ИНФРАСТРУКТУРА ЭЛЕКТРОННОГО ПРАВИТЕЛЬСТВА

ВЫПОЛНЕНИЕ РАБОТ ПО РАЗВИТИЮ ТИПОВОГО ТИРАЖИРУЕМОГО ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ ВИТРИН ДАННЫХ

Руководство по установке Типового ПО «Витрина данных НСУД»

Версия 1.14.0

Листов 138

Москва, 2024

СОДЕРЖАНИЕ

Аннотация

Настоящий документ является руководством по установке модернизированного программного обеспечения «Витрина данных НСУД» (далее – программа).

Перед началом работы необходимо ознакомиться с документом 83219291.62.01.11.A003.РСП.01.1 «Руководство системного программиста» на ПО «Витрина данных НСУД», разработанным в рамках выполнения государственного контракта № 0173100007520000024_144316 от 13 ноября 2020 года, на выполнение работ по модернизации федеральной государственной информационной системы «Единая информационная платформа Национальной системы управления данными» в части выделения программного обеспечения компонента «Витрина данных» и его доработки, необходимой для подготовки к публикации на условиях открытой лицензии.

В разделе «Общие сведения» указаны назначение и возможности Программы.

В разделе «Подготовка к установке» описаны предварительные действия на аппаратных и программных средствах перед установкой Программы.

В разделе «Настройка на состав технических средств» приведены требования к серверам программы, перечень сетевых портов, используемых аппаратными средствами.

В разделе «Установка программы» описаны порядок установки программных компонентов Программы и действия по установке Программы.

В разделе «Проверка программы» описаны действия по проверке корректной работы Программы.

В разделе «Обновление программы» описаны действия по обновлению программы на новую версию.

Оформление программного документа «Руководство по установке» произведено по требованиям ЕСПД (ГОСТ 19.101-77, ГОСТ 19.103-77, ГОСТ 19.104-78, ГОСТ 19.105-78, ГОСТ 19.106-78, ГОСТ 19.503-79, ГОСТ 19.604-78).

1 ОБЩИЕ СВЕДЕНИЯ О ПРОГРАММЕ

1.1 Обозначение и наименование программы

Полное наименование: «Типовое тиражируемое программное обеспечение Витрина данных НСУД».

Условное обозначение: ПО «Витрина данных».

1.2 Назначение программы

Национальная система управления данными (далее – НСУД) представляет собой систему, состоящую из взаимосвязанных элементов информационно-технологического, организационного, методологического, кадрового и нормативно-правового характера и обеспечивающую достижение целей и выполнение задач, обозначенных в Концепции Национальной системы управления данными, утвержденной распоряжением Правительства Российской Федерации от 3 июня 2019 года № 1189-р.

НСУД предназначена для управления информацией, содержащейся в информационных системах органов и организаций государственного сектора, а также в информационных ресурсах, созданных в целях реализации полномочий органов и организаций государственного сектора (далее – государственные данные) и для осуществления информационного обмена между Поставщиками и Получателями данных, присоединившимися к НСУД (далее – Участники НСУД).

Управление процессами информационного обмена между Участниками НСУД осуществляется средствами федеральной государственной информационной системы «Единая информационная платформа Национальной системы управления данными» (далее – ФГИС «ЕИП НСУД»).

Для передачи данных между Участниками НСУД используется среда взаимодействия НСУД, состоящая из Системы межведомственного электронного взаимодействия 3.0 (далее – СМЭВ) и (или) подсистемы обеспечения доступа к данным СМЭВ (далее – ПОДД СМЭВ) (СМЭВ 4.0), обеспечивающих транспорт и процессинг данных, а также агентов ПОДД СМЭВ, устанавливаемых на стороне Участников НСУД.

Для формирования и (или) для получения данных с использованием среды взаимодействия НСУД необходим комплекс программных и технических средств в составе информационно-телекоммуникационной инфраструктуры участника НСУД, описываемое в данном документе «Витрина данных НСУД», но возможно и применение «Витрина данных НСУД». Данный документ описывает применение именно ПО среды взаимодействия НСУД.

Программа «Витрина данных НСУД» является частью НСУД и предназначена для загрузки публикуемых данных в отдельную БД на стороне Поставщика данных. Программа представляет собой типовое программное обеспечение, устанавливаемое на стороне поставщиков/потребителей данных.

1.3 Возможности программы

В настоящий момент реализовано две конфигурации Программы:

- Стандарт;
- Лайт.

Возможности конфигурации Стандарт

Программа обеспечивает выполнение следующих задач:

- описание логической модели данных;
- настройка программы и структуры таблиц в ее БД для хранения публикуемых данных;
- загрузка и хранение публикуемых данных в БД программы;
- извлечение данных из внешних систем (внешних ИС по отношению к Витрине данных НСУД);
- выполнение запросов в соответствии с протоколом ПОДД через механизмы ПОДД СМЭВ:
    - поддержка протокола коммуникации Агента СМЭВ4;
    - предоставление публикуемых данных (в т. ч. BLOB-объектов и/или с использованием табличных параметров);
    - генерация формируемых документов на основании публикуемых данных;
    - репликация публикуемых данных (в качестве витрины-поставщика);
    - получение реплицируемых данных (в качестве витрины-получателя).
- обмен в соответствии с протоколом СМЭВ3:
    - подключение к СМЭВ3 как информационной системы участника взаимодействия;
    - обработку запросов на предоставление публикуемых данных (видов сведений), в т.ч. BLOB-объектов;
    - инициативная рассылка оповещений об обновлении публикуемых данных.
- публикация конечных точек API для обработки запросов с использованием спецификации OpenAPI версии 3;
- предоставление публикуемых данных информационным системам с использованием интерфейса REST-запросов;
- восстановление данных в непротиворечивое состояние после сбоев;
- поддержка языка SQL;
- журналирование событий функциональных блоков;
- мониторинг информации о работоспособности экземпляра Программы.

Возможности конфигурации Лайт

Программа обеспечивает выполнение следующих задач:

- автоматическая настройка взаимосвязей между компонентами программы;
- автоматический запуск всех необходимых компонентов программы после установки;
- автоматическая настройка витрины и структуры ее таблиц на основании содержимого XML-файла, загружаемого через пользовательский web-интерфейс;
- выгрузка шаблона через графический интерфейс (для упрощения процесса подготовки загружаемых данных);
- загрузка данных в витрину:
    - через графический интерфейс;
    - REST API;
    - файловый обмен.

– настройка параметров работы витрины через графический интерфейс;
– выполнение запросов на предоставление данных в соответствии с протоколом ПОДД через механизмы СМЭВ ПОДД.

1.4 Компоненты программы

Состав компонентов дистрибутива программы приведен в разделе Состав компонентов в дистрибутиве документа «Техническое описание программы ПО «Витрина данных НСУД»».

2 ПОДГОТОВКА К УСТАНОВКЕ

2.1 Предварительные действия

1. Примечание:
2. Установка Программы производится в закрытом контуре (без необходимости доступа к сети Интернет).

Предварительные действия для конфигурации Стандарт

1. Установить на серверы одну из поддерживаемых операционных систем (см. раздел Установка операционной системы).
2. Проверить настройки Firewall и отключить при необходимости (см. раздел Настройка межсетевого экрана).
3. Выключить SELinux (см. раздел Отключение SELinux (только для CentOS)).
4. Указать соответствующий местоположению сервера часовой пояс (см. раздел Выбор часового пояса).
5. Проверить, что на всех серверах установлен сервис синхронизации времени (см. раздел Установка сервиса синхронизации времени).
6. Проверить, что имена хостов (FQDN) серверов могут получать IP по имени со всех машин (см. раздел Настройка имен хостов (FQDN) на серверах);
7. Установить/обновить Java SE Development Kit 17.0.7 (см. раздел Установка Java SE Development Kit 17.0.7)

   Дополнительно устанавливаются:
   - компонент сбора данных запросов и ответов Витрины (см. раздел Установка компонента сбора данных запросов и ответов Витрины данных);
   - сервис журналирования (см. раздел Настройка сервиса журналирования);
   - подсистему мониторинга (см. раздел Настройка сервиса мониторинга).

Предварительные действия для конфигурации Лайт

1. Установить на серверы одну из поддерживаемых операционных систем (см. раздел Установка операционной системы).
2. Создать пользователя **datamart** с правами `sudo` (см. раздел Создание пользователя).
3. Проверить настройки Firewall и отключить при необходимости (см. раздел Настройка межсетевого экрана).
4. Выключить SELinux (см. раздел Отключение SELinux (только для CentOS)).
5. Указать соответствующий местоположению сервера часовой пояс (см. раздел Выбор часового пояса).
6. Настроить SSH-подключение к серверу, на котором планируется установка (см. раздел Подключение к серверу через SSH-клиент PuTTY).
7. Создать приватный и публичный SSH-ключ для пользователя *datamart* (см. раздел Создание SSH-ключей).
8. Скопировать публичный ключ на сервер (см. раздел Копирование ssh-ключей на сервер).
9. Скопировать архив программы (см. раздел Копирование архива программы).
10. Установить Docker (см. раздел Установка Docker).
11. Установить библиотеку *python-docker-py* (см. раздел Установка библиотеки python-docker-py).

12. Настроить логирование (см. раздел Настройка логирования в Docker).
13. Перезапустить Docker (см. раздел Перезапуск Docker).
14. Добавить пользователя *datamart* в группу *docker* (см. раздел Добавление пользователя в группу docker).
15. Настроить Ansible (см. раздел Настройка Ansible).

2.1.1 Установка операционной системы

Программа может работать на одной из операционных систем:

- Centos 7.9;
- Astra Linux Special Edition 1.7 (уровень защищенности «Воронеж»);
- Alt 8 SP Server;
- РЕД ОС, версии 7.2.

Подробная инструкция по установке операционной системы Centos 7.9 приведена на официальном сайте разработчика: https://docs.centos.org/en-US/centos/install-guide/.

