Содержание

Введение

Задача по сбору и анализу трафика может возникнуть в различных ситуациях. Например, при наблюдении за поведением пользователей во внутренней сети, попытке отследить вредоносное программное обеспечение, обращающееся к внешнему сервису, или при контроле квоты использования интернета. Большой объем данных, проходящих через сетевые узлы, накладывает некоторые ограничения на инструменты, используемые при решении поставленной задачи.

Целью данной работы является реализация программного комплекса для сбора, хранения и анализа информации о трафике, проходящего через некоторый сетевой узел.

Для достижения поставленной цели необходимо решить следующие задачи:

- проанализировать предметную область;
- спроектировать программный комплекс;
- реализовать спроектированную систему.

1 Аналитический раздел

В данном разделе будет поставлена задача, рассмотрены требования к системе и проведен анализ существующих решений.

1.1 Постановка задачи

Необходимо разработать программный комплекс, предоставляющий возможность собирать, хранить и просматривать информацию о трафике, проходящем через сетевой узел.

Предусмотреть наличие нескольких ролей пользователей с разным уровнем привилегий:

- пользователь "Гость" с возможностью просмотра базовых таблиц;
- пользователь "Аналитик" с возможностью запуска сложных запросов и наличием тех же прав, что и у "Гость";
- пользователь "Администратор" с возможностью управления аккаунтами пользователей и наличием тех же прав, что и у "Аналитик".

1.2 Требования к системе

- система должна обладать возможностью масштабирования на несколько сетевых узлов, кластеров хранения данных;
- необходимо обеспечить долгосрочное хранение больших объемов данных (трафик за несколько лет) с сохранением возможности поиска предыдущих записей за приемлемое время;

- предусмотреть возможность непрерывной работы подсистемы сбора данных в течении длительного времени.

1.3 Общие сведения

Информация о трафике обычно представляется с помощью NetFlow протокола[1]. Существуют различные версии протокола, но принцип работы у всех них один. Для сбора данных с помощью NetFlow используют следующие компоненты:

- сенсор;
- коллектор;
- анализатор.

Сенсор служит для сбора статистики о проходящем через него трафике. Данные, собранные сенсором отправляются в коннектор, который агрегирует данные с нескольких сенсоров и записывает их в хранилище. Анализатор предоставляет возможность работы с данными. Схема архитектуры NetFlow представлена на рисунке 1.

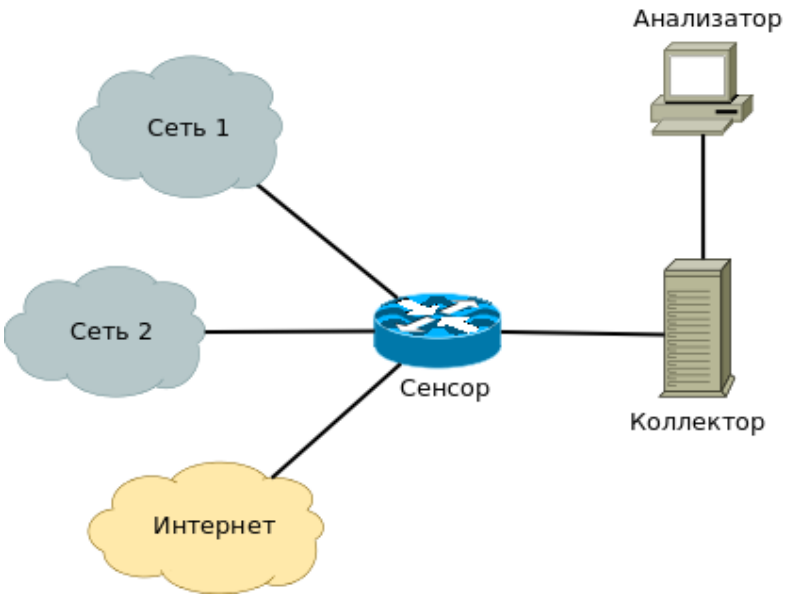


Рисунок 1 – Архитектура NetFlow.

1.4 Анализ существующих решений

Исходя из изложенных выше сведений можно выделить два направления поиска возможного решения:

- приложение, реализующее в себе все компоненты архитектуры NetFlow;
- система, использующая отдельные компоненты вместе.

1.4.1 Приложение

По первому направлению сравнения можно выделить два самых видных решения:

- Wireshark;
- tcpdump.

Wireshark

Бесплатный, мультиплатформенный (работает на Windows, macOS и семействе Linux) анализатор трафика с открытым исходным кодом, предоставляющий возможность быстро проанализировать трафик сети[2]. Есть возможность фильтровать и сортировать трафик. Имеет интуитивно понятный и информативный интерфейс. Однако позволяет работать только с локальными интерфейсами, а записанные данные сохраняет в простой файл, что делает его неприменимым для долгосрочного сбора данных и не дает возможности масштабирования.

tcpdump

Бесплатный анализатор трафика[3]. Работает на Linux, но есть несколько портов для Windows. Предоставляет базовую функциональность через консольный интерфейс. Минусы такие же как и у Wireshark - отсутствие возможности масштабирования и работа только с локальными интерфейсами.

В таблице 1 представлено сравнение Wireshark и tcpdump

Таблица 1 – Сравнение Wireshark и tcpdump.

	Wireshark	tcpdump
Возможность просмотра информации о трафике	+	+
Наличие графического интерфейса	+	-
Возможность записи данных в бд	-	-
Возможность отправки данных	-	-
Работа с удаленными интерфейсами	-	-
Разграничение сбора и анализа информации	-	-

Как видно, готовые программы анализа информации о трафике имеют общие недостатки, не позволяющие использовать их для решения поставленной задачи.

1.4.2 Система

По второму направлению сравнения выделить готовых систем не удалось, но можно выделить некоторые популярные решения отдельных мо-

дулей.

1.5 Анализ моделей СУБД

Модель данных определяет логическую структуру базы данных, то, каким образом данные могут храниться, организовываться и обрабатываться. Существуют два больших класса СУБД - SQL и NoSQL.

1.5.1 SQL

Наиболее популярным и самым распространенным типом СУБД является РСУБД (реляционная СУБД) или SQL-СУБД. Такие СУБД построены на основе реляционной модели, поддерживают стандарт SQL и поэтому имеют много схожих характеристик.

В реляционной модели данные представляются в виде строк в таблицах, соотносящих атрибуты и значения. Реляционная модель строго определяет структуру данных, ограничения целостности над данными, теоретико-множественные операции над ними. В реалиционной модели определены два базовых требования обеспечения целостности:

- ссылочная целостность (гарантия отсутствия ссылок на несуществующие объекты);
- целостность сущностей (гарантия уникальности объектов).

Вследствие вышесказанного SQL СУБД хорошо проходит для хранения данных, имеющих строго определенную структуру.

1.5.2 NoSQL

NoSQL СУБД появились не так давно, но уже обрели достаточно большую популярность в силу своих особенностей. NoSQL СУБД не имеют одной общей модели данных, единственное, что их объединяет это базированность не на основе реляционной модели.

