МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра МО ЭВМ

ИНДИВИДУАЛЬНОЕ ДОМАШНЕЕ ЗАДАНИЕ

по дисциплине «Введение в нереляционные базы данных»

Тема: Система мониторинга параметров оборудования.

Студент гр. 1303 _____ Кузнецов Н.А.

Студент гр. 1303 _____ Депрейс А.С.

Студентка гр. 1384 _____ Найдёнова Е.В.

Преподаватель _____ Заславский М.М.

Санкт-Петербург

2024

ЗАДАНИЕ

Студенты

Кузнецов Н.А.

Депрейс А.С.

Найдёнова Е.В.

Группы 1303, 1384

Тема работы : Система мониторинга параметров оборудования

Исходные данные:

1. Данные оборудования:
   Электрические параметры (ток, напряжение, мощность)
   Характеристики оборудования (производитель, год выпуска и др.)
2. Требования:
   Мониторинг превышений пороговых значений
   Визуализация данных
   Система предупреждений

Содержание пояснительной записки:

Введение

Сценарий использования

Модель данных

Разработка приложения

Выводы

Приложение

Литература

Предполагаемый объем пояснительной записки:

Не менее 10 страниц.

Дата выдачи задания: 01.09.2024

Дата сдачи реферата: 24.12.2024

Дата защиты реферата: 24.12.2024

Студент гр. 1303	_____	Кузнецов Н.А.
Студент гр. 1303	_____	Депрейс А.С.
Студентка гр. 1384	_____	Найдёнова Е.В.
Преподаватель	_____	Заславский М.М.

АННОТАЦИЯ

В рамках проекта разработана система мониторинга параметров промышленного оборудования с использованием современного технологического стека. Система обеспечивает сбор и анализ данных о токе, напряжении и мощности оборудования в режиме реального времени. Для хранения данных использованы InfluxDB (временные ряды) и MongoDB (метаданные). Реализован веб-интерфейс на Flutter с бэкендом на Go, предоставляющий функционал визуализации данных, управления оборудованием и пользователями, систему предупреждений. В результате создана платформа для мониторинга и анализа работы промышленного оборудования.

SUMMARY

The project developed a monitoring system for industrial equipment parameters using modern technology stack. The system provides real-time collection and analysis of current, voltage, and power equipment data. InfluxDB (time series) and MongoDB (metadata) are used for data storage. A web interface implemented in Flutter with Go backend provides functionality for data visualization, equipment and user management, and warning system. As a result, a platform for monitoring and analyzing industrial equipment operation has been created.

СОДЕРЖАНИЕ

	Введение	4
1.	Сценарии использования	5
2.	Модель данных	7
3.	Разработка приложения	26
4.	Выводы	27
5.	Приложения	28
6.	Литература	29

ВВЕДЕНИЕ

Актуальность решаемой проблемы:

В современной промышленности критически важно отслеживать параметры работы оборудования в реальном времени для предотвращения аварийных ситуаций и оптимизации производственных процессов.

Постановка задачи:

Разработать систему мониторинга параметров промышленного оборудования с функциями анализа данных и оповещения о превышении пороговых значений.

Предлагаемое решение:

Создание распределенной системы на основе:

- InfluxDB для хранения временных рядов данных
- MongoDB для хранения метаданных
- Backend на Go для обработки запросов
- Frontend на Flutter для визуализации

Качественные требования к решению:

- Удобный пользовательский интерфейс
- Оперативное оповещение о превышениях
- Возможность анализа исторических данных

1. СЦЕНАРИЙ ИСПОЛЬЗОВАНИЯ

Макет UI (см. Рисунок 1)

