

**ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«МОСКОВСКИЙ АВИАЦИОННЫЙ ИНСТИТУТ (НАЦИОНАЛЬНЫЙ
ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ)»**

Отчет по учебно-производственной практике

Обучающийся Журавков Владислав Максимович (Ф.И.О.)

Институт №3 «Системы управления, информатика и электроэнергетика»

Кафедра 307 «Цифровые технологии и информационные системы»

Учебная группа МЗО-312Б-19

Направление подготовки (специальность) 09.03.02
(шифр)

«Информационные системы и технологии»
(название направления, специальности)

Вид практики учебно-производственная практика
(учебной, производственной, преддипломной или другой вид практики)

Руководитель практики от МАИ

Скле́ймин Ю́рий Бори́сович _____
(фамилия, имя, отчество) (подпись)

_____/ Журавков В.М. _____ / «26» июля 2022 г.
(подпись обучающегося) (фамилия, инициалы) (дата)

1. Место и сроки проведения практики

Сроки проведения практики:

-дата начала практики 29 июня 2022г.
-дата окончания практики 26 июля 2022г.

Наименование предприятия ФГБОУ ВО МАИ (НИУ)
Название структурного подразделения (отдел, лаборатория) Кафедра 307 «Цифровые технологии и информационные системы»

ЦЕЛИ ПРОХОЖДЕНИЯ ПРАКТИКИ

Содержание практики охватывает круг вопросов, связанных с: получением, хранением, преобразованием, передачей и использованием информации, созданием информационных моделей предприятия и предметной области, созданием макетов программных модулей и разработки программ их реализующих, оценкой технико-экономических параметров создаваемых программных модулей.

Целью учебно-производственной практики является достижение следующих результатов освоения (РО):

№	Шифр	Результат освоения
1	В-6 (ОК-10)	Владеть основными навыками письма, необходимыми для подготовки публикаций, тезисов и ведения переписки
2	У-28 (ОПК-9)	Уметь работать в качестве пользователя персонального компьютера, использовать внешние носители информации для обмена данными между машинами, создавать резервные копии архивов данных и программ, использовать языки и системы программирования для решения профессиональных задач, работать с программными средствами и пакетами общего назначения
3	В-44 (ПК-41)	Владеть навыками применения инструментальных средств при эксплуатации и проектировании информационных систем различного назначения;
4	В-45 (ПК-42)	Владеть навыками построения моделей представления знаний, подходами и техникой решения задач искусственного интеллекта
5	З-62 (ПКС-1)	Знать теоретические и экспериментальные способы и методы оценки надежности, основные и эксплуатационные характеристики надежности, способы и методы их обеспечения в процессе проектирования, изготовления и эксплуатации информационных систем

6	3-72 (ПКС-9)	Знать функциональную и структурную организации информационных систем аэрокосмических комплексов, ее основные блоки и элементы;
7	3-141 (ПКС-8)	Знать информационные закономерности, специфику информационных объектов и ресурсов, информационных потребностей в предметной области.
8	У-115 (ПКС-11)	Уметь проводить выбор интерфейсных средств при построении сложных профессионально-ориентированных информационных систем.
9	У-116 (ПКС-11)	Уметь ставить и решать задачи, связанные с организацией диалога между человеком и информационной системой.
10	У-117 (ПКС-11)	Уметь формулировать основные технико-экономические требования к проектируемым профессионально-ориентированным информационным системам.
11	У-118 (ПКС-11)	Уметь создавать профессионально-ориентированные информационные системы.
12	В-61 (ПКС-17)	Владеть навыками разработки прикладных Web-приложений, использования инструментальных средств разработки фреймворков и др.;

Перечисленные РО являются основой для формирования следующих компетенций:

