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.

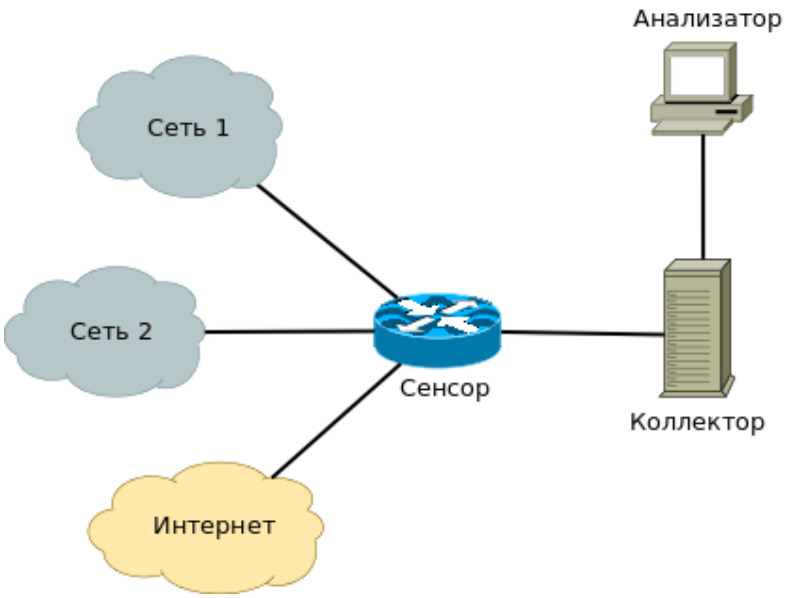


Рисунок 1 – Архитектура NetFlow.

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.

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.

Утилита fprobe широко применяется в качестве сенсора, так как он прост в использовании.

Стек ELK(Elasticksearch, Logstash, Kibana) применяется в качестве коллектора и анализатора. Logstash позволяет не терять данные в случае временного выхода из строя Elasticksearch, а Kibana визуализирует данные, выступая в роли анализатора. Стоит заметить, что БД Elasticksearch это документо-ориентированная NoSQL база данных, что делает её очень эффективной при работе с большими данными[4].

Другая NoSQL база данных Clickhouse является столбцово-ориентированной и её производительность выше чем у Elasticksearch[7].

Так же в качестве Коллектора может использоваться Kafka, предоставляющий ту же функциональность что и Logstash, но имеющий более гибкий инструментарий для масштабирования.

У Clickhouse есть нативная интеграция с Kafka, что, вкупе с преимуществами этих сервисов по отдельности и требованиями к системе, делает выбор этой связки оправданым для решения поставленной задачи[6].

1.5 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 к определенным точкам назначения.

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.

2.2 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 Проектирование базы данных

Необходимо выделить и описать сущности и ролевую модель.

2.3.1 Формализация сущностей системы

На рисунке 2 представлена ER-диаграмма системы.

