

**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: Сервис сбора и визуализации логов Apache2**

Студенты гр. 0381

Самойлов З.А.

Павлов Е.А.

Преподаватель

Заславский М.М.

Санкт-Петербург

2023

## ЗАДАНИЕ

Студенты

Павлов Е. А.

Самойлов З. А.

Группа 0381

Тема работы: Сервис сбора и визуализации логов Apache2

Исходные данные:

Задача - создать приложение, которое агрегирует логи Apache2 в influx. Необходимо поддерживать одновременно все файлы логов apache - access.log, error.log, other\_vhosts\_access.log, а также время загрузки страницы как один из элементов данных. Необходимые (но не достаточные) фичи - таблица поиска по всем логам с фильтром, страница отдельной записи в логе, кастомизируемая статистика (по хостам, ip клиентов, кодам ошибок, времени загрузки страниц)

Содержание пояснительной записки:

«Содержание»

«Введение»

«Качественные требования к решению»

«Сценарии использования»

«Модель данных»

«Разработка приложения»

«Вывод»

«Приложение»

Дата выдачи задания: 22.09.2023

Дата сдачи реферата: 25.12.2023

Дата защиты реферата: 25.12.2023

Студенты гр. 0381		Самойлов З. А.
		Павлов Е. А.
Преподаватель		Заславский М.М.

## АННОТАЦИЯ

В рамках данного курса предполагалось разработать какое-либо приложение в команде на одну из поставленных тем. Была выбрана тема создания для агрегации и визуализации apache логов. В качестве СУБД была выбрана InfluxDB [1], поскольку она лучше подходит для разработки такой системы. Во внимание будут приниматься такие аспекты как производительность и удобство разработки. Найти исходный код и всю дополнительную информацию можно по ссылке: <https://github.com/moevm/nosql2h23-apache-logs>

## **ANNOTATION**

As part of this course, it was supposed to develop an application in a team on one of the given topics. The theme of creating aggregation and visualization of Apache logs was chosen. InfluxDB [1] was chosen as the DBMS because it is better suited for developing such a system. Aspects such as performance and ease of development will be taken into account. You can find the source code and all additional information at the link: <https://github.com/moevm/nosql2h23-apache-logs>

## Оглавление

1. Введение.....	7
2. Качественные требования к решению.....	8
3. Сценарии использования.....	9
4. Модель данных.....	14
5. Разработанное приложение.....	26
6. Вывод.....	28
7. Приложения.....	29
8. Используемая литература.....	30

## **1. Введение**

Цель работы – создать высокопроизводительное и удобное решение для хранения и анализа логов Apache.

Было решено разработать приложение, которое позволит хранить логи Apache, при этом позволяющее с ними удобно взаимодействовать.

## **2. Качественные требования к решению**

Требуется разработать приложение с использованием СУБД InfluxDB. Backend должен быть реализован с использованием Apache и Telegraf. Frontend должен быть реализован с помощью Grafana.



### 3. Сценарии использования

Здесь представлены лишь несколько сценариев использования, которые отображают примеры с конкретными переменными/данными. Количество возможных сценариев намного больше, в том числе из-за возможности кастомизации.

1. Отображение данных за определенный временной период.

Предусловие: пользователь находится на странице логов в Grafana.

Пользователь: кликает на меню выбора временного периода в правом верхнем углу.

Система: отображает модальное окошко с выбором периода.

Пользователь: выбирает предопределенные периоды или создает свой собственный.

Система: закрывает модальное окошко и обновляет данные в соответствии с периодом.