Подробная инструкция по установке операционной системы Astra Linux Special Edition 1.7 приведена на официальном сайте разработчика: https://astralinux.ru/products/astra-linux-special-edition/documents-astra-se/

Подробная инструкция по установке операционной системы Alt 8 SP Server приведена на официальном сайте разработчика: https://www.basealt.ru/alt-8-sp-sertifikat-fstehk/docs

Подробная инструкция по установке операционной системы РЕД ОС, версии 7.2 приведена в документе «Руководство администратора по РЕД ОС 7.2»

2.1.2 Создание пользователя datamart

Внимание:
Данный пункт только для **конфигурации Лайт**

Для установки программы конфигурации Лайт рекомендуется создать отдельного пользователя, для этого следует выполнить следующие действия:

- создать пользователя **datamart**;
- отключить для пользователя пароль при вызове `sudo` (необходимо для автоматической установки);
- добавление пользователя в группу администраторов `sudo`.

2.1.2.1 Создание пользователя

Чтобы создать пользователя **datamart** и установить для него пароль, выполните команды:

```
sudo useradd datamart
sudo password for user:
sudo passwd datamart
sudo password for user:
Changing password for user datamart.
New password:
Retype new password:
```

После успешных действий система выведет сообщение:

```
passwd: all authentication tokens updated successfully.
```

2.1.2.2 Отключение пароля

Чтобы отключить пароль `sudo` для пользователя **datamart**, надо добавить в настройки (пользователя или группы) директиву `NOPASSWD`. Для этого последовательно выполните команду:

```
sudo visudo
```

В открывшемся конфигурационном файле, с помощью команд редактора `vim`

Для CentOS и РЕД ОС

Отредактируйте следующие записи:

```
datamart ALL=(ALL) NOPASSWD: ALL
```

Для АЛЬТ ОС

Отредактируйте следующие записи:

```
# uncomment line
WHEEL_USERS ALL=(ALL) NOPASSWD: ALL
```

Сохраните изменения и закройте файл.

2.1.2.3 Добавление пользователя в группу администраторов

Чтобы добавить пользователя в группу администраторов, у которых есть права выполнения команды `sudo`, выполните следующую команду:

```
sudo usermod -aG wheel datamart
```

Для проверки вы можете переключиться в учетную запись **datamart** и вывести список содержимого директории `/root`, которое обычно доступно только для пользователя `root user`:

```
su -datamart
Password:
sudo ls -la /root
```

2.1.3 Настройка межсетевого экрана

2.1.3.1 Для CentOS и RedOS

Для корректной установки потребуется отключить службу `FirewallD` операционной системы CentOS.

Что просмотреть текущий статус работы приложения используйте команду `firewall-cmd`:

```
sudo firewall-cmd --state
```

В случае, если служба `FirewallD` запущена, команда выше выведет следующее сообщение:

```
running
```

Вы можете временно остановить службу `FirewallD` для этого выполните следующую команду:

```
sudo systemctl stop firewalld
```

Следует учитывать, что данная команда только временно отключит службу, при последующей перезагрузке служба `FirewallD` снова будет запущена.

Чтобы отключить автоматический запуск службы `FirewallD` при загрузке операционной системы выполните команду:

```
sudo systemctl disable firewalld
```

После отключения проверьте, что статус службы изменился на `not running`, для этого выполните команду:

```
sudo firewall-cmd --state
not running
```

2.1.3.2 Для AltOS

По умолчанию выключен. Никаких действий не требуется.

2.1.4 Отключение SELinux (только для CentOS)

Для корректной установки CentOS необходимо отключить SELinux, для этого выполните следующие действия:

1. Проверьте параметры запуска `SELinux` при загрузке системы. Для этого выполните следующую команду:

```
cat /etc/selinux/config
```

2. Если параметр `SELINUX` имеет значение `enforcing`, отключите запуск `SELinux` при загрузке системы. Для этого следует в файле `/etc/selinux/config` указать значение `SELINUX=disabled` и перезагрузите сервер. `SELinux` будет отключен.

Открыть и отредактировать файл `/etc/selinux/config` можно с помощью редактора Vim, для этого выполните команду:

```
sudo vi /etc/selinux/config
```

1. Проверьте, что служба отключена. Для этого выполните команду

```
sestatus
```

В ответ вы должны получить

```
SELinux status: disabled
```

2.1.5 Выбор часового пояса

Проверьте, что установлен нужный часовой пояс. В нашем случае, на команду `timedatectl`, должна выводиться строка `Time zone: Europe/Moscow (MSK, +0300)`.

Пример команды:

```
timedatectl
```

Пример ответа:

```
Local time: Mon 2021-12-20 12:06:39 MSK
Universal time: Mon 2021-12-20 09:06:39 UTC
RTC time: Mon 2021-12-20 09:06:49
Time zone: Europe/Moscow (MSK, +0300)
NTP enabled: n/a
NTP synchronized: no
RTC in local TZ: no
DST active: n/a
```

Если результат отличается, укажите соответствующий местоположению сервера часовой пояс.

Пример команды для московского часового пояса:

```
sudo timedatectl set-timezone Europe/Moscow
```

2.1.6 Установка сервиса синхронизации времени

Внимание:
Данный пункт только для **конфигурации Стандарт**

Для корректной установки программы необходимо убедиться, что на всех серверах установлен сервис синхронизации времени.

2.1.7 Настройка имен хостов (FQDN) на серверах

Внимание:
Данный пункт только для **конфигурации Стандарт**

Для корректной установки программы необходимо проверить, что имена хостов (FQDN) серверов могут взаимно получать IP по имени со всех машин. Имена хостов меняются согласно документации установленной ОС.

2.1.8 Установка Java SE Development Kit 17.0.7

Внимание:
JДанный пункт только для **конфигурации Стандарт**

Установка Java SE Development Kit 17.0.7 осуществляется согласно официальной документации: https://www.oracle.com/java/technologies/javase/jdk17-archive-downloads.html

2.1.9 Подключение к серверу через SSH-клиент PuTTY

Внимание:
Данный пункт только для **конфигурации Лайт**

Настройку SSH-подключения к серверу можно выполнить, используя клиент удалённого доступа - PuTTY.

2.1.10 Создание SSH-ключей

Внимание:
Данный пункт для **конфигурации Лайт**

Внимание:
Действия этого раздела необходимо выполнять под созданной учетной записью **datamart** (см. раздел Создание пользователя).

Для переключения на учетную запись **datamart** выполните команду:

```
sudo su - datamart
```

Для подключения Ansible к серверу по протоколу SSH необходимо создать SSH-ключи.

Для аутентификации пользователя на сервере используются два ключа:

- приватный;
- публичный.

Чтобы сгенерировать SSH-ключи для операционной системы Linux, выполните следующие действия:

1. Откройте терминал на компьютере и выполните команду:

```
ssh-keygen -t rsa
```

Следует оставить все значения по умолчанию.

3. Примечание:
4. Если команда `ssh-keygen` не найдена, установите пакет `openssh`.

1. После выполнения команды будет предложено указать имена файлов, в которые будут сохранены ключи и ввести пароль для закрытого ключа. По умолчанию используется имя `id_rsa`, ключи будут созданы в директории `~/.ssh`.

   Публичная часть ключа будет сохранена в файле с названием `<имя_ключа>.pub`.
3. Будет выведено следующее сообщение:

```
Generating public/private rsa key pair.
Enter file in which to save the key (/home/datamart/.ssh/id_rsa).
```

4. Нажмите клавишу **Enter**. После этого ключ будет сохранен в указанную директорию по умолчанию. Далее вам будет предложено ввести кодовое слово для дополнительной защиты ключа. Вы можете пропустить данный шаг и нажать **Enter**.
5. На запрос указать кодовое слово, не вводя его нажмите клавишу **Enter**.

Процедура создания ключей завершена, ключи сохранены в директории `~/.ssh/ `` в файлах ``id_rsa` и `id_rsa.pub`.

2.1.11 Копирование SSH-ключей на сервер

Внимание:

Данный пункт для **конфигурации Лайт**

Внимание:

Действия этого раздела необходимо выполнять под созданной учетной записью **datamart** (см. раздел Создание пользователя).

Для переключения на учетную запись **datamart** выполните команду:

```
sudo su - datamart
```

Для копирования SSH-ключей в список разрешенных ключей выполните команды:

```
cat ~/.ssh/id_rsa.pub >> ~/.ssh/authorized_keys
chmod 600 ~/.ssh/authorized_keys
```

2.1.12 Копирование архива программы

Внимание:

Данный пункт для **конфигурации Лайт**

1. Для загрузки на сервер файла с архивом программы используйте SFTP-клиент (например, WinSCP или Filezilla). Для авторизации используйте логин и пароль учетной записи администратора (**datamart**) созданной при установке ОС (см. Раздел 2.2.2). Загрузите файл с архивом программы в домашнюю директорию администратора (`~/`).
2. Подключитесь по SSH к серверу (см. раздел Подключение к серверу через SSH-клиент

PuTTY), используя логин и пароль учетной записи администратора.

3. Переместите файл с архивом программы в домашнюю директорию пользователя **datamart** командой:

```
mv ~/dtm-lite-1.13.1.tgz /home/datamart/
```

где,

- `dtm-lite-1.13.1.tgz` - название архива программы.
- `datamart` - имя пользователя.

5. Примечание:

6. Обратите внимание, что название и версия файла с архивом может отличаться в зависимости от версии программы.

2.1.13 Распаковка архива

Внимание:

Данный пункт для **конфигурации Лайт**

Внимание:

Действия этого раздела необходимо выполнять под созданной учетной записью **datamart** (см. раздел Создание пользователя).

Для переключения на учетную запись **datamart** выполните команду:

```
sudo su - datamart
```

Чтобы распаковать архив, выполните команду:

```
tar -xzvf dtm-lite-1.13.1.tgz
```

7. Примечание:

8. Обратите внимание, что название и версия файла с архивом может отличаться в зависимости от версии программы.

