Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение высшего образования
«Московский государственный технический университет имени Н.Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ «Информатика и системы управления»

КАФЕДРА «Теоретическая информатика и компьютерные технологии»

Рубежный контроль № 1

по курсу «Разработка мобильных приложений»

Студент группы ИУ9-71Б Окутин Д. А.

Преподаватель Посевин Д. П.

Москва 2024

1 Цель

Реализовать мобильное приложение для визуализации значений полинома 3 степени.

2 Задание

Через форму пользователем вручную должны заполняться коэфициенты полинома и на графике должна быть видна зависимость значения полинома в точке x от 0 до 100.

3 Реализация

Исходный код представлен в листинге 1.

Листинг 1: Исхоный код программы

```
1
2  import 'package:flutter/material.dart';
3  import 'dart:math';
4
5  void main() {
6    runApp(PolynomialGraphApp());
7  }
8
9  class PolynomialGraphApp extends StatelessWidget {
10   @override
11   Widget build(BuildContext context) {
12     return MaterialApp(
13       debugShowCheckedModeBanner: false,
14       title: 'Polynomial Graph',
15       theme: ThemeData(
16         primarySwatch: Colors.blue,
17       ),
18       home: PolynomialInput(),
19     );
20   }
21 }
22
23 class PolynomialInput extends StatefulWidget {
24   @override
25   _PolynomialInputState createState() => _PolynomialInputState();
```

```
26 }
27
28 class _PolynomialInputState extends State<PolynomialInput> {
29   final TextEditingController _xController = TextEditingController();
30   final List<TextEditingController> _coeffControllers =
31       List.generate(4, (index) => TextEditingController());
32   List<double> _coefficients = [0, 0, 0, 0];
33
34   void _generateGraph() {
35     setState(() {
36       double x = double.tryParse(_xController.text) ?? 0;
37
38       _coefficients = _coeffControllers.map((controller) {
39         double value = double.tryParse(controller.text) ?? 0;
40         return value;
41       }).toList();
42
43       print('x: $x');
44       print('                          : $_coefficients');
45     });
46   }
47
48   @override
49   Widget build(BuildContext context) {
50     return Scaffold(
51       appBar: AppBar(
52         title: Text('Polynomial Graph'),
53       ),
54       body: Padding(
55         padding: const EdgeInsets.all(16.0),
56         child: Column(
57           children: <Widget>[
58             SizedBox(height: 10),
59             for (int i = 0; i < 4; i++)
60               TextField(
61                 controller: _coeffControllers[i],
62                 decoration: InputDecoration(labelText: '
                                a${i}'),
63                 keyboardType: TextInputType.number,
64               ),
65             SizedBox(height: 10),
66             ElevatedButton(
67               onPressed: _generateGraph,
68               child: Text('                                    '),
69             ),
70             SizedBox(height: 20),
```

```
71              Expanded(
72                child: CustomPaint(
73                  painter: PolynomialGraphPainter(coefficients:
    _coefficients),
74                  child: Container(),
75                ),
76              ),
77            ],
78          ),
79        ),
80      );
81    }
82  }
83
84  class PolynomialGraphPainter extends CustomPainter {
85    final List<double> coefficients;
86
87    PolynomialGraphPainter({required this.coefficients});
88
89    void paint(Canvas canvas, Size size) {
90      final paint = Paint()
91        ..color = Colors.blue
92        ..style = PaintingStyle.stroke
93        ..strokeWidth = 4.0;
94
95      final axisPaint = Paint()
96        ..color = Colors.black
97        ..style = PaintingStyle.stroke
98        ..strokeWidth = 2.0;
99
100     //
101     const double yMax = 10000; //

              Y
102     const double xMax = 10; //                                    X
103
104     final textStyle = TextStyle(
105       color: Colors.black,
106       fontSize: 12,
107     );
108     final textPainter = TextPainter(
109       textDirection: TextDirection.ltr,
110     );
111
112     //
```

```
113     canvas.drawLine(Offset(0, size.height), Offset(size.width, size.
        height), axisPaint); //          X
114     canvas.drawLine(Offset(0, 0), Offset(0, size.height), axisPaint); //
              Y
115
116     //
117     final path = Path();
118     for (double x = 0; x <= xMax; x++) {
119       double y = _calculatePolynomialValue(x);
120       double normalizedY = size.height - (y / yMax) * size.height;
121
122       if (x == 0) {
123         path.moveTo((x / xMax) * size.width, normalizedY);
124       } else {
125         path.lineTo((x / xMax) * size.width, normalizedY);
126       }
127     }
128
129     canvas.drawPath(path, paint);
130
131     //                                                  X
132     for (int i = 0; i <= 10; i++) { //                          , 10
                                    X
133       double x = i * 10;
134       double xPos = (x / 100) * size.width;
135
136       textPainter.text = TextSpan(
137         text: x.toString(),
138         style: textStyle,
139       );
140       textPainter.layout();
141       textPainter.paint(canvas, Offset(xPos - textPainter.width / 2,
        size.height));
142     }
143   }
144
145
146
147   double _calculatePolynomialValue(double x) {
148     double result = 0.0;
149     for (int i = 0; i < coefficients.length; i++) {
150       result +=
151           coefficients[i] * pow(x, i); // y = a0 + a1 * x + a2 * x^2 +
        a3 * x^3
152     }
153     return result;
```

```
154     }
155
156     @override
157     bool shouldRepaint(CustomPainter oldDelegate) {
158       return true; //
159     }
160 }
161 \begin{figure}
162     \centering
163     \includegraphics[width=0.5\linewidth]{latex//images/img1.png}
164 \end{figure}
```