№	Шифр	Компетенция
1	ОК-10	Способность логически верно, аргументированно и ясно строить устную и письменную речь на русском языке
2	ОПК-9	Готовность применять основы информатики и программирования для решения типовых профессиональных задач;
3	ПКС-1	Способность оценивать надежность и качество функционирования объекта проектирования
4	ПКС-8	Способность разрабатывать информационно-логическую, функциональную и объектно-ориентированную модели информационной системы, модели данных информационных систем .
5	ПКС-9	Готовность осуществлять организацию сбора, коммутации, формирования групповых сигналов и контроль качества входных данных при проектировании ИС АКК.
6	ПКС-11	Готовность участвовать в разработке и эксплуатации систем наблюдения и мониторинга с использованием технологий дистанционного зондирования Земли (ДЗЗ).
7	ПКС-17	Способность создавать Web-интерфейс для информационных систем
8	ПК-41	Способность использовать инструментальные средства информационных систем в своей профессиональной деятельности
9	ПК-42	Способность решать прикладные вопросы интеллектуальных систем и владеть подходами и техникой решения задач искусственного интеллекта

3. Индивидуальное задание обучающемуся

Изучение особенностей работы с time-series базами данных на примере InfluxDB

4. План выполнения индивидуального задания

№ п/п	Наименование этапов выполнения работы	Срок выполнения этапов работы
1.	Проведение инструктажа по технике безопасности	29.06
2.	Общее ознакомление с понятием time-series баз данных	30.06
3.	Сбор информации об организации ФГБОУ ВО МАИ, изучение организационно-управленческой структуры, подчиненности, содержания устава (положения о подразделении) организации	1.07-3.07
4.	Исследование особенностей работы с InfluxDB, изучение языка скриптов Flux	4.07-8.07
5.	Выполнение обзора научной литературы и электронных информационно-образовательных ресурсов	9.07-12.07
6.	Подготовка среды для работы с InfluxDB	13.07
7.	Проектирование программ мониторинга ресурсов системы, погоды и курса валют в реальном времени	14.07-17.07
8.	Интеграция InfluxDB с Grafana	18.07-19.07
9.	Анализ и обобщение результатов работ по проектированию программы мониторинга	20.07-22.07
10.	Подготовка презентации программы мониторинга	23.07-24.07
11.	Написание индивидуального отчета о прохождении практики	25.07-26.07

5. Содержательное описание работы

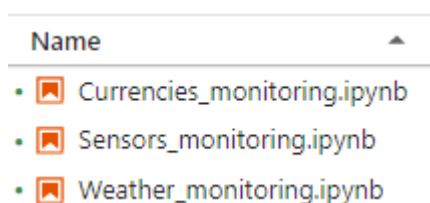
Разработка приложений для мониторинга ресурсов системы, погоды и курса валют в реальном времени

(Разработка приложения производится на ОС Windows 11).

1) Создание проекта

Для удобства будем использовать широко известную IDE JupyterLab.

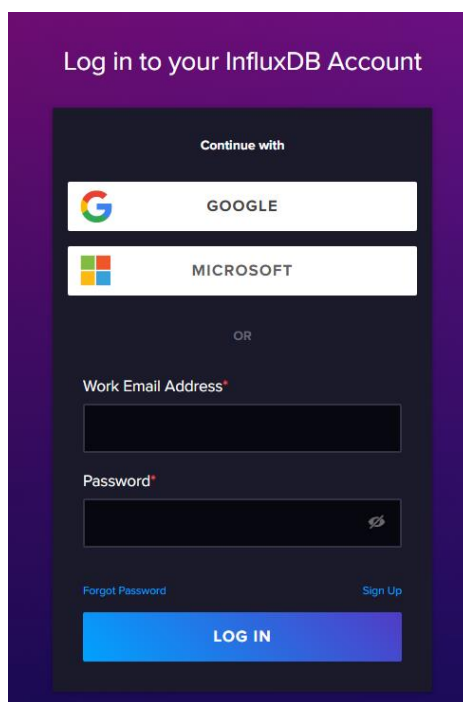
Создадим блокноты JupyterNotebook для каждого из приложений



2) Создание базы данных InfluxDB в официальном облачном сервисе

Проходим регистрацию на официальном сайте облачного сервиса

<https://cloud2.influxdata.com/>



Install Dependencies

First, you need to install the `influxdb-client` module. Run the command below in your terminal.

```
pip3 install influxdb-client
```

COPY TO CLIPBOARD

Create Bucket

X

Name*

Practice

✓

Data Retention Preferences

NEVER DELETE

DELETE OLDER THAN

CANCEL

CREATE

```
[1]: from influxdb_client import InfluxDBClient, Point  
from influxdb_client.client.write_api import SYNCHRONOUS
```

```
[2]: token = "XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX"  
org = "XXXXXXXXXX"  
url = "https://eu-central-1-1.aws.cloud2.influxdata.com"
```

```
client = InfluxDBClient(url=url, token=token, org=org)
```

3) Создание приложения для мониторинга погоды в реальном времени с помощью OpenWeatherMap API.