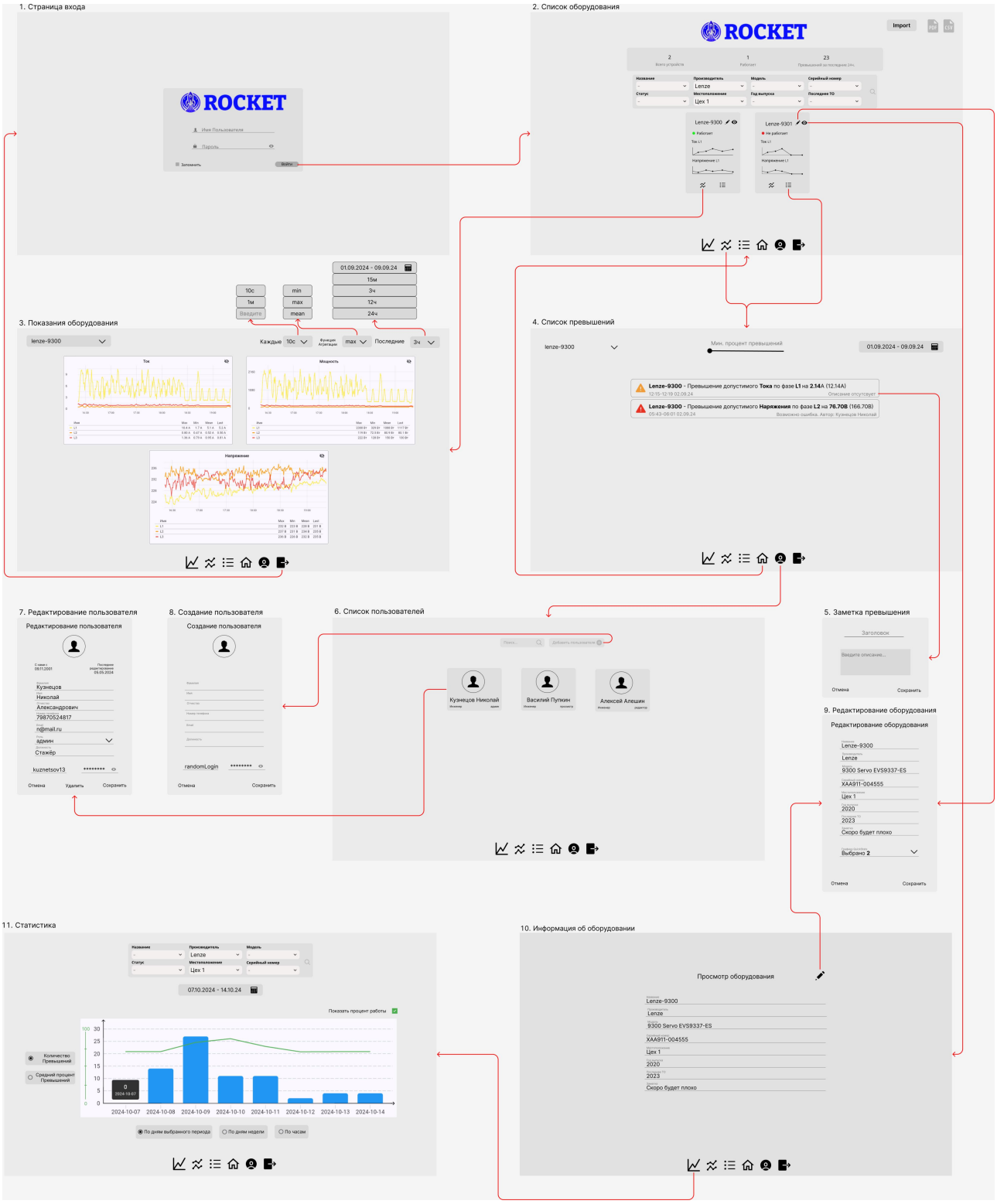

Рисунок 1 - Макет UI

Сценарии использования для задачи.

1. Импорт данных:

Массовый: Пользователь может импортировать данные в формате txt через страницу “Настройки”.

2. Представление данных:

- Визуализация через графики на странице показаний оборудования
- Фильтрация по временному диапазону
- Возможность скрывать/показывать отдельные графики
- Статистические графики с настраиваемыми осями X и Y
- Фильтрация превышений по величине и времени

3. Анализ данных:

- Построение статистических графиков с выбором характеристик оборудования
- Группировка данных по временным интервалам
- Анализ превышений пороговых значений
- Возможность добавления заметок к зафиксированным превышениям

4. Экспорт данных:

- Экспорт в формат txt со страницы “Настройки”.

Вывод об операциях:

В системе будут преобладать операции чтения данных, так как основной функционал направлен на:

- Постоянный мониторинг показаний
- Визуализацию данных через графики
- Анализ статистики
- Просмотр превышений

Операции записи будут происходить реже и включают:

- Импорт данных
- Добавление заметок к превышениям
- Управление пользователями

2. МОДЕЛЬ ДАННЫХ

Нереляционная модель данных.

Графическое представление. (см. Рисунок 2)

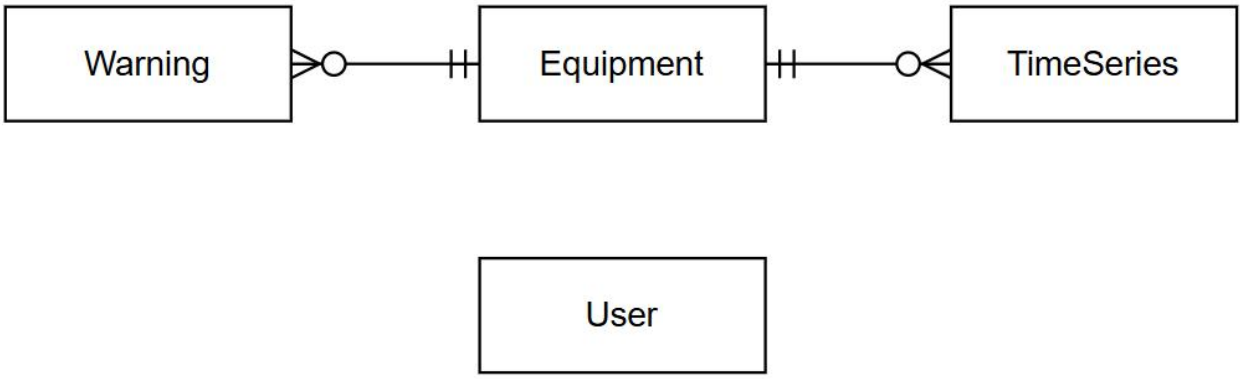

Рисунок 2 – Нереляционная модель данных

Описание назначений коллекций, типов данных и сущностей

1. Оборудование будет храниться в формате bson в MongoDb

· _id: Уникальный идентификатор

· key: Ключ для идентификации оборудования

· name: Название оборудования

· details: Информация об оборудовании (серийный номер, местоположение, производитель, модель, статус, год, группа)

· parameters: Параметры оборудования с их единицами измерения, пороговыми значениями и подпараметрами

· working_parameter: Основной рабочий параметр оборудования с пороговым значением

2. Предупреждения будут храниться в формате bson в MongoDb

· _id: Уникальный идентификатор

· date_from: Время начала предупреждения

· date_to: Время окончания предупреждения

· equipment: Ключ оборудования
· text: Описание предупреждения
· value: Значение при превышении
· type: Тип события
· excess_percent: Процент превышения
· title, description: Дополнительная информация
· username: Автор описания