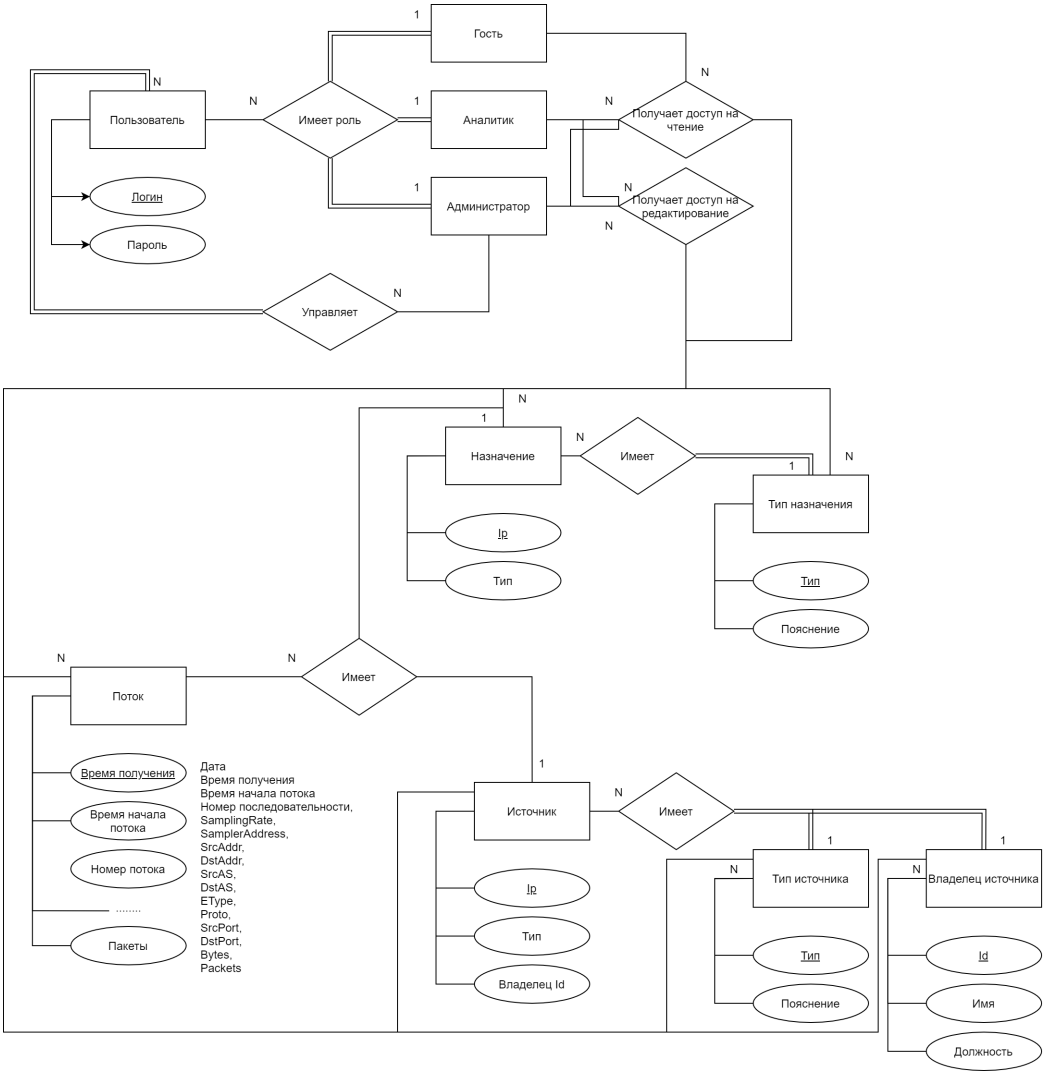


Рисунок 2 – ER-диаграмма системы.

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.

2.3.2 Функции и триггеры

Для того, чтобы автоматически сортировать потоки трафика, необходимо создать триггер на добавление в таблицу потоков. Функции же нужны для аналитических запросов.

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 база данных и спроектировано приложение.

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 Средства реализации поставленной задачи

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[8].

В качестве среды разработки была выбрана "Microsoft Visual Studio 2019"[9], поскольку она имеет много полезных возможностей при написании кода на C#[9].

В качестве сенсора будет использоваться fprobe так как он прост в использовании. В качестве коллектора будет применяться связка Kafka + Clickhouse из-за преимуществ, описанных в аналитической части. В качестве коллектора для Kafka будет использован Goflow, преобразующий все реализации протокола Netflow к формату protobuf, который будет использоваться в дальнейшем.

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

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 3 представлены сущности БД. В приложении А приведен листинг создания БД.