OpenWeatherMap – сервис, предоставляющий API для для получения разнообразных данных о погоде, такими как:

- Интерактивная карта с данными о текущей погоде
- Прогноз на неделю в городе
- Исторические данные о погоде в 120 000 городах мира.
- Данные от 40 000 метеостанций по всему миру получаемые практически в режиме online. (Задержка от секунд до часа)
- Много различных web карт, включая карты облаков, осадков, ветра, температуры и т.п.

Проходим простую регистрацию, оформление подписки OneCall API 3.0 (<https://openweathermap.org/api>) и используя полученный API ключ, получаем текущую температуру в выбранном городе.

```
r = requests.get("https://api.openweathermap.org/data/2.5/weather?q=Moscow,RU&units=metric&appid=XXXXXXXXXXXXXXXXXXXX")
weather = r.json()
current_temperature = weather['main']['temp']
```

Библиотека requests позволяет совершать HTTP запросы к веб серверам. В данном случае мы совершаем GET запрос к выше описанному API и полученный ответ представляем в Json формате, из которого и получаем текущую температуру.

Затем, записываем полученную температуру в базу данных InfluxDB

```
bucket="Practice"

write_api = client.write_api(write_options=SYNCHRONOUS)
point = Point("Weather").tag("Location", "Moscow").field("Temperature", current_temperature)
write_api.write(bucket=bucket, org="XXXXXXXXXXXX", record=point)
```

Мы создаём экземпляр клиента записи, используя объект клиента и метод write_api.

Создаём объект point, имеющий вид:

Point(“измерение”).tag(“название тега”, “значение тега”).field(“название поля для записи”, “значение”)

Измерение - часть структуры InfluxDB, которая описывает данные, хранящиеся в связанных полях. Измерения - это строки.

Тэг - пара ключ-значение в структуре данных InfluxDB, которая записывает метаданные. Теги являются необязательной частью структуры данных InfluxDB, но они полезны для хранения часто запрашиваемых метаданных

Поле - пара ключ-значение в структуре данных InfluxDB, которая записывает метаданные и фактическое значение данных. Поля обязательны для заполнения в структуре данных InfluxDB, и они не индексируются - запросы к значениям полей сканируют все точки, соответствующие указанному временному диапазону, и, как следствие, не являются производительными по отношению к тегам.

Записываем полученный объект в InfluxDB, используя метод `write` объекта API `writer`.

Исходный код:

```
[1]: import requests  
import time  
  
from influxdb_client import InfluxDBClient, Point  
from influxdb_client.client.write_api import SYNCHRONOUS
```

```
[2]: token = "eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.e30.fg8tLH6TqkZCQWwE7XvR-  
org = "my-org"  
url = "https://eu-central-1-1.aws.cloud2.influxdata.com"
```

```
client = InfluxDBClient(url=url, token=token, org=org)
```

```
[*]: while True:  
    r = requests.get("https://api.openweathermap.org/data/2.5/weather?q=Moscow,RU&units=meteric&appid=  
weather = r.json()  
current_temperature = weather['main']['temp']  
  
bucket="Practice"  
  
write_api = client.write_api(write_options=SYNCHRONOUS)  
point = Point("Weather").tag("Location", "Moscow").field("Temperature", current_temperature)  
write_api.write(bucket=bucket, org="my-org", record=point)  
time.sleep(300)
```

4) Создание приложения для мониторинга ресурсов компьютера в реальном времени

Для получения данных о ресурсах компьютера используется библиотека `psutil`.

```
cpu_usage = psutil.cpu_percent()
ram_usage = psutil.virtual_memory().percent
disk_usage = psutil.disk_usage("C:\\").percent
battery_charge = psutil.sensors_battery().percent
```

В данном фрагменте кода реализовано получение текущей нагрузки на процессор, оперативную память в процентном соотношении, а также степень загрузки диска и заряда батареи.

