|  |  |
| --- | --- |
| Gerb-BMSTU_01 | **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» |

**Отчет по самостоятельной работе**

**«Запустить MQTT клиент»**

***по курсу***

***«Разработка мобильных приложений»***

Выполнил:

Студент группы ИУ9-72Б

Караник А.А.

Проверил:

Посевин Д.П

1. *г.*

## Цель

Целью данной работы является изучение работы с протоколом MQTT (Message Queuing Telemetry Transport) и создание простого MQTT-клиента на языке Dart с использованием фреймворка Flutter.

## Практическая реализация

Код приложения:

import 'dart:async';

import 'dart:convert'; // <-- для JSON

import 'dart:io';

import 'package:flutter/material.dart';

import 'package:http/http.dart' as http;

import 'package:mqtt\_client/mqtt\_client.dart';

import 'package:mqtt\_client/mqtt\_server\_client.dart';

class MyForm extends StatefulWidget {

  @override

  State<StatefulWidget> createState() => MyFormState();

}

class MyFormState extends State {

  final \_formKey = GlobalKey<FormState>();

  String \_body = "";

  final client = MqttServerClient('test.mosquitto.org', '');

  var pongCount = 0; // Pong counter

  Future AAA(String message) async {

    client.logging(on: true);

    client.setProtocolV311();

    client.keepAlivePeriod = 20;

    client.onDisconnected = onDisconnected;

    client.onConnected = onConnected;

    client.onSubscribed = onSubscribed;

    client.pongCallback = pong;

    print('Mosquitto client connecting....');

    try {

      await client.connect();

    } on NoConnectionException catch (e) {

      print('client exception - $e');

      client.disconnect();

    } on SocketException catch (e) {

      print('socket exception - $e');

      client.disconnect();

    }

    if (client.connectionStatus!.state == MqttConnectionState.connected) {

      print('Mosquitto client connected');

    } else {

      print('ERROR Mosquitto client connection failed - disconnecting, status is ${client.connectionStatus}');

      client.disconnect();

      exit(-1);

    }

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

      final recMess = c![0].payload as MqttPublishMessage;

      final pt = MqttPublishPayload.bytesToStringAsString(recMess.payload.message);

      print('Change notification:: ---------------> topic is <${c[0].topic}>, payload is <-- $pt -->');

      \_body = "--> ${pt}";

      print('');

    });

    client.published!.listen((MqttPublishMessage message) {

      print('Published notification:: topic is ${message.variableHeader!.topicName}, with Qos ${message.header!.qos}');

    });

    const pubTopic = 'IU/9';

    final builder = MqttClientPayloadBuilder();

    builder.addString('Dart say ${message}');

    \_body = "--> ${message}";

    print('Subscribing to the UI/9 topic');

    client.subscribe(pubTopic, MqttQos.exactlyOnce);

    print('Publishing our topic');

    client.publishMessage(pubTopic, MqttQos.exactlyOnce, builder.payload!);

    print('Sleeping.... 60 sec');   /// Ok, we will now sleep a while, in this gap you will see ping request/response messages being exchanged by the keep alive mechanism.

    await MqttUtilities.asyncSleep(60);

    print('Awaked');

    print('Unsubscribing....');

    client.unsubscribe(pubTopic);

    await MqttUtilities.asyncSleep(2); /// Wait for the unsubscribe message from the broker if you wish.

    print('Disconnecting ...');

    client.disconnect();

    print('Stopped! Bye!....');

  }

  void onSubscribed(String topic) {

    print('Subscription confirmed for topic $topic');

  }

  void onDisconnected() {

    print('OnDisconnected client callback - Client disconnection');

    if (client.connectionStatus!.disconnectionOrigin ==

        MqttDisconnectionOrigin.solicited) {

      print('OnDisconnected callback is solicited, this is correct');

    } else {

      print('OnDisconnected callback is unsolicited or none, this is incorrect - exiting');

      exit(-1);

    }

    if (pongCount == 3) {

      print('Pong count is correct');

    } else {

      print('Pong count is incorrect, expected 3. actual $pongCount');

    }

  }

  void onConnected() {

    print('OnConnected client callback - Client connection was successful');

  }

  void pong() {

    print('Ping response client callback invoked');

    \_body = 'Ping response client callback invoked';

    pongCount++;

  }

  Widget build(BuildContext context) {

    return Container(padding: EdgeInsets.all(10.0), child: new Form(key: \_formKey, child: new Column(children: <Widget>[

      new Text('Тестовое поле:', style: TextStyle(fontSize: 20.0),),

      new TextFormField(validator: (value) {

        if (value == null || value.isEmpty)

        {

          return 'Тестовое поле - не заполнено!';

        }

        else

        {

          print('---->'+value);

          \_body = value;

          AAA(value);

        }

      }),

      new SizedBox(height: 20.0),

      //     new RaisedButton(onPressed: (){

      //       if(\_formKey.currentState!.validate()) Scaffold.of(context).showSnackBar(SnackBar(content: Text('Форма заполнена!'+\_body), backgroundColor: Colors.red,));