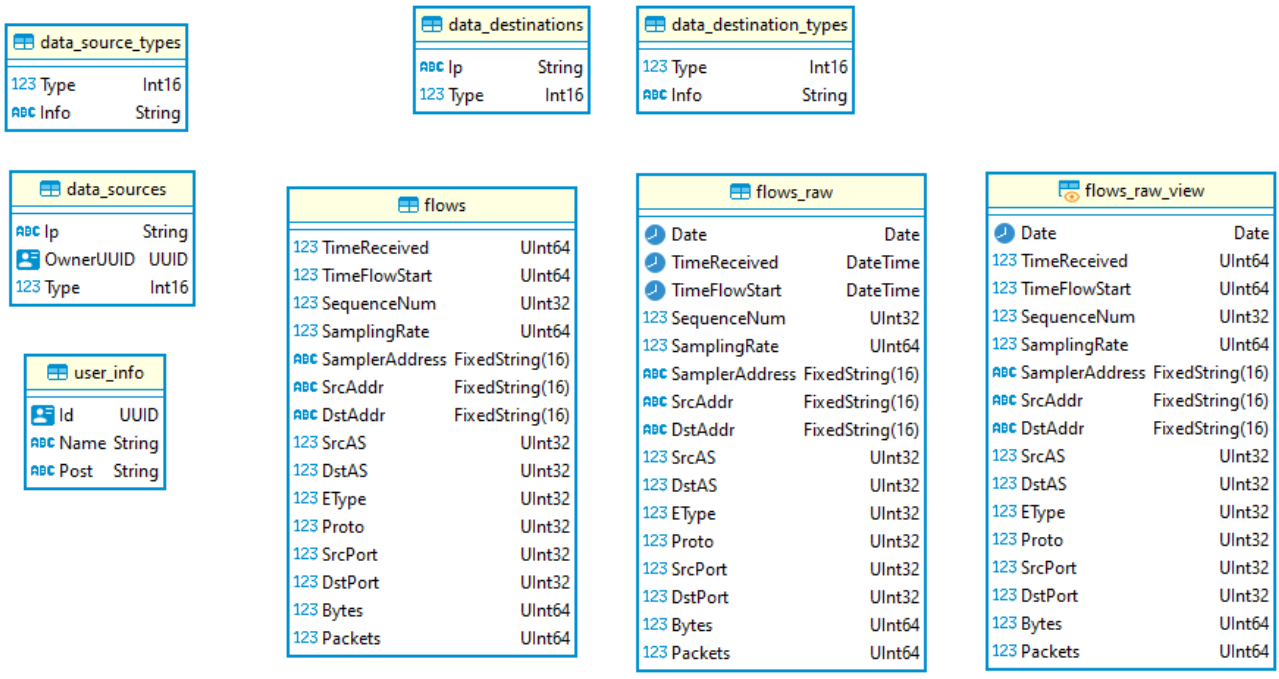


Рисунок 3 – Диаграмма БД.

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 4 представлена UML-диаграмма компонента доступа к данным.