У NoSQL СУБД есть ряд преимуществ над SQL СУБД:

- возможность хранения больших объемов неструктурированной информации;
- большая скорость обработки больших объемов данных;
- отсутствие проблем с масштабируемостью.

NoSQL СУБД хорошо подходят для хранения больших данных, так как имеют большую скорость обработки, позоляют легко смасштабировать систему и позволяют хранить неструктурированную информацию.

В таблице Представлено сравнение SQL и NoSQL СУБД.

Таблица 2 – Сравнение SQL и NoSQL.

	SQL	NoSQL
Поддержка SQL	+	частично
Легкость масштабирования	-	+
Высокая скорость обработки больших данных	-	+
Поддержка связанности	+	-
Поддержка хранимых процедур и тригеров	+	-

Исходя из требований к системе и сущности обрабатываемых данных был выбран NoSQL подход.

1.6 Вывод

В данном разделе была поставлена задача, рассмотрены требования к системе, проанализированы существующие решения.

2 Конструкторский раздел

В данном разделе будут спроектированы база данных и приложение.

2.1 Сценарии использования

Необходимо формализовать требуемую функциональность.

2.1.1 Гость

У пользователя с ролью "Гость" есть только самые базовые права - права на просмотр существующие данные, без возможности их как-либо менять или делать сложные запросы. Разрешаемые действия:

1) просмотр всех источников;
2) просмотр всех типов источников;
3) просмотр всех владельцев источников;
4) просмотр всех точек назначения;
5) просмотр всех типов точек назначений;
6) просмотр потока данных за последние n минут.

2.1.2 Аналитик

Функциональность, предоставляемая аналитику расширяет возможности гостя:

1) просмотр потока определенного типа;

2) просмотр потока из определенного интервала;

3) получение суммы трафика от источников.

2.1.3 Администратор

Администратор, помимо предоставляемого аналитику функционала, может управлять аккаунтами пользователей:

1) просмотреть всех пользователей;

2) удалить пользователя;

3) создать пользователя;

4) выдать права пользователю;

5) забрать права у пользователя.

На рисунке 2 приведена use-case диаграмма.

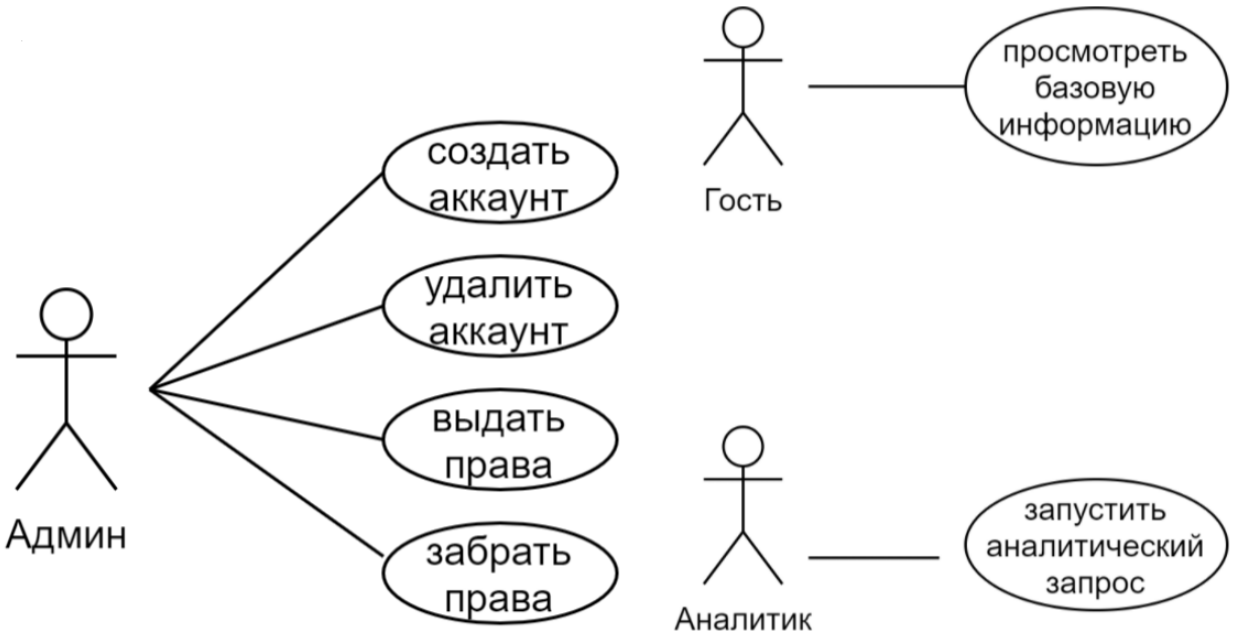


Рисунок 2 – Схема потоков данных системы.

2.2 Проектирование системы

На этапе анализа были выделены три сущности:

- Сенсор
- Коллектор
- Анализатор

В проектируемой системе было решено использовать базу данных и клиентское приложение. На рисунке 3 представлена схема потоков данных системы.

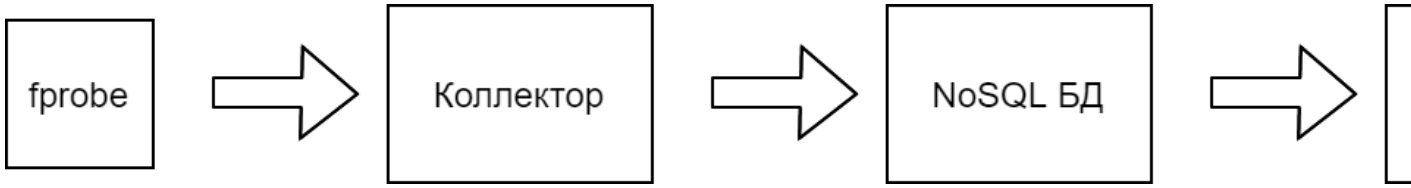


Рисунок 3 – Схема потоков данных системы.

2.3 Проектирование базы данных

Для проектирования базы данных необходимо выделить и описать сущности системы.

2.3.1 Ролевая модель

На уровне базы данных введены следующие роли:

1) guest;
2) analyst;
3) admin.

Права приведенных ролей совпадают с возможностями пользователей этой роли.

2.3.2 Формализация сущностей системы

На рисунке 4 представлена ER-диаграмма системы.

