



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»

Летучка
«Работа с базой данных MySQL во
Flutter-приложении»
по курсу
«Разработка мобильных приложений»

Выполнил:
студент группы ИУ9-72Б
Караник А.А.

Проверено:
Посевин Д.П.

Москва, 2024

Цель работы

Цель данной работы - разработка мобильного приложения на Flutter, позволяющего пользователям вводить данные (имя, почта и возраст), добавлять их в базу данных MySQL, а также отображать список всех добавленных пользователей в виде таблицы. Также реализована функциональность удаления всех пользователей из базы данных и обновления отображаемой таблицы.

Реализация

Исходный код программы:

```
import 'package:flutter/material.dart';
import 'package:font_awesome_flutter/font_awesome_flutter.dart';
import 'package:http/http.dart' as http;
import 'dart:math';
import 'dart:convert';
import 'dart:async';
import 'dart:io';
import 'package:mqtt_client/mqtt_client.dart';
import 'package:mqtt_client/mqtt_server_client.dart';
import 'package:mysql1/mysql1.dart';

void main() {
  runApp(MyApp());
}

class MyApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: 'FlutterApp',
      debugShowCheckedModeBanner: false,
      theme: ThemeData(
        primarySwatch: Colors.blue,
        scaffoldBackgroundColor: Colors.white,
        useMaterial3: true,
        colorScheme: ColorScheme.fromSeed(
          seedColor: Color(0xFF13519F),
          primary: Color(0xFF13519F),
          onPrimary: Colors.white,
          background: Colors.white)),
      home: HomeScreen(),
    );
  }
}

class HomeScreen extends StatelessWidget {
  final GlobalKey<ScaffoldState> _scaffoldKey = GlobalKey<ScaffoldState>();

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      key: _scaffoldKey,
      appBar: AppBar(
        title: Text('Главный экран'),
        leading: IconButton(
          icon: Icon(Icons.menu),
          onPressed: () {
            _scaffoldKey.currentState?.openDrawer();
          },
        ),
      ),
    );
  }
}
```

```

    ),
  ),
  body: Center(
    child: Text(
      'Откройте меню',
      style: TextStyle(fontSize: 24),
    ),
  ),
  drawer: Drawer(
    child: ListView(
      padding: EdgeInsets.zero,
      children: <Widget>[
        DrawerHeader(
          decoration: BoxDecoration(
            color: Theme.of(context).primaryColor,
          ),
          child: Text(
            'Меню',
            style: TextStyle(
              color: Colors.white,
              fontSize: 24,
            ),
          ),
        ),
        ListTile(
          leading: FaIcon(
            FontAwesomeIcons.flaskVial,
            color: Theme.of(context).primaryColor,
          ),
          title: Text(
            'lab2',
            style: TextStyle(color: Theme.of(context).primaryColor),
          ),
          onTap: () {
            Navigator.of(context).pop();
            Navigator.of(context).push(
              MaterialPageRoute(builder: (context) => Lab2Screen()),
            );
          },
        ),
        Divider(color: Colors.black12, thickness: 1),
        ListTile(
          leading: FaIcon(
            FontAwesomeIcons.flaskVial,
            color: Theme.of(context).primaryColor,
          ),
          title: Text(
            'lab3',
            style: TextStyle(color: Theme.of(context).primaryColor),
          ),
          onTap: () {
            Navigator.of(context).pop();
            Navigator.of(context).push(
              MaterialPageRoute(builder: (context) => Lab3Screen()),
            );
          },
        ),
        Divider(color: Colors.black12, thickness: 1),
        ListTile(
          leading: FaIcon(
            FontAwesomeIcons.flaskVial,
            color: Theme.of(context).primaryColor,
          ),
          title: Text(
            'animation-controller',