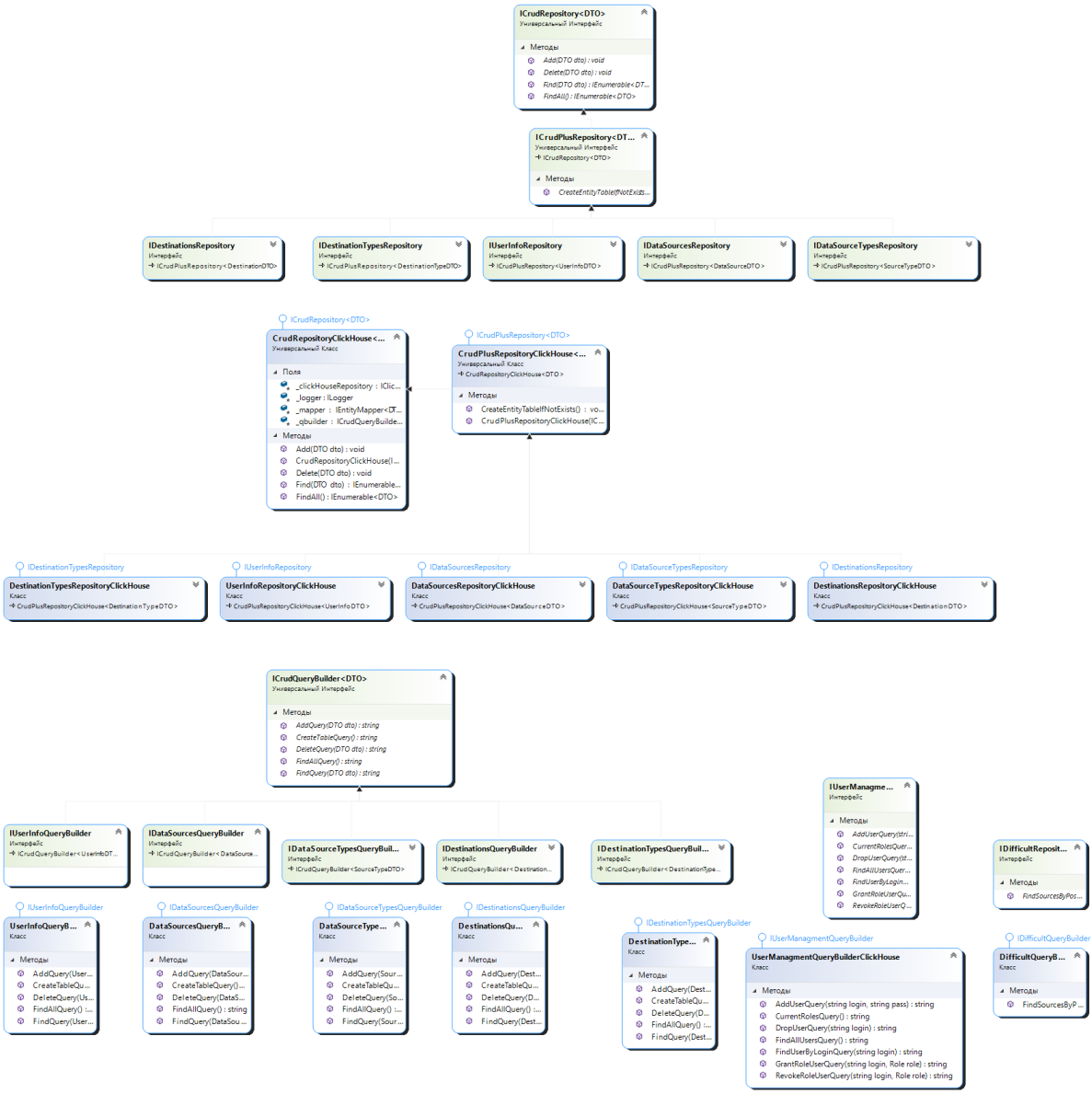


Рисунок 4 – Компонент доступа к данным.

3.3.2 Компонент бизнес-логики

На рисунке 5 представлена UML-диаграмма компонента бизнес-логики.