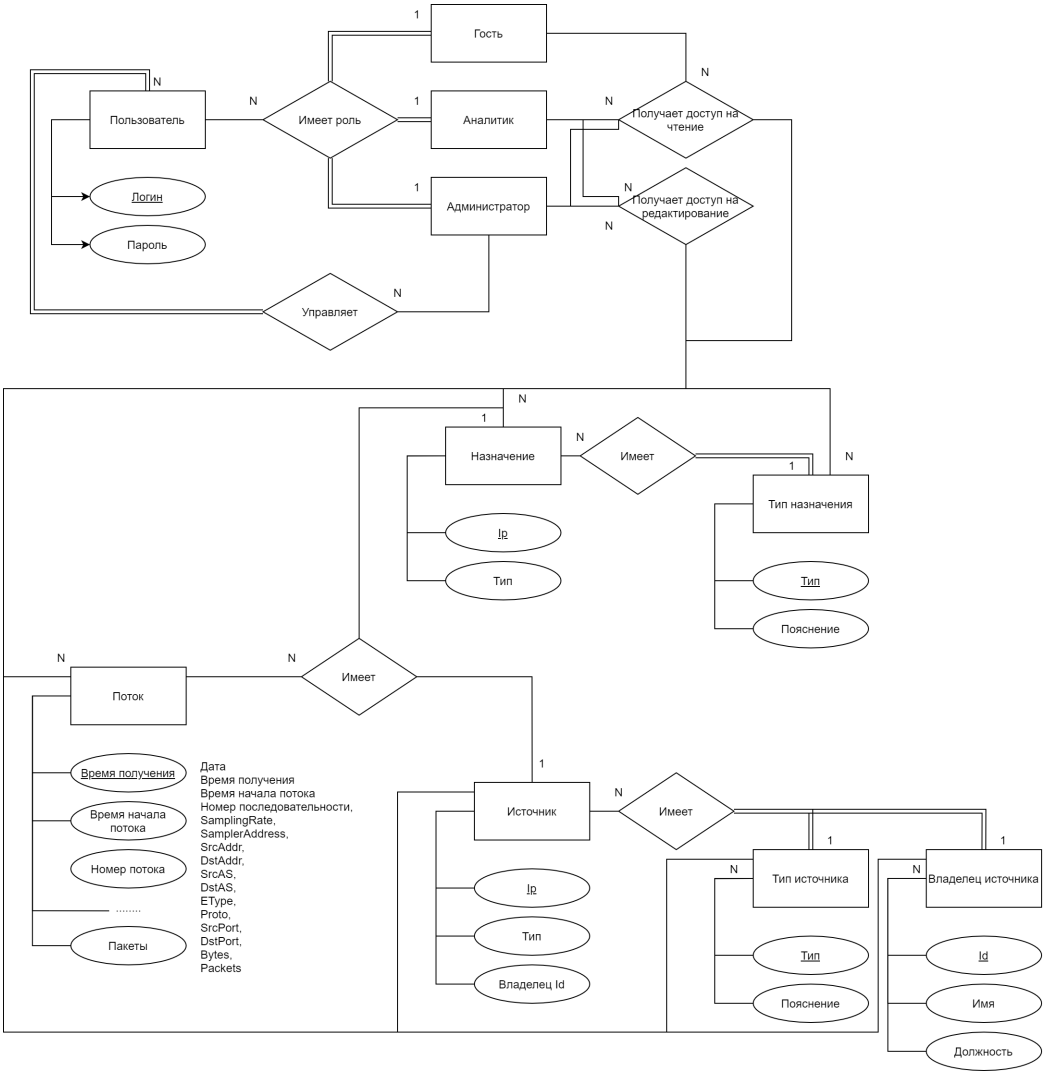


Рисунок 4 – ER-диаграмма системы.

Выделенные сущности:

1) Пользователь - пользователь системы;

2) Администратор, Гость, Аналитик - роли;

3) Поток - поток данных;

4) Источник - источник потока;

5) Тип источника - тип источника;

6) Владелец источника - так как предполагается, что мы находимся во внутренней сети, у источника есть известное нам расположение и владелец;

7) Назначение - место назначение потока данных.

8) Тип назначения - тип точки назначения.

2.3.3 Материализованное представление

Для того, чтобы автоматически сортировать потоки трафика, необходимо создать материализованное представление, способное агрегировать информацию из нескольких таблиц для записи в таблицу агрегированных данных.

2.4 Проектирование приложения

Приложение анализатор NetFlow спроектировано по архитектурному паттерну MVC. Были выделены три компонента:

1. доступа к данным;

2. бизнес-логики;

3. интерфейса.

Компонент доступа к данным предоставляет интерфейс для взаимодействия с базой данных. В компоненте бизнес-логики содержатся контроллеры, использующиеся в компоненте интерфейса.

2.5 Вывод

В данном разделе были рассмотрены сценарии использования, спроектирована система и компоненты системы.

3 Технологический раздел

В данном разделе будут выбраны средства реализации поставленной задачи, создана база данных и ролевая модель, разработаны компоненты и описан порядок работы.

3.1 Выбор средств реализации поставленной задачи

3.1.1 Выбор сенсора

Были выделены две утилиты, предоставляющие возможность сбора и отправки пакетов netflow: fprobe и nProbe. В таблице 3 представлено их сравнение

Таблица 3 – Сравнение netflow и nProbe.

	nProbe	fprobe
Бесплатный	-	+
Возможность работать с различными версиями NetFlow	v5/v9/IPFIX	v1/v5/v7
Мультиплатформенность	Linux, FreeBSD и Windows	Linux
Поддержка IPv4/v6	+	-

Так как утилита fprobe бесплатна и предоставляет всю необходимую функциональность для разработки была выбрана она.

3.1.2 Выбор СУБД

Так как в аналитической части была выбрана модель данных NoSQL будут рассмотрены только NoSQL СУБД.

Одной из известных NoSQL базой данных является Elasticksearch. Это документо-ориентированная база данных, разрабатываемая компанией Elastic и имеющая все плюсы NoSQL базы данных. Является частью так называемого ElastickStack - программного комплекса для записи, хранения и просмотра данных. В ElastickStack или ELK(Elasticksearch, Logstash, Kibana) - Logstash позволяет не терять данные в случае временного выхода из строя Elasticksearch, а Kibana визуализирует их. У Elasticksearch есть open-source версия с урезанной функциональностью под лицензией Apache 2.0, бесплатная версия под версий под лицензией Elastic и несколько коммерческих версий под лицензией Elastic License.

Другая NoSQL база данных Clickhouse является столбцово-ориентированной и её производительность выше чем у Elasticksearch[7]. Разрабатывается компанией Yandex, является open-source под лицензией Apache 2.0

В таблице 4 представлено их сравнение.

Таблица 4 – Сравнение Elasticksearch и ClickHouse.

	Elasticksearch	ClickHouse
NoSQL	+	+
Парадигма	Документо-ориентированная	Столбцово-ориентированная
Бесплатная	-/+	+
Open-source	-/+	+
Скорость обработки больших данных	Большая	Ещё больше

Ввиду урезанной в open-source версии Elasticsearch функциональности (например, отсутствие ролевой модели), большей скорости работы с данными в ClickHouse, и специфики данных (в системе используются структурированные данные) был выбран Clickhouse.

3.1.3 Выбор брокера

Для агрегации данных с сенсоров нужна прослойка между базой данных и сенсорами, агрегирующая данные. В качестве такой прослойки было выбрано использовать брокер сообщений.