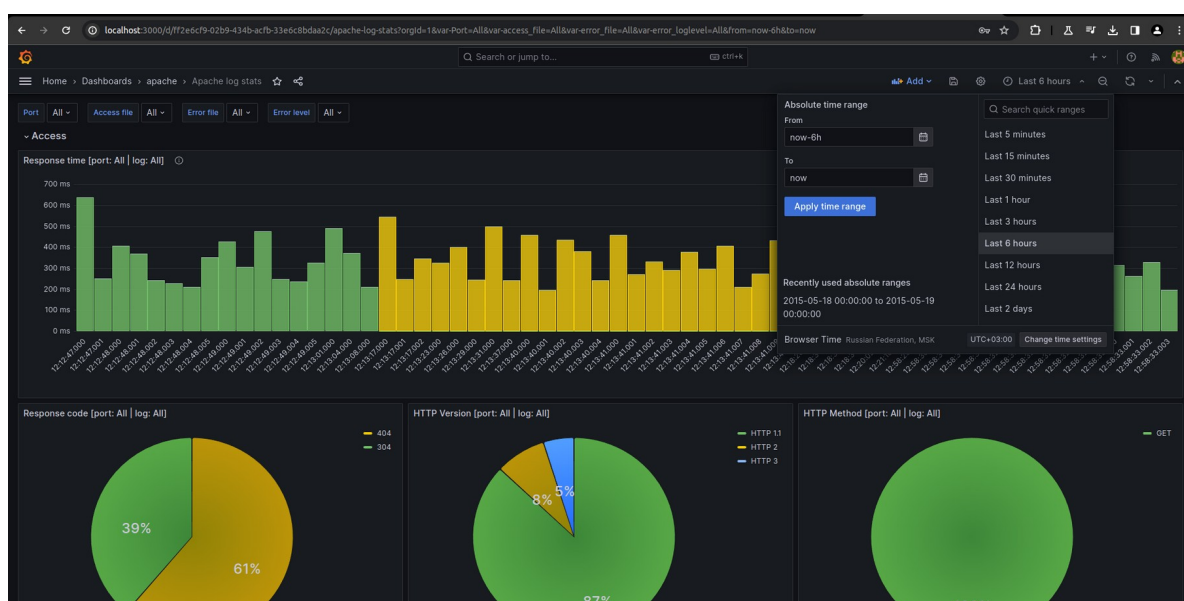


Рисунок 1 – Отображение данных во временном периоде. Открытие модального окна.

2. Отображение данных по определенному порту.

Предусловие: пользователь находится на странице логов в Grafana.

Пользователь: кликает на меню выбора порта в левом верхнем углу.

Система: отображает модальное окошко с выбором порта.

Пользователь: выбирает порт из доступных.

Система: закрывает модальное окошко и обновляет данные в соответствии с выбранным портом.

