Оглавление

[Установка PostgreSQL (При необходимости) 2](#_Toc151648474)

[Приложение на С# 3](#_Toc151648475)

[Приложение на Python 7](#_Toc151648476)

[Docker 8](#_Toc151648477)

## Установка PostgreSQL (При необходимости)

Дописать

## Приложение на С#

1. Запускаем приложение: AppARM.exe, откроется окно (рис 1).

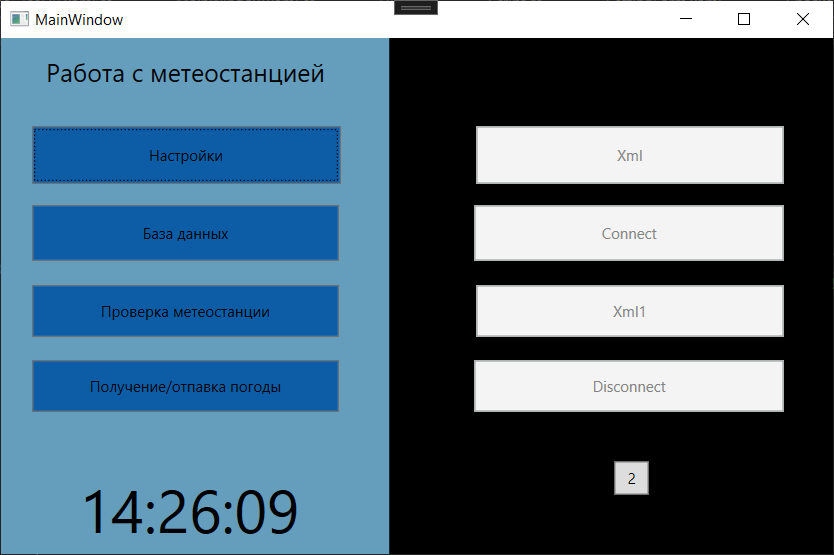


Рис.1 Стартовое окно приложения.

1. Необходимо проверить настройки приложения. Необходимо нажать кнопку **«настройки»**. При необходимости, нужно внести корректировки (рис.2).

**Данные для базы данных:**

**Сервер**: localhost,

**Порт**: 5432,

**Логин**: postgres,

**Пароль**: 111111.

После внесения изменений необходимо нажать кнопку **«OK»**, для сохранения изменений. Кнопка **«Cancel»**, закрывает данное окно.

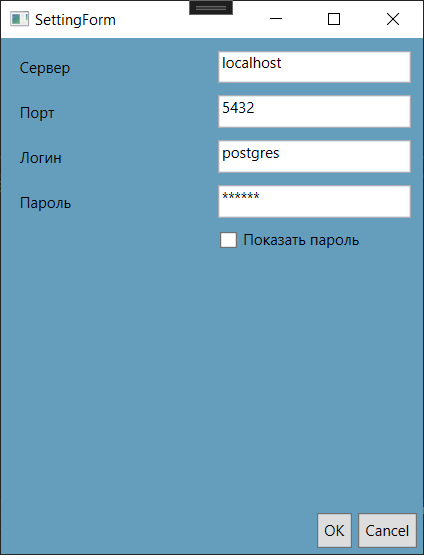
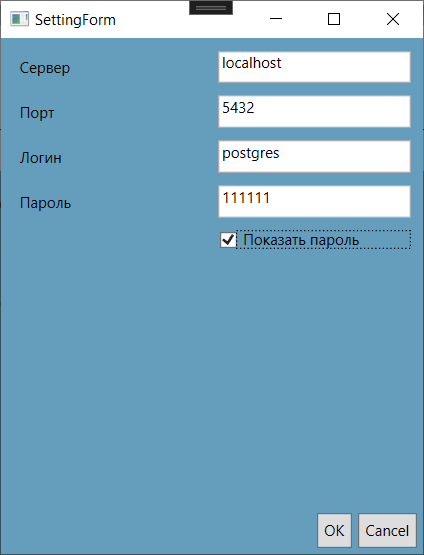
 

Рис.2 Окно настройки.

1. Проверка данных в базе данных

Для этого необходимо на главном окне нажать кнопку **«База данных»**, откроется окно (рис 3).