```

```

        style: TextStyle(color: Theme.of(context).primaryColor),
      ),
      onTap: () {
        Navigator.of(context).pop();
        Navigator.of(context).push(
          MaterialPageRoute(builder: (context) => AnimScreen()),
        );
      },
    ),
    Divider(color: Colors.black12, thickness: 1),
    ListTile(
      leading: FaIcon(
        FontAwesomeIcons.flaskVial,
        color: Theme.of(context).primaryColor,
      ),
      title: Text(
        'mqtt-send-lab5',
        style: TextStyle(color: Theme.of(context).primaryColor),
      ),
      onTap: () {
        Navigator.of(context).pop();
        Navigator.of(context).push(
          MaterialPageRoute(builder: (context) => Mqtt1Screen()),
        );
      },
    ),
    Divider(color: Colors.black12, thickness: 1),
    ListTile(
      leading: FaIcon(
        FontAwesomeIcons.flaskVial,
        color: Theme.of(context).primaryColor,
      ),
      title: Text(
        'mqtt-get-lab5',
        style: TextStyle(color: Theme.of(context).primaryColor),
      ),
      onTap: () {
        Navigator.of(context).pop();
        Navigator.of(context).push(
          MaterialPageRoute(builder: (context) => Mqtt2Screen()),
        );
      },
    ),
    Divider(color: Colors.black12, thickness: 1),
    ListTile(
      leading: FaIcon(
        FontAwesomeIcons.flaskVial,
        color: Theme.of(context).primaryColor,
      ),
      title: Text(
        'fly-mysql',
        style: TextStyle(color: Theme.of(context).primaryColor),
      ),
      onTap: () {
        Navigator.of(context).pop();
        Navigator.of(context).push(
          MaterialPageRoute(builder: (context) => UserScreen()),
        );
      },
    ),
    Divider(color: Colors.black12, thickness: 1),
  ],
),
),
);

```

```

    }
}

class Lab2Screen extends StatefulWidget {
  @override
  _Lab2ScreenState createState() => _Lab2ScreenState();
}

class _Lab2ScreenState extends State<Lab2Screen> {
  String response = "Здесь будет ответ";
  bool isSwitched = false;

  Future<void> requestOff() async {
    final response = await http
      .get(Uri.parse('http://iocontrol.ru/api/sendData/karanik/value/0'));

    if (response.statusCode == 200) {
      setState(() {
        this.response = jsonDecode(response.body).toString();
      });
    } else {
      setState(() {
        this.response = 'Failed';
      });
    }
  }

  Future<void> requestOn() async {
    final response = await http
      .get(Uri.parse('http://iocontrol.ru/api/sendData/karanik/value/1'));

    if (response.statusCode == 200) {
      setState(() {
        this.response = jsonDecode(response.body).toString();
      });
    } else {
      setState(() {
        this.response = 'Failed';
      });
    }
  }
}

@override
Widget build(BuildContext context) {
  return Scaffold(
    appBar: AppBar(title: Text("lab2"), leading: BackButton()),
    body: Padding(
      padding: const EdgeInsets.all(16.0),
      child: Column(
        mainAxisAlignment: MainAxisAlignment.center,
        children: <Widget>[
          // Переключатель on/off
          Row(
            mainAxisAlignment: MainAxisAlignment.center,
            children: [
              Text("OFF"),
              Switch(
                value: isSwitched,
                inactiveThumbColor: Colors.grey,
                inactiveTrackColor: Colors.grey[300],
                onChanged: (value) {
                  setState(() {
                    isSwitched = value;
                    if (isSwitched) {
                      requestOn();
                    }
                  });
                },
              ),
            ],
          ),
        ],
      ),
    ),
  );
}

```

```

        } else {
            requestOff();
        }
    });
},
),
Text("ON"),
],
),
SizedBox(height: 16),
Text(response),
],
),
),
);
}
}

class Lab3Screen extends StatefulWidget {
  @override
  _Lab3ScreenState createState() => _Lab3ScreenState();
}

class _Lab3ScreenState extends State<Lab3Screen> {
  int _counter = 0;
  String _serverResponse = '';
  final TextEditingController _textController = TextEditingController();

  void _incrementCounter() {
    setState(() {
      _counter++;
    });
    _sendCounterToServer();
  }

  void _decrementCounter() {
    setState(() {
      _counter--;
    });
    _sendCounterToServer();
  }

  // Отправка значения на сервер (POST)
  Future<void> _sendCounterToServer() async {
    final url = Uri.parse('http://194.67.88.154:8100/${_counter}');
    try {
      final response = await http.post(url);

      if (response.statusCode == 200) {
        setState(() {
          _serverResponse = 'Значение отправлено: $_counter';
        });
      } else {
        setState(() {
          _serverResponse =
            'Не удалось отправить значение. Сервер ответил кодом статуса:
             ${response.statusCode}';
        });
      }
    } catch (e) {
      setState(() {
        _serverResponse = 'Ошибка отправки значения: $e';
      });
    }
  }
}