      //     }, child: Text('Отправить данные'), color: Colors.blue, textColor: Colors.white,),

      ElevatedButton(

        child: Text('Button'),

        onPressed: () {

          if(\_formKey.currentState!.validate()) ScaffoldMessenger.of(context).showSnackBar(SnackBar(content: Text('Форма заполнена!'+\_body), backgroundColor: Colors.red,));

        },

        style: ElevatedButton.styleFrom(

            backgroundColor: Colors.purple,

            padding: EdgeInsets.symmetric(horizontal: 50, vertical: 20),

            textStyle: TextStyle(

                fontSize: 30,

                fontWeight: FontWeight.bold)),

      ),

    ],)));

  }

}

void main() => runApp(

    new MaterialApp(

        debugShowCheckedModeBanner: false,

        home: new Scaffold(

            appBar: new AppBar(title: new Text('IU9 - Форма ввода')),

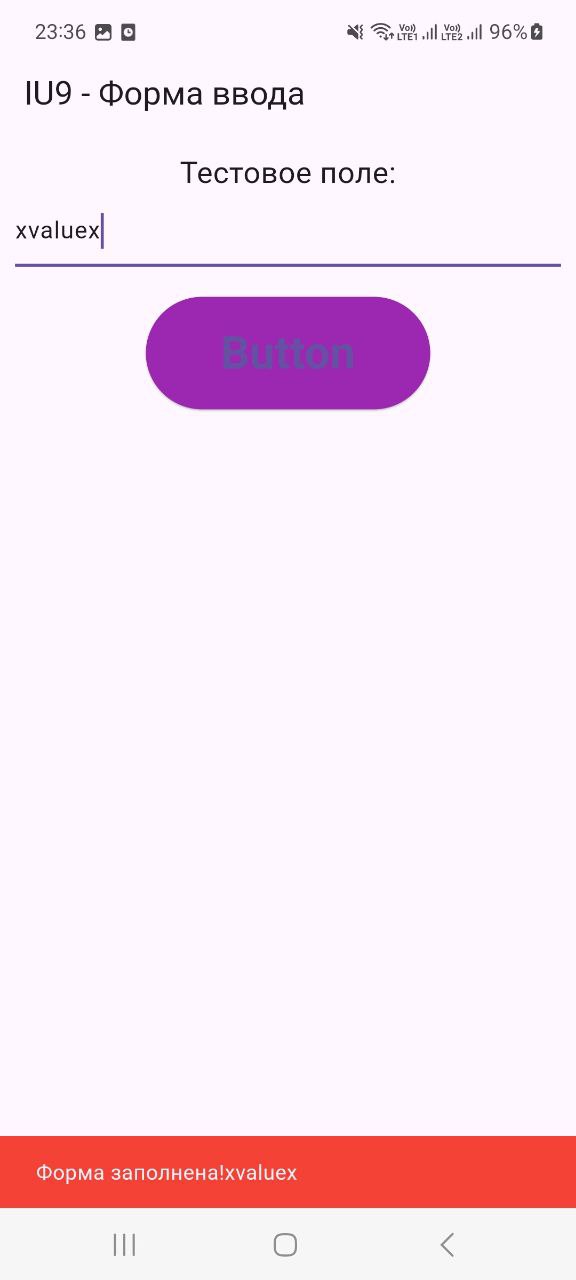
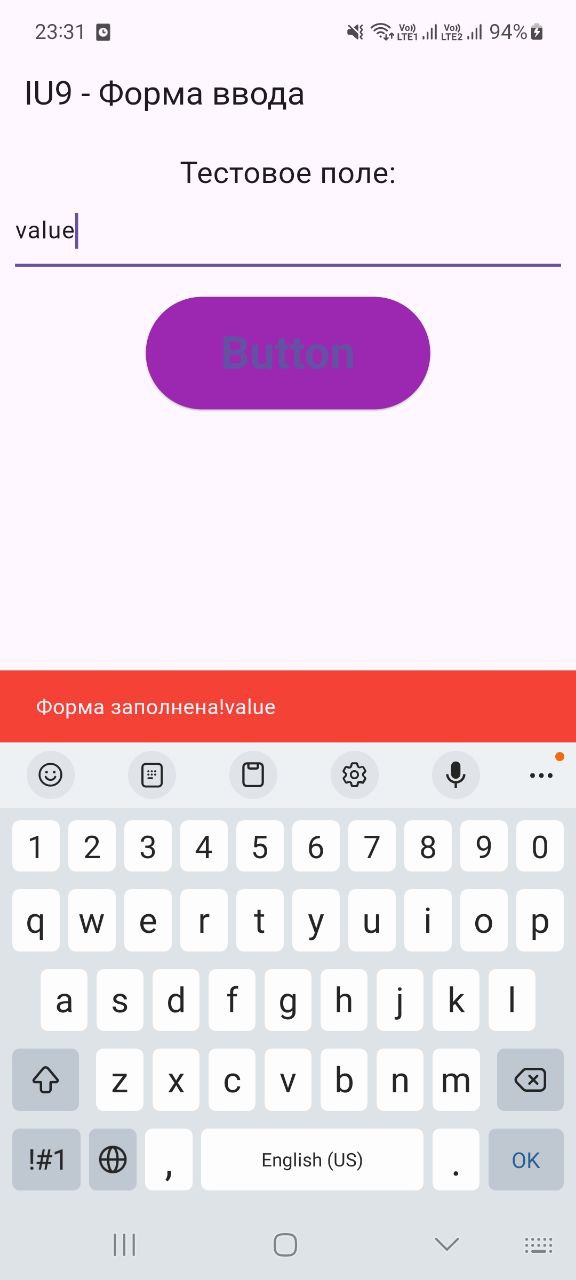
            body: new MyForm()