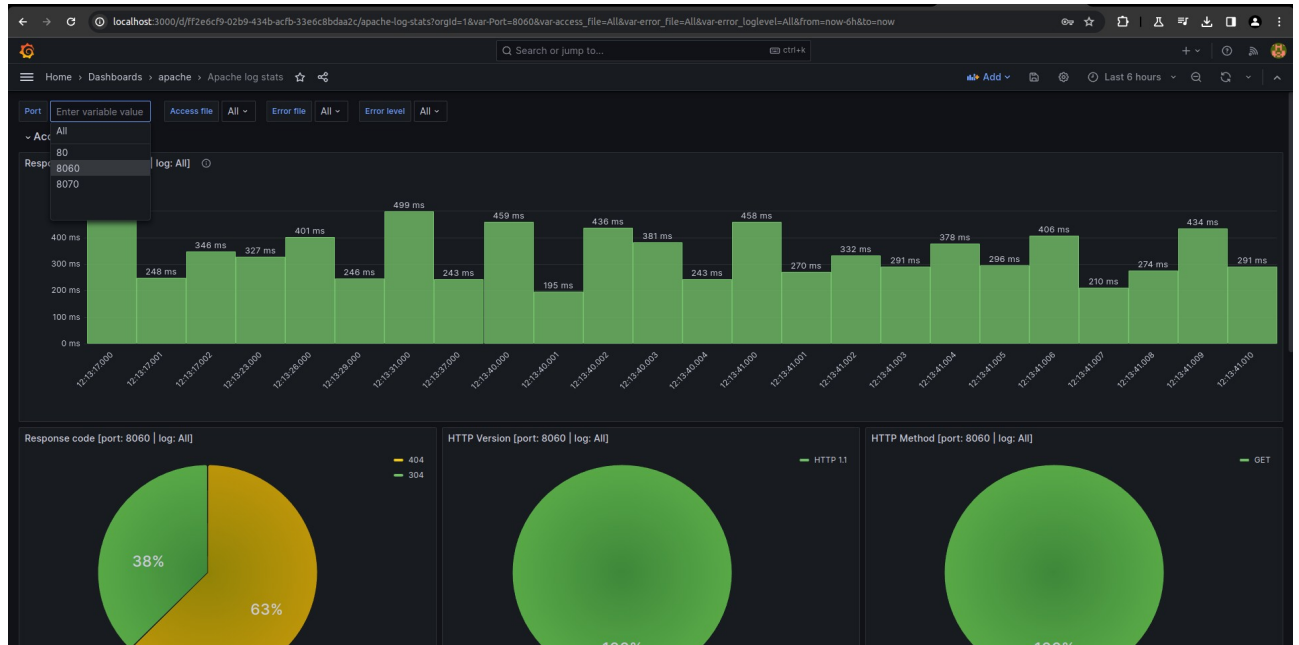


Рисунок 2 – Отображение данных по определенному порту.

### 3. Просмотр частоты ошибки в error логах.

Предусловие: пользователь находится на странице логов в Grafana.

Пользователь: кликает на строку лога в разделе error.

Система: отображает тэги лога.

Пользователь: выбирает кнопку статистики рядом с тегом ошибок.

Система: отображает статистику по частоте данной ошибки.

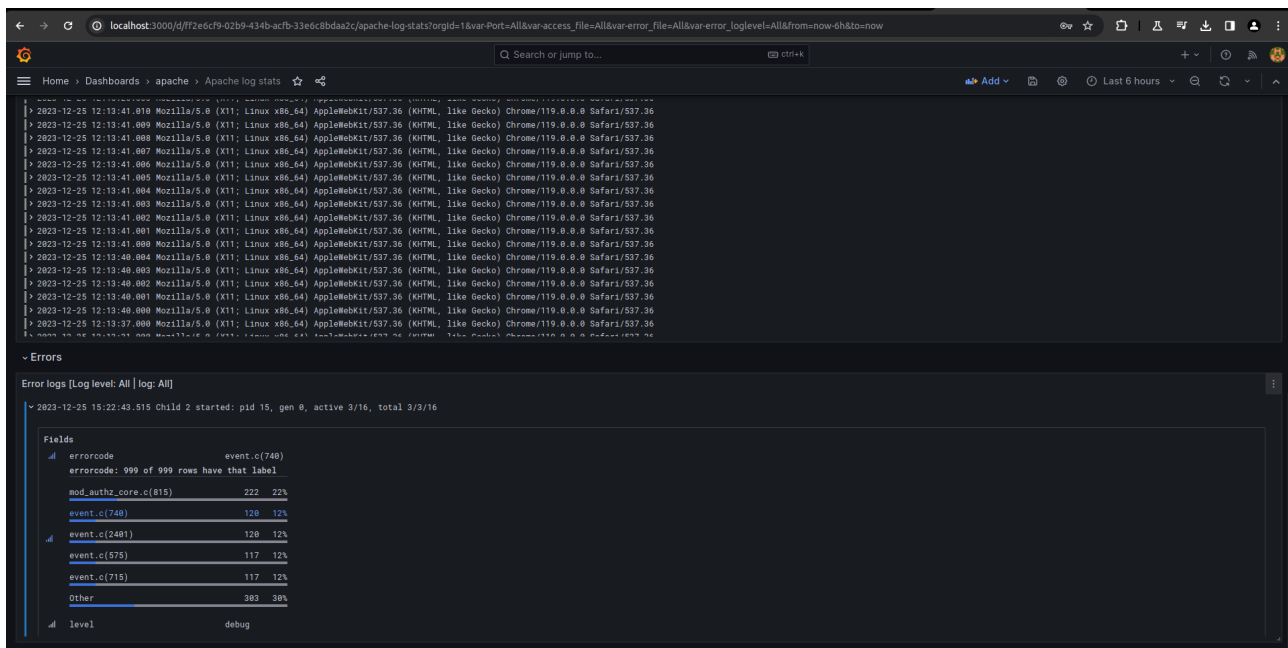


Рисунок 3 – Просмотр статистики по ошибке в error.