3. Пользователи будут храниться в формате bson в MongoDb
· _id: Уникальный идентификатор
· Личные данные: имя, фамилия, отчество, дата рождения
· Рабочая информация: место работы, должность, группа электробезопасности
· supervisor_id: ID руководителя
· Контакты: email, телефон
· Системные данные: роль, логин, пароль, аватар
· Метки времени: created, last_modified

4. Временные ряды по параметрам оборудования будут храниться в InfluxDb
· measurement: Название измерения
· tags: Теги для идентификации параметра
· fields: Значения измерений
· time: Временная метка

Оценка удельного объема информации, хранимой в модели (сколько потребуется памяти, чтобы сохранить объекты, как объем зависит от количества объектов)

1. Оборудование:

- _id: 12 байт
  - $oid: 12 байт
- key: 20 байт
- name: 30 байт
- details: 200 байт
  - SN: 20 байт
  - location: 30 байт
  - manufacturer: 30 байт
  - model: 30 байт
  - status: 50 байт
  - year: 4 байта
  - group: 36 байт
- parameters: 300 байт (3 параметра по 100 байт каждый)
  - [имя_параметра]: 100 байт
    - translate: 20 байт
    - unit: 10 байт
    - threshold: 8 байт
    - subparameters: 60 байта
      - [имя_подпараметра]: ~20 байт
        - translate: 10 байт
        - topic: 10 байт
- working_parameter: 50 байт
  - parameter: 20 байт
  - threshold: 8 байт
  - subparameter: 22 байта

Общий размер для одного объекта оборудования: 612 байт

2. Предупреждения:

- _id: 12 байт
  - $oid: 12 байт
- date_from: 16 байт
  - $date: 16 байт
- date_to: 16 байт
  - $date: 16 байт
- equipment: 20 байт
- text: 100 байт
- value: 20 байт
- type: 20 байт
- excess_percent: 8 байт
- title: 50 байт
- description: 200 байт
- username: 30 байт

Общий размер для одного предупреждения: 492 байта

3. Пользователи:
- _id: 12 байт
  - $oid: 12 байт
- surname: 30 байт
- patronymic: 30 байт
- birthday: 10 байт
- work_place: 50 байт
- position: 50 байт
- electrical_safety_group: 4 байта
- supervisor_id: 12 байт
  - $oid: 12 байт
- email: 50 байт

- phone_number: 20 байт
- role: 20 байт
- name: 30 байт
- avatar: 100 байт
- username: 20 байт
- password: 60 байт
- last_modified: 16 байт
  - $date: 16 байт
- created: 16 байт
  - $date: 16 байт

Общий размер для одного пользователя: 548 байт

4. Временные ряды в InfluxDB:

Для каждой точки данных:

- measurement: ~20 байт
- tags:
  - topic: ~30 байт
- fields:
  - value: 8 байт (число с плавающей точкой)
- time: 8 байт (timestamp)

Общий размер одной точки данных: ~66 байт

Расчет объема данных для временных рядов за день:

- Каждое оборудование имеет 3 параметра, каждый параметр имеет 3 субпараметра
- Всего 9 временных рядов на оборудование
- Данные записываются раз в секунду

- За день: 9 * 86400 * 66 = 51,408,000 байт ≈ 50 МБ на оборудование в день

Общая формула выражена через две переменные, так как используются две базы данных.

Пусть n - количество объектов оборудования, d - количество дней хранения данных

1. Оборудование: 612n байт
2. Предупреждения: 492 * 2d*n = 984nd байт (предполагая по 2 предупреждения по каждому оборудованию в день)
3. Пользователи: 548 * (n/2) = 274n байт (предполагая 5 пользователей на каждые 10 объектов оборудования)
4. Временные ряды: 51,408,000 * n * d байт

Итоговая формула:

V = 612n + 984nd + 274n + 51,408,000nd байт
V ≈ 886n Байт + 50nd МБ

Избыточность данных

1. Расчет избыточных данных:

- Оборудование: 12 байт (_id) * n = 12n байт
- Предупреждения: 12 байт (_id) * 2dn = 24dn байт
- Пользователи: 12 байт (_id) * (n/2) = 6n байт

Общий объем избыточных данных:
I = 12n + 24dn + 6n = 18n + 24dn байт

2. Общий объем данных (из предыдущего расчета):

$V \approx 886n$ байт + $50nd$ МБ

$V \approx (886n + 51{,}200{,}000nd)$ байт (переведем МБ в байты для удобства расчетов)

3. "Чистый" объем данных:

$C = V - I = (886n + 51{,}200{,}000nd) - (18n + 24dn)$ байт

$C = 868n + 51{,}199{,}976nd$ байт

Итоговая формула избыточности:

$R = (886 + 51{,}200{,}000d) / (868 + 51{,}199{,}976d)$

Интерпретация:

1. При $d \to \infty$, $R \to 1$, что означает, что при увеличении периода хранения данных избыточность стремится к нулю.

2. При $d = 1$, $R \approx 1.0000352$, то есть при хранении данных за один день избыточность составляет около 0.00352%.

3. При $d = 0$ (гипотетически, если бы у нас были только метаданные без временных рядов), $R \approx 1.0207$, что соответствует избыточности около 2.07%.

Запросы к модели, с помощью которых реализуются сценарии использования

1. Получение всех оборудований

```
equipment_collection = get_collection_by_token(token, "EquipmentList")
res = list(equipment_collection.find({}))
```

Количество запросов: 1

Количество задействованных коллекций: 1

2. Получение превышений

Для выбора превышений по признакам нужно выполнить запрос 1 для получения списка оборудования, далее выбрать нужное оборудование и сделать такой запрос:

```
    if excess_percent:
        query["excess_percent"] = {'$gte': float(excess_percent)}
    if equipment_key:
        query["equipment"] = equipment_key
    if start_date and end_date:
        start_datetime = datetime.datetime.strptime(start_date,
'%Y-%m-%d %H:%M')
        end_datetime = datetime.datetime.strptime(end_date, '%Y-
%m-%d %H:%M')
        query["date_from"] = {'$gte': start_datetime, '$lte':
end_datetime}

    total = warnings_list.count_documents(query)
    warnings = warnings_list.find(query).sort('timestamp', -
1).skip((page - 1) * per_page).limit(per_page)
```

Количество запросов: 2

Количество задействованных коллекций: 2

3. Получение данных для построение графика.