4 Результаты

Результаты представлен на рисунке 1.

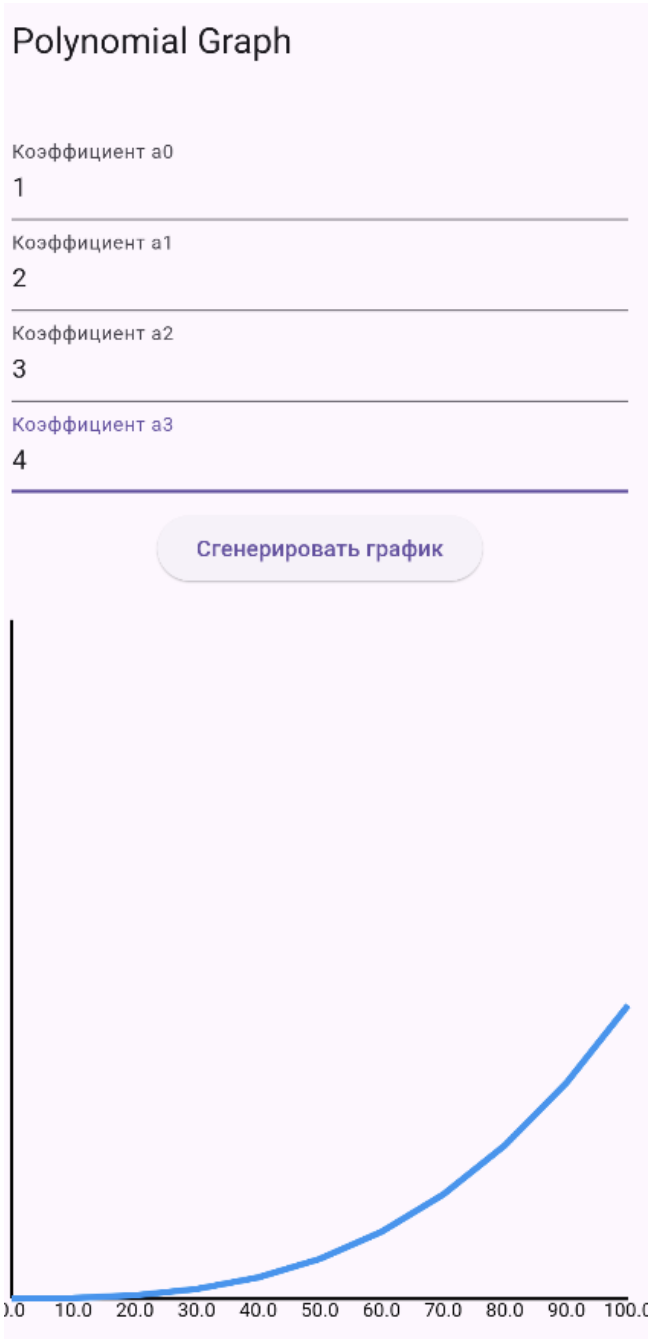

Рис. 1 — Интерфейс взаимодействия

5 Выводы

В результате работы было создано приложение для визуализации полинома 3 степени с коэффициентами, задаваемыми пользователем.