2.1.14 Установка Docker

2.1.14.1 Установка Docker в CentOS

Внимание:

Действия этого раздела необходимо выполнять под созданной учетной записью **datamart** (см. раздел Создание пользователя).

Для переключения на учетную запись **datamart** выполните команду:

```
sudo su - datamart
```

Полную инструкцию по установке Docker можно просмотреть на официальном сайте разработчиков: https://docs.docker.com/engine/install/centos/.

Приведем описание основных шагов инструкции.

Проверяем, установлен ли Docker командой:

```
sudo systemctl status docker
```

Если ответ содержит сообщение:

```
"Unit docker.service could not be found."
```

Значит Docker не найден и нужно его установить.

Добавляем *docker в автозагрузку:

```
sudo systemctl enable docker
```

Запускаем Docker:

```
sudo systemctl start docker
```

Проверяем работоспособность *Docker* командой:

```
sudo systemctl status docker
```

2.1.14.2 Установка Docker в RedOS

> Внимание:
>
> Действия этого раздела необходимо выполнять под созданной учетной записью **datamart** (см. раздел Создание пользователя).

Для переключения на учетную запись **datamart** выполните команду:

```
sudo su - datamart
```

Для установки Docker выполните команду:

```
sudo yum install docker-ce
```

Добавляем Docker в автозагрузку:

```
sudo systemctl enable docker
```

Запускаем Docker:

```
sudo systemctl start docker
```

2.1.14.3 Установка Docker в AltOS

> Внимание:
>
> Действия этого раздела необходимо выполнять под созданной учетной записью **datamart** (см. раздел Создание пользователя).

Для переключения на учетную запись **datamart** выполните команду:

```
sudo su - datamart
```

Установить docker можно следующей командой:

```
sudo apt-get install docker-ce
```

Удалить сервис `containerd`:

```
sudo rm -f /lib/systemd/system/containerd.service
```

Затем необходимо запустить соответствующую службу:

```
sudo systemctl unmask docker
sudo systemctl start docker
```

2.1.15 Установка библиотеки python-docker-py

> Внимание:
>
> Данный пункт для **конфигурации Лайт**

Для CentOS

Для установки библиотек `python-docker-py` выполните следующие команды:

```
sudo yum install epel-release
sudo yum install python-docker-py
```

Для РЕД ОС

Для установки библиотек `python2-docker` выполните следующую команду:

```
sudo yum install python2-docker
```

Для АЛЬТ ОС

> Внимание:
>
> В некоторых версиях «АЛЬТ Сервер 8 СП» (например, АЛЬТ Сервер 8.4 СП) в лицензионный диск с операционной системой не входят пакеты *python3-module-docker* и *python3-websocket-client*, поэтому они будут установлены из сертифицированного репозитория компании-разработчика операционной системы, для этого необходим доступ в Интернет!

Для установки выполните следующую команду:

```
sudo apt-get install python3-module-docker
```

2.1.16 Настройка логирования в Docker

> Внимание:
>
> Данный пункт для **конфигурации Лайт**

Для CentOS и РЕД ОС

Настройка логирования в Docker осуществляется с помощью файла конфигурации. Путь к файлу конфигурации — `/etc/docker/daemon.json`. Если этого файла не существует, его необходимо создать.

Добавьте в файл следующие настройки логирования:

```
{
  "log-opts": {
    "max-file": "1",
    "max-size": "300m"
  }
}
```

где,

- `max-file` - ограничение по количеству файлов (настройки ротации). Максимальное количество файлов журнала, которые могут быть созданы. Если при просмотре журналов создаются лишние файлы, самый старый файл удаляется. Действует только тогда, когда `max-size` (см. ниже) также установлен. Положительное целое число. По умолчанию `1`.
- `max-size` - устанавливает ограничение по размеру лог-файла (`k`, `m` или `g`). По умолчанию - `1` (неограниченно).

Для АЛЬТ ОС

Выполните команду:

```
sudo sed -i 's/journald/json-file/' /etc/docker/daemon.json
```

2.1.17 Перезапуск Docker

> Внимание:
>
> Данный пункт для **конфигурации Лайт**

Для применения настроек, выполненных на предыдущем шаге необходимо перезапустите Docker.

Для CentOS и РЕД ОС

Выполните команду:

```
sudo systemctl restart docker
```

Для АЛЬТ ОС

Выполните команду:

```
sudo systemctl restart docker
```

2.1.18 Добавление пользователя в группу docker

> Внимание:
>
> Данный пункт для **конфигурации Лайт**

Далее, нужно добавить пользователя **datamart** в группу `docker`. Для этого подключитесь к серверу по SSH (например, через Putty) и выполните команду:

```
sudo usermod -aG docker datamart
```

> Внимание:
>
> Для применения настроек изменения группы выполните повторную авторизацию под пользователем **datamart**!

2.1.19 Настройка Ansible

> Внимание:
>
> Данный пункт для **конфигурации Лайт**

> Внимание:
>
> Действия этого раздела необходимо выполнять под созданной в учетной записью **datamart** (см. раздел Создание пользователя).

Для переключения на учетную запись **datamart** выполните команду:

```
sudo su - datamart
```

Для загрузки docker-образа выполните команду:

```
docker image load -i images/ansible-2.9-centos-7.tar
```

Чтобы создать `alias` для вызова Ansible выполните команду:

```
echo "alias docker-ansible-cmd='docker run --rm -it -v $(pwd)/ansible:/ansible -v
~/.ssh/id_rsa:/root/.ssh/id_rsa --workdir=/ansible
cr.yandex/crpl8ogf99r0eaq13vkd/ansible:2.9-centos-7 '" >> .bashrc
. .bashrc
```

Далее нужно перечитать конфигурационный файл, чтобы применить созданные `alias`. для этого выполните команду:

```
source .bashrc
```

Чтобы проверить установку Ansible в контейнере, выполните команду **Ansible**, позволяющую вывести номер версии:

```
docker-ansible-cmd ansible --version
```

3 ТРЕБОВАНИЯ К СЕРВЕРАМ КОНФИГУРАЦИИ СТАНДАРТ

Внимание:
Данный раздел только для **конфигурации Стандарт**

3.1 Требования к серверу Prostore

Сервер Prostore должен отвечать следующим минимальным аппаратным требованиям:

- ядро системы: 4 CPU, 16 RAM, 20 HDD;
- сервис мониторинга статусов Kafka: 2 CPU, 8 RAM, 20 HDD.

Минимальные системные требования приведены в разделе Минимальные системные требования документации Prostore.

3.2 Требования к серверу Arenadata Cluster Manager (ADCM)

При условии использования CentOS 7.9 требуется:

- виртуальная или физическая машина с операционной системой на базе Linux (kernel 3.10 и yum/rpm);
- не менее 5 ГБ свободного пространства в директории `/home`.

3.3 Требования к кластеру серверов Arenadata Streaming (ADS)

1. На серверах должен быть настроен протокол **NTP**;
2. В кластере должны быть открыты порты, представленные в таблице ниже (см. Таблица 3.1).

Таблица 3.1 Перечень сетевых портов, используемых серверами ADS

Номер порта	Описание использования порта
22	SSH
81	(при установке ADS без подключения к сети Интернет) TCP-порт репозиториев Arenadata Enterprise Tools
2015	TCP-порт для отправки метрик на сервер мониторинга
2016	UDP-порт для отправки метрик на сервер мониторинга
8000	TCP-порт для отправки статусов компонентов кластера в ADCM
2181	Порт доступа к сервису ZooKeeper
2888	Порт межсерверного взаимодействия для компонентов кворума ZooKeeper
3888	Порт межсерверного взаимодествия для компонентов кворума ZooKeeper
9092	HTTP-порт доступа к сервису Kafka
9093	HTTPS-порт доступа к сервису Kafka
8081	Порт доступа к сервису Schema-Registry
8082	Порт доступа к сервису Kafka REST Proxy
8088	Порт доступа к компоненту KSQL Server
9898	Порт доступа к сервису Kafka-Manager
9997	JMX-порт для доступа к метрикам сервиса Schema-Registry
9998	JMX-порт для доступа к метрикам сервиса Kafka REST Proxy
9999	JMX-порт для доступа к метрикам сервиса Kafka

3.4 Требования к серверу «СМЭВ3-адаптер»

1. На серверах должен быть настроен протокол **NTP**;
2. Снаружи сервер должен быть доступен по порту 22 (SSH);
3. Необходим сетевой доступ до сервера СМЭВ. Например, если используется адрес `http://smev3-n0.test.gosuslugi.ru:5000/transport_1_0_2/`, необходимо организовать к нему доступ.
4. Если для шифрования и подписи сообщений используется КриптоПро JCP 2.0.41618.
5. Необходимо так же подготовить ключи (контейнер) для КриптоПро и лицензию на JCP. В случае отсутствия лицензии будет использоваться пробная лицензия.
6. Если для оформления цифровой подписи используется **VipNet PKI** необходимо обеспечить сетевой доступ до его сервера.

3.5 Требования к сервису «Агент СМЭВ4»

1. На серверах должен быть настроен протокол **NTP**;
2. Снаружи сервер должен быть доступен по порту 22 (SSH);
3. Необходим сетевой доступ до сервера **ПОДД**. Например, если используется адрес http://smev3-n0.test.gosuslugi.ru:5000/transport_1_0_2/, то необходимо организовать доступ до хоста `smev3-n0.test.gosuslugi.ru` по порту `5000`.

3.6 Требования к серверу «ETL»

1. На серверах должен быть настроен протокол **NTP**;
2. Должны быть открыты порты, представленные в таблице ниже (см. Таблица 3.2).