Для построения графика нужно получить данные оборудования, сделав запрос 1, далее для получения временного ряда сделать такой запрос:

```
    query := fmt.Sprintf(`from(bucket:"%s")
                        |> range(start: -%ds)
                        |> filter(fn: (r) => r.topic == "%s")
                        |> filter(fn: (r) => r._field == "value")
                        |> aggregateWindow(every: %ds, fn: mean,
createEmpty: false)
                        `, customerClients.Config.InfluxBucket,
request.Interval, topic, every)

                              results, err :=
queryAPI.Query(context.Background(), query)
```

Количество запросов: 2

Количество задействованных баз данных: 2 (1 коллекция, 1 бакет)

Реляционная модель данных.

Графическое представление. (см. Рисунок 3)

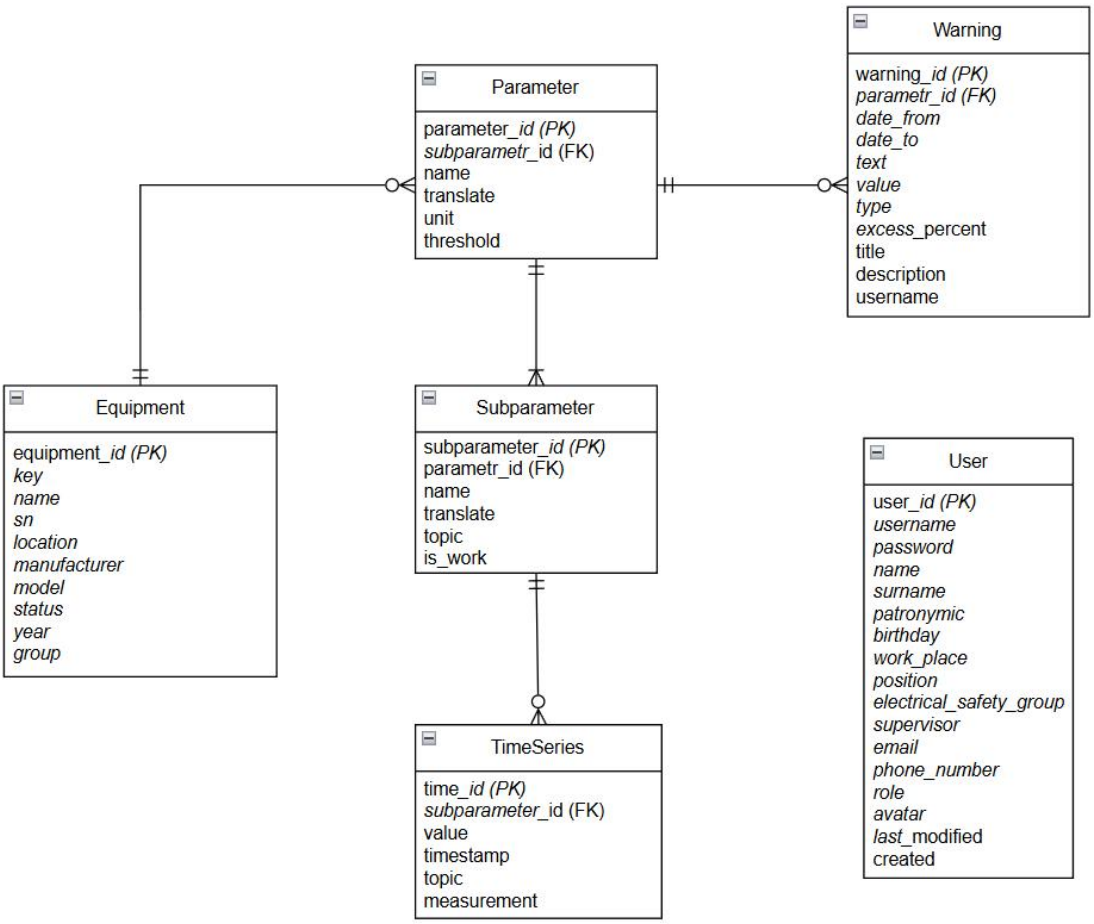

Рисунок 3 - Реляционная модель данных

Описание назначений коллекций, типов данных и сущностей

1. Equipment:

- equipment_id: BIGINT PRIMARY KEY
- key: VARCHAR(20)
- name: VARCHAR(30)
- sn: VARCHAR(20)
- location: VARCHAR(30)
- manufacturer: VARCHAR(30)
- model: VARCHAR(30)
- status: VARCHAR(50)
- year: INTEGER

- group: VARCHAR(36)

2. Parameter:
- parameter_id: BIGINT PRIMARY KEY
- subparametr_id: BIGINT FOREIGN KEY
- name: VARCHAR(100)
- translate: VARCHAR(20)
- unit: VARCHAR(10)
- threshold: DECIMAL(10,2)

3. Subparameter:
- subparameter_id: BIGINT PRIMARY KEY
- parametr_id: BIGINT FOREIGN KEY
- name: VARCHAR(20)
- translate: VARCHAR(10)
- topic: VARCHAR(10)
- is_work: BOOLEAN

