Содержание отчёта  
Титульный лист

Задание

Создать систему анализа данных, хранящихся в базе. Система должна рассчитывать, как минимум две различных характеристики, например среднее и максимальное значение или применять модель для предсказания значений. Рассчитанные характеристики должны быть использованы для автоматизации какого-либо процесса. Например, среднее значение датчика может быть использовано, как опорное значение, в зависимости от которого может приниматься решение о адекватности получаемых данных.  
Код классов и приложения

App.py

from flask import Flask, render\_template, request

from threading import Timer

import things

import logger

import re

import numpy as np

import matplotlib.pyplot as plt

from sklearn.linear\_model import LinearRegression

logger = logger.Logger('IOT\_Log')

app = Flask(\_\_name\_\_)

sensor\_1 = things.Sensor('wet\_1')

sensor\_2 = things.Sensor('wet\_2')

sensor\_3 = things.Sensor('wet\_3')

sensor\_4 = things.Sensor('wet\_4')

sensor\_5 = things.Sensor('wet\_5')

sensor\_6 = things.Sensor('wet\_6')

sensor\_7 = things.Sensor('wet\_7')

sensor\_8 = things.Sensor('wet\_8')

sensor\_9 = things.Sensor('wet\_9')

sensors1 = (sensor\_5, sensor\_6, sensor\_7, sensor\_8, sensor\_9)

humidifier = things.Humidifier('Main\_humidifier', 25)

sensors2 = (sensor\_1, sensor\_2, sensor\_3, sensor\_4)

humidifier\_2 = things.Humidifier('Bedroom\_humidifier', 25)

def log\_wet():

    logger.insert\_wet(sensor\_1, sensor\_2, sensor\_3,

                              sensor\_4, sensor\_5, sensor\_6,

                              sensor\_7, sensor\_8, sensor\_9)

    Timer(5, log\_wet).start()

log\_wet()

@app.route('/connect\_interface')

def connect\_interface():

    return render\_template('connect\_interface.html')

@app.route('/connect')

def connect():

    cursor = logger.read\_data('wet')

    print(cursor)

    time = []

    avg\_wet = []

    for item in cursor:

        time.append(item['timeStamp'])

        avg\_wet.append(np.average(list(item.values())[2:]))

    # x = np.arange(0, len(time)).reshape((-1, 1))

    x = np.arange(0, len(time))

    y = np.array(avg\_wet)

    plt.plot(x, y)

    plt.xticks(rotation=90)

    plt.ylim(0, 30)

    # model = LinearRegression().fit(x, y)

    # x = np.arange(len(time), len(time) \* 2).reshape((-1, 1))

    # y\_pred = model.predict(x)

    # plt.plot(x, y\_pred)

    # plt.xticks(rotation=90)

    # plt.ylim(0, 30)

    # plt.show()

    x = np.arange(len(time), len(time)\*2)

    y\_pred = np.poly1d(np.polyfit(x, y, 30))

    plt.plot(x, y\_pred(x))

    plt.xticks(rotation=90)

    plt.ylim(0, 30)

    plt.show()

    return {}

@app.route('/')

def start\_page():

    humidifier.humidify(\*sensors1)

    result = {}

    for sensor in sensors1:

        result[sensor.name] = sensor.value

    return result

@app.route('/bedroom')

def bedroom\_page():

    humidifier\_2.humidify(\*sensors2)

    result = {}

    for sensor in sensors2:

        result[sensor.name] = sensor.value

    return result

if \_\_name\_\_ == '\_\_main\_\_':

    app.run()

Things.py

import abc

class Thing(abc.ABC):

    @abc.abstractmethod

    def \_\_init\_\_(self, name):

        self.name = name

        print(f"Create thing {self.name}")