```

```

Future<void> _sendValueToServer(int value) async {
  final url = Uri.parse('http://194.67.88.154:8100/$value');
  try {
    final response = await http.post(url);

    if (response.statusCode == 200) {
      setState(() {
        _serverResponse = 'Значение отправлено: $value';
        _counter = value;
      });
    } else {
      setState(() {
        _serverResponse =
          'Не удалось отправить значение. Сервер ответил кодом статуса:
           → ${response.statusCode}';
      });
    }
  } catch (e) {
    setState(() {
      _serverResponse = 'Ошибка отправки значения: $e';
    });
  }
}

// Получение значения с сервера (GET)
Future<void> _getValueFromServer() async {
  final url = Uri.parse('http://194.67.88.154:8100');
  try {
    final response = await http.get(url);

    if (response.statusCode == 200) {
      setState(() {
        _counter = int.parse(response.body);
        _serverResponse = 'Счетчик обновлен с сервера: $_counter';
      });
    } else {
      setState(() {
        _serverResponse =
          'Не удалось получить значение. Сервер ответил кодом статуса:
           → ${response.statusCode}';
      });
    }
  } catch (e) {
    setState(() {
      _serverResponse = 'Ошибка получения значения: $e';
    });
  }
}

@override
Widget build(BuildContext context) {
  return Scaffold(
    appBar: AppBar(title: Text("lab3"), leading: BackButton()),
    body: Center(
      child: Column(
        mainAxisAlignment: MainAxisAlignment.center,
        children: <Widget>[
          const SizedBox(height: 20),
          TextField(
            controller: _textController,
            decoration: const InputDecoration(
              labelText: 'Введите init value',
              border: OutlineInputBorder(),
            ),
          ),
        ],
      ),
    ),
  );
}

```

```

    ),
    const SizedBox(height: 10),
    ElevatedButton(
      onPressed: () {
        _sendValueToServer(int.parse(_textController.text));
      },
      child: const Text('POST INIT'),
    ),
    const Text(
      'You have pushed the button this many times:',
    ),
    Text(
      '$_counter',
      style: Theme.of(context).textTheme.headlineMedium,
    ),
    const SizedBox(height: 20),
    ElevatedButton(
      onPressed: _getValueFromServer,
      child: const Text('GET Counter'),
    ),
    const SizedBox(height: 20),
    Text(
      _serverResponse,
      style: const TextStyle(color: Colors.lightGreen),
    ),
  ],
),
),
floatingActionButton: Column(
  mainAxisAlignment: MainAxisAlignment.end,
  children: <Widget>[
    FloatingActionButton(
      onPressed: _incrementCounter,
      tooltip: 'Increment',
      child: const Icon(Icons.add),
    ),
    const SizedBox(height: 10),
    FloatingActionButton(
      onPressed: _decrementCounter,
      tooltip: 'Decrement',
      child: const Icon(Icons.remove),
    ),
  ],
),
);
}

class AnimScreen extends StatefulWidget {
  @override
  _AnimScreenState createState() => _AnimScreenState();
}

class _AnimScreenState extends State<AnimScreen>
  with SingleTickerProviderStateMixin {
  double a = 1;
  double b = 0;
  double c = 0;

  late AnimationController _controller;

  @override
  void initState() {
    super.initState();
    _controller =