4. Warning:
- warning_id: BIGINT PRIMARY KEY
- parametr_id: BIGINT FOREIGN KEY
- date_from: TIMESTAMP
- date_to: TIMESTAMP
- text: VARCHAR(100)
- value: VARCHAR(20)
- type: VARCHAR(20)
- excess_percent: DECIMAL(10,2)
- title: VARCHAR(50)
- description: TEXT
- username: VARCHAR(30)

5. User:

- user_id: BIGINT PRIMARY KEY
- username: VARCHAR(20) UNIQUE
- password: VARCHAR(60)
- name: VARCHAR(30)
- surname: VARCHAR(30)
- patronymic: VARCHAR(30)
- birthday: DATE
- work_place: VARCHAR(50)
- position: VARCHAR(50)
- electrical_safety_group: INTEGER
- supervisor: VARCHAR(30)
- email: VARCHAR(50)
- phone_number: VARCHAR(20)
- role: VARCHAR(20)
- avatar: VARCHAR(100)
- last_modified: TIMESTAMP
- created: TIMESTAMP

6. TimeSeries:

- time_id: BIGINT PRIMARY KEY
- subparameter_id: BIGINT FOREIGN KEY
- value: DECIMAL(10,2)
- timestamp: TIMESTAMP
- topic: VARCHAR(30)
- measurement: VARCHAR(20)

Оценка удельного объема информации, хранимой в модели (сколько потребуется памяти, чтобы сохранить объекты, как объем зависит от количества объектов)

Пусть n - количество оборудования, d - количество дней хранения

1. Equipment: (8 + 20 + 30 + 20 + 30 + 30 + 30 + 50 + 4 + 36) * n = 258n байт

2. Parameter: (8 + 8 + 100 + 20 + 10 + 8) * 3n = 462n байт

3. Subparameter: (8 + 8 + 20 + 10 + 10 + 1) * 9n = 513n байт

4. Warning: (8 + 8 + 8 + 8 + 100 + 20 + 20 + 8 + 50 + 200 + 30) * 2dn = 920dn байт

5. User: (8 + 20 + 60 + 30 + 30 + 30 + 4 + 50 + 50 + 4 + 30 + 50 + 20 + 20 + 100 + 8 + 8) * (n/2) = 261n байт

6. TimeSeries: 9 * (8 + 8 + 8 + 8 + 30 + 20) * 86400 * n * d = 63,763,200nd байт

Общая формула:

V = (258 + 462 + 513 + 261)n + (920 + 63,763,200)nd

V = 1,494n + 63,764,120nd байт

V ≈ 1.5KB * n + 60.8MB * n * d

Избыточность данных:

Избыточные данные формируются за счет:

- PK (8 байт на каждую связь)
- FK (8 байт на каждый индекс)

Избыточные данные:

(8 * 4)n + (8 * 2)nd = 32n + 16nd байт

Формула избыточности:

R = (1,494 + 63,764,120d) / (1,462 + 63,764,104d)

Интерпретация:

1. При $d \to \infty$, $R \to 1$, что означает, что при увеличении периода хранения данных избыточность стремится к нулю.

2. При $d = 1$, $R \approx 1.0000049$, то есть при хранении данных за один день избыточность составляет около 0.00049%.

3. При $d = 0$ (гипотетически), $R \approx 1.022$, что соответствует избыточности около 2.2%.

Запросы к модели, с помощью которых реализуются сценарии использования.

1. Получение всего оборудования:

```
SELECT * FROM Equipment;
```

Количество запросов: 1

Количество задействованных таблиц: 1

2. Получение предупреждений по оборудованию с фильтрами:

```
SELECT w.*
FROM Warning w
JOIN Parameter p ON w.parametr_id = p.parameter_id
WHERE w.excess_percent >= :threshold
  AND w.date_from BETWEEN :start_date AND :end_date;
```

Количество запросов: 1

Количество задействованных таблиц: 2

3. Получение данных временного ряда для графика:

```
SELECT ts.timestamp, ts.value
FROM TimeSeries ts
JOIN Subparameter sp ON ts.subparameter_id =
sp.subparameter_id
```

```
WHERE sp.topic = :topic
  AND ts.timestamp >= :start_time
  AND ts.timestamp <= :end_time
ORDER BY ts.timestamp;
```

Количество запросов: 1

Количество задействованных таблиц: 2

Сравнение моделей

Удельный объем информации:

1. Нереляционная модель:

V ≈ 886n байт + 50nd МБ

2. Реляционная модель:

V ≈ 1433n байт + 61nd МБ

Реляционная модель требует больше места как для метаданных (≈1.6 раз), так и для временных рядов (≈1.22 раза).

Запросы по юзкейсам:

1. Нереляционная модель требует больше запросов (1-2 запроса), но данные получаются в полном объеме без дополнительных JOIN-операций.

2. Реляционная модель использует меньше запросов (всегда 1 запрос), но требует JOIN-операций для получения связанных данных.