Таблица 3.2 Перечень сетевых портов, используемых серверами ETL

Номер порта	Описание использования порта
22	SSH
8080	TCP-порт, WEB - интерфейс для Apache Spark master
6066	TCP-порт, отправка задач в кластер через REST API для Apache Spark master
7077	TCP-порт, отправка задач в кластер регистрация исполнителей Apache Spark worker для spark master
8081	TCP-порт, WEB-интерфейс Apache Spark worker
8080	TCP-порт, WEB – интерфейс Apache Airflow
5555	TCP-порт, мониторинг задач Apache Airflow
5432	TCP-порт, Airflow Postgres для хранения метаданных
6379	TCP-порт, Redis для хранения очередей

3.6.1 Требования к серверу «REST-адаптер»

1. На серверах должен быть настроен протокол **NTP**;
2. Снаружи сервер должен быть доступен по следующим портам:
   - 22 (SSH)
   - 8080

9. Примечание:

10. Установка витрины данных производится без необходимости доступа к сети Интернет для скачивания компонент.

4.1 Установка ПО конфигурации Стандарт

4.1.1 Порядок установки

1. Проверить соответствие серверов техническим характеристикам (см. раздел Требования к серверам конфигурации Стандарт).
2. Выполнить предварительные действия перед установкой программы (см. раздел Предварительные действия).
3. Установить на серверы, в соответствующем порядке, следующее программное обеспечение:
   - Arenadata Cluster Manager (ADCM) - при условии использования CentOS 7.9;
   - Arenadata Streaming (ADS) - при условии использования CentOS 7.9;
   - коннектор Kafka-Postgres;
   - ProStore;
   - СМЭВ QL Сервер;
   - СМЭВ3-адаптер;
   - ПОДД-адаптер - Модуль исполнения запросов;
   - ПОДД-адаптер – Модуль MPPR;
   - ПОДД-адаптер – Модуль MPPW;
   - ПОДД-адаптер – Модуль импорта данных табличных параметров;
   - ПОДД-адаптер – Модуль группировки данных табличных параметров;
   - ПОДД-адаптер – ПОДД-адаптер – Wrapper;
   - Data-uploader – Модуль исполнения асинхронных заданий;
   - REST-uploader – Модуль асинхронной загрузки данных из сторонних источников;
   - ПОДД-адаптер – Модуль подписки;
   - BLOB-адаптер;
   - Сервис формирования документов;
   - ETL;
   - CSV-uploader;
   - REST-адаптер;
   - Counter-provider
   - Backup manager.
4. Проверить работу всех компонентов программы (см. раздел Проверка программы).

Версии компонентов программы приведены в Состав компонентов в дистрибутиве документа «Техническое описание программы ПО «Витрина данных НСУД»».

Описание настроек модулей приведено в Руководстве администратора ПО «Витрина данных НСУД».

4.1.2 Установка ПО ProStore

Установка ProStore должна осуществляться после установки СУБД и брокера сообщений.

ПО ProStore поставляется в виде дистрибутива с компонентами в jar-файлах.

Процесс установки состоит из следующих действий приведен в разделе Сборка и развертывание документации **Prostore**.

4.1.3 Установка СМЭВ QL Сервера

4.1.3.1 Процесс установки

Действия по установке выполняются через SSH консоль технологического пользователя.

Общий процесс установки состоит из следующих действий:

1. Настроить конфигурацию модуля.
2. Создать на сервере директорию для загрузки файлов модуля.
3. Загрузить файлы модуля в созданную директорию.
4. Запустить модуль.
5. Проверить установку модуля.

4.1.3.1.1 Настройка конфигурации

Настройка конфигурации выполняется путем редактирования параметров файлов:

- `application.yaml` - конфигурирует поведение сервера;
- `credentials.yaml` - конфигурирует представление сервера.

Пример файлов конфигурации и возможные настройки модуля см. в разделе Конфигурирование сервера Руководства администратора.

4.1.3.1.2 Загрузка JAR-файла на сервер

Для загрузки файла на сервер выполните команду:

```
scp file.jar user_name@IP:/home/dir
```

где,

- `file.jar` - название JAR-файла;
- `user_name` - имя пользователя, например, `sudo` или `root`;
- `IP` - адрес сервера;
- `/home/dir` - директория на сервере, в которую будет загружен файл.

4.1.4 Установка СМЭВ3-адаптера

4.1.4.1 Установка модуля

Действия по установке выполняются через SSH консоль технологического пользователя.

4.1.4.1.1 Установка КриптоПро

Установка СМЭВ3-адаптера возможна, только если были добавлены ключи (контейнер закрытого ключа) для сервиса **КриптоПро**.

В случае использования для электронной подписи сервиса **КриптоПро**, необходимо предварительно выполнить шаги:

- скачать **КриптоПро JCP** (https://www.cryptopro.ru/download?pid=129) для Java JDK на сервер, где будет установлен СМЭВ3–адаптер;
- установить загруженное ПО, следуя инструкции и документации на официальном

сайте;

– получить сертификат для установки от уполномоченных лиц и установить его в **КриптоПро**.

4.1.4.2 Процесс установки

Модуль СМЭВ3-адаптер поставляется в виде JAR-файла. В поставку также входят следующие файлы:

– файл настроек конфигурации модуля СМЭВ3-адаптер `application.yml`;

– файлы для подключения к ProStore: JDBC–драйвер (`dtm-jdbc-driver-*.*.*.jar`) и `commons-lang3-3.12.0.jar`;

– сконфигурированные pebble-шаблоны.

Общий процесс установки состоит из следующих действий:

1. Настроить конфигурацию модуля.
2. Создать на сервере директорию для загрузки файлов модуля.
3. Загрузить файлы модуля в созданную директорию.
4. Запустить JAR-файл модуля.
5. Проверить установку модуля.

4.1.4.2.1 Настройка конфигурации

Настройка конфигурации выполняется путем редактирования параметров файла конфигурации `application.yml`.

Пример файла `application.yml` и возможные настройки конфигурации модуля см. в разделе Конфигурация СМЭВ3-адаптер (application.yml) Руководства администратора.

4.1.4.2.2 Добавление папки для загрузки файлов модуля

Создайте на сервере папку, в которую будут загружены файлы модуля, например, `/opt/smev3-adapter`.

В случае, если ранее была установлена старая версия **СМЭВ3-адаптера**, сделайте его резервную копию.

4.1.4.2.3 Загрузка файлов на сервер

Загрузите в созданную на предыдущем шаге папку:

– JAR-файл модуля;

– файл настроек конфигурации модуля СМЭВ3-адаптер (`application.yml`);

– файлы JDBC-драйвер (`dtm-jdbc-driver-*.*.*.jar`) и `commons-lang3-3.12.0.jar` для подключения к ProStore;

– сконфигурированные pebble-шаблоны.

4.1.4.2.4 Кластеризация модуля

Кластеризация модуля достигается путем запуска копии экземпляра данного модуля. Оптимальным вариантом является использование оркестраторов, например:

– **Kubernetes;**

– **Openshift;**

– **Docker-swarm;**

– **Nomad**.

4.1.5 Установка ПОДД-адаптера - Модуля исполнения запросов

4.1.5.1 Процесс установки

Действия по установке выполняются через SSH консоль технологического пользователя.

Общий процесс установки состоит из следующих действий:

1. Настроить конфигурацию модуля.
2. Создать на сервере директорию для загрузки файлов модуля.
3. Загрузить файлы модуля в созданную директорию.
4. Запустить модуль.
5. Проверить установку модуля.

4.1.5.1.1 Настройка конфигурации

Настройка конфигурации выполняется путем редактирования параметров файла конфигурации `application.yml`.

Пример файла `application.yml` для **Модуля исполнения запросов** см. в разделе Конфигурация ПОДД-адаптера - Модуль исполнения запросов (application.yml) Руководства администратора ПО «Витрина данных НСУД».

11. Примечание:
12. Значения настроек `MYSQL_USER` для MariaDB и `SUBSCR_DB_USER` для *Модуля исполнения запросов*, а также `MYSQL_PASSWORD` и `SUBSCR_DB_PASS`, должны совпадать.

4.1.5.1.2 Кластеризация модуля

Кластеризация модуля достигается путем запуска копии экземпляра данного модуля. Оптимальным вариантом является использование оркестраторов, например:

- **Kubernetes;**
- **Openshift;**
- **Docker-swarm;**
- **Nomad.**

4.1.6 Установка ПОДД-адаптера – Модуля MPPR

4.1.6.1 Процесс установки

Общий процесс установки состоит из следующих действий:

1. Настроить конфигурацию модуля.
2. Создать на сервере директорию для загрузки файлов модуля.
3. Загрузить файлы модуля в созданную директорию.
4. Запустить модуль.
5. Проверить установку модуля.

4.1.6.1.1 Настройка конфигурации

Настройка конфигурации выполняется путем редактирования параметров файла `application.yml`, пример которого и возможные настройки конфигурации модуля см. в разделе Конфигурация ПОДД-адаптера - Модуль MPPR (application.yml) Руководства администратора ПО «Витрина данных НСУД».

4.1.6.1.2 Загрузка JAR-файла на сервер

Для загрузки файла на сервер выполните команду:

```
scp file.jar user_name@IP:/home/dir
```

где,

- `file.jar` - название jar-файла;
- `user_name` - имя пользователя, например, `sudo` или `root`;
- `IP` - адрес сервера;
- `/home/dir` - директория на сервере, в которую будет загружен файл.

4.1.6.1.3 Кластеризация модуля

Кластеризация модуля достигается путем запуска копии экземпляра данного модуля. Оптимальным вариантом является использование оркестраторов, например:

- **Kubernetes;**
- **Openshift;**
- **Docker-swarm;**
- **Nomad**.

4.1.7 Установка ПОДД-адаптера - Модуля MPPW

4.1.7.1 Установка модуля

Общий процесс установки состоит из следующих действий:

1. Настроить конфигурацию модуля.
2. Создать на сервере директорию для загрузки файлов модуля.
3. Загрузить файлы модуля в созданную директорию.
4. Запустить модуль.
5. Проверить установку модуля.

4.1.7.1.1 Настройка конфигурации

Настройка конфигурации выполняется путем редактирования параметров файла `application.yml`, пример которого и возможные настройки конфигурации модуля см. в разделе см. Конфигурация модуля ПОДД-адаптер - Модуль MPPW (application.yml) Руководства администратора ПО «Витрина данных НСУД».