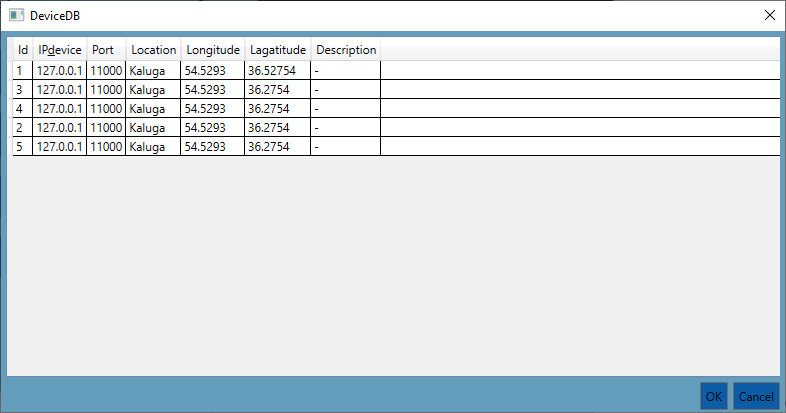


Рис.3 Окно базы данных.

Для добавление новой базы данных, необходимо нажать правой кнопкой по полю. Появится выпадающий список, состоящий из добавления/удаление/вывод. Выбираем вариант **«добавить»** (Рис.4).

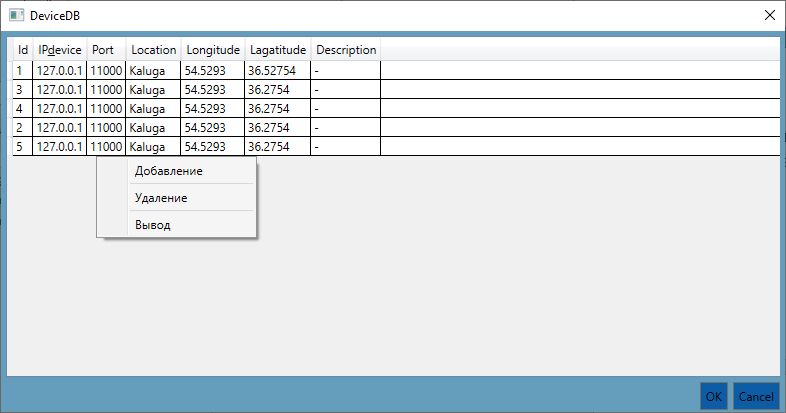


Рис.4 Выпадающий список.

После нажатия **«Добавить»** появится тестовая запись (Рис 5). Для редактирования необходимо выбрать интересующее поле и изменить его. Подвердить изменение можно нажатием кнопки **«Enter»** (Рис 6).