Вывод:

NoSQL модель предпочтительнее для данной задачи, так как:

- Занимает меньше места на диске
- Лучше подходит для хранения иерархических данных параметров
- Более гибкая схема данных
- InfluxDB специально оптимизирована для временных рядов
- Несмотря на большее количество запросов, они проще и эффективнее чем JOIN-операции в SQL

3. РАЗРАБОТАННОЕ ПРИЛОЖЕНИЕ

Система построена на микросервисной архитектуре и состоит из следующих модулей:

1. Frontend модуль:

Web-интерфейс на Flutter

Отвечает за визуализацию данных и взаимодействие с пользователем

Реализует все описанные пользовательские сценарии

2. Backend модуль (Go):

API-сервер для обработки запросов

Бизнес-логика приложения

Управление авторизацией

Обработка импорта/экспорта данных

3. Модуль хранения данных:

InfluxDB для временных рядов (показания оборудования)

MongoDB для метаданных (информация о пользователях, оборудовании, настройки)

Использованные технологии:

Frontend: Flutter Web

Backend: Go

Базы данных: InfluxDB, MongoDB

API: REST

Контейнеризация: Docker

Полученные результаты (см. Рисунок 4 - 8)

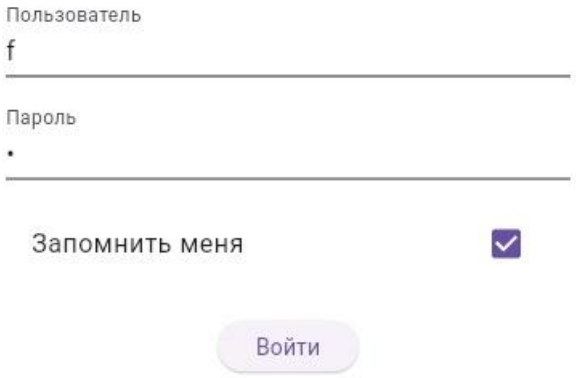

Рисунок 4 - Авторизация