    @abc.abstractmethod

    def print\_name(self):

        print(f'name this device is {self.name}')

class Sensor(Thing):

    def \_\_init\_\_(self, name):

        super().\_\_init\_\_(name)

        self.value = 20

        print(f"Create new device {self.name}")

    def print\_name(self):

        super().print\_name()

class Humidifier(Thing):

    def \_\_init\_\_(self, name, focus\_wet):

        super().\_\_init\_\_(name)

        self.focus\_wet = focus\_wet

    def print\_name(self):

        super().print\_name()

    def humidify(self, \*sensors):

        for sensor in sensors:

            if sensor.value < self.focus\_wet:

                sensor.value += 1

            else:

                sensor.value -= 1

Logger.py

import pymongo

import datetime

class Logger:

    def \_\_init\_\_(self, db\_name):

        self.client = pymongo.MongoClient('mongodb://localhost:27017/')

        self.db = self.client[db\_name]

    def insert\_wet(self, \*sensors):

        result = {'timeStamp': datetime.datetime.now().strftime("%Y-%m-%d %H:%M:%S")}

        for sensor in sensors:

            result[sensor.name] = sensor.value

        return self.db['wet'].insert\_one(result)

    def read\_data(self, collection, value={}, field={}):

        return self.db[collection].find(value, field)

Script.js

function send\_data() {

    $.ajax({

        type: 'GET', //тип запроса

        url: '/connect', // адрес, на который отправлен запрос

        dataType: 'json', //тип данных, ожидаемый от сервера

        conectType:'application/json', //тип передаваемых данных

        data:{ //данные запроса

            "value": document.getElementById("value").value,

            "name": document.getElementById("name").value,

            "check": Number(document.getElementById("check").checked)

        },

        success: function (response) {

            console.log(response)

        }

    });

}

Index.html

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <title>Smart home</title>

    <link rel="stylesheet" type="text/css" href="../static/css/main.css">

    <script src="http://code.jquery.com/jquery-3.3.1.js"></script>

    <script src="../static/js/scripts.js"></script>

</head>

<body>

    <div class="text-field">

      <label class="text-field\_\_label" for="name">Name sensor</label>

      <input class="text-field\_\_input" type="text" id="name" placeholder="Type sensor">

    </div>

    <div class="text-field">

      <label class="text-field\_\_label" for="name">Value</label>

      <input class="text-field\_\_input" type="text" id="value" placeholder="Type value">