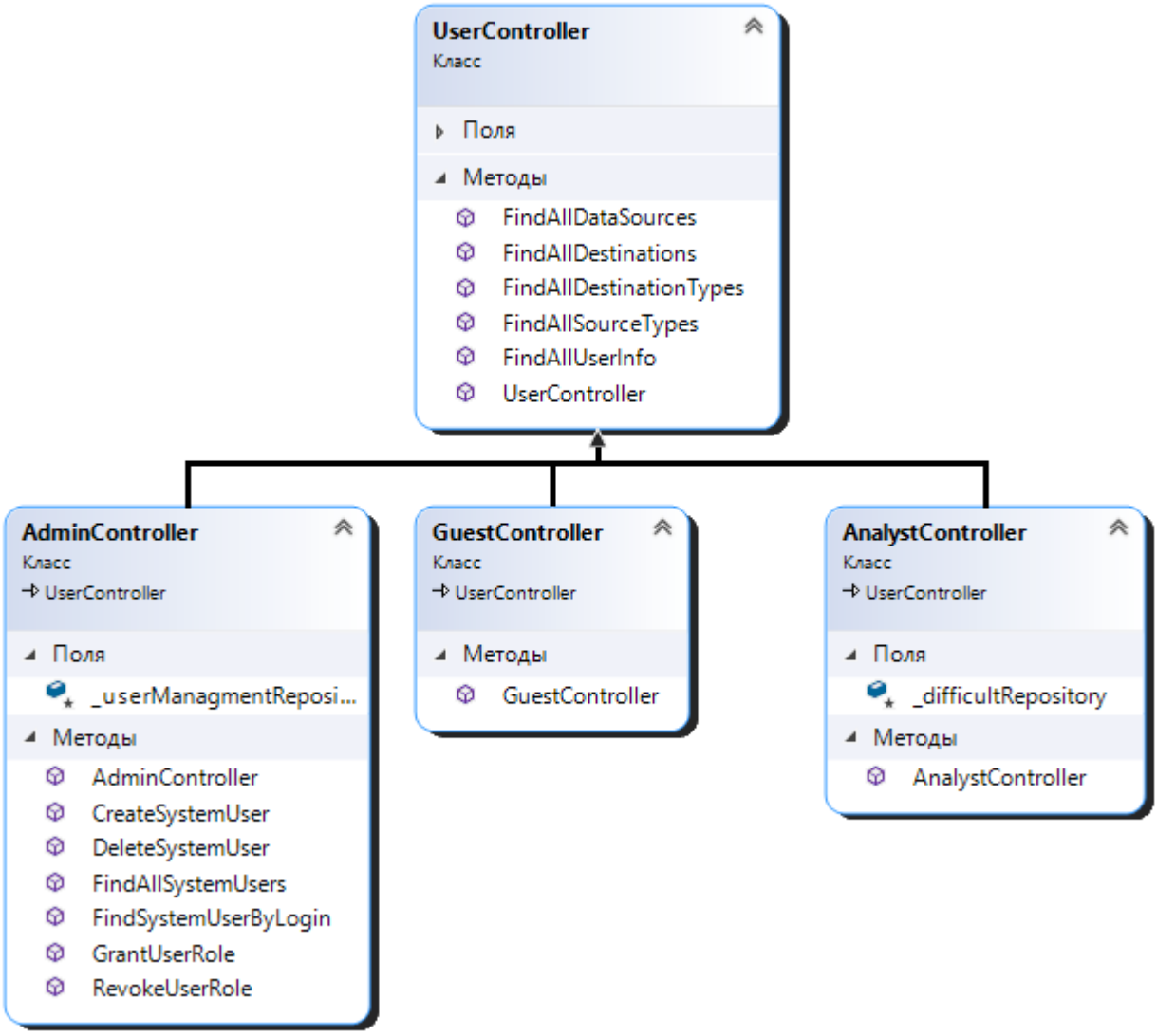


Рисунок 5 – Компонент бизнес-логики.

3.4 Интерфейс приложения

На рисунках ниже показан пользовательский интерфейс.

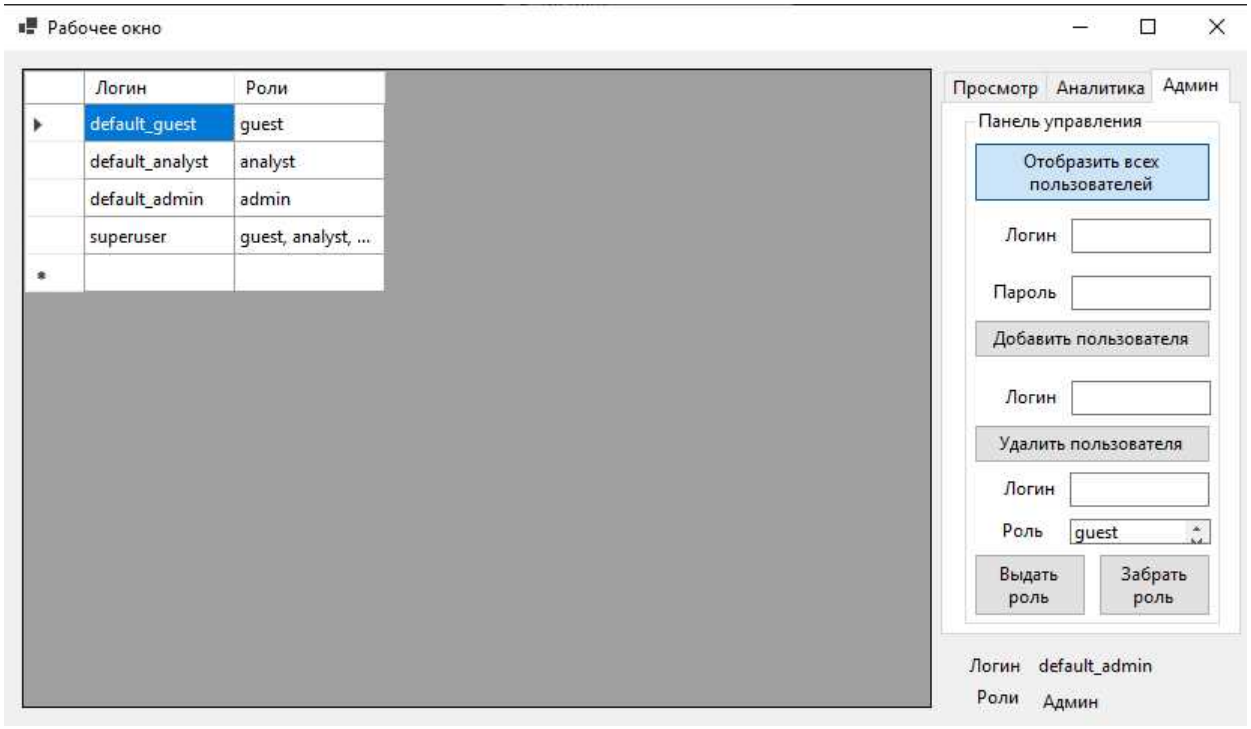


Рисунок 6 – Вкладка админа.