## 4. Модель данных

### Нереляционная модель

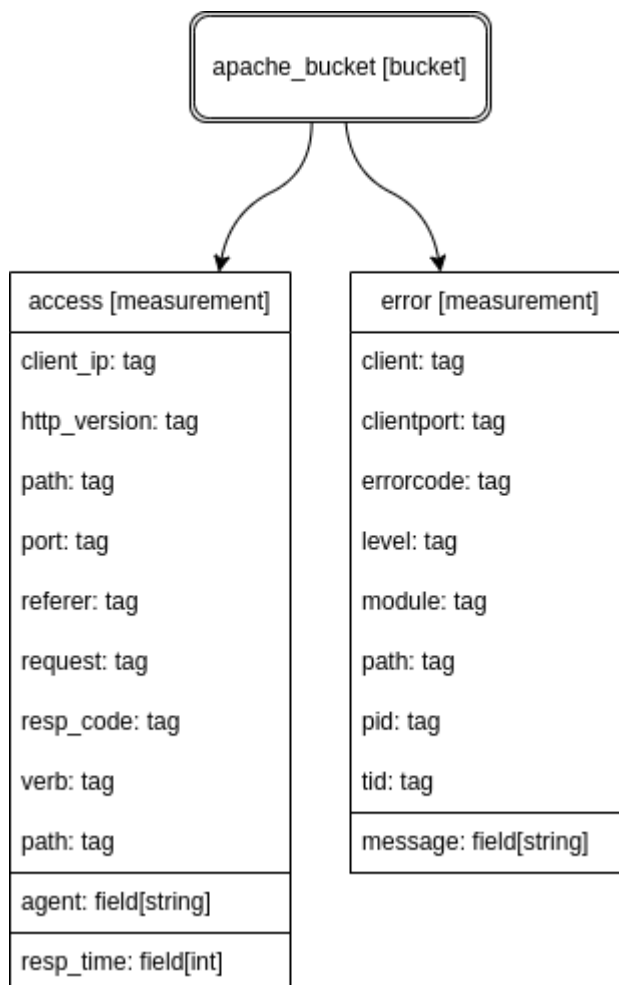


Рисунок 4 – Графическое представление нереляционной модели

### Описание и оценка объёма

Решаемая задача: агрегация и визуализация apache - логов

Сущность: - один файл с логами. По сути, массив из записей логгера.

Пример одной строки с логами из access.log (здесь 646 - время ответа):

InfluxDB - база данных, предназначенная для хранения временных рядов.

В нашем случае, в качестве временного ряда выступают записи от логгера apache. Всего будет два measurement - access и error.

В access у нас будет два поля - response time и agent, остальные данные будем записывать в качестве тэгов. В InfluxDB все тэги хранятся как строки, только \_value может иметь другие типы данных, например, int.

Тэги:

- path - путь до файла с логами
- client\_ip - ip, с которого пришёл запрос
- http\_version - версия http для данного запроса
- port - порт, на который пришёл запрос
- referer - адрес, с которого пришёл запрос
- request - запрашиваемый ресурс
- resp\_code - статус код ответа
- verb - метод REST API
- \_value - значение (т.е response time или agent, в зависимости от \_field),  
типы данных - int или string

### **Избыточность модели**

Избыточность данных будет проявляться в хранении одинаковых тэгов для разных полей. Фактически избыточность равна количеству полей (за вычетом памяти используемой для самих полей).

### **Направление роста модели**

Примеры запросов из Grafana (могут использоваться переменные Grafana).