    </div>

    <label class="switch">

        <input type="checkbox" id="check">

        <span class="slider round"></span>

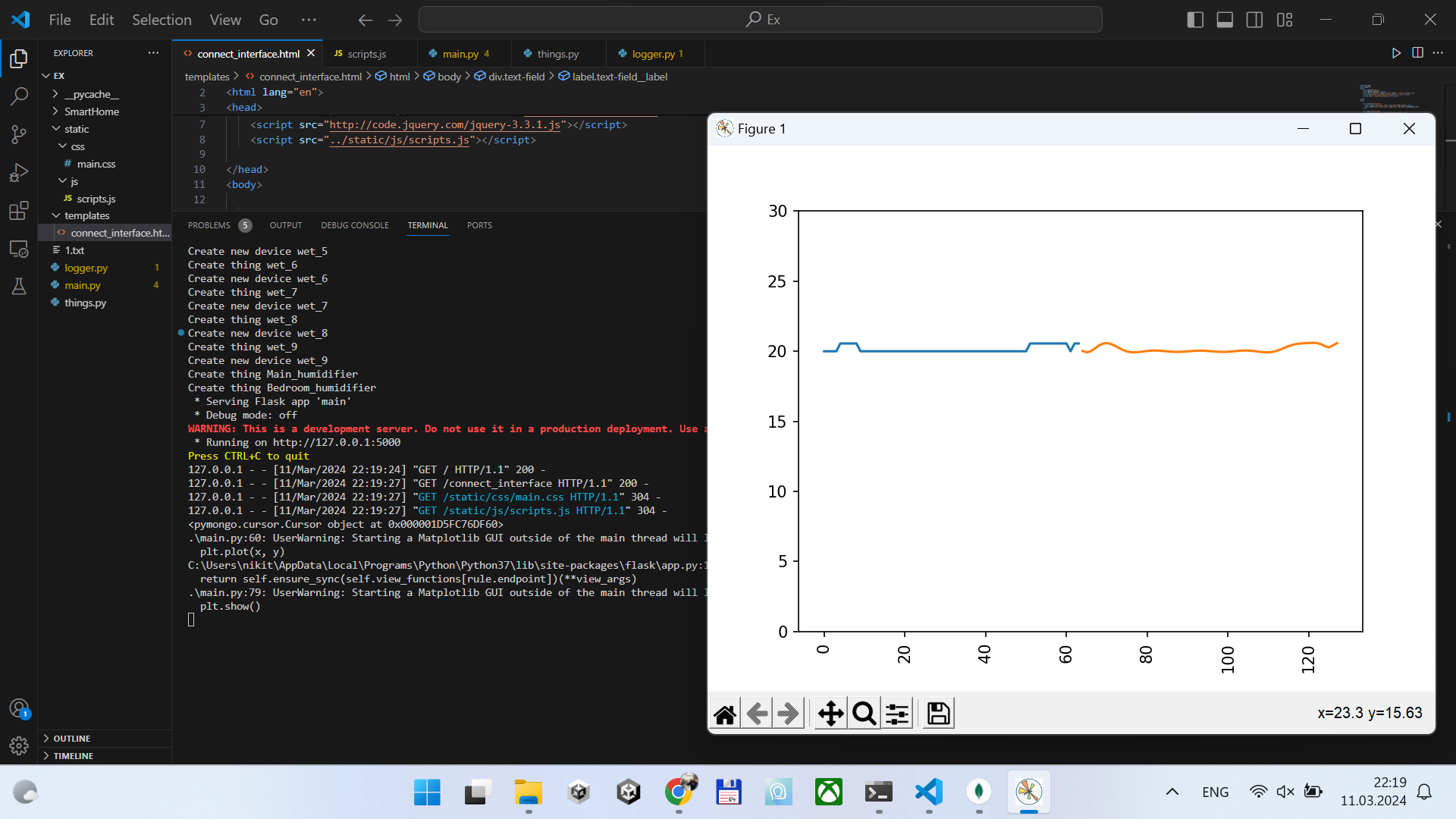
    </label>

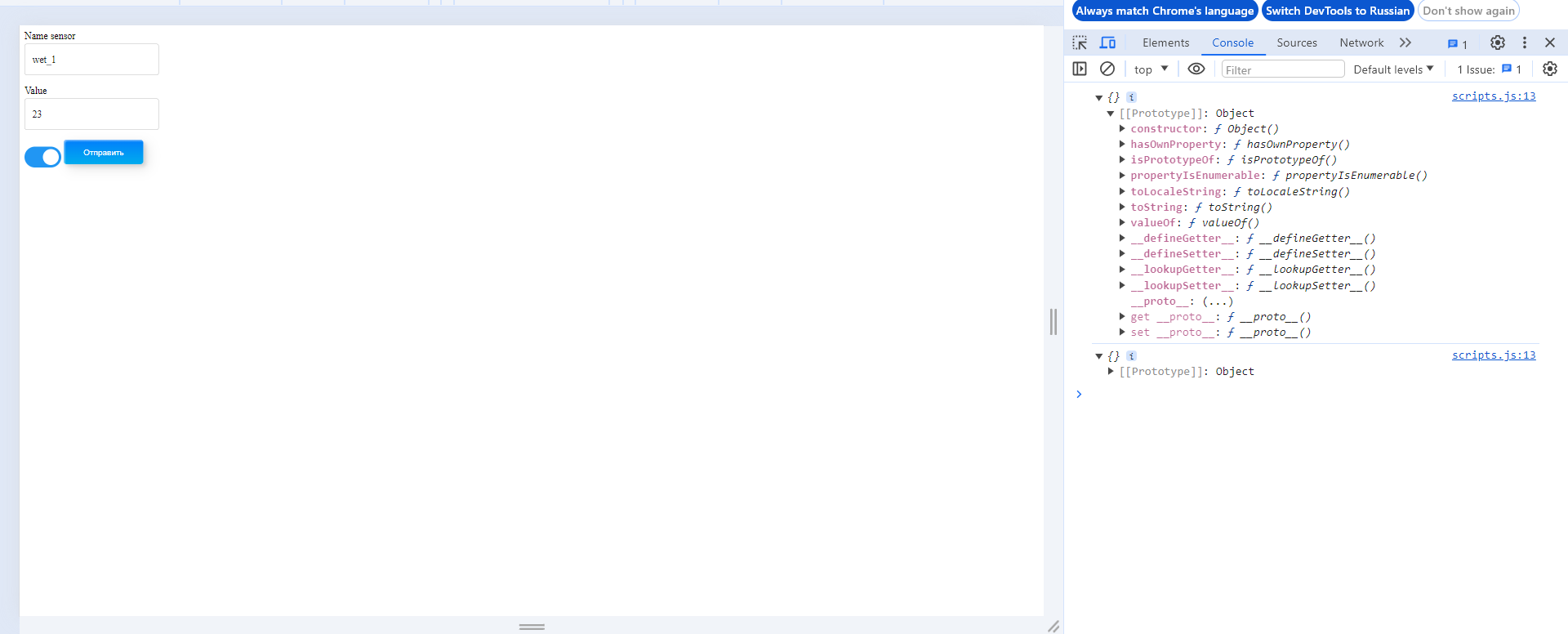
    <button class="custom-btn btn-3" onclick="send\_data()"><span>Отправить</span></button>