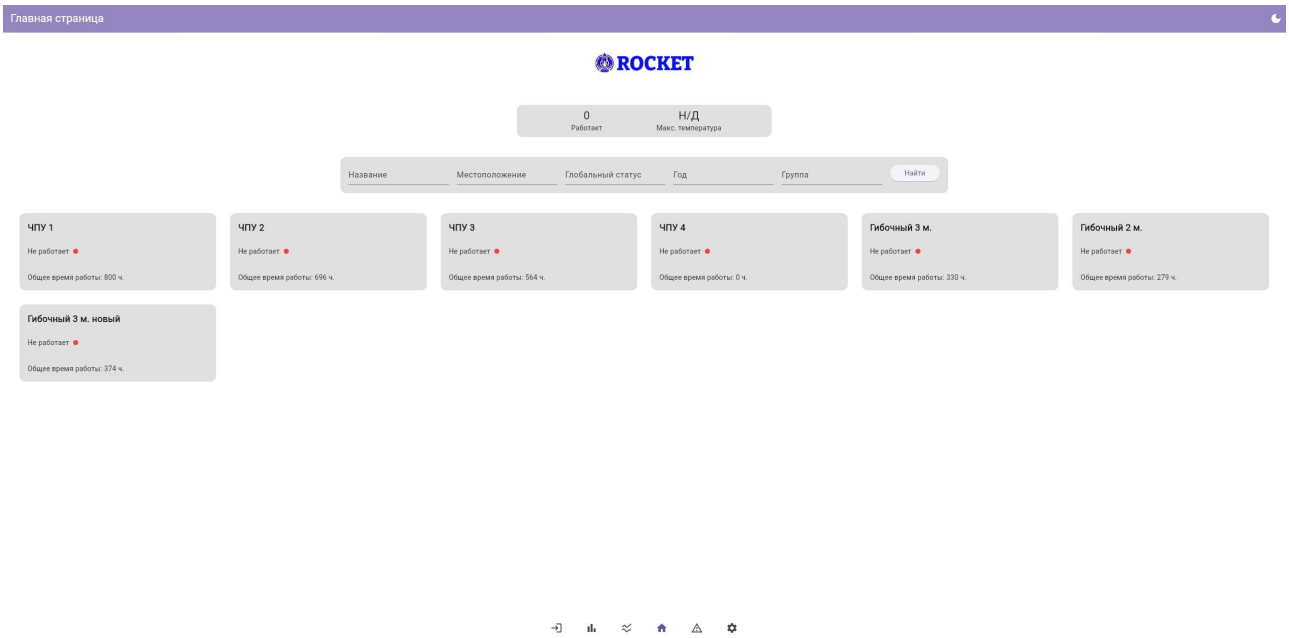

Рисунок 5 - Главная страница

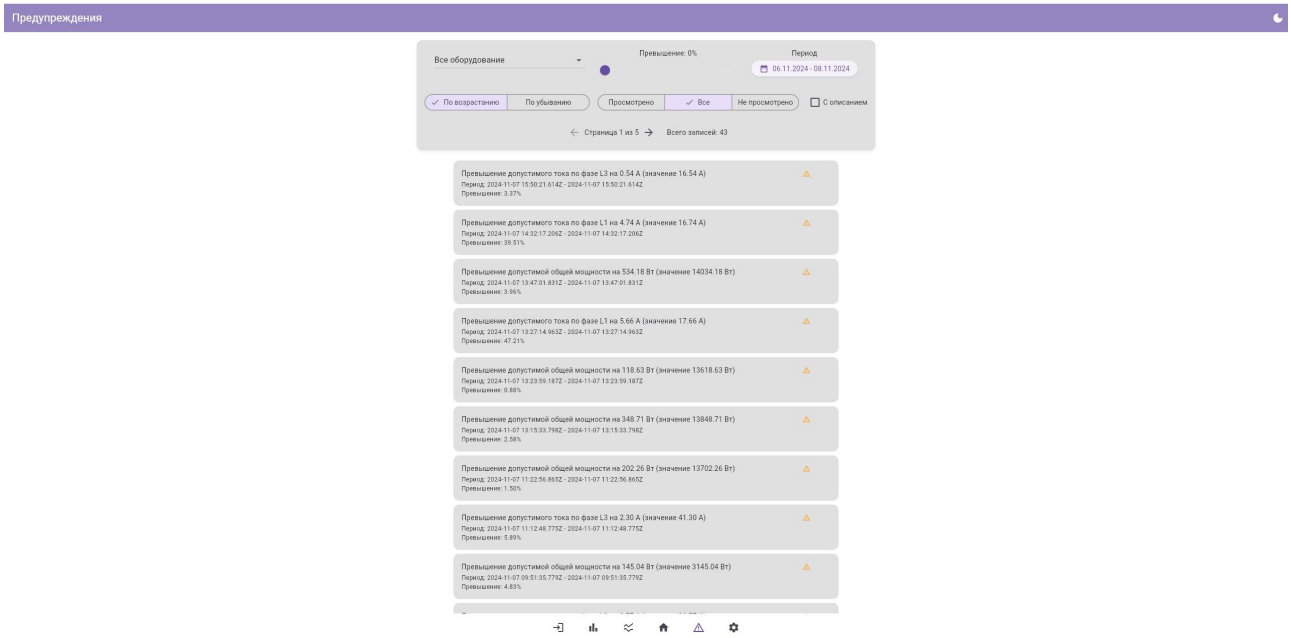

Рисунок 5 - Предупреждения

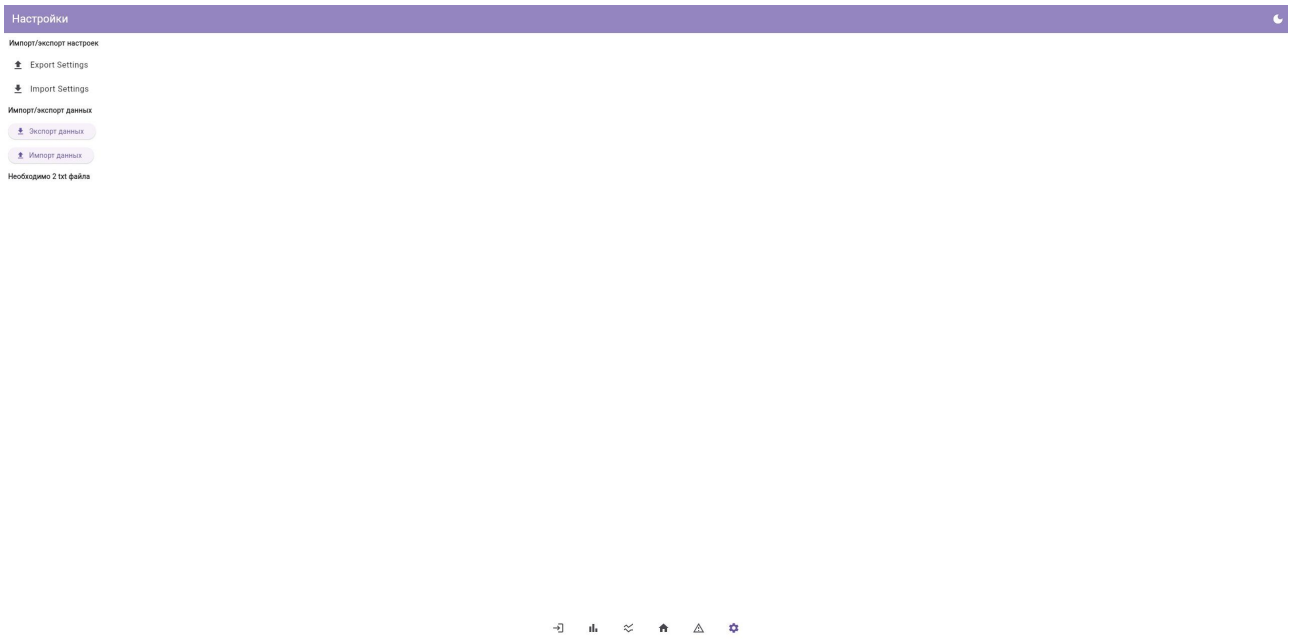

Рисунок 6 - Настройки

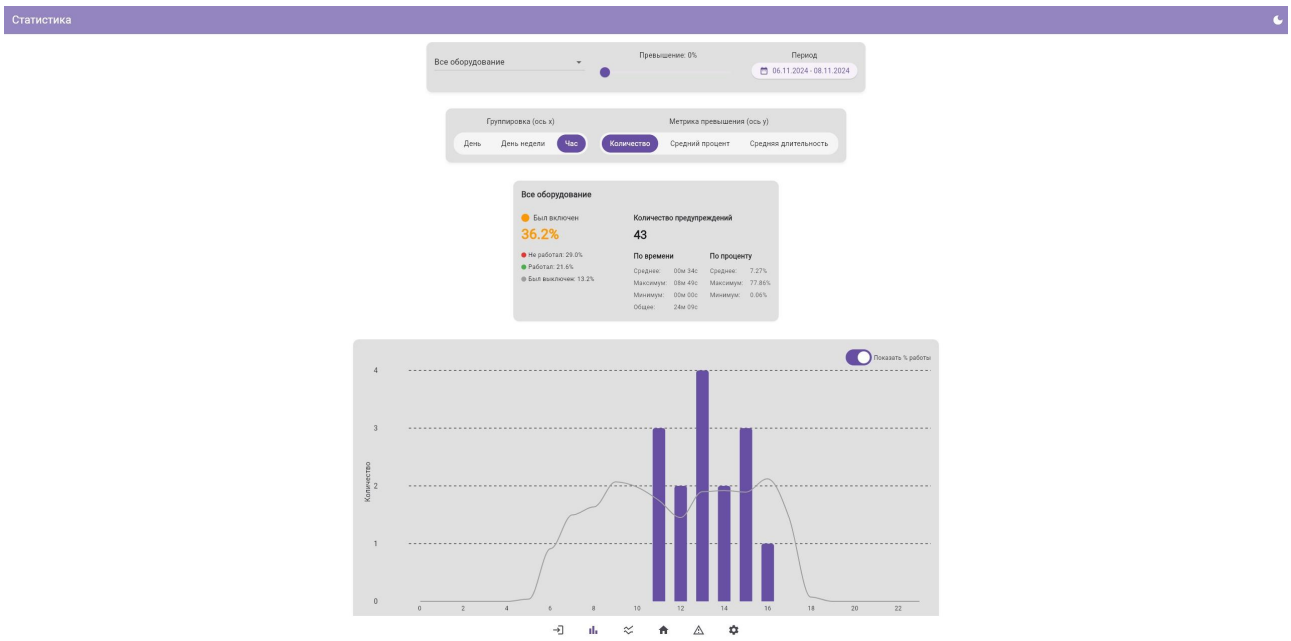

Рисунок 7 - Статистика

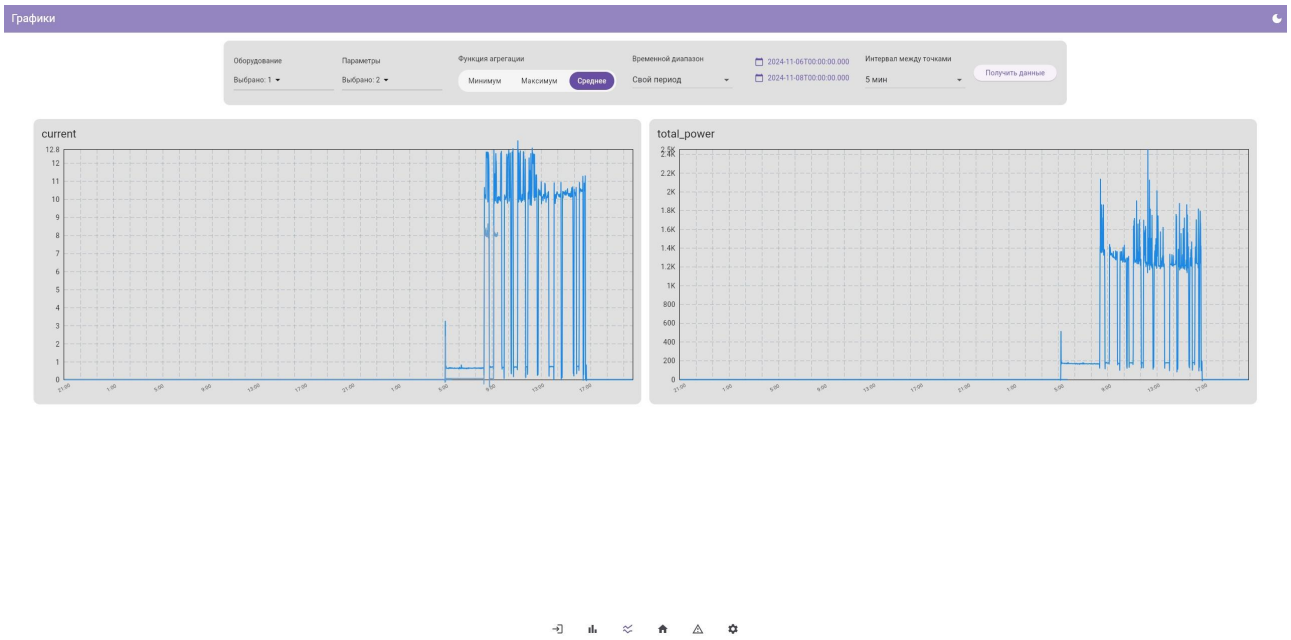

Рисунок 8 - Графики

4. ВЫВОДЫ

Достигнутые результаты:

- Разработана система мониторинга параметров оборудования
- Внедрена система предупреждений о превышениях
- Создан функционал анализа и визуализации данных
- Обеспечена возможность импорта/экспорта данных

Недостатки и пути улучшения:

- Отсутствие многопользовательского доступа с разграничением прав
- Нет автоматического масштабирования системы

Будущее развитие решения:

- Добавление REST API для внешних интеграций
- Разработка мобильного приложения
- Добавление автоматического масштабирования
- Внедрение расширенной аналитики и отчетности
- Реализация многопользовательского доступа с разграничением прав

5. ПРИЛОЖЕНИЕ

Запуск приложения

1. Клонировать репозиторий и запустить:
git clone https://github.com/moevm/nosql2h24-factory.git
docker compose build --no-cache && docker compose up