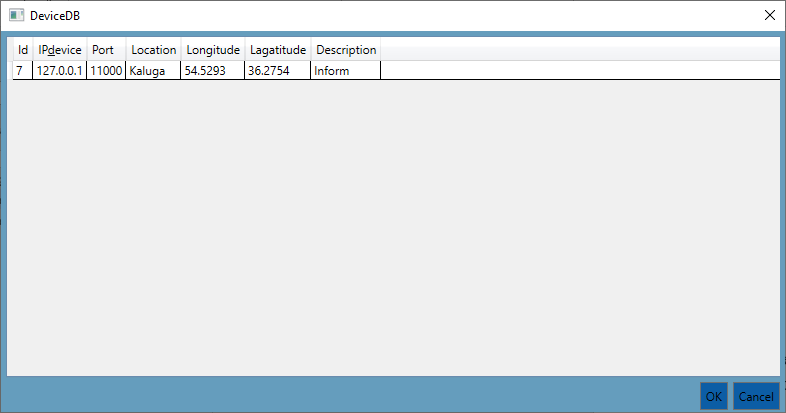


Рис.5 Тестовая запись

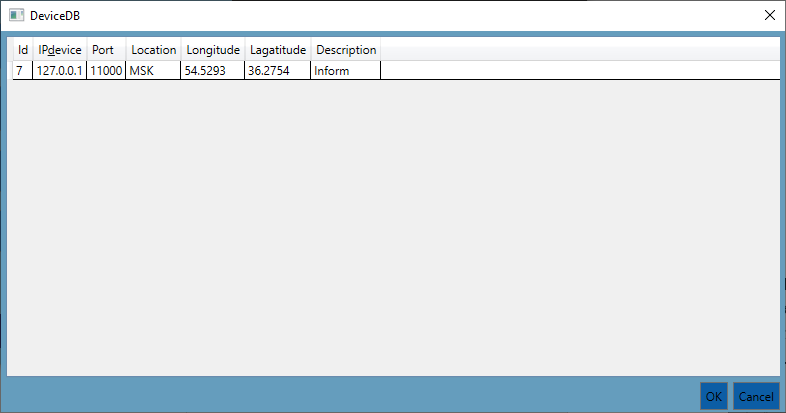


Рис.6 Изменение поля «Location»

После нажатия кнопки **«Enter»**. Появится сообщение об изменении (Рис.7).

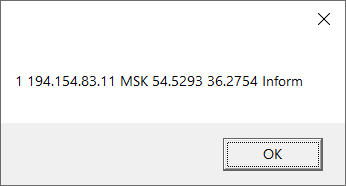


Рис.7 Информация об изменению

Если в случае добавления, произойдет добавление записи с отрицательным **Id**, значит не верные данные подключения к базе данных (Рис.8). Если база данных была заполнена, а при выводе таблица пуста, так же нужно проверить данные подключения к базе данных.

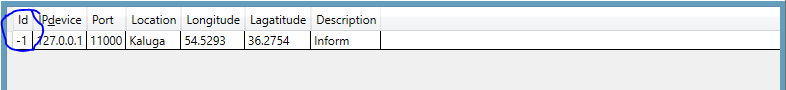


Рис.8 Некорректное добавление

1. При необходимости можно проверить работу каждой метеостанции. Для этого необходимо нажать кнопку «проверка метеостанции» на главном окне (рис 4).

## Приложение на Python (Приемка данных)

*"""Скрипт реализует веб-сервис для отрисовки интерактивной карты с метеостанциями  
map\_url: http://127.0.0.1:8000/map/  
"""*# https://plotly.com/python/filled-area-on-mapbox/  
  
import math  
from dataclasses import dataclass  
from typing import List  
  
import pandas as pd  
import plotly.graph\_objects as go  
  
import uvicorn  
from dash import Dash, dcc, html, Input, Output, State, callback  
from fastapi import FastAPI  
from fastapi.middleware.wsgi import WSGIMiddleware  
  
# Настройки моделя  
UPDATE\_INTERVAL = 2 \* 1000  
#MAP\_TILE\_SERVER\_URL = "http://basemap.nationalmap.gov/arcgis/rest/services/USGSImageryOnly/MapServer/tile/{z}/{y}/{x}"  
MAP\_TILE\_SERVER\_URL = "http://127.0.0.1:8080/tile/{z}/{x}/{y}.png"  
INITIAL\_ZOOM = 3  
#http:/127.0.0.1/tile/5/16/11.png  
#http:localhost:8000/tile/{1}/{1}/{1}.png"  
@dataclass  
class Weather:  
 Temperature: float  
 Wind\_speed: float  
 Direction: str  
 Parameter: str  
  
@dataclass  
class Station:  
 Name: str  
 Device\_ip: str  
 Longitude : float  
 Latitude: float  
 Weather: Weather  
  
"""Начальное значение точек (области), сделано для теста, может быть пустым"""  
station\_list = [  
 Station(  
 Name = 'Калуга',  
 Device\_ip = '127.0.0.1',  
 Longitude = 36.2753,  
 Latitude = 54.5292,  
 Weather = Weather(  
 Temperature = 12.0,  
 Wind\_speed = 3.2,  
 Direction = 'SW',  
 Parameter = None)  
 )  
]  
  
  
def test\_data() -> pd.DataFrame:  
 *"""Подготавливает набор тестовых данных в виде элипса, только для проверки"""* r = 0.5  
 n = 20  
 x\_scale = 3.0  
 x0 = 36.3  
 y0 = 54.5  
 x = [x0 + r \* x\_scale \* math.cos(2\*math.pi\*i/n) for i in range(n)]  
 y = [y0 + r \* math.sin(2\*math.pi\*i/n) for i in range(n)]  
  
 df = pd.DataFrame()  
  