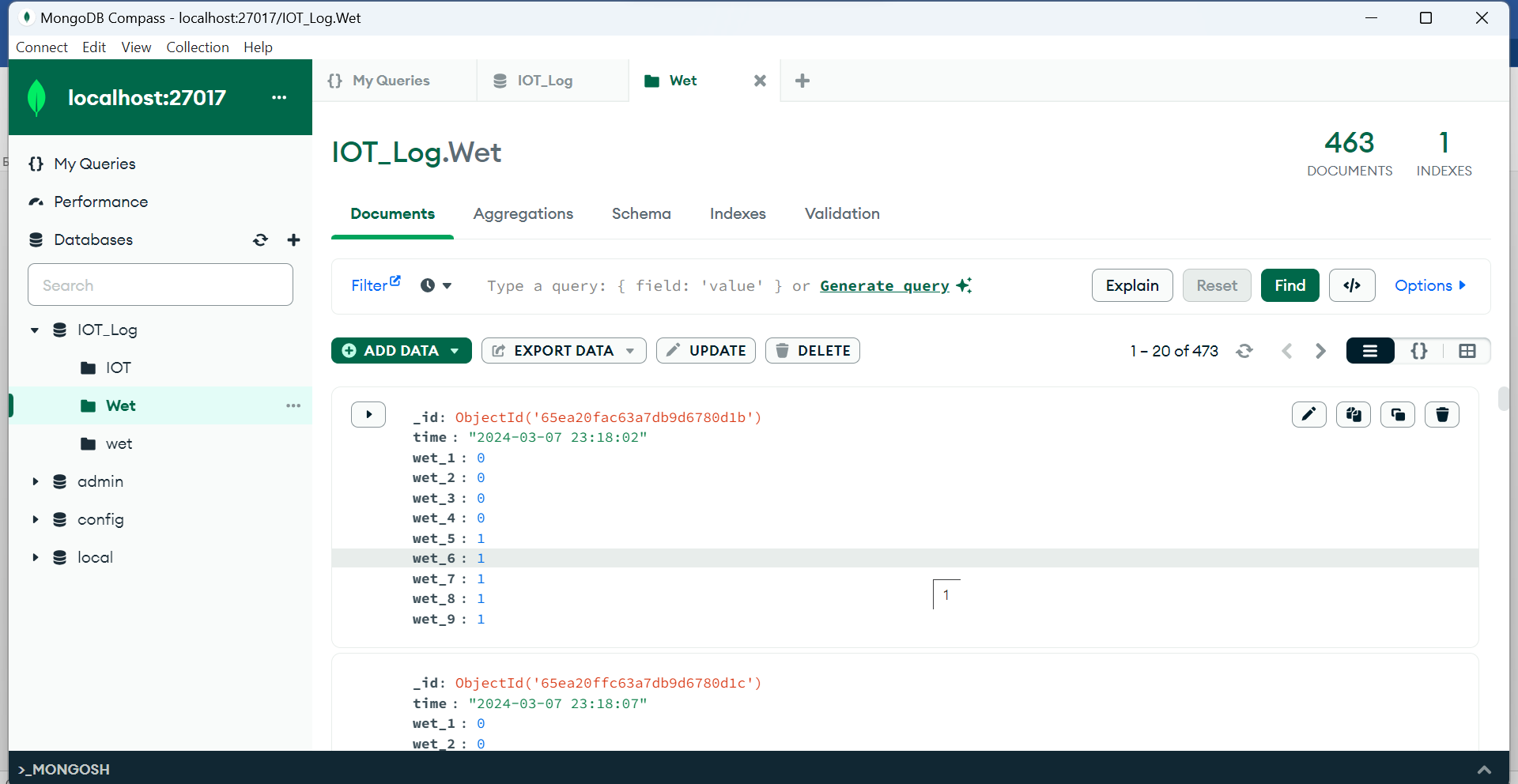
</body>

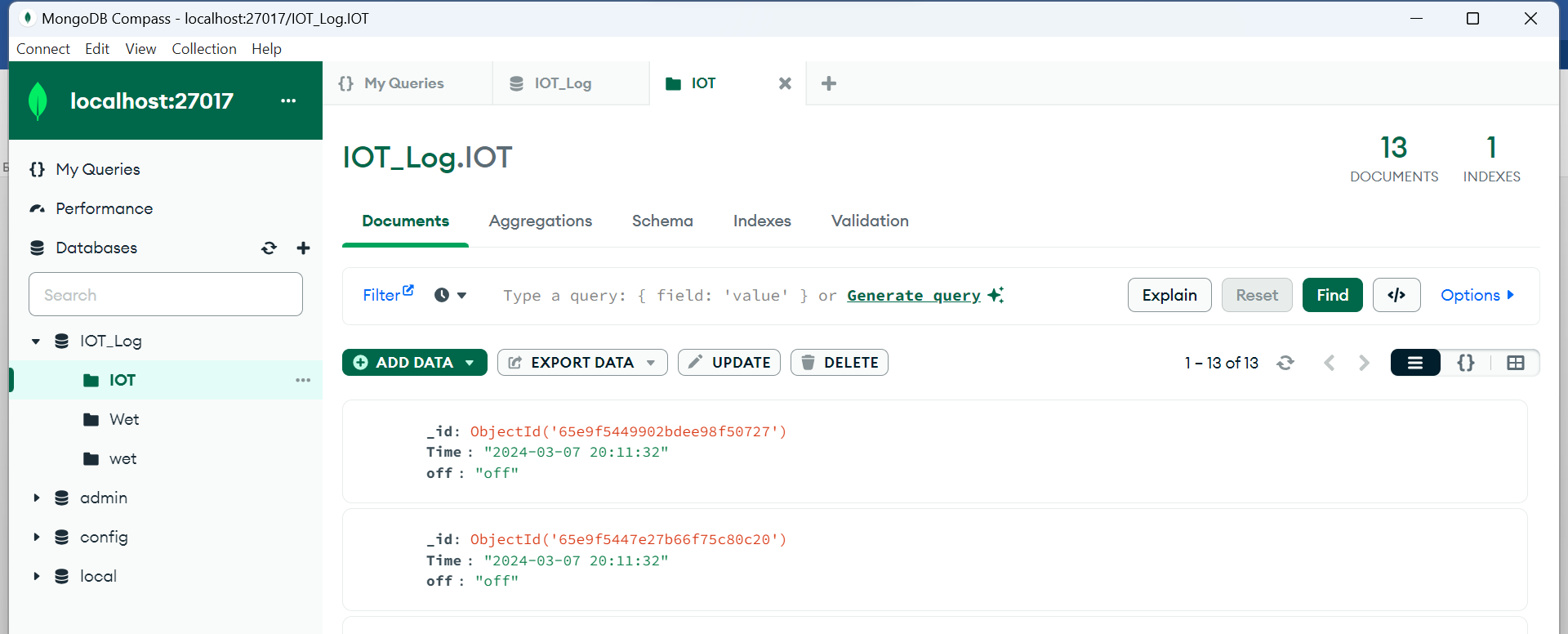
</html>

Скриншоты









Вывод

В ходе выполнения лабораторной работы мной была создана система анализа данных, способная рассчитывать различные характеристики, такие как среднее и максимальное значение, а также применять модели для предсказания значений. Эти характеристики в дальнейшем могут быть использованы для автоматизации процессов, что значительно упрощает принятие решений на основе данных. Разработанная система открывает новые возможности для оптимизации работы и улучшения эффективности процессов, требующих анализа данных. Созданная система анализа данных также может быть интегрирована в умный дом для отслеживания уровня влажности в помещении. Данная функциональность позволит автоматически контролировать и реагировать на изменения влажности, что является важным аспектом для поддержания комфортных условий проживания. Благодаря логированию данных о влажности, пользователь сможет в любой момент получить информацию о динамике изменения этого параметра в помещении. Такой подход позволит не только контролировать микроклимат в доме, но также предотвращать возможные проблемы, связанные с избыточной или недостаточной влажностью. В итоге, интеграция системы анализа данных в умный дом обеспечит еще больший уровень удобства и безопасности для пользователей.