4.1.7.1.2 Добавление папки для загрузки файлов модуля

Создайте на сервере папку, в которую будут загружены файлы модуля, например, `/opt/podd-adapter-mppw`.

В случае, если ранее была установлена старая версия **Модуль MPPW**, сделайте его резервную копию.

4.1.7.1.3 Загрузка файлов на сервер

Для загрузки файла на сервер выполните команду:

```
scp file.jar user_name@IP:/opt/podd-adapter-mppw
```

где,

- `file.jar` - название jar-файла модуля;
- `user_name` - имя пользователя, например, `sudo` или `root`;
- `IP` - адрес сервера;
- `/opt/podd-adapter-mppw` - директория на сервере, в которую будет загружен файл.

4.1.7.1.4 Кластеризация модуля

Кластеризация модуля достигается путем запуска копии экземпляра данного модуля. Оптимальным вариантом является использование оркестраторов, например:

- **Kubernetes;**
- **Openshift;**
- **Docker-swarm;**
- **Nomad**.

4.1.8 Установка ПОДД-адаптера – Модуля импорта данных табличных параметров

4.1.8.1 Установка модуля

Действия по установке выполняются через SSH консоль технологического пользователя.

Общий процесс установки состоит из следующих действий:

1. Настроить конфигурацию модуля.
2. Создать на сервере директорию для загрузки файлов модуля.
3. Загрузить файлы модуля в созданную директорию.
4. Запустить модуль.
5. Проверить установку модуля.

4.1.8.1.1 Настройка конфигурации

Настройка конфигурации выполняется путем редактирования параметров файла конфигурации `application.yml`.

Пример файла `application.yml` и возможные настройки конфигурации модуля см. в разделе Конфигурация модуля ПОДД-адаптер - Модуль импорта данных табличных параметров (application.yml) Руководства администратора ПО «Витрина данных НСУД».

4.1.8.1.2 Добавление папки для загрузки файлов модуля

Создайте на сервере папку, в которую будут загружены файлы модуля, например, `/opt/podd-adapter-import-tp`.

В случае, если ранее была установлена старая версия **Модуля импорта данных табличных параметров**, сделайте его резервную копию.

4.1.8.1.3 Загрузка файлов на сервер

Для загрузки файла на сервер выполните команду:

```
scp file.jar user_name@IP:/podd-adapter-import-tp
```

где,

- `file.jar` - название jar-файла модуля;
- `user_name` - имя пользователя, например, `sudo` или `root`;
- `IP` - адрес сервера;
- `/opt/podd-adapter-import-tp` - директория на сервере, в которую будет загружен файл.

4.1.8.1.4 Кластеризация модуля

Кластеризация модуля достигается путем запуска копии экземпляра данного модуля. Оптимальным вариантом является использование оркестраторов, например:

- **Kubernetes;**

- **Openshift;**
- **Docker-swarm;**
- **Nomad.**

4.1.9 Установка ПОДД-адаптера – Модуля группировки данных табличных параметров

4.1.9.1 Установка модуля

Действия по установке выполняются через SSH консоль технологического пользователя.

Общий процесс установки состоит из следующих действий:

1. Настроить конфигурацию модуля.
2. Создать на сервере директорию для загрузки файлов модуля.
3. Загрузить файлы модуля в созданную директорию.
4. Запустить модуль.
5. Проверить установку модуля.

4.1.9.1.1 Настройка конфигурации

Настройка конфигурации выполняется путем редактирования параметров файла конфигурации `application.yml`.

Пример файла `application.yml` и возможные настройки конфигурации модуля см. в разделе Конфигурация модуля ПОДД-адаптер – Модуль группировки данных табличных параметров (application.yml) Руководства администратора ПО «Витрина данных НСУД».

4.1.9.1.2 Добавление папки для загрузки файлов модуля

Создайте на сервере папку, в которую будут загружены файлы модуля, например, `/opt/podd-adapter-group-tp`.

В случае, если ранее была установлена старая версия **Модуля группировки данных табличных параметров**, сделайте его резервную копию.

4.1.9.1.3 Загрузка файлов на сервер

Для загрузки файла на сервер выполните команду:

```
scp file.jar user_name@IP:/opt/podd-adapter-group-tp
```

где,

- `file.jar` - название jar-файла модуля;
- `user_name` - имя пользователя, например, `sudo` или `root`;
- `IP` - адрес сервера;
- `/opt/podd-adapter-group-tp` - директория на сервере, в которую будет загружен файл.

4.1.9.1.4 Кластеризация модуля

Кластеризация модуля достигается путем запуска копии экземпляра данного модуля. Оптимальным вариантом является использование оркестраторов, например:

- **Kubernetes;**
- **Openshift;**
- **Docker-swarm;**
- **Nomad**.

4.1.10 Установка ПОДД-адаптера – Wrapper

4.1.10.1 Установка модуля

Действия по установке выполняются через SSH консоль технологического пользователя.

Общий процесс установки состоит из следующих действий:

1. Настроить конфигурацию модуля.
2. Создать на сервере директорию для загрузки файлов модуля.
3. Загрузить файлы модуля в созданную директорию.
4. Запустить модуль.
5. Проверить установку модуля.

4.1.10.1.1 Настройка конфигурации

Настройка конфигурации выполняется путем редактирования параметров файла конфигурации `application.yml`.

Пример файла `application.yml` и возможные настройки конфигурации модуля см. в разделе Конфигурация модуля ПОДД-адаптер - Wrapper (application.yml) Руководства администратора ПО «Витрина данных НСУД».

4.1.10.1.2 Добавление папки для загрузки файлов модуля

Создайте на сервере папку, в которую будут загружены файлы модуля, например, `/opt/podd-avro-defragmentator`.

В случае, если ранее была установлена старая версия модуля **ПОДД-адаптер - Wrapper**, сделайте его резервную копию.

4.1.10.1.3 Загрузка файлов на сервер

Загрузите в созданную на предыдущем шаге папку:

– JAR-файл модуля;

– файл настроек конфигурации модуля **ПОДД-адаптер - Wrapper** (application.yml);

Для загрузки файла на сервер выполните команду:

```
scp file.jar user_name@IP:/opt/podd-avro-defragmentator
```

где,

– `file.jar` - название JAR-файла модуля;

– `user_name` - имя пользователя, например, `sudo` или `root`;

– `IP` - адрес сервера;

– `/opt/podd-avro-defragmentator` - директория на сервере, в которую будет загружен файл.

4.1.11 Установка модуля группировки чанков репликации

4.1.11.1 Процесс установки

Общий процесс установки состоит из следующих действий:

1. Настроить конфигурацию модуля.
2. Создать на сервере директорию для загрузки файлов модуля.
3. Загрузить файлы модуля в созданную директорию.
4. Запустить модуль.
5. Проверить установку модуля.

4.1.11.1.1 Настройка конфигурации

Настройка конфигурации выполняется путем редактирования параметров файла `application.yml`, пример которого и возможные настройки конфигурации модуля см. в разделе Конфигурация Модуля группировки чанков репликации (application.yml) Руководства администратора ПО «Витрина данных НСУД».

4.1.11.1.2 Загрузка JAR-файла на сервер

Для загрузки файла на сервер выполните команду:

```
scp file.jar user_name@IP:/home/dir
```

где,

- `file.jar` - название jar-файла;
- `user_name` - имя пользователя, например, `sudo` или `root`;
- `IP` - адрес сервера;
- `/home/dir` - директория на сервере, в которую будет загружен файл.

4.1.12 Установка DATA-uploder – Модуля исполнения асинхронных заданий

4.1.12.1 Процесс установки

Действия по установке выполняются через SSH консоль технологического пользователя.

Общий процесс установки состоит из следующих действий:

1. Настроить конфигурацию модуля.
2. Создать на сервере директорию для загрузки файлов модуля.
3. Загрузить файлы модуля в созданную директорию.
4. Запустить модуль.
5. Проверить установку модуля.

4.1.12.1.1 Настройка конфигурации

Настройка конфигурации выполняется путем редактирования параметров файла конфигурации `application.yml`.

Пример файла `application.yml` и возможные настройки конфигурации модуля см. в разделе Конфигурация модуля DATA-Uploader (application.yml) Руководства администратора ПО «Витрина данных НСУД».

4.1.12.1.2 Загрузка JAR-файла на сервер

Для загрузки файла на сервер выполните команду:

```
scp file.jar user_name@IP:/home/dir
```

где,

- `file.jar` - название JAR-файла;
- `user_name` - имя пользователя, например, `sudo` или `root`;
- `IP` - адрес сервера;
- `/home/dir` - директория на сервере, в которую будет загружен файл.

4.1.13 Установка REST-uploader – Модуля асинхронной загрузки данных из сторонних источников

4.1.13.1 Процесс установки

Действия по установке выполняются через SSH консоль технологического пользователя.

Общий процесс установки состоит из следующих действий:

1. Настроить конфигурацию модуля.
2. Создать на сервере директорию для загрузки файлов модуля.
3. Загрузить файлы модуля в созданную директорию.
4. Запустить модуль.
5. Проверить установку модуля.

4.1.13.1.1 Настройка конфигурации

Настройка конфигурации выполняется путем редактирования параметров файла конфигурации `application.yml`.

Пример файла `application.yml` и возможные настройки конфигурации модуля см. в разделе Конфигурация модуля REST-Uploader (application.yml) Руководства администратора ПО «Витрина данных НСУД».

4.1.13.1.2 Загрузка JAR-файла на сервер

Для загрузки файла на сервер выполните команду:

```
scp file.jar user_name@IP:/home/dir
```

где,

- `file.jar` - название JAR-файла;
- `user_name` - имя пользователя, например, `sudo` или `root`;
- `IP` - адрес сервера;
- `/home/dir` - директория на сервере, в которую будет загружен файл.

4.1.14 Установка ПОДД-адаптера – Модуля подписки

4.1.14.1 Установка модуля

Действия по установке выполняются через SSH консоль технологического пользователя.