Можно выделить два брокер сообщений для анализа. RedisMS и Kafka.

Kafka - распределенный горизонтально масштабируемый отказоустойчивый программный брокер сообщений. Приложения (генераторы) посылают сообщения (записи) на узел Kafka (брокер), и указанные сообщения

обрабатываются другими приложениями, так называемыми потребителями. Указанные сообщения сохраняются, а потребители подписываются для получения новых сообщений. Kafka гарантирует, что все сообщения будут упорядочены именно в той последовательности, в которой поступили. Kafka не отслеживает, какие записи считываются потребителем и после этого удаляются, а просто хранит их в течение заданного периода времени. Потребители сами опрашивают Kafka, не появилось ли у него новых сообщений, и указывают, какие записи им нужно прочесть[8]. Между ClickHouse и Kafka существует нативная интеграция.

Приложения (publishers) посылают сообщения на узел RabbitMQ (exchange), при этом RabbitMQ отсылает обратно приложениям подтверждение, что сообщение получено. Получатели (consumers) постоянно соединены с RabbitMQ по TCP и ждут, когда RabbitMQ протолкнет (push) им сообщения. Получатели подтверждают получение сообщения или сообщают о неудаче. Если доставка неудачна, то RabbitMQ проталкивает сообщение до тех пор, пока оно не будет доставлено. После успешной доставки сообщение удаляется из очереди[8].

В таблице 5 представлено их сравнение.

Таблица 5 – Сравнение RabbitMQ и Kafka.

	Redis	Kafka
Масштабируем	+	+
Отказоустойчив	+	+
Поддерживает UDP протокол получения пакетов	-	-
Существует коллектор для записи Netflow	-	+
Поддерживает интеграцию с ClickHouse	-	+

Так как Kafka поддерживает интеграцию в ClickHouse без сторонних модулей и существует модуль для записи Netflow в Kafka (Goflow)[**?**], был выбран брокер сообщений Kafka.

3.1.4 Выбор инструментов для реализации приложения

В качестве языка программирования был выбран язык C#, так как он поддерживает парадигму ООП, имеет большой набор библиотек, легкий в использовании конструктор интерфейса для WinForms[9].

В качестве среды разработки была выбрана "Microsoft Visual Studio 2019"[10], поскольку она имеет много полезных возможностей при написании кода на C#[10].

3.1.5 Остальные выбранные технологии и инструменты

В качестве инструмента развертывания был выбран Docker, так как он позволяет быстро и без установки сторонних модулей развернуть приложение на новой машине.

Разработка ведется на Linux Manjaro 5.11 И Windows 10.

3.1.6 Диаграмма потока данных

Для общего представления о системе на рисунке 5 представлена диаграмма потока данных с учетом используемых технологий.

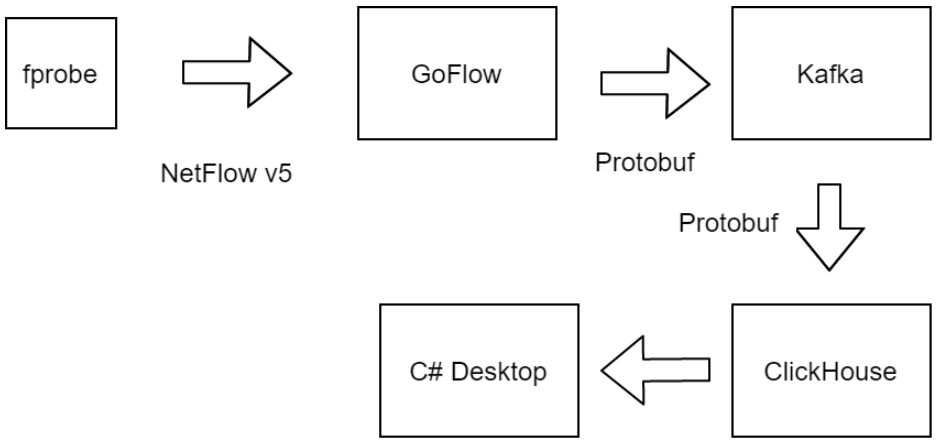


Рисунок 5 – Схема потоков данных системы.

3.2 Создание базы данных

Специфика Clickhouse наложила некоторые ограничения. Clickhouse не поддерживает PrimaryKey и ForeignKey, не поддерживает уникальность значений в таблице, отсутствует поддержка транзакций и другие особенности[5]. В связи с этим нельзя сказать, что одна таблица ссылается на другую и

нельзя построить диаграмму БД. На рисунке 6 представлены сущности БД.

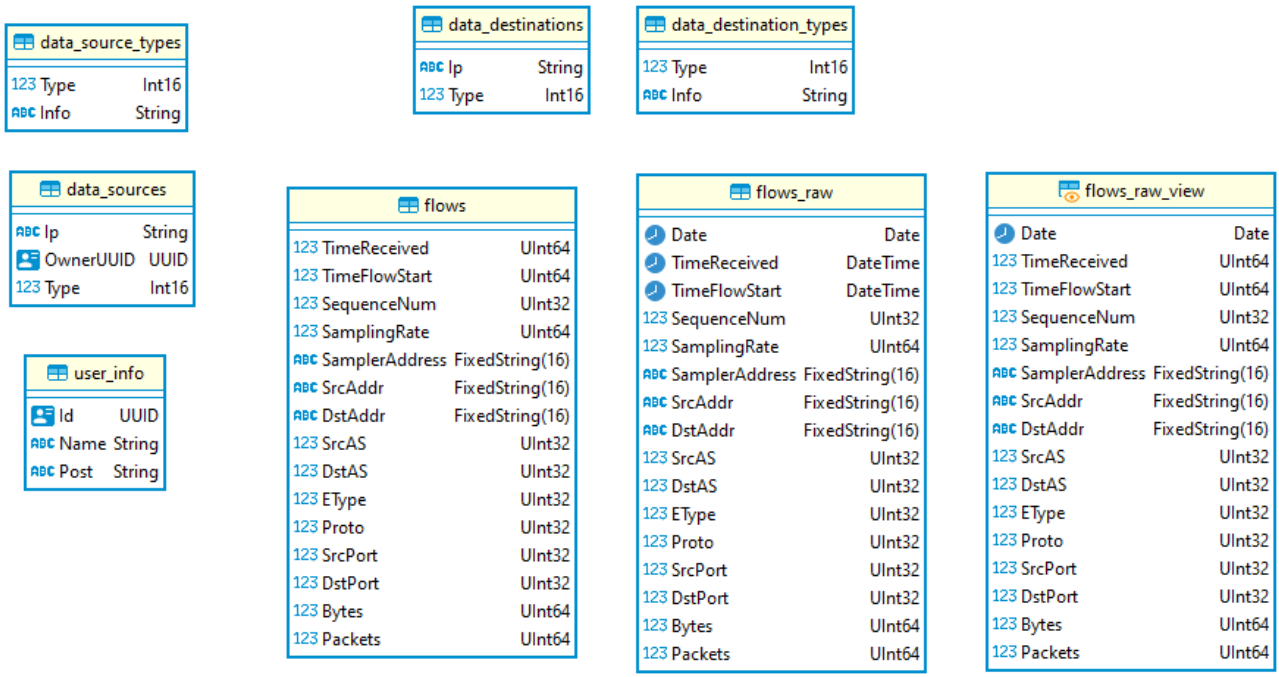


Рисунок 6 – Диаграмма БД.