Аналогичным с предыдущим пунктом методом, производим запись полученных данных в базу данных InfluxDB

```
cpu_point = Point("Monitoring").tag("CPU", "cpu_total").field("cpu_usage", cpu_usage)
ram_point = Point("Monitoring").tag("RAM", "ram_total").field("ram_usage", ram_usage)
disk_point = Point("Monitoring").tag("Disk", "disk_total").field("disk_usage", disk_usage)
battery_point = Point("Monitoring").tag("Battery", "battery_total").field("battery_charge", battery_charge)

write_api = client.write_api(write_options=SYNCHRONOUS)
write_api.write(bucket=bucket, org="XXXXXXXXXXXXXXXXXXXX", record=cpu_point)
write_api.write(bucket=bucket, org="XXXXXXXXXXXXXXXXXXXX", record=ram_point)
write_api.write(bucket=bucket, org="XXXXXXXXXXXXXXXXXXXX", record=disk_point)
write_api.write(bucket=bucket, org="XXXXXXXXXXXXXXXXXXXX", record=battery_point)
```

Исходный код:

```
[1]: import psutil
import time

from influxdb_client import InfluxDBClient, Point
from influxdb_client.client.write_api import SYNCHRONOUS


[2]: token = "XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX"
org = "XXXXXXXXXX"
url = "https://eu-central-1-1.aws.cloud2.influxdata.com"

client = InfluxDBClient(url=url, token=token, org=org)


[ ]: bucket="Monitoring"

while True:
    cpu_usage = psutil.cpu_percent()
    ram_usage = psutil.virtual_memory().percent
    disk_usage = psutil.disk_usage("C:\\").percent
    battery_charge = psutil.sensors_battery().percent

    cpu_point = Point("Monitoring").tag("CPU", "cpu_total").field("cpu_usage", cpu_usage)
    ram_point = Point("Monitoring").tag("RAM", "ram_total").field("ram_usage", ram_usage)
    disk_point = Point("Monitoring").tag("Disk", "disk_total").field("disk_usage", disk_usage)
    battery_point = Point("Monitoring").tag("Battery", "battery_total").field("battery_charge", battery_charge)

    write_api = client.write_api(write_options=SYNCHRONOUS)
    write_api.write(bucket=bucket, org="XXXXXXXXXX", record=cpu_point)
    write_api.write(bucket=bucket, org="XXXXXXXXXX", record=ram_point)
    write_api.write(bucket=bucket, org="XXXXXXXXXX", record=disk_point)
    write_api.write(bucket=bucket, org="XXXXXXXXXX", record=battery_point)

    time.sleep(5)
```

5) Создание приложения для мониторинга курса валют в реальном времени с помощью Currency Data API

Currency Data API предоставляет простой REST API с обменными курсами в реальном времени и историческими курсами для 168 мировых валют, предоставляя валютные пары в универсальном формате JSON.

Оформляем бесплатную подписку на Currency Data API (https://apilayer.com/marketplace/currency_data-api#pricing)

```

url = "https://api.apilayer.com/currency_data/live?source=RUB&currencies=USD%2C%20EUR%2C%20BTC"

payload = {}
headers = {
    "apikey": "████████████████████████████████████████"
}

response = requests.request("GET", url, headers=headers, data = payload)

status_code = response.status_code
result = response.json()
USD_RUB = 1/float(result["quotes"]["RUBUSD"])
EUR_RUB = 1/float(result["quotes"]["RUBEUR"])
BTC_RUB = 1/float(result["quotes"]["RUBBTC"])

```

С помощью полученного API ключа, делаем GET запрос, после чего представляем полученный ответ в Json формате, из которого и получаем текущие курсы доллара, евро и биткойна.