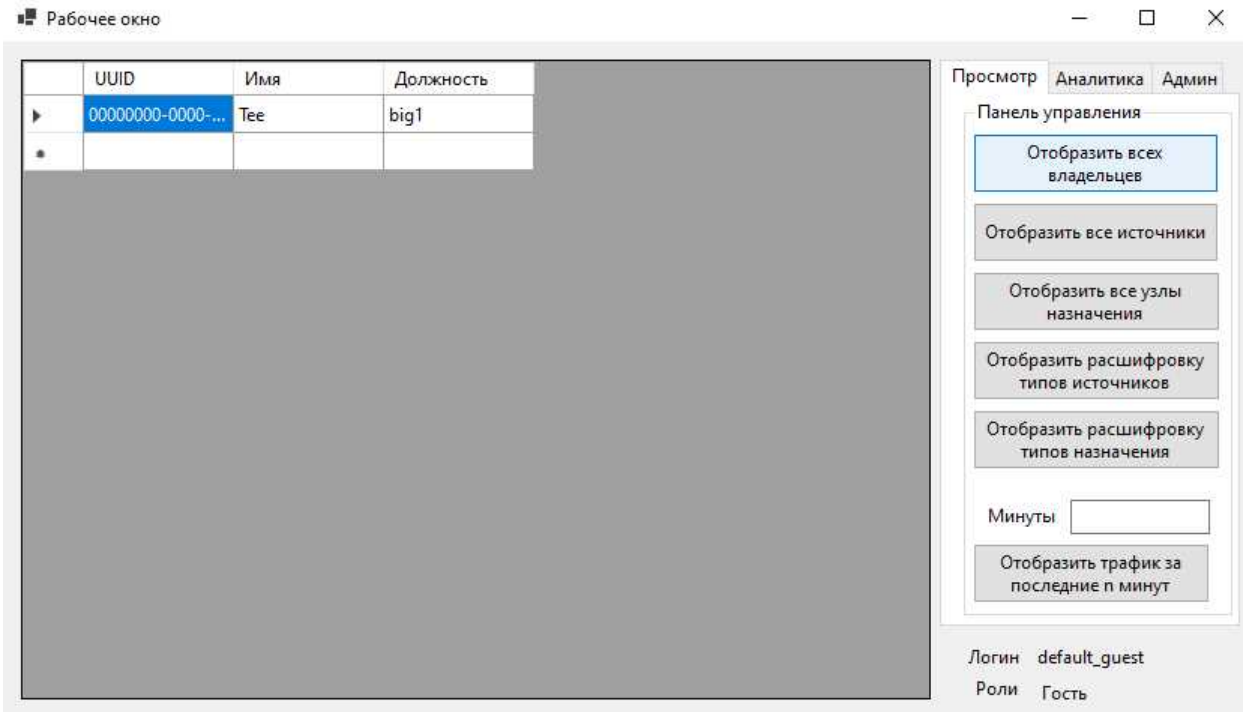


Рисунок 7 – Общедоступная вкладка.

3.5 Вывод

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

Заключение

В процессе выполнения курсовой работы задание было формализовано, был проведен анализ предметной области, были выбраны подходящие инструменты для решения задачи и был спроектирован и реализован программный комплекс.

Цель курсовой работы достигнута.

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

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

Приложение А.