3.2.1 Описание полей базы данных

В таблице flows представлена часть структуры Netflow пакета, значения полей которого определяются стандартом Netflow. Из полученного пакета будут выбраны следующие поля:

- TimeReceived - Время получения пакета
- TimeFlowStart - Время начала потока
- SequenceNum - Время пакета в текущей записываемой последовательности

- SamplingRate - Пакеты для семплирования
- SamplerAddres - Адресс семплера потока
- SrcAddr - Адресс источника пакета
- DstAddr - Адрес назначения пакета
- SrcAS - BGP автономной системы
- DstAS - BGP автономной системы
- EType - Протокол третьего уровня
- Proto - Протокол четвертого уровня
- SrcPort - Порт источника
- DstPort - Порт назначения
- Bytes - Размер потока в байтах
- Packets - Количество агрегированных пакетов

Материализованное представление flow_raw_view и таблица flows_raw дублируют эти поля, добавляя поле Date - Дату получения пакета.

Остальные таблицы хранят информацию для категорирования входящих пакетов. Таблица data_sources - хранит адрес источника пакета, тип источника и UUID владельца источника. Таблица data_sources_type - хранит типы источников. Таблица data_destinations - хранит адрес назначения пакета и его тип. Таблица data_destinations_type - хранит типы назначений.

3.2.2 Типы таблиц

В базе данных используется три типа таблиц.

Движок Kafka

В ClickHouse Существует понятие движка(engine), который определяет:

- как и где хранятся данные, куда их писать и откуда читать;
- какие запросы поддерживаются и каким образом;
- конкурентный доступ к данным;
- использование индексов, если есть;
- возможно ли многопоточное выполнение запроса;
- параметры репликации данных.

Движок Kafka определяет, что данные берутся из Kafka и записываются в таблицу. Рассмотри создание таблицы на листинге 3.2.2.

```
0 CREATE TABLE IF NOT EXISTS flows
1 (
2     TimeReceived UInt64,
3     TimeFlowStart UInt64,
4     SequenceNum UInt32,
5     SamplingRate UInt64,
6     SamplerAddress FixedString(16),
7     SrcAddr FixedString(16),
8     DstAddr FixedString(16),
```

```
9     SrcAS UInt32,
10    DstAS UInt32,
11    EType UInt32,
12    Proto UInt32,
13    SrcPort UInt32,
14    DstPort UInt32,
15    Bytes UInt64,
16    Packets UInt64
17 )
18 ENGINE = Kafka
19 SETTINGS kafka_broker_list = '192.168.1.53:9092',
20  kafka_topic_list = 'flows',
21  kafka_group_name = 'clickhouse',
22  kafka_format = 'Protobuf',
23  kafka_schema = './flow.proto:FlowMessage';
```

В строках 2-17 определяется структура таблицы, в строке 18 указывается, что используется движок Kafka. В строках 19-23 определяются следующие настройки:

- kafka_broker_list - список адресов брокеров Kafka
- kafka_topic_list - название топика Kafka;
- kafka_group_name - название группы при подключении;
- kafka_format - формат получаемых данных;
- kafka_schema - схема получаемых данных.

Материализованное представление

Таблица flows_raw_view является материализованным представлением. Рассмотрим создание таблицы на листинге 3.2.2.

```
0  CREATE MATERIALIZED VIEW IF NOT EXISTS default.flows_raw_view TO default.
       flows_raw
1  (
2      `Date` Date,
3      `TimeReceived` UInt64,
4      `TimeFlowStart` UInt64,
5      `SequenceNum` UInt32,
6      `SamplingRate` UInt64,
7      `SamplerAddress` FixedString(16),
8      `SrcAddr` FixedString(16),
9      `DstAddr` FixedString(16),
10     `SrcAS` UInt32,
11     `DstAS` UInt32,
12     `EType` UInt32,
13     `Proto` UInt32,
14     `SrcPort` UInt32,
15     `DstPort` UInt32,
16     `Bytes` UInt64,
17     `Packets` UInt64
18 ) AS
19 SELECT
20     toDate(TimeReceived) AS Date, *
21 FROM default.flows;
```

В строке 1 указывается, что создается материализованное представление, результат которого будет вставлен в таблицу flows_raw. MATERIALIZED - Материализованное выражение, всегда вычисляется.

Строки 2-18 описывают строение. Далее идет описание SELECT из default.flows.

Обычная таблица

Таблица flows_raw создается как обычная таблица. Рассмотрим создание таблицы на листинге 3.2.2.

```
0  CREATE TABLE IF NOT EXISTS flows_raw
1  (
2      Date Date,
3      TimeReceived DateTime,
4      TimeFlowStart DateTime,
5      SequenceNum UInt32,
6      SamplingRate UInt64,
7      SamplerAddress FixedString(16),
8      SrcAddr FixedString(16),
9      DstAddr FixedString(16),
10     SrcAS UInt32,
11     DstAS UInt32,
12     EType UInt32,
13     Proto UInt32,
14     SrcPort UInt32,
15     DstPort UInt32,
16     Bytes UInt64,
17     Packets UInt64
18 )
19 ENGINE = MergeTree
20 PARTITION BY Date
21 ORDER BY TimeReceived;
```

В данном случае используется движок MergeTree. Движок MergeTree, а также другие движки этого семейства (*MergeTree) — это наиболее функциональные движки таблиц ClickHouse.

Основная идея, заложенная в основу движков семейства MergeTree следующая. Когда у вас есть огромное количество данных, которые должны быть

вставлены в таблицу, вы должны быстро записать их по частям, а затем объединить части по некоторым правилам в фоновом режиме. Этот метод намного эффективнее, чем постоянная перезапись данных в хранилище при вставке[11].

Строка PARTITION BY Date указывает, что создается партицирование по ключу Date. ClickHouse поддерживает операции над партициями, которые намного эффективнее обычных. Так же, если задан ключ партицирования в запросе, ClickHouse сразу выберет нужную партицию, что также увеличивет эффективность выполнения запросов.

Создание остальных таблиц приведено в приложении А.

В данном случае таблица flows напрямую берет данные из Kafka, но не хранит в себе данные предыдущего запроса к Kafka, а только последнего. Таблица же flows_raw, дублируя flows, не удаляет старые данные при получении новых. Синхронизировать запись в flows_raw при добавлении во flows помогает flows_raw_view - материализованное представление, ещё одна особенность Clickhouse. В Clickhouse отсутствуют триггеры и функции, определяемые пользователем, но вместо этого существуют материализованные представления, которые работают схожим образом с триггером добавления.