```



```

        AnimationController(vsync: this, duration: Duration(seconds: 1))
            ..repeat();
    }

    @override
    void dispose() {
        _controller.dispose();
        super.dispose();
    }

    @override
    Widget build(BuildContext context) {
        return Scaffold(
            appBar: AppBar(title: Text('Anim'), leading: BackButton()),
            body: Column(
                children: [
                    Expanded(
                        child: AnimatedBuilder(
                            animation: _controller,
                            builder: (context, child) {
                                return CustomPaint(
                                    painter: ParabolaPainter(a, b, c),
                                    child: Container(),
                                );
                            },
                        ),
                    ),
                    _buildSlider('a', a, -10, 10, (val) => setState(() => a = val)),
                    _buildSlider('b', b, -10, 10, (val) => setState(() => b = val)),
                    _buildSlider('c', c, -10, 10, (val) => setState(() => c = val)),
                ],
            ),
        );
    }

    Widget _buildSlider(String label, double value, double min, double max,
        ValueChanged<double> onChanged) {
        return Padding(
            padding: const EdgeInsets.symmetric(horizontal: 20, vertical: 10),
            child: Row(
                children: [
                    Text('$label: ', style: TextStyle(fontSize: 18)),
                    Expanded(
                        child: Slider(
                            value: value,
                            min: min,
                            max: max,
                            divisions: 100,
                            label: value.toStringAsFixed(2),
                            onChanged: onChanged,
                        ),
                    ),
                ],
            ),
        );
    }
}

class ParabolaPainter extends CustomPainter {
    final double a;
    final double b;
    final double c;

    ParabolaPainter(this.a, this.b, this.c);
}

```

```

@override
void paint(Canvas canvas, Size size) {
  final paint = Paint()
    ..color = Colors.blue
    ..strokeWidth = 2
    ..style = PaintingStyle.stroke;

  final centerX = size.width / 2;
  final centerY = size.height / 2;

  canvas.drawLine(Offset(0, centerY), Offset(size.width, centerY), paint);
  canvas.drawLine(Offset(centerX, 0), Offset(centerX, size.height), paint);

  paint.color = Colors.green;

  final path = Path();
  for (double x = -centerX; x <= centerX; x += 1) {
    double y = a * pow(x / 50, 2) + b * (x / 50) + c;
    if (x == -centerX) {
      path.moveTo(centerX + x, centerY - y * 50);
    } else {
      path.lineTo(centerX + x, centerY - y * 50);
    }
  }

  canvas.drawPath(path, paint);
}

@override
bool shouldRepaint(covariant CustomPainter oldDelegate) {
  return true;
}
}

class Mqtt1Screen extends StatefulWidget {
  @override
  Mqtt1ScreenState createState() => Mqtt1ScreenState();
}

class Mqtt1ScreenState extends State<Mqtt1Screen> {
  final _formKey = GlobalKey<FormState>();
  String _valueA = "";
  String _valueB = "";
  String _valueC = "";

  final client = MqttServerClient('broker.emqx.io', '');

  @override
  void initState() {
    super.initState();
    setupMqttClient();
  }

  Future<void> setupMqttClient() async {
    client.logging(on: true);
    client.setProtocolV311();
    client.keepAlivePeriod = 20;
    client.onDisconnected = onDisconnected;
    client.onConnected = onConnected;
    client.onSubscribed = onSubscribed;

    try {
      await client.connect();
    } catch (e) {
      print('Connection exception - $e');
    }
  }
}

```

```

        client.disconnect();
    }
}

void sendMessage(String topic, String message) {
    if (client.connectionStatus!.state == MqttConnectionState.connected) {
        final builder = MqttClientPayloadBuilder();
        builder.addString(message);
        client.publishMessage(topic, MqttQos.exactlyOnce, builder.payload!);
        print('Message "$message" sent to topic "$topic"');
    } else {
        print('MQTT Client is not connected');
    }
}

void onSubscribed(String topic) {
    print('Subscription confirmed for topic $topic');
}

void onDisconnected() {
    print('Disconnected from the broker');
}

void onConnected() {
    print('Connected to the broker');
}

@override
Widget build(BuildContext context) {
    return Scaffold(
        appBar: AppBar(title: Text('MQTT Publisher'), leading: BackButton()),
        body: Container(
            padding: EdgeInsets.all(10.0),
            child: Form(
                key: _formKey,
                child: Column(
                    children: <Widget>[
                        TextFormField(
                            decoration: InputDecoration(
                                labelText: 'Введите значение для топика A'),
                                onSave: (value) => _valueA = value!,
                            ),
                        TextFormField(
                            decoration: InputDecoration(
                                labelText: 'Введите значение для топика B'),
                                onSave: (value) => _valueB = value!,
                            ),
                        TextFormField(
                            decoration: InputDecoration(
                                labelText: 'Введите значение для топика C'),
                                onSave: (value) => _valueC = value!,
                            ),
                        SizedBox(height: 20.0),
                        ElevatedButton(
                            child: Text('Отправить'),
                            onPressed: () {
                                if (_formKey.currentState!.validate()) {
                                    _formKey.currentState!.save();

                                    sendMessage('IU9/test/a', _valueA);
                                    sendMessage('IU9/test/b', _valueB);
                                    sendMessage('IU9/test/c', _valueC);

                                    ScaffoldMessenger.of(context).showSnackBar(SnackBar(
                                        content: Text('Сообщения отправлены'),