- Получение возможных значений переменной Port

```
import "influxdata/influxdb/v1"

v1.tagValues(
  bucket: v.bucket,
  tag: "port",
  predicate: (r) => true,
  start: -1d)
```

- Получение возможных значений переменной `access_file`

```
from(bucket: "apache_bucket")
|> range(start: -24h)
|> filter(fn: (r) => r["_measurement"] == "access" and r["_field"] ==
"resp_time")
|> group(columns: ["path"])
|> distinct(column: "path")
|> keep(columns: ["_value"])
```

- Получение возможных значений переменной `error_file`

```
from(bucket: "apache_bucket")
|> range(start: -24h)
|> filter(fn: (r) => r["_measurement"] == "error" and r["_field"] == "message")
|> group(columns: ["path"])
|> distinct(column: "path")
|> keep(columns: ["_value"])
```

- Получение возможных значений переменной `error_loglevel`

```
from(bucket: "apache_bucket")
|> range(start: -24h)
|> filter(fn: (r) => r["_measurement"] == "error" and r["_field"] == "message")
|> group(columns: ["level"])
|> distinct(column: "level")
|> keep(columns: ["_value"])
```

- Response time

```
keeps = if "${Port}" == "All" then ["_time", "_value", "port"] else ["_time",
"_value"]
```

```
from(bucket: "apache_bucket")
```

```

|> range(start: v.timeRangeStart, stop:v.timeRangeStop)
|> filter(
  fn: (r) => if "${Port}" == "All" then
    r["_measurement"] == "access" and r["_field"] == "resp_time" and
(r["path"] == "${access_file}" or "${access_file}" == "All")
  else
    r["_measurement"] == "access" and r["_field"] == "resp_time" and r["port"]
== "${Port}" and (r["path"] == "${access_file}" or "${access_file}" == "All")
)
|> keep(columns: keeps)
|> sort(columns: ["_time"])

```

- Response code

```

from(bucket: "apache_bucket")
|> range(start: v.timeRangeStart, stop:v.timeRangeStop)
|> filter(fn: (r) => r["_measurement"] == "access" and r["_field"] ==
"resp_time" and (r["path"] == "${access_file}" or "${access_file}" == "All")
and (r["port"] == "${Port}" or "${Port}" == "All"))
|> group(columns: ["resp_code"], mode: "by")
|> count()
|> keep(columns: ["resp_code", "_value"])
|> rename(columns: {_value: ""})

```

- HTTP Version

```

from(bucket: "apache_bucket")
|> range(start: v.timeRangeStart, stop:v.timeRangeStop)
|> filter(fn: (r) => r["_measurement"] == "access" and r["_field"] ==
"resp_time" and (r["path"] == "${access_file}" or "${access_file}" == "All")
and (r["port"] == "${Port}" or "${Port}" == "All"))
|> group(columns: ["http_version"], mode: "by")

```

```
|> count()
|> keep(columns: ["http_version", "_value"])
|> rename(columns: {_value: "HTTP "})
```

- HTTP Method

```
from(bucket: "apache_bucket")
|> range(start: v.timeRangeStart, stop:v.timeRangeStop)
|> filter(fn: (r) => r["_measurement"] == "access" and r["_field"] ==
"resp_time" and (r["path"] == "${access_file}" or "${access_file}" == "All")
and (r["port"] == "${Port}" or "${Port}" == "All"))
|> group(columns: ["verb"], mode: "by")
|> count()
|> keep(columns: ["verb", "_value"])
|> rename(columns: {_value: ""})
```