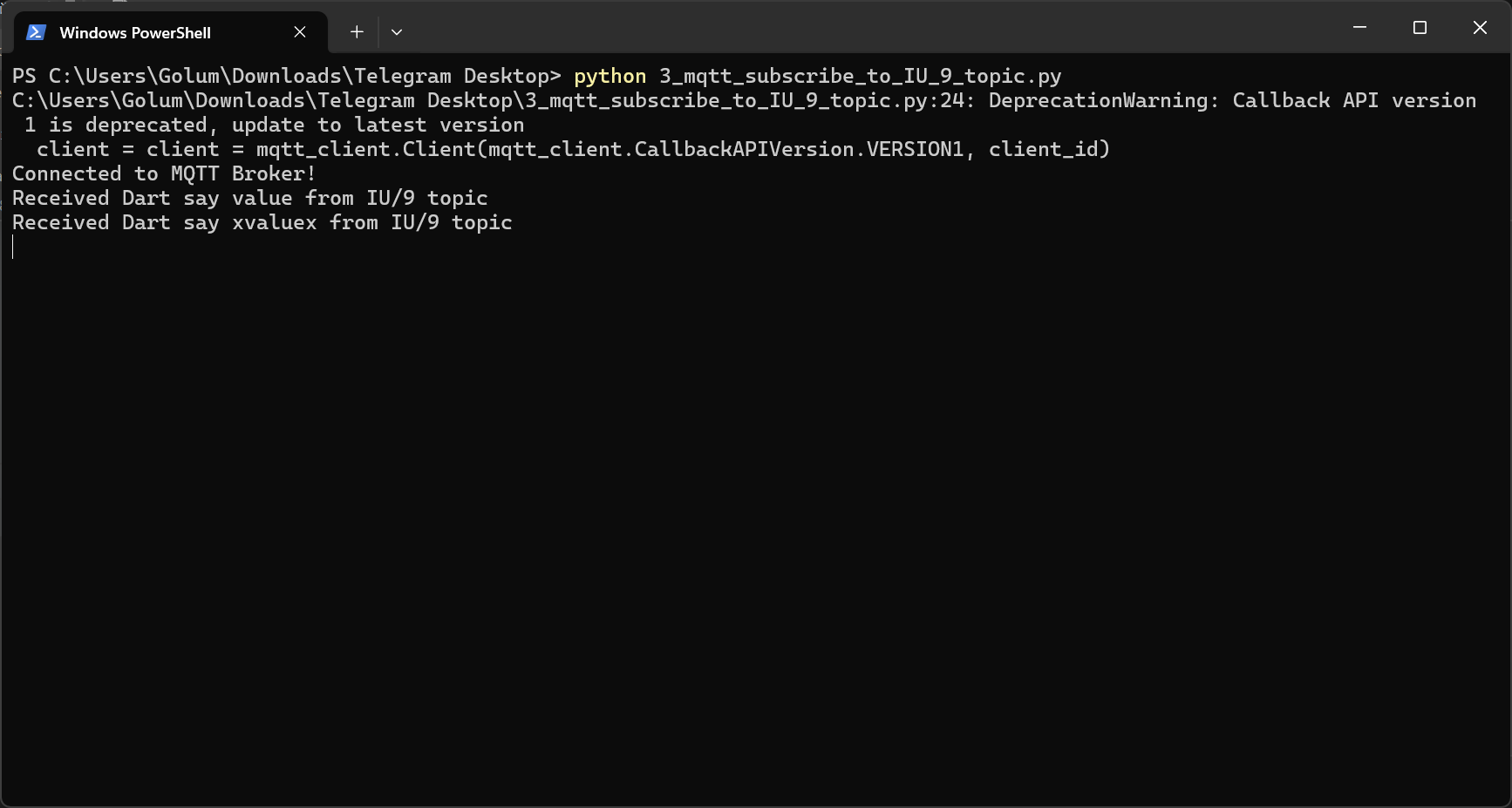
        )

    )

);

## Результаты



. 

## Вывод

В ходе выполнения работы был разработан работающий MQTT-клиент, который успешно выполняет функции подключения к брокеру, публикации и подписки на сообщения. Данная работа обеспечила понимание основ работы с MQTT, а также практический опыт, который может быть применён для создания более сложных приложений, использующих обмен данными в реальном времени.