2. Открыть в браузере:
http://127.0.0.1:81

3. Войти в систему:
Логин: f
Пароль: f

Функционал

- Мониторинг оборудования
- Просмотр параметров и графиков
- Система предупреждений

Требования

- Docker
- Ubuntu 22.04+
- Свободные порты 81 и 8080

6. ЛИТЕРАТУРА

1. Проект системы мониторинга оборудования на производстве. GitHub Repository. https://github.com/moevm/nosql2h24-factory

2. Das, P., & Kumar, S. (2023). "InfluxDB: A Comprehensive Study on Time Series Database Management." International Journal of Database Management Systems, 15(2), 45-62.

3. Zhang, Y., et al. (2024). "MongoDB Performance Optimization in Industrial Applications." Journal of Database Systems, 8(1), 112-128.

3. Smith, R., & Johnson, M. (2023). "Flutter for Enterprise Web Applications: Best Practices and Performance Analysis." IEEE Software Engineering Conference, 234-249.

4. Chen, L., & Wang, H. (2024). "Microservices Architecture with Go: A Case Study in Industrial Monitoring Systems." ACM Computing Surveys, 56(3), 1-34.

5. Anderson, K., et al. (2023). "Real-time Data Processing in Industrial IoT: Architectures and Challenges." Industrial Internet of Things Journal, 12(4), 78-95.

.