```

```

        ));
    },
    style: ElevatedButton.styleFrom(
        padding: EdgeInsets.symmetric(
            horizontal: 50, vertical: 20),
        textStyle: TextStyle(
            fontSize: 20, fontWeight: FontWeight.bold)),
    ),
    1,
    ))));
}
}

class Mqtt2Screen extends StatefulWidget {
  @override
  Mqtt2ScreenState createState() => Mqtt2ScreenState();
}

class Mqtt2ScreenState extends State<Mqtt2Screen> {
  String _valueA = "Ожидание данных...";
  String _valueB = "Ожидание данных...";
  String _valueC = "Ожидание данных...";

  final client = MqttServerClient('broker.emqx.io', '');

  @override
  void initState() {
    super.initState();
    setupMqttClient();
  }

  Future<void> setupMqttClient() async {
    client.logging(on: true);
    client.setProtocolV311();
    client.keepAlivePeriod = 20;
    client.onDisconnected = onDisconnected;
    client.onConnected = onConnected;
    client.onSubscribed = onSubscribed;

    try {
      print('Подключение к MQTT брокеру...');
      await client.connect();
    } catch (e) {
      print('Ошибка подключения - $e');
      client.disconnect();
      return;
    }

    if (client.connectionStatus!.state == MqttConnectionState.connected) {
      print('Подключение установлено!');
    } else {
      print('Подключение не удалось. Статус: ${client.connectionStatus}');
      return;
    }
  }

  Future<void> getValueFromTopic() async {
    if (client.connectionStatus!.state == MqttConnectionState.connected) {
      print('Подписываемся на топики');
      client.subscribe('IU9/test/a', MqttQos.atLeastOnce);
      client.subscribe('IU9/test/b', MqttQos.atLeastOnce);
      client.subscribe('IU9/test/c', MqttQos.atLeastOnce);

      client.updates!.listen((List<MqttReceivedMessage<MqttMessage?>>? c) {

```

```

final recMess = c![0].payload as MqttPublishMessage;
final pt =
    MqttPublishPayload.bytesToStringAsString(recMess.payload.message);

print('Получено сообщение: topic: ${c[0].topic}, payload: $pt');

setState(() {
  if (c[0].topic == 'IU9/test/a') {
    _valueA = pt;
  } else if (c[0].topic == 'IU9/test/b') {
    _valueB = pt;
  } else if (c[0].topic == 'IU9/test/c') {
    _valueC = pt;
  }
});
});
} else {
  print('Клиент не подключен к брокеру');
}
}

void onSubscribed(String topic) {
  print('Subscription confirmed for topic $topic');
}

void onDisconnected() {
  print('Disconnected from the broker');
}

void onConnected() {
  print('Connected to the broker');
}

@override
Widget build(BuildContext context) {
  return Scaffold(
    appBar: AppBar(title: Text('MQTT Subscriber'), leading: BackButton()),
    body: Container(
      padding: EdgeInsets.all(10.0),
      child: Column(
        crossAxisAlignment: CrossAxisAlignment.start,
        children: <Widget>[
          Text(
            'Значение для топика A:',
            style: TextStyle(fontSize: 20.0),
          ),
          Text(
            _valueA,
            style: TextStyle(fontSize: 18.0),
          ),
          SizedBox(height: 20.0),
          Text(
            'Значение для топика B:',
            style: TextStyle(fontSize: 20.0),
          ),
          Text(
            _valueB,
            style: TextStyle(fontSize: 18.0),
          ),
          SizedBox(height: 20.0),
          Text(
            'Значение для топика C:',
            style: TextStyle(fontSize: 20.0),
          ),
          Text(