3.3 Разработка компонентов

При разработке приложения были встречены трудности, так же обусловленные выбором Clickhouse.

3.3.1 Компонент доступа к данным

В C# есть фрэймворк EntityFramework и его модули, предоставляющие ORM для многих популярных баз данных, тем самым существенно упрощая процесс разработки. Но для Clickhouse нет готовой ORM, поэтому компонент доступа данным использует query билдеры. На рисунке 7 представлена UML-диаграмма компонента доступа к данным.

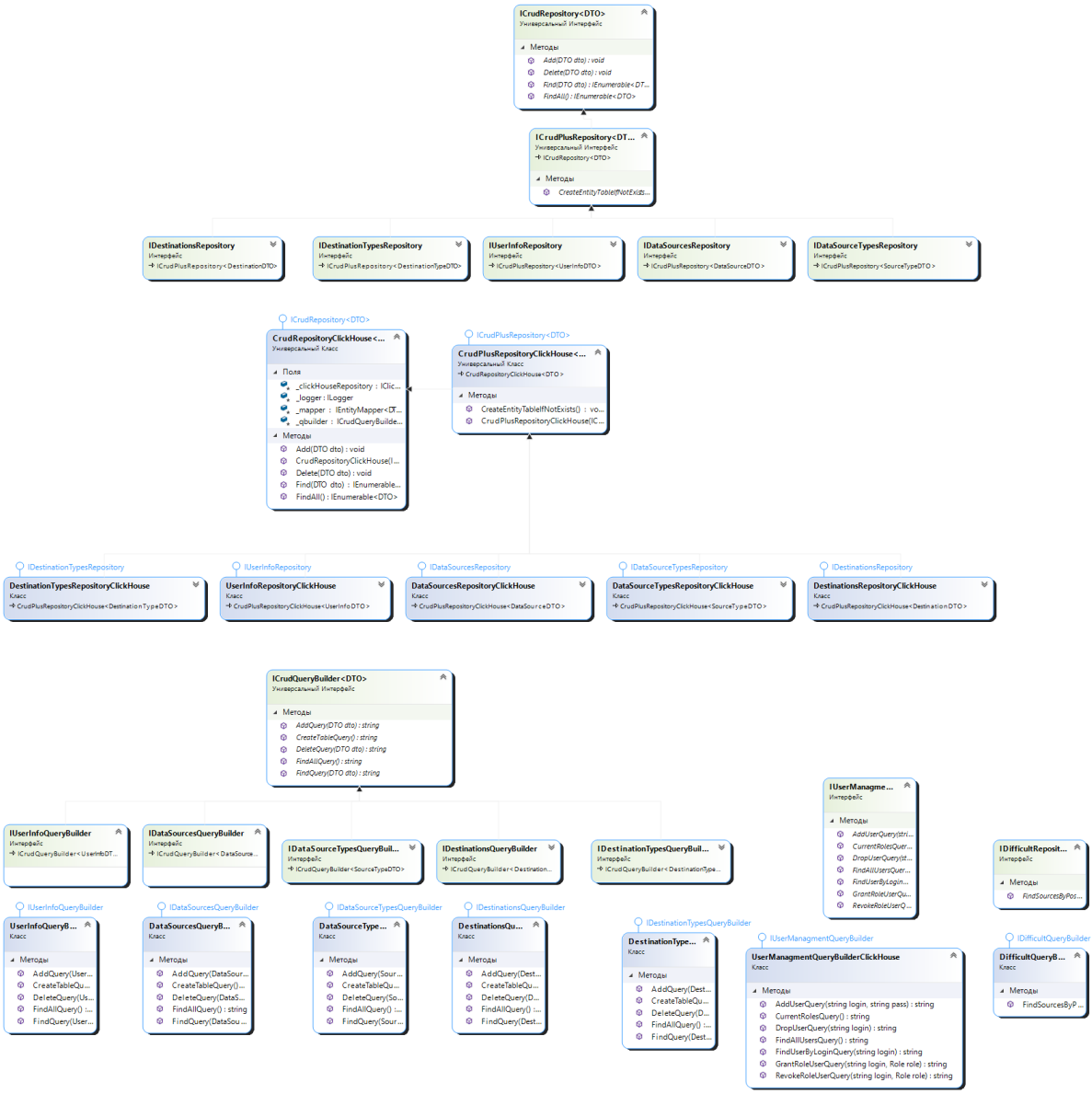


Рисунок 7 – Компонент доступа к данным.

3.3.2 Компонент бизнес-логики

На рисунке 8 представлена UML-диаграмма компонента бизнес-логики.