- Access logs

```
import "join"
```

```
left = from(bucket: "apache_bucket")
|> range(start: v.timeRangeStart, stop:v.timeRangeStop)
|> filter(fn: (r) => r["_measurement"] == "access" and r["_field"] ==
"resp_time" and (r["path"] == "${access_file}" or "${access_file}" == "All"))
|> pivot(rowKey: ["_time"], columnKey: ["_field"], valueColumn: "_value")
|> keep(columns: ["_time", "resp_time"])
|> group(columns: ["_time"])
```

```
right = from(bucket: "apache_bucket")
|> range(start: -24h)
|> filter(fn: (r) => r["_measurement"] == "access" and r["_field"] == "agent"
and (r["path"] == "${access_file}" or "${access_file}" == "All"))
```



```
|> group(columns:["_time"])
```

```
join.right(  
  left: left,  
  right: right,  
  on: (l, r) => l._time == r._time,  
  as: (l, r) => ({r with resp_time: l.resp_time}),  
)  
|> group()  
|> drop(columns:["_field", "_measurement", "_start", "_stop"])
```

- Error logs

```
from(bucket: "apache_bucket")  
|> range(start: v.timeRangeStart, stop:v.timeRangeStop)  
|> filter(fn: (r) => r["_measurement"] == "error"  
  and (r["path"] == "${error_file}" or "${error_file}" == "All")  
  and (r["level"] == "${error_loglevel}" or "${error_loglevel}" == "All")  
)
```