```

```

        _valueC,
        style: TextStyle(fontSize: 18.0),
    ),
    SizedBox(height: 20.0),
    ElevatedButton(
        child: Text('Получить значения'),
        onPressed: () {
            getValueFromTopic();
            ScaffoldMessenger.of(context).showSnackBar(SnackBar(
                content: Text('Значения обновляются...'),
            ));
        },
        style: ElevatedButton.styleFrom(
            padding:
                EdgeInsets.symmetric(horizontal: 50, vertical: 20),
            textStyle:
                TextStyle(fontSize: 20, fontWeight: FontWeight.bold)),
    ),
],
)));
}
}

class UserScreen extends StatefulWidget {
  @override
  _UserScreenState createState() => _UserScreenState();
}

class _UserScreenState extends State<UserScreen> {
  final TextEditingController _nameController = TextEditingController();
  final TextEditingController _emailController = TextEditingController();
  final TextEditingController _ageController = TextEditingController();
  List<User> _users = [];

  @override
  void initState() {
    super.initState();
    _fetchUsers();
  }

  Future<void> _fetchUsers() async {
    final conn = await MySqlConnection.connect(ConnectionSettings(
      host: 'students.yss.su',
      port: 3306,
      user: 'iu9mobile',
      db: 'iu9mobile',
      password: 'bmstubmstu123'));

    var results = await conn.query('select id, name, email, age from users');
    setState(() {
      _users = results
        .map((row) => User(
          id: row[0],
          name: row[1],
          email: row[2],
          age: row[3],
        ))
        .toList();
    });

    await conn.close();
  }

  Future<void> _addUser() async {
    final conn = await MySqlConnection.connect(ConnectionSettings(

```

```

        host: 'students.yss.su',
        port: 3306,
        user: 'iu9mobile',
        db: 'iu9mobile',
        password: 'bmstubmstu123'));

var result = await conn
    .query('insert into users (name, email, age) values (?, ?, ?)', [
        _nameController.text,
        _emailController.text,
        int.tryParse(_ageController.text) ?? 0
    ]);

print('Inserted row id=${result.insertId}');

await conn.close();
_clearFields();
}

Future<void> _deleteAllUsers() async {
    final conn = await MySqlConnection.connect(ConnectionSettings(
        host: 'students.yss.su',
        port: 3306,
        user: 'iu9mobile',
        db: 'iu9mobile',
        password: 'bmstubmstu123'));

    await conn.query('delete from users');
    _fetchUsers();

    await conn.close();
}

void _clearFields() {
    _nameController.clear();
    _emailController.clear();
    _ageController.clear();
}

@override
Widget build(BuildContext context) {
    return Scaffold(
        appBar: AppBar(title: Text('FLY MYSQL'), leading: BackButton()),
        body: Padding(
            padding: const EdgeInsets.all(16.0),
            child: Column(
                children: [
                    TextField(
                        controller: _nameController,
                        decoration: InputDecoration(labelText: 'Имя'),
                    ),
                    TextField(
                        controller: _emailController,
                        decoration: InputDecoration(labelText: 'Почта'),
                    ),
                    TextField(
                        controller: _ageController,
                        decoration: InputDecoration(labelText: 'Возраст'),
                        keyboardType: TextInputType.number,
                    ),
                    SizedBox(height: 20),
                    ElevatedButton(
                        onPressed: _addUser,
                        child: Text('Добавить пользователя'),
                    ),
                ],
            ),
        ),
    );
}