Производим запись полученных данных в базу данных InfluxDB

```

write_api = client.write_api(write_options=SYNCHRONOUS)
usd_point = Point("Currency").tag("Currency", "USD_RUB").field("Currency", USD_RUB)
eur_point = Point("Currency").tag("Currency", "EUR_RUB").field("Currency", EUR_RUB)
btc_point = Point("Currency").tag("Currency", "BTC_RUB").field("Currency", BTC_RUB)

bucket = "Practice"
write_api.write(bucket=bucket, org="████████████████████████████████████████", record=usd_point)
write_api.write(bucket=bucket, org="████████████████████████████████████████", record=eur_point)
write_api.write(bucket=bucket, org="████████████████████████████████████████", record=btc_point)

```

Исходный код:

```

[1]: import requests
import time

from influxdb_client import InfluxDBClient, Point
from influxdb_client.client.write_api import SYNCHRONOUS

[2]: token = "████████████████████████████████████████████████████████████████████████████████"
org = "████████████████████"
url = "https://eu-central-1-1.aws.cloud2.influxdata.com"

client = InfluxDBClient(url=url, token=token, org=org)

[ ]: while True:
    url = "https://api.apilayer.com/currency_data/live?source=RUB&currencies=USD%2C%20EUR%2C%20BTC"

    payload = {}
    headers = {
        "apikey": "████████████████████████████████████████"
    }

    response = requests.request("GET", url, headers=headers, data = payload)

    status_code = response.status_code
    result = response.json()
    USD_RUB = 1/float(result["quotes"]["RUBUSD"])
    EUR_RUB = 1/float(result["quotes"]["RUBEUR"])
    BTC_RUB = 1/float(result["quotes"]["RUBBTC"])

    write_api = client.write_api(write_options=SYNCHRONOUS)
    usd_point = Point("Currency").tag("Currency", "USD_RUB").field("Currency", USD_RUB)
    eur_point = Point("Currency").tag("Currency", "EUR_RUB").field("Currency", EUR_RUB)
    btc_point = Point("Currency").tag("Currency", "BTC_RUB").field("Currency", BTC_RUB)

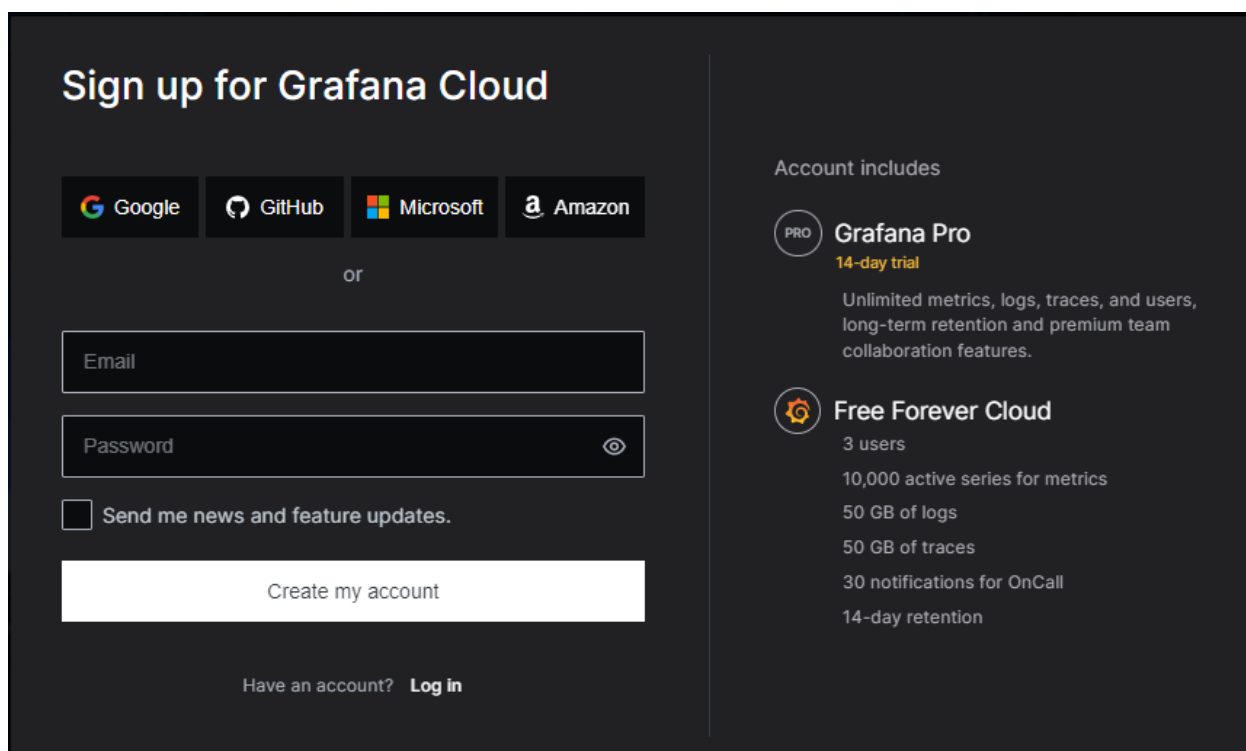
    bucket = "Practice"
    write_api.write(bucket=bucket, org="████████████████████████████████████████", record=usd_point)
    write_api.write(bucket=bucket, org="████████████████████████████████████████", record=eur_point)
    write_api.write(bucket=bucket, org="████████████████████████████████████████", record=btc_point)
    time.sleep(900)