 df.drop(df.index, inplace=True)  
 df['Longitude'] = x  
 df['Latitude'] = y  
 df['Device\_ip'] = '127.0.0.1'  
 df['Name'] = [f'Kaluga-{p}' for p in df.index]  
 df['Temperature'] = 12  
 df['Wind\_speed'] = 3.2  
 df['Direction'] = 'SW'  
 df['Parameter'] = None  
  
 return df  
  
def data() -> pd.DataFrame:  
 *"""Подготавливает набор тестовых данных в виде элипса, только для проверки"""* df = pd.DataFrame()  
 df.drop(df.index, inplace=True)  
  
 df['Longitude'] = [x.Longitude for x in station\_list]  
 df['Latitude'] = [x.Latitude for x in station\_list]  
 df['Device\_ip'] = [x.Device\_ip for x in station\_list]  
 df['Name'] = [x.Name for x in station\_list]  
 df['Temperature'] = [x.Weather.Temperature for x in station\_list]  
 df['Wind\_speed'] = [x.Weather.Wind\_speed for x in station\_list]  
 df['Direction'] = [x.Weather.Direction for x in station\_list]  
 df['Parameter'] = [x.Weather.Parameter for x in station\_list]  
  
 return df  
  
  
def get\_figure():  
 *"""Постороение фигуры (карты)"""* df = data()  
  
 names = df['Name'].to\_list()  
 temps = df['Temperature'].to\_list()  
 winds = df['Wind\_speed'].to\_list()  
 dirs = df['Direction'].to\_list()  
 customdata = [(names[i], temps[i], winds[i], dirs[i]) for i in range(0, len(names))]  
  
 fig = go.Figure()  
  
 fig.add\_trace(  
 go.Scattermapbox(lat=df['Latitude'], lon=df['Longitude'],  
 customdata=customdata,  
 name='',  
 mode="markers",  
 marker=dict(size=20, color="red", opacity=0.5),  
 hovertemplate="<br>".join([  
 "Name: <b>%{customdata[0]}</b>",  
 "Temperature: %{customdata[1]}",  
 "Wind\_speed: %{customdata[2]}",  
 "Direction: %{customdata[3]}"  
 ])  
 )  
 )  
  
 fig.layout.update(  
 title=  
 {  
 'xanchor': 'center',  
 'yanchor': 'top'  
 },  
 showlegend=False,  
 margin= {"r": 0, "t": 0, "l": 0, "b": 0 },  
 mapbox=go.layout.Mapbox(  
 uirevision="foo",  
 style="white-bg",  
 #mapbox\_style="open-street-map",  
 zoom=INITIAL\_ZOOM,  
 center\_lat=station\_list[0].Latitude if len(station\_list) > 0 else 0,  
 center\_lon=station\_list[0].Longitude if len(station\_list) > 0 else 0,  
 layers=[{  
 "below": 'traces',  
 "sourcetype": "raster",  
 "source": [MAP\_TILE\_SERVER\_URL]  
 }]  
 ))  
  
 return fig  
  
  
app\_dash = Dash(requests\_pathname\_prefix='/map/')  
app\_dash.layout = html.Div([  
 dcc.Graph(id='live-update-graph', figure=get\_figure()),  
 dcc.Interval(  
 id='interval-component',  
 interval=UPDATE\_INTERVAL,  
 n\_intervals=0  
 )  
])  
  
@app\_dash.callback(Output('live-update-graph', 'figure'),  
 Input('interval-component', 'n\_intervals'),  
 State('live-update-graph', 'figure'))  
def update\_graph\_live(n\_intervals, fig):  
 df = data()  
 fig['data'][0]['lat'] = df['Latitude'].to\_list()  
 fig['data'][0]['lon'] = df['Longitude'].to\_list()  
 fig['data'][0]['customdata'] = df.apply(  
 lambda x: [x['Name'], x['Temperature'], x['Wind\_speed'], ['Direction']], axis=1).to\_list()  
  
 return fig  
  
app = FastAPI()  
app.mount("/map", WSGIMiddleware(app\_dash.server))  
  
@app.post("/set\_stations/")  
def set\_stations(stations: List[Station]):  
 global station\_list  
 station\_list = stations  
 return 'OK'  
  
  
if \_\_name\_\_ == "\_\_main\_\_":  
 uvicorn.run(app, host="localhost")

## Docker