from tkinter.tix import Tree

from PySide2.QtCore import \*

from PySide2.QtGui import \*

from PySide2.QtWidgets import \*

import os

import sys

from email.mime import message

from typing\_extensions import Self

import paho.mqtt.client as mqtt

from pyowm.owm import OWM

from pyowm.utils.config import get\_default\_config

from datetime import datetime

import threading

import psutil

import time

from ui\_client import Ui\_MainWindow

import pygame

import socket

//главный класс

class MainWindow(QMainWindow):

//инициализация конструктора

    def \_\_init\_\_(self):

        QMainWindow.\_\_init\_\_(self)

Объект класса ui main window

        self.ui = Ui\_MainWindow()

        self.ui.setupUi(self)

//обращение к элементу для закрытия приложения

        self.ui.btn\_close.clicked.connect(self.close)

        self.ui.btn\_ghost.clicked.connect(lambda: self.showMinimized())

        self.ui.lineEdit\_for\_writetext.setPlaceholderText('Ввести текст для mqtt/example1')

        self.ui.textEdit\_for\_view.setPlaceholderText( 'Здесь отображаются сообщения ' + '\n'

      ' \_\_    \_\_     \_\_\_\_\_\_     \_\_\_\_\_\_   \_\_     \n'

      '/\ -./  \   /\  \_\_ \   /\\_\_  \_\ /\\_\_  \_\  \n'

      '\ \ \-./\ \  \ \ \/\\_\  \/\_/\ \/ \/\_/\ \/  \n'

      ' \ \\_\ \ \\_\  \ \\_\_\_\\_\    \ \\_\    \ \\_\  \n'

      '  \/\_/  \/\_/   \/\_\_\_/\_/     \/\_/     \/\_/  \n'

        + '\n')

//обращение к кнопкам для перехода по страницам

        self.ui.pushButton\_page1.clicked.connect(lambda: self.ui.stackedWidget.setCurrentWidget(self.ui.page\_1))

        self.ui.pushButton\_page2.clicked.connect(lambda: self.ui.stackedWidget.setCurrentWidget(self.ui.page\_3))

        self.setWindowFlags(Qt.FramelessWindowHint)

        self.setAttribute(Qt.WA\_TranslucentBackground)

        self.ui.\_old\_pos = None

        self.center()

//обращение к брокеру mosquitto.org

self.client = mqtt.Client()

        self.client.connect("test.mosquitto.org")

        self.start\_take()

        pygame.mixer.init()

        self.playing = False

        self.ui.textEdit\_for\_view.setReadOnly(True)

        self.hdd = psutil.disk\_usage('/')

        self.ui.btn\_for\_sendmessage.clicked.connect(self.start\_send)

//список подписок

        list\_for\_publish\_client\_1 = [

            'mqtt/example1',

            'device/ip',

            'mqtt/picture',

            'mqtt/get\_weather/temp',

            'mqtt/get\_weather/status',

            'android/vibro',

        ]

        self.ui.comboBox\_for\_select\_topic.addItems(list\_for\_publish\_client\_1)

//функции, отключающие панель навигации

    def center(self):

        qr = self.frameGeometry()

        cp = QDesktopWidget().availableGeometry().center()

        qr.moveCenter(cp)

        self.move(qr.topLeft())

    def mousePressEvent(self, event):

        self.oldPos = event.globalPos()

    def mouseMoveEvent(self, event):

        try:

            delta = QPoint(event.globalPos() - self.oldPos)

            self.move(self.x() + delta.x(), self.y() + delta.y())

            self.oldPos = event.globalPos()

        except AttributeError:

            pass

//создание нового процесса, который отправляет сообщения

    def start\_send(self):

        threading.Thread(target=self.public(self.client),daemon=True).start()

    def public(self,client):

        now = datetime.now()

        cur\_time = now.strftime("%H:%M:%S")

        text = self.ui.lineEdit\_for\_writetext.text()

        pubtop = self.ui.comboBox\_for\_select\_topic.currentText()

//отображение информации на клиенте об отправляемой реакции

        if pubtop == 'android/vibro':

            self.ui.textEdit\_for\_view.insertPlainText('['+ cur\_time + '] ' + '<вибрация> ' + '\n')

        if pubtop == 'mqtt/example1':

            self.ui.textEdit\_for\_view.insertPlainText('['+ cur\_time + '] ' +'<' + text + '> ' + '\n')

        if pubtop == 'device/ip':

            h\_name = socket.gethostname()

            IP\_addres = socket.gethostbyname(h\_name)

            self.ui.textEdit\_for\_view.insertPlainText('['+ cur\_time + '] ' + '<' + IP\_addres + '> '+ '\n')

        if pubtop == 'mqtt/picture':

            self.ui.textEdit\_for\_view.insertPlainText('['+ cur\_time + '] ' + '<Picture1.jpg> ' + '\n')

        if pubtop == 'mqtt/get\_weather/temp':

            self.ui.textEdit\_for\_view.insertPlainText('['+ cur\_time + '] ' + '<запрос на температуру> ' + '\n')

        if pubtop == 'mqtt/get\_weather/status':

            self.ui.textEdit\_for\_view.insertPlainText('['+ cur\_time + '] ' + '<запрос на статус погоды> ' + '\n')

        self.client.publish(pubtop,text)

        self.ui.lineEdit\_for\_writetext.clear()