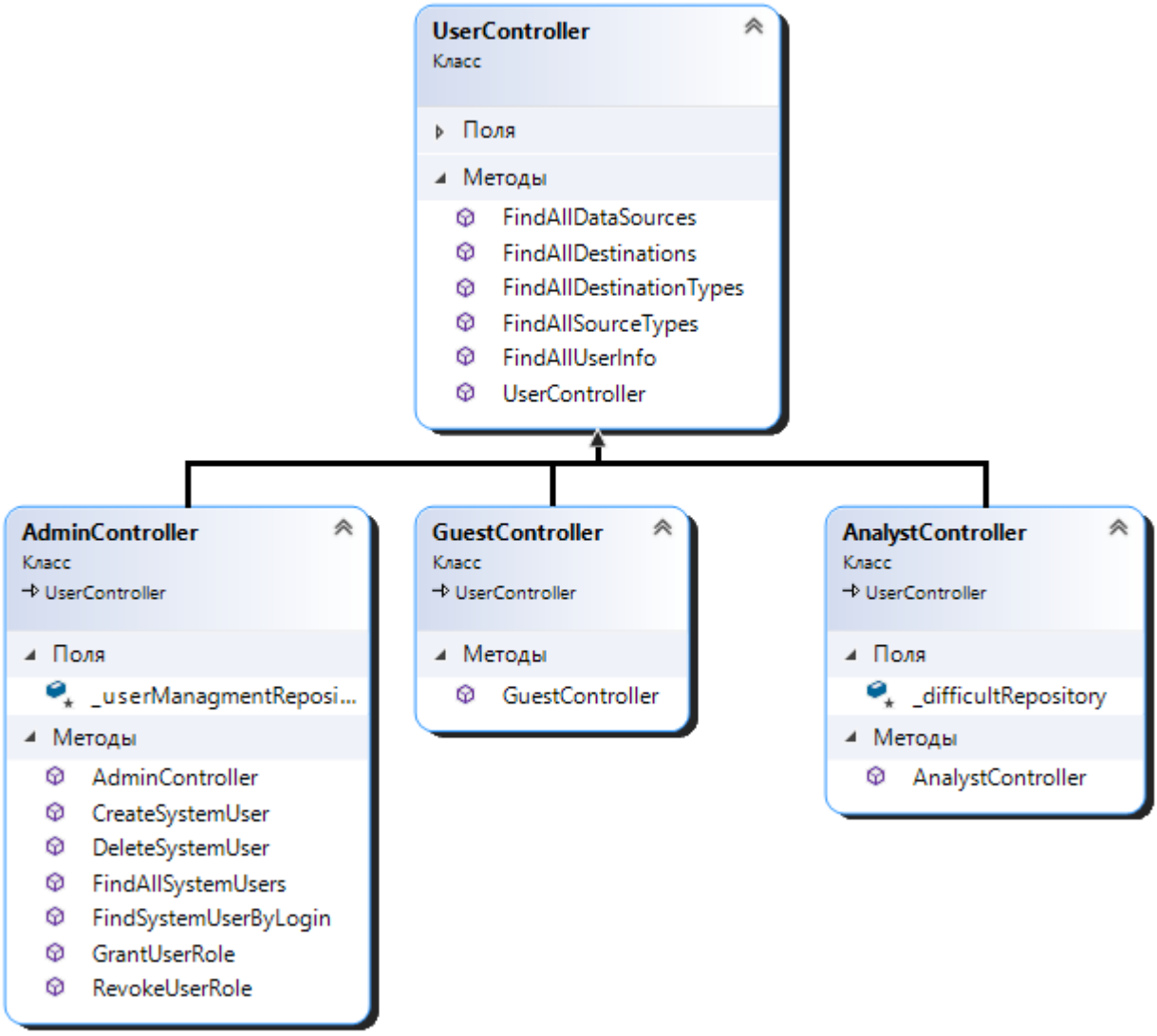


Рисунок 8 – Компонент бизнес-логики.

3.4 Интерфейс приложения

На рисунках ниже показан пользовательский интерфейс.

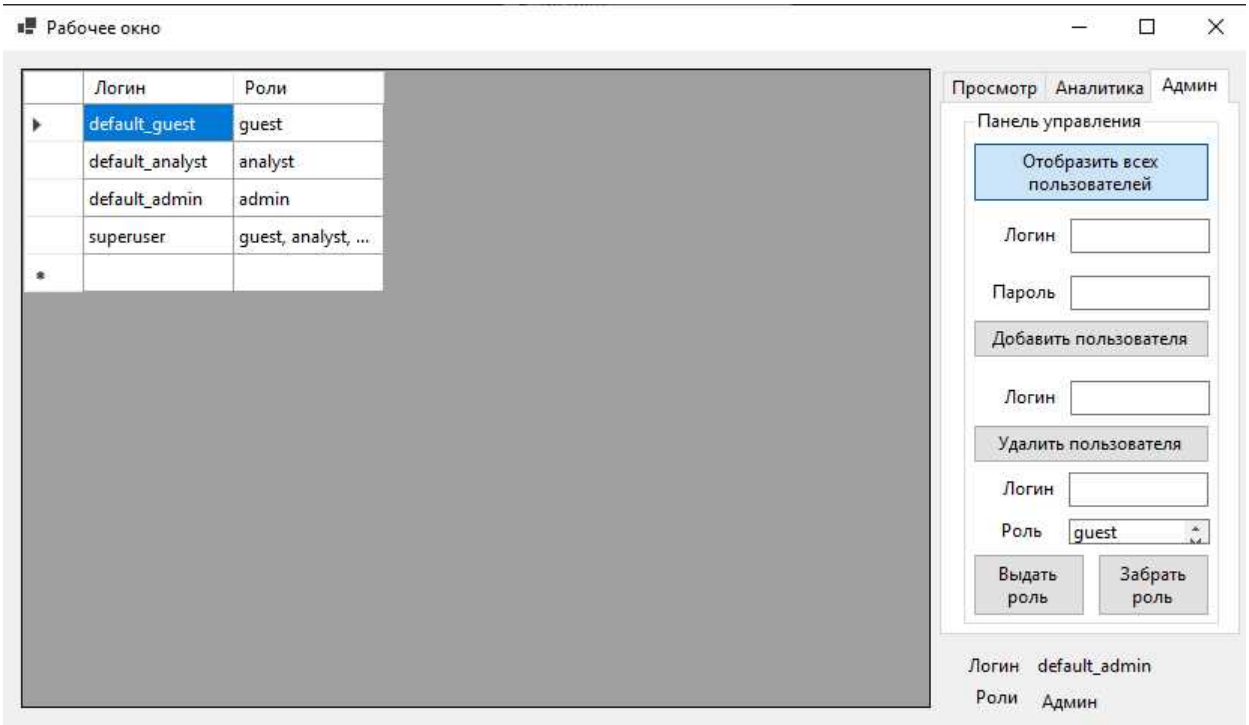


Рисунок 9 – Вкладка адмна.

На этой вкладке можо получить доступ к функциям адмна, добавлению и удалению пользователей, выдачу и отзыв ролей.

Рисунок 10 – Общедоступная вкладка.

На этой вкладке можо получить доступ к функциям гостя. Каждый пользователь может это сделать. К функциям гостя относят только базовые функции для просмотра данных.

3.5 Docker

При развертовании системы был использован докер, что позволит разворачивать и настраивать систему не зависимо от машины, упростит масштабирование.

3.6 Вывод

В данном разделе были выбраны средства реализации поставленной задачи, создана база данных, разработано приложение.

Заключение

В процессе выполнения курсовой работы задание было выполнено следующее:

- проведен анализ предметной области;
- были выбраны подходящие инструменты для решения задачи;
- реализован программный комплекс;

Цель курсовой работы достигнута, все задачи выполнены.

В процессе разработки была заложена возможность масштабирования системы. В качестве дальнейшего развития можно предложить:

- введение системы в эксплуатацию;
- добавление возможности управлять сенсором через приложение;
- доработка UI приложения и логики добавления новых запросов.

Список литературы

[1] Introduction to Cisco IOS NetFlow [Электронный ресурс] URL: https://www.cisco.com/c/en/us/products/collateral/ios-nx-os-software/ios-netflow/prod_white_paper0900aecd80406232.html (дата обращения: 09.06.2021).

[2] Wireshark [Электронный ресурс] URL: https://www.wireshark.org/ (дата обращения: 09.06.2021).

[3] Man page of tcpdump [Электронный ресурс] URL: https://www.tcpdump.org/manpages/tcpdump.1.html (дата обращения: 09.06.2021).

[4] Elasticksearch as NoSQL [Электронный ресурс] URL: https://www.elastic.co/blog/found-elasticsearch-as-nosql (дата обращения: 09.06.2021).

[5] ClickHouse : Введение [Электронный ресурс] URL: https://clickhouse.tech/docs/ru/introduction/distinctive-features/ (дата обращения: 09.06.2021).

[6] ClickHouse : Kafka [Электронный ресурс] URL: https://clickhouse.tech/docs/en/engines/table-engines/integrations/kafka/ (дата обращения: 09.06.2021).