Общий процесс установки состоит из следующих действий:

1. Настроить конфигурацию модуля.
2. Создать на сервере директорию для загрузки файлов модуля.
3. Загрузить файлы модуля в созданную директорию.
4. Запустить модуль.
5. Проверить установку модуля.

4.1.14.1.1 Настройка конфигурации

Настройка конфигурации выполняется путем редактирования параметров файла конфигурации `application.yml`.

Пример файла `application.yml` и возможные настройки конфигурации модуля см. в разделе Конфигурация модуля ПОДД-адаптер - Модуль подписок (application.yml) Руководства администратора ПО «Витрина данных НСУД».

4.1.14.1.2 Добавление папки для загрузки файлов модуля

Создайте на сервере папку, в которую будут загружены файлы модуля, например,

/opt/podd-adapter-replicator.

В случае, если ранее была установлена старая версия **Модуля подписки**, сделайте его резервную копию.

4.1.14.1.3 Загрузка файлов на сервер

Для загрузки файла на сервер выполните команду:

```
scp file.jar user_name@IP:/opt/podd-adapter-replicator
```

где,

- file.jar - название JAR-файла модуля;
- user_name - имя пользователя, например, sudo или root;
- IP - адрес сервера;
- /opt/podd-adapter-replicator - директория на сервере, в которую будет загружен файл.

4.1.14.1.4 Кластеризация модуля

Кластеризация модуля достигается путем запуска копии экземпляра данного модуля. Оптимальным вариантом является использование оркестраторов, например:

- **Kubernetes;**
- **Openshift;**
- **Docker-swarm;**
- **Nomad**.

4.1.15 Установка BLOB-адаптера

4.1.15.1 Установка модуля

Модуль BLOB-адаптер поставляется в виде jar-файла.

Общий процесс установки состоит из следующих действий:

1. Настроить конфигурацию модуля.
2. Создать на сервере директорию для загрузки файлов модуля.
3. Загрузить файлы модуля в созданную директорию.
4. Запустить jar-файл модуля.
5. Проверить установку модуля.

4.1.15.1.1 Настройка конфигурации

Настройка конфигурации выполняется путем редактирования параметров файла конфигурации application.yml.

Пример файла application.yml и возможные настройки конфигурации модуля см. Конфигурация BLOB-адаптера (application.yml) Руководства администратора ПО «Витрина данных НСУД».

4.1.15.1.2 Добавление папки для загрузки файлов модуля

Создайте на сервере папку, в которую будут загружены файлы модуля, например, /opt/blob-adapter. В случае, если ранее была установлена старая версия модуля **BLOB-адаптер**, сделайте его резервную копию.

4.1.15.1.3 Загрузка файлов на сервер

Загрузите в созданную на предыдущем шаге папку:

- jar-файл модуля;

– файл настроек конфигурации модуля BLOB-адаптер (`application.yml`).

4.1.16 Установка Сервиса формирования документов

4.1.16.1 Установки модуля

Действия по установке выполняются через SSH консоль технологического пользователя.

4.1.16.1.1 Установка модуля

Модуль Сервис формирования документов поставляется в виде JAR-файла.

В поставку также входят следующие файлы:

– файл настроек конфигурации **Сервиса формирования документов** (`application.yml`);
– файлы для подключения к Prostore: JDBC-драйвер (`dtm-jdbc-driver-*.*.*.jar` и `commons-lang3-3.12.0.jar`).

Общий процесс установки состоит из следующих действий:

1. Настроить конфигурацию модуля.
2. Создать на сервере директорию для загрузки файлов модуля.
3. Загрузить файлы модуля в созданную директорию.
4. Запустить JAR-файл модуля.
5. Проверить установку модуля.

4.1.16.1.2 Настройка конфигурации

Настройка конфигурации выполняется путем редактирования параметров файла конфигурации `application.yml`.

Пример файла `application.yml` и возможные настройки конфигурации модуля см. в разделе Конфигурация Сервиса Формирования документов (application.yml) Руководства администратора ПО «Витрина данных НСУД».

4.1.16.1.3 Добавление папки для загрузки файлов модуля

Создайте на сервере папку, в которую будут загружены файлы модуля, например, `/opt/printable-form-service`. В случае, если ранее была установлена старая версия **Сервиса формирования документов**, сделайте его резервную копию.

4.1.16.1.4 Загрузка файлов на сервер

Загрузите в созданную на предыдущем шаге папку:

– JAR-файл модуля;
– файл настроек конфигурации модуля Сервис формирования документов (`application.yml`);
– файлы JDBC-драйвер (`dtm-jdbc-driver-*.*.*.jar`) и `commons-lang3-3.12.0.jar` для подключения к Prostore.

4.1.16.1.5 Кластеризация модуля

Кластеризация модуля достигается путем запуска копии экземпляра данного модуля. Оптимальным вариантом является использование оркестраторов, например:

– **Kubernetes;**
– **Openshift;**
– **Docker-swarm;**
– **Nomad.**

4.1.17 Загрузка и удаление данных через ETL

4.1.17.1 Общее описание

4.1.17.1.1 Общие положения

Базовый сервис загрузки данных предоставляет возможность асинхронного приёма данных из сторонних источников с целью последующей загрузки их в Витрину данных. Загрузка/обновление данных осуществляется в соответствии с заранее подготовленными Avro-схемами.

Сервис загрузки данных реализуется компонентом **ETL**, предоставляющей REST API. Доступ к REST API должен осуществляться через Proxy API Datamart Studio. Перед загрузкой необходимо получить токен Proxy API, который в дальнейшем используется для авторизации при операциях, описанных в разделах ниже. При получении ошибки авторизации необходимо повторить авторизацию, получить новый токен и использовать его для дальнейшей загрузки.

Для взаимодействия с REST API (через Proxy API) в продуктивной среде на стороне источника данных требуется использовать сертифицированную версию ОС, а также механизмы, соответствующие требованиям безопасности, установленным для эксплуатации системы (например, через программный продукт Postman).

На стороне источника данных должны соблюдаться следующие требования к механизму взаимодействия:

1. Запрещается хранение логина и пароля в открытом виде на диске. Логин и пароль должны выгружаться из безопасного хранилища в память ВМ (или контейнера) при старте взаимодействия с Proxy API для дальнейшего использования, сама ВМ должна находиться в закрытом контуре ИС или ведомства.
2. Рекомендуется реализовать механизм стирания логина и пароля из памяти после успешной аутентификации через Proxy API.

В целях тестирования взаимодействия на тестовой среде может использоваться утилита curl.

4.1.17.2 Особенности реализации

4.1.17.2.1 Основные требования к исходным файлам

Загрузка данных в систему производится в виде сообщений, каждое из которых имеет структуру, представленную на Рисунок - 4.1

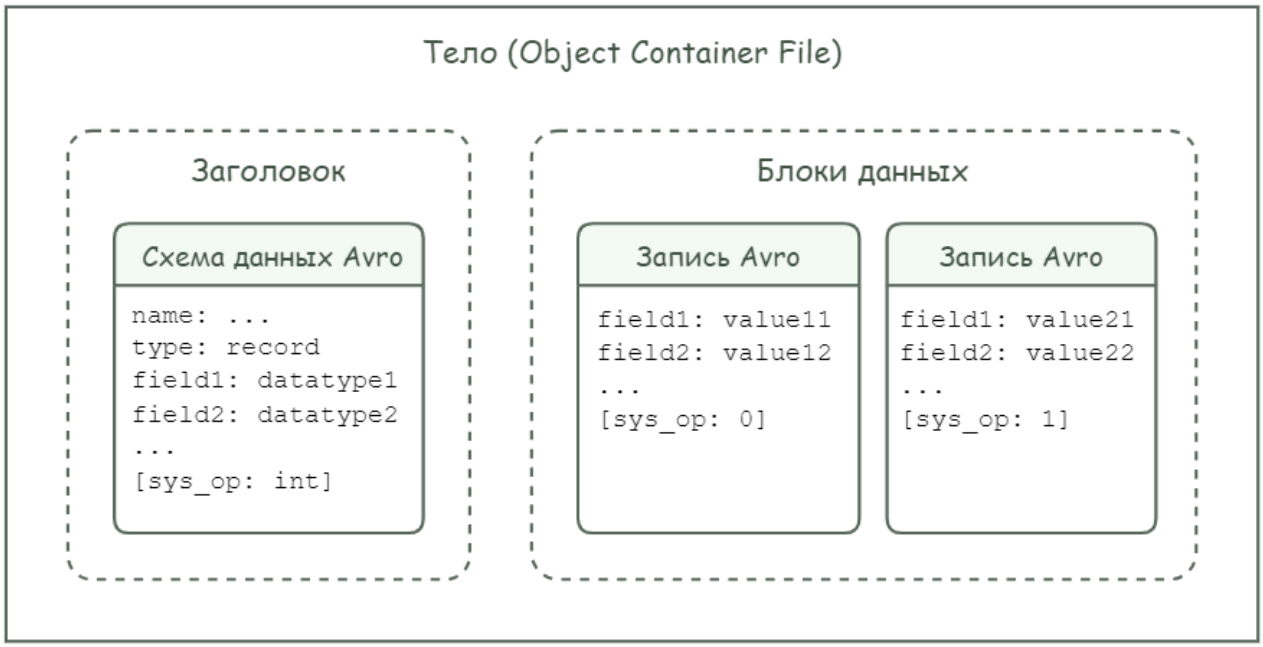

Рисунок - 4.1 Структура загружаемых сообщений

Для успешной загрузки данные должны соответствовать следующим условиям:

1. Тело сообщения представляет собой файл Avro (**Object Container File**), который состоит из заголовка и блоков данных.
2. Заголовок файла содержит схему данных Avro.
3. Схема данных содержит следующие элементы:
   - имя;
   - тип “record”;
   - перечень полей.
4. Последним полем схемы должно быть указано служебное поле `sys_op` с типом данных `avro.int`.
5. Каждый блок данных содержит запись, представленную в бинарной кодировке.
6. Каждая запись содержит перечень полей и их значений.
7. Состав и порядок полей должны совпадать во всех следующих объектах:
   - в схеме данных заголовка файла Avro;
   - в наборе загружаемых записей;
   - во внешней таблице загрузки (поле `sys_op` должно отсутствовать);
   - в таблице-приемнике данных (поле `sys_op` должно отсутствовать).