```

6) Интеграция InfluxDB с Grafana

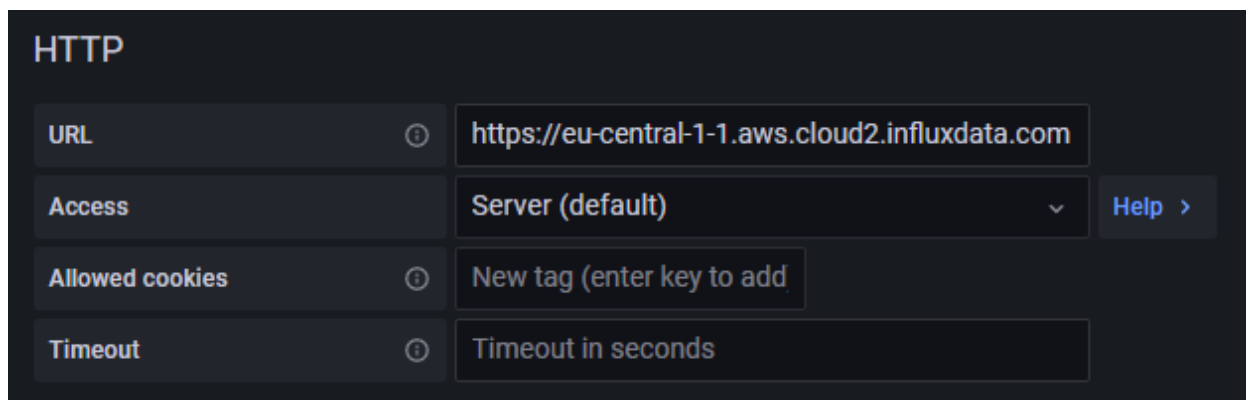
Grafana — это платформа для мониторинга, анализа и визуализации данных. Grafana не собирает и не хранит данные, а представляет «сухие» цифры в виде красивых диаграмм и графиков. Для Grafana источник данных — системы мониторинга и хранилища систематических данных. Таким образом, платформу можно подключить к InfluxDB и многим другим движкам.

Проходим процедуру регистрации на платформе Grafana



The image shows the 'Sign up for Grafana Cloud' page. It features a dark theme with white text. At the top, it says 'Sign up for Grafana Cloud'. Below this, there are four buttons for social login: Google, GitHub, Microsoft, and Amazon. A separator 'or' is placed below these buttons. Then, there are two input fields: 'Email' and 'Password'. Below the password field is a checkbox labeled 'Send me news and feature updates.' and a large white button labeled 'Create my account'. At the bottom, there is a link 'Have an account? Log in'. On the right side, under 'Account includes', there are two options: 'Grafana Pro' (14-day trial) with unlimited metrics, logs, traces, and users, and 'Free Forever Cloud' with 3 users, 10,000 active series for metrics, 50 GB of logs, 50 GB of traces, 30 notifications for OnCall, and 14-day retention.

Создаём источник данных для Grafana, выбрав опцию InfluxDB.