## Реляционная модель

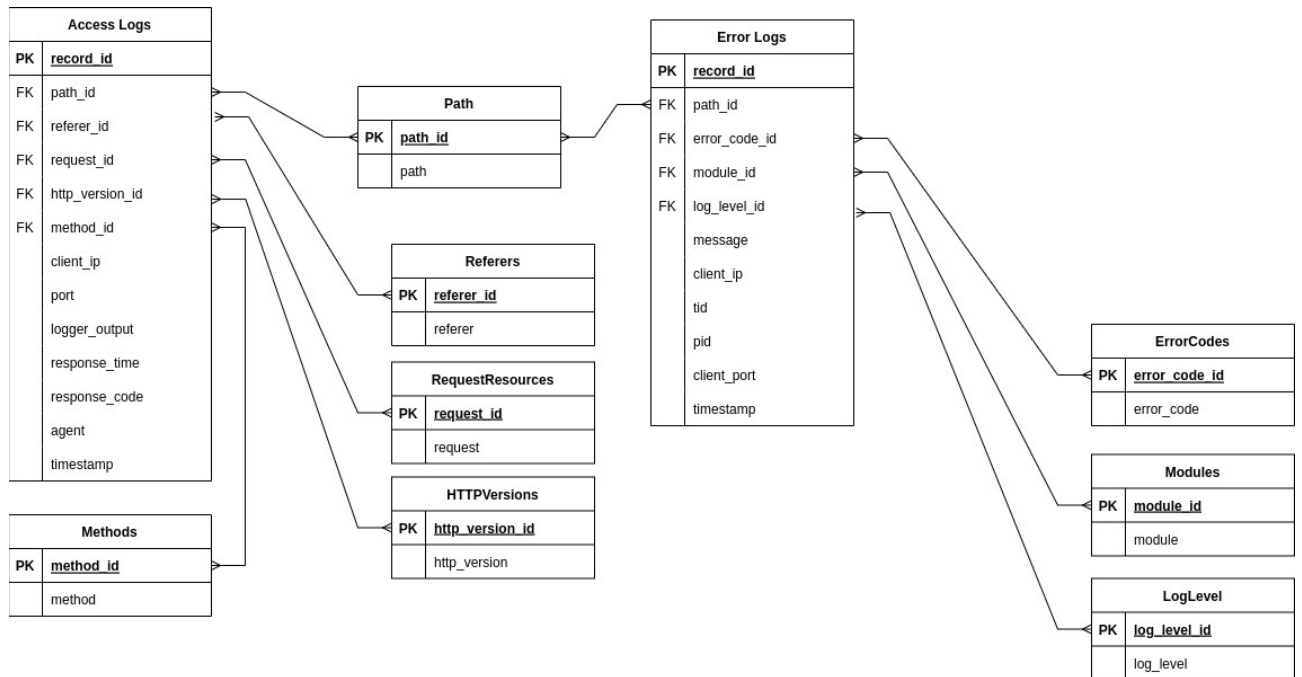


Рисунок 5 – Графическое представление реляционной модели

## Описание и оценка объема

Поле	Описание	Тип	Размер, байт
record_id	Идентификатор записи	INT	4
path_id	Идентификатор пути лога	INT	4
referer_id	Идентификатор referer	INT	4
request_id	Идентификатор запроса	INT	4
http_version_id	Идентификатор версии HTTP	INT	4
method_id	Идентификатор метода	INT	4
client_ip	IP клиента	INT	4
port	Порт хоста в Apache	INT	4
logger_output	Строка лога	STRING	200
response_time	Время ответа сервера	INT	4
response_code	Код ответа	INT	4
agent	Агент клиента	STRING	100
timestamp	Временная точка	TIMESTAMP	8

Таблица 1. Сущность AccessLogs.

Поле	Описание	Тип	Размер, байт
path_id	Идентификатор пути лога	INT	4
path	Путь лога	STRING	100

Таблица 2. Сущность Paths.

Поле	Описание	Тип	Размер, байт
referer_id	Идентификатор referer	INT	4
referer	Поле Referer заголовка запроса	STRING	50

Таблица 3. Сущность Referers.

Поле	Описание	Тип	Размер, байт
request_id	Идентификатор запроса	INT	4
Request	Запрос	STRING	50

Таблица 4. Сущность Requests.

Поле	Описание	Тип	Размер, байт
http_version_id	Идентификатор версии HTTP	INT	4
http_version	Версия HTTP протокола	STRING	10

Таблица 5. Сущность HTTPVersions.

Поле	Описание	Тип	Размер, байт
method_id	Идентификатор метода	INT	4
Method	Метод	STRING	10

Таблица 6. Сущность Methods.

Поле	Описание	Тип	Размер, байт
record_id	Идентификатор записи	INT	4
path_id	Идентификатор пути лога	INT	4
error_code_id	Идентификатор кода ошибки	INT	4
module_id	Идентификатор модуля	INT	4
log_level_id	Идентификатор уровня логирования	INT	4
Message	Сообщение ошибки	STRING	50
client_ip	IP клиента	INT	4
client_port	Порт клиента	INT	4
tid	TID	INT	4
pid	PID	INT	4
timestamp	Временная точка	TIMESTAMP	8

Таблица 7. Сущность ErrorLogs.

Поле	Описание	Тип	Размер, байт
error_code_id	Идентификатор кода ошибки	INT	4
error_code	Код ошибки	STRING	20

Таблица 8. Сущность ErrorCodes.

Поле	Описание	Тип	Размер, байт
module_id	Идентификатор модуля	INT	4
Module	Модуль Apache	STRING	20

Таблица 9. Сущность Modules.

Поле	Описание	Тип	Размер, байт
log_level_id	Идентификатор уровня логирования	INT	4
log_level	Уровень логирования	STRING	10

Таблица 10. Сущность LogLevels.

\*Женя\* - оценка объема.

### Избыточность модели

\*Женя\* - избыточность.

Избыточность может составлять от  $N$  до  $2N$ , где  $N$  - количество дополнительных таблиц для id, в данный момент — 10.

### Направление роста модели

\*Женя\* - направление роста.

## Запросы

\*Женя\* - запросы?

## Сравнение моделей

\*Женя\* - сравнение.

## Вывод

\*Женя\* - вывод по сравнению.

## Примеры хранения данных

[illegible]

Рисунок 6 – Экспорт базы данных в CSV формат.

## 5. Разработанное приложение

## Краткое описание

Разработанное приложение представляет из себя надстройку над apache, telegraf, influxdb и grafana. Apache поставляет логи запросов, которые telegraf парсит и отправляет в influxdb. Далее в grafana строятся визуализации с использованием данных, получаемых запросами в influxdb. Каждое из приложений завёрнуто в свой docker контейнер, а запуск и настройка итогового приложения осуществляется с помощью docker compose.

## Схема экранов приложения

Экраны приложения и переходы между ними отображены на рис. 12.

