dispose();

super.dispose();

}

@override

void didUpdateWidget(QuestionText oldWidget) {

super.didUpdateWidget(oldWidget);

if (oldWidget.\_question != widget.\_question) {

\_fontSizeAnimationController.reset();

\_fontSizeAnimationController.forward();

}

}

@override

Widget build(BuildContext context) {

return new Material(

color: Colors.white,

child: new Padding(

padding: new EdgeInsets.symmetric(vertical: 20.0),

child: new Center(

child: new Text("Statement " + widget.\_questionNumber.toString() + ": " + widget.\_question,

style: new TextStyle(fontSize: \_fontSizeAnimation.value \* 15),

),

)

),

);

}

}

**Question.dart**

class Question {

final String question;

final bool answer;

Question(this.question, this.answer);

}

**Quiz.dart**

class Quiz {

List<Question> \_questions;

int \_currentQuestionIndex = -1;

int \_score = 0;

Quiz(this.\_questions) {

\_questions.shuffle();

}

List<Question> get questions => \_questions;

int get length => \_questions.length;

int get questionNumber => \_currentQuestionIndex+1;

int get score => \_score;

Question get nextQuestion {

\_currentQuestionIndex++;

if (\_currentQuestionIndex >= length) return null;

return \_questions[\_currentQuestionIndex];

}

void answer(bool isCorrect) {

if (isCorrect) \_score++;

}

}

**Линк към github repository -** [**https://github.com/KrasimirEtov/TU-Sofia**](https://github.com/KrasimirEtov/TU-Sofia)

**Използвани източници**

* **Youtube -** [**Flutter crash course**](https://www.youtube.com/watch?v=jBBl1tYkUnE)
* **Stackoverflow -** [**https://stackoverflow.com/**](https://stackoverflow.com/)