[7] ClickHouse vs Elasticsearch [Электронный ресурс] URL: https://altinity.com/faqs/clickhouse-and-elasticsearch-faqs (дата обращения: 09.06.2021).

[8] Kafka vs Redis [Электронный ресурс] URL: https://www.upsolver.com/blog/kafka-versus-rabbitmq-architecture-performance-use-case (дата обращения: 10.06.2021).

[9] Документация по C# [Электронный ресурс] URL: https://docs.microsoft.com/ru-ru/dotnet/csharp/ (дата обращения: 09.06.2021).

[10] Документация по семейству продуктов Visual Studio [Электронный ресурс] URL: https://docs.microsoft.com/ru-ru/visualstudio/?view=vs-2019 (дата обращения: 09.06.2021).

[11] ClickHouse : MergeTree [Электронный ресурс] URL:https://clickhouse.tech/docs/ru/engines/table-engines/mergetree-family/mergetree (дата обращения: 10.06.2021).

Приложение А.

Создание таблиц базы данных.

```
0  CREATE TABLE IF NOT EXISTS data_sources (
1                          Ip String,
2                          OwnerUUID UUID,
3                          Type Int16
4                          )
5                          ENGINE=MergeTree()
6                          ORDER BY (Ip);
7
8  CREATE TABLE IF NOT EXISTS data_source_types (
9                          Type Int16,
10                         Info String
11                         )
12                         ENGINE=MergeTree()
13                         ORDER BY (Type);
14
15    CREATE TABLE IF NOT EXISTS data_destinations (
16                         Ip String,
17                         Type Int16
18                         )
19                         ENGINE=MergeTree()
20                         ORDER BY (Ip);
21
22    CREATE TABLE IF NOT EXISTS data_destination_types (
23                         Type Int16,
24                         Info String
25                         )
26                         ENGINE=MergeTree()
27                         ORDER BY (Type);
28
29    CREATE TABLE IF NOT EXISTS user_info (
30                         Id UUID,
31                         Name String,
```

```
32                          Post String
33                          )
34                          ENGINE=MergeTree()
35                          ORDER BY (Id);
```

Приложение Б. Презентация

Разработка системы сбора и анализа информации о трафике

Студент: Хетагуров П.К
Научный руководитель: Павельев А.А

Цель и задачи

Цель работы - разработать систему сбора и анализа информации о трафике. Для достижения поставленной цели необходимо решить следующие задачи:

- проанализировать предметную область;
- спроектировать программный комплекс;
- реализовать спроектированную систему.

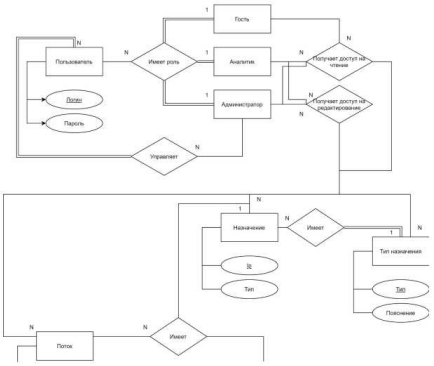
2/10

Анализ существующих решений

	Wireshark	tcpdump
Возможность просмотра информации о трафике	+	+
Наличие графического интерфейса	+	-
Возможность записи данных в бд	-	-
Возможность отправки данных	-	-
Работа с удаленными интерфейсами	-	-
Разграничение сбора и анализа информации	-	-

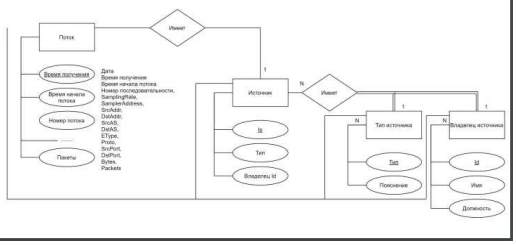
3/10

ER-диаграмма



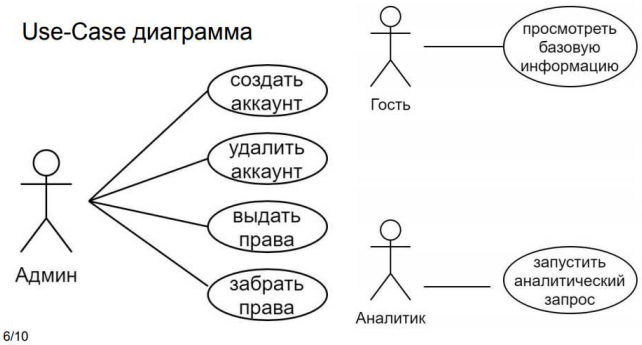
4/10

Продолжение



5/10

Use-Case диаграмма



6/10

Выбор СУБД

	Реляционные	Elasticsearch	Clickhouse
Поддержка SQL	+	+/-	+/-
Фокус на масштабируемости	-	+	+
Высокая скорость обработки больших данных	-	+	+
Поддержка связанности	+	-	-
Поддержка хранимых процедур и триггеров	+	-	-

7/10

Технологический стек

Проект представляет собой комплекс модулей, предоставляющий возможности сбора, хранения и анализа информации о трафике. Проект разработан с использованием следующих технологий:

- Clickhouse
- Kafka
- fprobe
- Go
- C#
- Docker

8/10

Заключение

Цель курсовой работы достигнута. В ходе работы были решены задачи анализа существующих решений, были формализованы сущности, спроектирована система, спроектированы и реализованы модули системы.

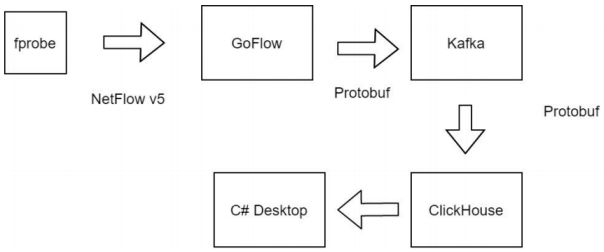
С использованием C# было реализовано пользовательское приложение.

Стек Clickhouse-Kafka вместе с Goflow реализуют модуль записи и хранения данных.

Утилита fprobe собирает данные для записи.

9/10

Поток данных



Дальнейшее развитие

При проектировании и реализации системы были заложены возможности для дальнейшего расширения. В будущем можно улучшить UI/UX, расширить поддерживаемые протоколы, расширить Clickhouse в объединение нескольких хранилищ в кластер, увеличить количество сетевых узлов, с которых собираются данные. Также возможна реализация управления сенсорами с общего клиентского приложения.

10/10

Дальнейшее развитие

При проектировании и реализации системы были заложены возможности для дальнейшего расширения. В будущем можно:

- улучшить UI/UX
- расширить поддерживаемые протоколы
- расширить Clickhouse в объединение нескольких хранилищ в кластер
- увеличить количество сетевых узлов, с которых собираются данные.

Также возможна реализация управления сенсорами с общего клиентского приложения.

10/10