Более подробно про формат данных Avro описано в источнике: https://avro.apache.org/docs/1.10.2/spec.html#Object+Container+Files

13. Примечание:
14. В загружаемой схеме данных Avro и записях Avro важны порядок и тип полей. Имена полей не сравниваются с именами полей внешней таблицы и таблицы-приемника.

Пример ниже содержит схему данных Avro, используемую для загрузки данных о сотрудниках в таблицу `staff`.

Для поля `date_of_birth` указан логический тип Avro, для поля `middle_name` — элемент

union (поле является не обязательным для заполнения, поэтому маркер `null` выведен в отдельный параметр).

Пример схемы данных Avro:

```
{
  "name": "staff",
  "type": "record",
  "fields": [
      {
      "name": "id",
      "type": "long"
      },
      {
      "name": "firstname",
      "type": "string"
      },
      {
      "name": "lastname",
      "type": "string"
      },
      {
      "name": "middle_name",
      "type": [
          "null",
          "string"
      ]
      },
      {
      "name": "date_of_birth",
      "type": {
      "logicalType": "timestamp-micros",
      "type": "long"
      }
      },
      {
      "name": "employee_position",
      "type": "string"
      },
      {
      "name": "department_category",
      "type": "long"
      },
      {
      "name": "sys_op",
      "type": "int"
      }
  ]
}
```

Пример ниже содержит набор записей о сотрудниках, загружаемых в таблицу `staff`. Для наглядности примера бинарные данные представлены в JSON-формате.

```
[
  {
      "id": 1000111,
      "firstname": "Елена",
      "lastname": "Фролова",
      "middle_name": "Андреевна",
      "date_of_birth": 4641084000000000,
      "employee_position": "Менеджер по подбору персонала",
      "department_category": 1,
```

```
    "description": "Сотрудники отдела кадров",
    "sys_op": 0
  },
  {
    "id": 1000005,
    "firstname": "Пётр",
    "lastname": "Платонов",
    "middle_name": "",
    "date_of_birth": 5639904000000000,
    "employee_position": "Руководитель отдела кадров",
    "department_category": 1,
    "description": "Сотрудники отдела кадров",
    "sys_op": 0
  }
]
```

4.1.17.2.2 Особенности реализации ETL

Функциональные особенности реализованного ETL включают в себя следующие пункты:

1. Генерация первичных ключей записей, передаваемых для загрузки, производится на стороне источника.
2. Каждая Avro-структура должна содержать данные только для одной таблицы Витрины.
3. В Avro-структурах данных источник заполняет тип операции `sys_op`:
   - 0 – для добавления новой или обновления существующей записи;
   - 1 – для удаления существующей записи (см. пример записей Avro в Раздел 4.1.17.2.1).
4. ETL не выполняет преобразования данных, предназначенных для загрузки в Витрину данных.
5. При выполнении операций, требующих консистентности данных, в рамках одной дельты могут быть только операции одного типа: либо добавления/обновления, либо удаления. При этом в рамках одной дельты первичные ключи всех записей должны быть уникальны.
6. Не должно быть двух версий одной записи в рамках одной дельты.
7. Нельзя менять порядок атрибутов в avro-схеме, поскольку данные при загрузке в БД распределяются в соответствии с тем перечнем, который был указан в avro-схеме.

4.1.17.2.3 Получение токена Proxy API

Этапы настройки Proxy API включают в себя следующие шаги:

1. Для начала необходимо пройти авторизацию в Datamart Studio. Сделать это можно, направив запрос ниже:

```
curl -X POST ¥
'http://<ip-studio>:8088/api/v1/auth_system' ¥
-d "username=<username>" ¥
-d "password=<password>" ¥
-d "organization_ogrn=<organization_ogrn>" ¥
-d "datamart_mnemonic=<datamart_mnemonic>"
```

где:

- `ip-studio` — IP-адрес Datamart Studio;
- `username` — имя пользователя IAM;
- `password` — пароль пользователя IAM;
- `organization_ogrn` — ОГРН организации, в рамках которой развёрнута Витрина

данных;

- `datamart_mnemonic` — мнемоника Витрины (пример: eduejd##, где ## – номер региона).

15. Примечание:

16. Безопасность передачи данных по протоколу HTTP обеспечивается защищенной сетью. Требуется обращать внимание на протокол запроса. Если он будет некорректным (например, HTTPS вместо HTTP), то ответ сервера будет содержать ошибку с кодом 500 - InternalServerError.

Пример успешного ответа Datamart Studio на запрос токена представлен ниже:

```
{
  "access_token":
"eyJhbGciOiJSUzI1NiIsInR5cCIgOiAiSldUIiwia2lkIiA6ICI5NGRuNzhOeVF2TjBhRG3NMcUdkc0tTOTNlT
WxjNkRwRjZ5V1NXSUo4In0.eyJleHAiOjE2OTMzMTg0NzIsImlhdCI6MTY5MzMxNDg3MiwianRpIjoiYTJkNTQ5
NDktZWYwZC00MzA1LWI5OTAtMDIxMTAyZDkzODU2IiwiaXNzIjoiaHR0cHM6Ly9rYy5kYXRhbWFydC5ydS9yZWF
sbXMvc3R1ZGlvLWRldiIsInN1YiI6IjE1NjdkYWI3LTc1OTAtNGM0Zi1iNWNhLWYzMmFkOTU1NThjYyIsInR5cC
I6IkJlYXJlciIsImF6cCI6InN0dWRpbyIsInNlc3Npb25fc3RhdGUiOiJlMGE0MjJlZC1hNDQxLTQzY2MtYTAxM
S01MzNiY2RiNTc5OGQiLCJhY3IiOiIxIiwicmVhbG1fYWNjZXNzIjp7InJvbGVzIjpbInN1cGVyYWRtaW4iLCJh
ZG1pbiJdfSwic2NvcGUiOiJvcGVuaWQgZW1haWwiLCJzaWQiOiJlMGE0MjJlZC1hNDQxLTQzY2MtYTAxMS01MzN
iY2RiNTc5OGQiLCJ1cG4iOiJhZG1pbiIsImVtYWlsX3ZlcmlmaWVkIjp0cnVlLCJhZGRyZXNzIjp7fSwibmFtZS
I6ImFkbWluIGFkbWluIiwiZ3JvdXBzIjpbInN1cGVyYWRtaW4iLCJhZG1pbiJdLCJwcmVmZXJyZWRfdXNlcm5hb
WUiOiJhZG1pbiIsImdpdmVuX25hbWUiOiJhZG1pbiIsIm9yZ2FuaXphdGlvbl9vZ3JuIjpbIjExMTIyMjMzMzQ0
NCIsIjExMTEiXSwiZmFtaWx5X25hbWUiOiJhZG1pbiIsImVtYWlsIjoiYWRtaW5AYWRtaW4uYWRtaW4ifQ.BC3s
REXC3nf2LNvBX8SiHKouVGqJVfUBokVJa-B-9YW0zLhnNTs7mGZVOnC-kM-
5mWE6bz8du0lvxQqiGpi3HRlAv1eedcGMTf_2TmjhohAaz--
zSCdLC5NSmI79r54XYTLORiWKXj5T_AY8efFwWnWgUJ1LEkd5BTQyGSTvaoJkMv7xextA_isx_WoReHC5_-
3GznNtcf_hOd2J1CfMHUFjhqMRSxMkIQDPHnspgU6WUz9IeVA1VWKh1GcggqYDtrruigQcl4_f7XeJQKJ49NNVd
hjHtywUVbTpEDKYh4FsgAbf3vIPYUVwGWFW0Qm7LgUCpB8UvMQfb4UYZMF4UA",
  "expires_in": 3600,
  "refresh_expires_in": 3600,
  "refresh_token":
"eyJhbGciOiJIUzI1NiIsInR5cCIgOiAiSldUIiwia2lkIiA6ICJmYmMzOWNmNi1kZTRkLTQ2YWEtYTAwZi1iZG
U3ZmFkNTJmNTgifQ.eyJleHAiOjE2OTMzMTg0NzIsImlhdCI6MTY5MzMxNDg3MiwianRpIjoiYWIxN2M0MzQtZG
I4Yi00N2QwLTkyM2YtMTFiYmM3NzBiNmE4IiwiaXNzIjoiaHR0cHM6Ly9rYy5kYXRhbWFydC5ydS9yZWFsbXMvc
3R1ZGlvLWRldiIsImF1ZCI6Imh0dHBzOi8va2MuZGF0YW1hcnQucnUvcmVhbG1zL3N0dWRpby1kZXYiLCJzdWIi
OiIxNTY3ZGFiNy03NTkwLTRjNGYtYjVjYS1mMzJhZDk1NTU4Y2MiLCJ0eXAiOiJSZWZyZXNoIiwiYXpwIjoic3R
1ZGlvIiwic2Vzc2lvbl9zdGF0ZSI6ImUwYTQyMmVkLWE0NDEtNDNjYy1hMDExLTUzM2JjZGI1Nzk4ZCIsInNjb3
BlIjoib3BlbmlkIGVtYWlsIiwic2lkIjoiZTBhNDIyZWQtYTQ0MS00M2NjLWEwMTEtNTMzYmNkYjU3OThkIn0.S
I6Tb6CJ5HIHwzETOyJZ-2nTLibGNq7JEQwW07fY-2M",
  "token_type": "Bearer",
  "id_token":
"eyJhbGciOiJSUzI1NiIsInR5cCIgOiAiSldUIiwia2lkIiA6ICI5NGRuNzhOeVF2TjBhRGtvX3NMcUdkc0tTOT
NlTWxjNkRwRjZ5V1NXSUo4In0.eyJleHAiOjE2OTMzMTg0NzIsImlhdCI6MTY5MzMxNDg3MiwiYXV0aF90aW1lI
jowLCJqdGkiOiI5NDM1ZDE3NS00MzE2LTQyN2QtODlkYi1lYTJkOWJlZjJmNWIiLCJpc3MiOiJodHRwczovL2tj
LmRhdGFtYXJ0LnJ1L3JlYWxtcy9zdHVkaW8tZGV2IiwiYXVkIjoic3R1ZGlvIiwic3ViIjoiMTU2N2RhYjctNzU
5MC00YzRmLWI1Y2EtZjMyYWQ5NTU1OGNjIiwidHlwIjoiSUQiLCJhenAiOiJzdHVkaW8iLCJzZXNzaW9uX3N0YX
RlIjoiZTBhNDIyZWQtYTQ0MS00M2NjLWEwMTEtNTMzYmNkYjU3OThkIiwiYXRfaGFzaCI6Ik9UZzZoUFQtWjhhW
HR1Yjl1VV91LWciLCJhY3IiOiIxIiwic2lkIjoiZTBhNDIyZWQtYTQ0MS00M2NjLWEwMTEtNTMzYmNkYjU3OThk
IiwidXBuIjoiYWRtaW4iLCJlbWFpbF92ZXJpZmllZCI6dHJ1ZSwiYWRkcmVzcyI6e30sIm5hbWUiOiJhZG1pbiB
hZG1pbiIsImdyb3VwcyI6WyJzdXBlcmFkbWluIiwiYWRtaW4iXSwicHJlZmVycmVkX3VzZXJuYW1lIjoiYWRtaW
4iLCJnaXZlbl9uYW1lIjoiYWRtaW4iLCJvcmdhbml6YXRpb25fb2dybiI6WyIxMTEyMjIzMzM0NDQiLCIxMTExI
l0sImZhbWlseV9uYW1lIjoiYWRtaW4iLCJlbWFpbCI6ImFkbWluQGFkbWluLmFkbWluIn0.K372NBffA3xtsxyM
3hixr5GF1ouHNdr8DFMBYnGQ-t-REbYcwOymvs-D-HYEsmaUhkCWjKSeLM9taLmAPloSt2hb8xN_VG4s-
gc_yvGs_aHkUehTqddjGMislPyAlydzCaDVxQ5Px-
TplsDzIAwm5P0V23LDU3qwnVxVR7P3Dbxi5YB84_38zjClNrDWt9YOxnPMCzDT4fnBjGXkDcQZoHo5jsbFP_K5y
mugsYEumKIZyekbY_l_A-XkRcTM6SMTRhKLAvQ3lq2YguLm2LFF3e-PrGsaEymeS-
Peuff5qJw5Vf9r3_nD3APCivVc0Kunl8miRpr3lsZgSAp-xi3Jow",
  "not-before-policy": 0,
  "session_state": "e0a422ed-a441-43cc-a011-533bcdb5798d",
  "scope": "openidemail"
}
```