```

```

        ElevatedButton(
            onPressed: _fetchUsers,
            child: Text('Получить всех пользователей'),
        ),
        SizedBox(height: 20),
        Expanded(
            child: SingleChildScrollView(
                scrollDirection: Axis.horizontal,
                child: DataTable(
                    columns: [
                        DataColumn(label: Text('ID')),
                        DataColumn(label: Text('Имя')),
                        DataColumn(label: Text('Почта')),
                        DataColumn(label: Text('Возраст')),
                    ],
                    rows: _users.map((user) {
                        return DataRow(
                            cells: [
                                DataCell(Text(user.id.toString())),
                                DataCell(Text(user.name)),
                                DataCell(Text(user.email)),
                                DataCell(Text(user.age.toString())),
                            ],
                        );
                    }).toList(),
                ),
            ),
        ),
        ElevatedButton(
            onPressed: _deleteAllUsers,
            child: Text('Удалить всех пользователей'),
        ),
    ],
),
);
}
}

class User {
    final int id;
    final String name;
    final String email;
    final int age;

    User({
        required this.id,
        required this.name,
        required this.email,
        required this.age,
    });
}

```


Результаты

The screenshot shows a mobile application interface titled "FLY MYSQL". At the top, the status bar displays the time 22:34, signal strength, and 47% battery. The app header includes a back arrow and the title "FLY MYSQL". Below the header, there are three input fields: "Имя" (Name) with the value "dmitry", "Почта" (Email) with the value "dmitry@gmail.com", and "Возраст" (Age) with the value "21". Each field is followed by a horizontal line. Below the input fields, there are three buttons: "Добавить пользователя" (Add user), "Получить всех пользователей" (Get all users), and "Удалить всех пользователей" (Delete all users). At the bottom of the screen, there is a numeric keypad with digits 1-9, 0, a backspace key (X), an "OK" key, and a "." key. The bottom of the screen also shows the standard Android navigation bar with three icons: a square, a circle, and a triangle.

Рис. 1: Добавление пользователя

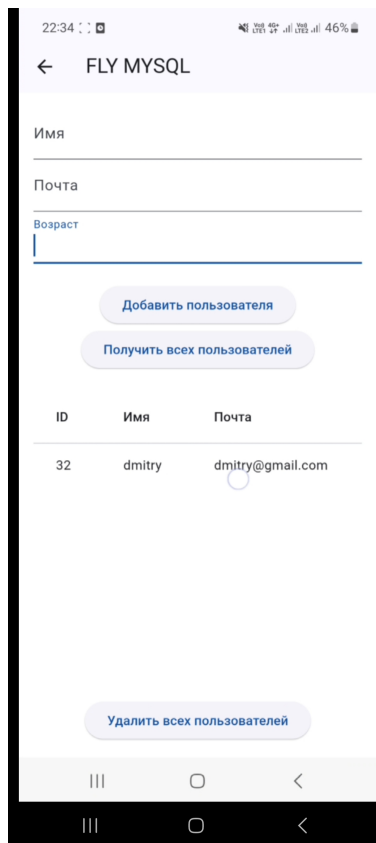


Рис. 2: Получение всех пользователей

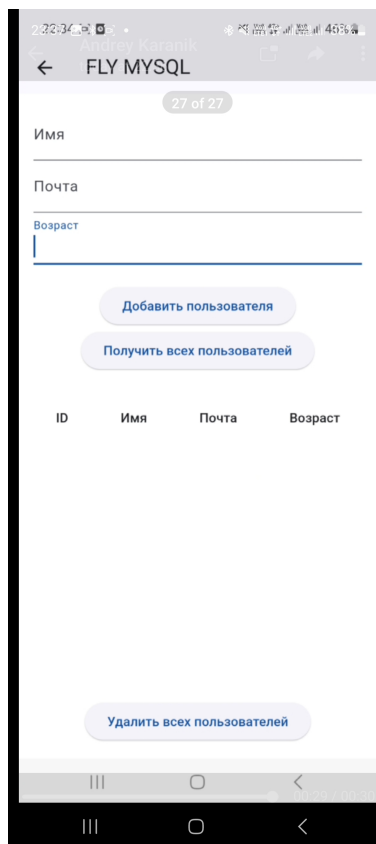


Рис. 3: Удаление всех пользователей

Вывод

В результате выполнения данной работы было создано простое и эффективное приложение на Flutter, которое демонстрирует взаимодействие с базой данных MySQL. Пользователи могут легко добавлять данные через интерфейс приложения, а также получать и удалять список всех пользователей в таблице.