Создание таблиц базы данных.

```
0
1  CREATE TABLE IF NOT EXISTS flows
2  (
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 )
19 ENGINE = Kafka
20 SETTINGS kafka_broker_list = '192.168.1.53:9092',
21  kafka_topic_list = 'flows',
22  kafka_group_name = 'clickhouse',
23  kafka_format = 'Protobuf',
24  kafka_schema = './flow.proto:FlowMessage';
25
26 CREATE TABLE IF NOT EXISTS flows_raw
27 (
28     Date Date,
29     TimeReceived DateTime,
30     TimeFlowStart DateTime,
31     SequenceNum UInt32,
```

```
32      SamplingRate UInt64,
33      SamplerAddress FixedString(16),
34      SrcAddr FixedString(16),
35      DstAddr FixedString(16),
36      SrcAS UInt32,
37      DstAS UInt32,
38      EType UInt32,
39      Proto UInt32,
40      SrcPort UInt32,
41      DstPort UInt32,
42      Bytes UInt64,
43      Packets UInt64
44  )
45  ENGINE = MergeTree
46  PARTITION BY Date
47  ORDER BY TimeReceived
48  SETTINGS index_granularity = 8192;
49
50  CREATE MATERIALIZED VIEW IF NOT EXISTS default.flows_raw_view TO default.
        flows_raw
51  (
52      `Date` Date,
53      `TimeReceived` UInt64,
54      `TimeFlowStart` UInt64,
55      `SequenceNum` UInt32,
56      `SamplingRate` UInt64,
57      `SamplerAddress` FixedString(16),
58      `SrcAddr` FixedString(16),
59      `DstAddr` FixedString(16),
60      `SrcAS` UInt32,
61      `DstAS` UInt32,
62      `EType` UInt32,
63      `Proto` UInt32,
64      `SrcPort` UInt32,
65      `DstPort` UInt32,
```

```
    `Bytes` UInt64,
    `Packets` UInt64
) AS
SELECT
    toDate(TimeReceived) AS Date, *
FROM default.flows;


CREATE TABLE IF NOT EXISTS data_sources (
                        Ip String,
                        OwnerUUID UUID,
                        Type Int16
                        )
                        ENGINE=MergeTree()
                        ORDER BY (Ip);

CREATE TABLE IF NOT EXISTS data_source_types (
                        Type Int16,
                        Info String
                        )
                        ENGINE=MergeTree()
                        ORDER BY (Type);

   CREATE TABLE IF NOT EXISTS data_destinations (
                        Ip String,
                        Type Int16
                        )
                        ENGINE=MergeTree()
                        ORDER BY (Ip);

   CREATE TABLE IF NOT EXISTS data_destination_types (
                        Type Int16,
                        Info String
                        )
                        ENGINE=MergeTree()
```

```
101                         ORDER BY (Type);
102
103     CREATE TABLE IF NOT EXISTS user_info (
104                         Id UUID,
105                         Name String,
106                         Post String
107                         )
108                         ENGINE=MergeTree()
109                         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.

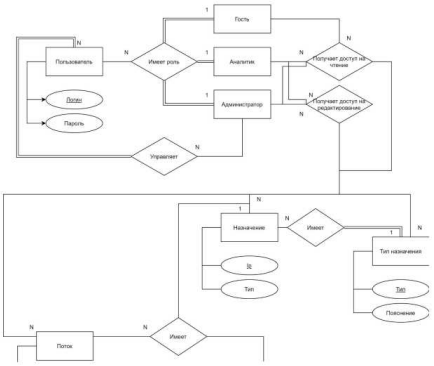
2/10

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	-	-

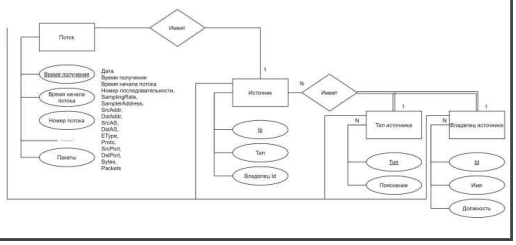
3/10

ER-диаграмма



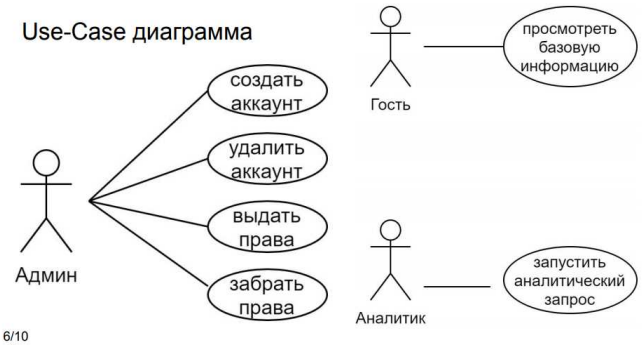
4/10

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

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