//методы, отвечающие за принятие сообщения с другого клиента

    def start\_track(self):

        pygame.mixer.music.load('Track1.mp3')

        pygame.mixer.music.play()

        self.playing = True

    def stop\_track(self):

        if self.playing:

            pygame.mixer.music.pause()

//новый процесс, который принимает сообщения

    def start\_take(self):

        threading.Thread(target=self.take\_message(self.client),daemon=True).start()

    def restart(self):

        os.system("shutdown /r /t 1")

    def get\_ascii(self):

        self.ui.textEdit\_for\_view.insertPlainText(

' \_\_    \_\_     \_\_\_\_\_\_     \_\_\_\_\_\_   \_\_\_\_ \n'

'/\ "-./  \   /\  \_\_ \   /\\_\_  \_\ /\\_\_  \_\  \n'

'\ \ \-./\ \  \ \ \/\\_\  \/\_/\ \/ \/\_/\ \/  \n'

' \ \\_\ \ \\_\  \ \\_\_\_\\_\    \ \\_\    \ \\_\  \n'

'  \/\_/  \/\_/   \/\_\_\_/\_/     \/\_/     \/\_/  \n'

 + '\n'

+ '\n' + '\n'

 )

    def on\_message(self,client,userdata,message):

        now = datetime.now()

        cur\_time = now.strftime("%H:%M:%S")

//если пришло сообщение, вызываем метод на клиенте

        if message.topic == 'device/memorystatus/harddrive/c':

            self.ui.textEdit\_for\_view.insertPlainText(message.topic + ' ' + str(self.hdd.free / (2\*\*30))+ '\n')

            self.ui.textEdit\_for\_view.setStyleSheet('background-color: red')

            time.sleep(3)

            self.ui.textEdit\_for\_view.setStyleSheet(

            'background-color: rgb(76, 79, 84);'

            'color: white;'

            'border-radius: 1px solid;')

//обработка принятого сообщения

        if message.topic == 'device/work/cpu':

            self.ui.textEdit\_for\_view.insertPlainText('['+ cur\_time + '] ' + message.topic + ' ' + str(psutil.cpu\_percent(interval=1))+ '%' + '\n')

        if message.topic == 'mqtt/text/chat':

            self.ui.textEdit\_for\_view.insertPlainText('['+ cur\_time + '] ' + message.topic + ' ' + message.payload.decode('utf-8') + '\n' )

        if message.topic == 'device/work/ram':

            values = psutil.virtual\_memory().percent

            self.ui.textEdit\_for\_view.insertPlainText('['+ cur\_time + '] ' + message.topic + ' ' + str(values) + '%' + '\n')

        if message.topic == 'music/track1/start':

            self.ui.textEdit\_for\_view.insertPlainText('['+ cur\_time + '] ' + message.topic + '  Track1 начал проигрывание' + '\n')

            self.start\_track()

        if message.topic == 'music/track1/stop':

            self.ui.textEdit\_for\_view.insertPlainText( '['+ cur\_time + '] '+ message.topic + '  Track1 закончил проигрывание' + '\n')

            self.stop\_track()

        if message.topic == 'mqtt/chat/client\_1/android':

            self.ui.textEdit\_for\_view.insertPlainText('['+ cur\_time + '] ' + message.topic + ' ' + message.payload.decode('utf-8') + '\n' )

        if message.topic == 'mqtt/pc/client\_1/restart':

            self.ui.textEdit\_for\_view.insertPlainText('['+ cur\_time + '] ' + message.topic + ' ' + '<Перезагрузка> ' + '\n' )

            self.restart()

        if message.topic == 'mqtt/file/client\_1/get\_text':

            self.ui.textEdit\_for\_view.insertPlainText('['+ cur\_time + '] ' + message.topic + ' ' + '<Получить ascii> ' + '\n' + '\n' )

            self.get\_ascii()

//метод подписки на сообщения

    def take\_message(self,client):

        self.client.subscribe('device/memorystatus/harddrive/c')

        self.client.subscribe('device/work/cpu')

        self.client.subscribe('mqtt/text/chat')

        self.client.subscribe('device/work/ram')

        self.client.subscribe('music/track1/start')

        self.client.subscribe('music/track1/stop')

        self.client.subscribe('mqtt/chat/client\_1/android')

        self.client.subscribe('mqtt/pc/client\_1/restart')

        self.client.subscribe('mqtt/file/client\_1/get\_text')

        self.client.loop\_start()

        client.on\_message = self.on\_message

//конструкция, которая даёт возможность запуска файла

if \_\_name\_\_ == "\_\_main\_\_":

    app = QApplication(sys.argv)

//создание объекта MainWindow и вызов метода Show

    window = MainWindow()

    window.show()

    sys.exit(app.exec\_())