The image shows the 'HTTP' data source configuration page in Grafana. It has a dark theme. The title 'HTTP' is at the top. Below it, there are four rows of configuration options, each with a label, a help icon, and a value field. The first row is 'URL' with the value 'https://eu-central-1-1.aws.cloud2.influxdata.com'. The second row is 'Access' with the value 'Server (default)' and a 'Help >' button. The third row is 'Allowed cookies' with the value 'New tag (enter key to add)'. The fourth row is 'Timeout' with the value 'Timeout in seconds'.

InfluxDB Details

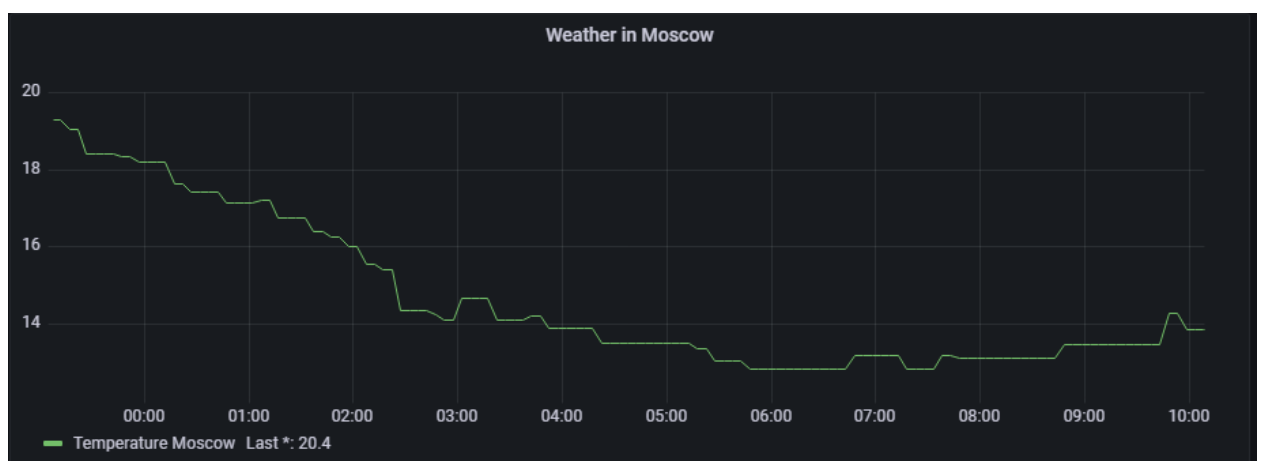
Organization	<input type="text"/>	
Token	<input type="text" value="configured"/>	<button>Reset</button>
Default Bucket	<input type="text" value="Monitoring"/>	
Min time interval	<input type="text" value="3"/>	
Max series	<input type="text" value="1000"/>	

Создаём новый Dashboard для мониторинга погоды и проводим необходимые настройки.

Для получения необходимых данных из базы данных InfluxDB, пишем запрос на языке Flux в формате:

```
from(bucket: "Practice")
|> range(start: -30d, stop: now())
|> filter(fn: (r) => r._measurement == "Weather")
|> filter(fn: (r) => r.Location == "Moscow")
|> filter(fn: (r) => r._field == "Temperature")
```

В результате получаем график изменения погоды на основании измерений, проведённых во время работы программы.



Проводим аналогичные действия для программы для мониторинга ресурсов системы



И мониторинга курса валют



6. Результаты практики.

Студент Журавков В.М. показал умение работать с большим объёмом информационного материала, смог качественно обработать и проанализировать научную информацию в рамках выполнения задач практики. Владение профессиональной терминологией свободное. Студент продемонстрировал высокий уровень теоретических знаний и умение использовать их для решения профессиональных задач. Без затруднений ориентируется в нормативной, научной и специальной литературе.

Программа практики и индивидуальное задание выполнено в полном объёме. Материалы, изложенные в отчёте студента, полностью соответствуют индивидуальному заданию. Все предусмотренные рабочей программой практики компетенции сформированы. Демонстрируется высокий уровень самостоятельности, высокая адаптивность практического навыка.

Рекомендованная оценка за практику «отлично».