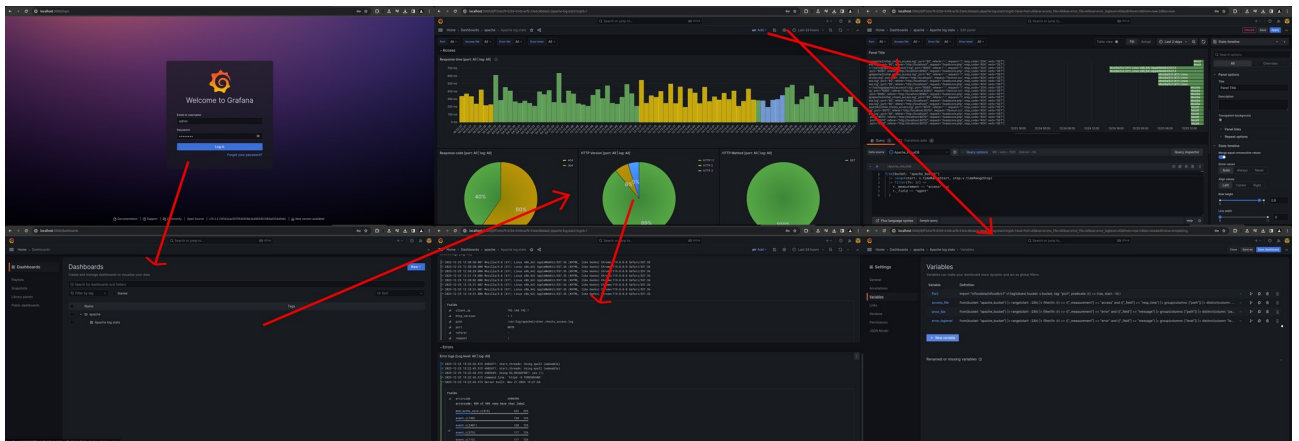


Рисунок 7 – Схема экранов приложения

## Использованные технологии

БД: InfluxDB.

Backend: Apache, Telegraf

Frontend: Grafana

### **Ссылка на Приложение**

Ссылка на GitHub: <https://github.com/moevm/nosql2h23-apache-logs>



## **6. Вывод**

### **Результаты**

В ходе работы разработано приложение, которое обрабатывает логи apache. Аггрегирует логи из разных файлов, считает статистики и строит визуализации

### **Недостатки и пути для улучшения полученного решения**

В данный момент решение использует специфические настройки логов apache и собственные grok паттерны для telegraf. Это может приводить к тому, что, в случае установки новых форматов логов, в некоторых сценариях пользователю придётся переопределить grok паттерны telegraf для корректной работы приложения. Из-за специфики выбранной БД, большое количество данных хранится как тэги, что представляется неоптимальным. Во-первых, с точки зрения потребления памяти, во-вторых, с точки зрения скорости работы из-за индексации тэгов.

### **Будущее развитие решения**

Выбор базы данных, лучше подходящий для хранения логов

## **7. Приложения**

### **Документация по сборке и развёртыванию приложений**

Для установки зависимостей необходимо выполнить:

`docker compose up (-d).`

Предполагается, что на машине установлен `docker` и `docker compose`.

## 8. Используемая литература

1. Документация к InfluxDB: <https://docs.influxdata.com/>
2. Паттерны Grok:  
[https://docs.influxdata.com/telegraf/v1/data\\_formats/input/grok/](https://docs.influxdata.com/telegraf/v1/data_formats/input/grok/)
3. Паттерны Grok в Telegraf:  
[https://github.com/influxdata/telegraf/blob/master/plugins/parsers/grok/influx\\_patterns.go](https://github.com/influxdata/telegraf/blob/master/plugins/parsers/grok/influx_patterns.go)
4. Флаги логов в Apache:  
[https://httpd.apache.org/docs/2.4/mod/mod\\_log\\_config.html#logformat](https://httpd.apache.org/docs/2.4/mod/mod_log_config.html#logformat)