2. С полученным токеном послать запрос для подключения к API инсталляции приложения типового ПО Витрины данных для выбранной организации:

```
curl -X <method>
'http://<ip-
studio>:8088/api/v1/secure/<organization_ogrn>/<datamart_mnemonic>/<installation_name>/
<installation_id>/<request_path>' ¥
-H "Authorization: Bearer <access_token>" ¥
-H "<headers>" ¥
-d "<data>"
```

где:

- `ip-studio` — ip-адрес Datamart Studio;
- `organization_ogrn` — ОГРН организации, в рамках которой развёрнута Витрина данных;
- `datamart_mnemonic` — мнемоника Витрины (пример: eduejd##, где ## – номер региона);
- `installation_name` — имя инсталляции в целевой Витрине;
- `installation_id` — идентификатор инсталляции (присутствует в её названии);
- `request_path` — URI оригинального API инсталляции;
- `access_token` — токен Proxy API;
- `headers` — заголовки запроса;
- `data` — данные запроса.

4.1.17.3 Загрузка / удаление данных

4.1.17.3.1 Загрузка/ удаление данных

Сначала источник данных формирует и передаёт файлы по REST API, которые накапливаются Компонентом ETL. Далее данные загружаются и вставляются в Витрину данных через внешние таблицы, или удаляются. Статусы обработки каждой операции необходимо отслеживать через Endpoint `/status`.

Информация о каждом шаге процесса содержится в подразделах ниже.

4.1.17.3.1.1 Начало операции загрузки / удаления согласованных данных (Endpoint – newDelta)

Под Endpoint'ом `/newDelta` регистрируется новая порция данных. Для того чтобы начать работу с данными, источнику данных необходимо сгенерировать UUID (идентификатор для новой порции данных) и вставить его в запрос с Endpoint'ом `/newDelta`.

Согласованные данные – это данные для нескольких таблиц, которые должны попасть в Витрину данных за одну операцию вставки (то есть в одной дельте), а значит будут доступны для потребителя одномоментно (такой способ загрузки актуален, когда необходимо обновить данные).

Пример набора данных, который будет загружен или удален в рамках одной дельты представлен ниже:

```
{
  "requestId": "6B29FC40-CA47-1067-B31D-00DD010662DA",
  "dataSetName": ["product", "stock"]
}
```

где:

- `requestId` — идентификатор порции изменений (дельты);
- `dataSetName` — массив имен набора данных (product и stock - имена таблиц).

Запрос с Endpoint'ом `/newDelta` будет иметь вид:

```
curl -X POST "http://<ip-
studio>:8088/api/v1/secure/<organization_ogrn>/<datamart_mnemonic>/<installation_name>/
<installation_id>/newDelta" -H "Authorization: Bearer <access_token>" -H 'Content-Type:
application/json' -d '{"requestId": "6B29FC40-CA47-1067-B31D-00DD010662DA",
"dataSetName": ["product", "stock"]}'
```

где:

- `requestId` — идентификатор порции изменений (дельты), который был ранее сгенерирован;
- `dataSetName` — имя набора данных (имена таблиц).

17. Примечание:

18. Данные для обновления/вставки и для удаления должны быть отдельно зарегистрированы в newDelta с разными requestId. Если требуется обновить/вставить данные, то сначала нужно зарегистрировать порцию данных через newDelta с requestId, затем прислать данные с зарегистрированным requestId через endpoint /partOfDelta (описан в Раздел 4.1.17.3.1.2). Допускается передача как в одном файле одним запросом, так и в двух файлах двумя запросами - это не имеет значения. Главное условие - нужно отправлять только данные на обновление и вставку, и они должны быть уникальны по первичному ключу. Также в них не должно быть одинаковых записей с одним первичным ключом. Пример: отправлены два обновления одной записи (с одинаковым первичным ключом). В этом случае в хранилище Prostore попадет только одна запись, и неизвестно какая, поэтому дублей по первичному ключу отправленных с одним requestId быть не должно!

Чтобы проверить статус выполнения запроса, необходимо направить запрос с Endpoint'ом `/status` (пример запроса описан в Раздел 5.1.18.1)

Если обработка запроса завершится успешно, то Витрина данных вернёт JSON-ответ, содержащий статус-сообщение об успешной операции:

```
{
    "requestId": "6B29FC40-CA47-1067-B31D-00DD010662DA",
    "dataSets": [
    "product",
        "stock"
    ],
    "inDeltaFlag": true,
    "statusCode": "SUCCESS",
    "statusMessage": "Загрузка порции данных успешно завершена"
    "errors": []
}
```

где:

- `requestId` — идентификатор порции изменений (дельты);
- `statusCode` — код возвращаемого статуса (SUCCESS – запрос выполнен успешно).

Пример JSON-ответа на проверку статуса, завершившегося ошибкой, приведен ниже:

```
{
    "requestId": "6B29FC40-CA47-1067-B31D-00DD010662DA",
    "statusCode": "REQUESTID_ALREADY_EXIST",
    "statusMessage": "Дельта с requestId = 6B29FC40-CA47-1067-B31D-00DD010662DA уже
существует. Статус загрузки дельты: SUCCESS"
}
```

где:

- `requestId` — идентификатор порции изменений (дельты) который необходимо проверить;
- `statusCode` — код возвращаемого статуса (REQUESTID_ALREADY_EXIST – идентификатор уже существует в БД);
- `statusMessage` — сообщение с описанием кода ошибки.

19. Примечание:
20. Для удаления записей необходимо зарегистрировать новую дельту во Endpoint'е `/newDelta` с новым `requestId`, и по зарегистрированному `requestId` должны быть присланы только данные на удаление.

4.1.17.3.1.2 Загрузка / удаление согласованных данных (Endpoint – partOfDelta)

На данном шаге выполняется накапливание порции данных, создаётся вставка в Витрину данных по Endpoint'у `isLastChunk`.

21. Примечание:
22. Если в процессе загрузки вызван метод `newDelta`, то текущая загрузка будет прервана и порция не попадет в Витрину данных.

Чтобы отправить последнюю порцию данных для таблицы product, необходимо направить запрос с Endpoint'ом `/partOfDelta` с указанием `dataSetName=product` и `isLastChunk=true` (что означает что данная порция данных - последняя). Обработка и загрузка данных не начнётся, пока не будет направлен такой же запрос, но уже по таблице `stock: dataSetName=stock, isLastChunk=true`.

Пример запроса на загрузку данных под ранее созданный набор данных:

```
curl -X POST "http://<ip-studio>:8088/api/v1/secure/<organization_ogrn>/<datamart_mnemonic>/<installation_name>/<installation_id>/partOfDelta" -H "Authorization: Bearer <access_token>" -F upload=@"./product.avro" -F dataSetName=product -F chunkNumber=0 -F isLastChunk=false -F requestId=a6212a7d-4526-4e2d-89a7-9828f380c91d
```

где:

- `upload` — загружаемый avro-файл (пример avro-файла с данными представлен в разделе 2);
- `dataSetName` — имя набора данных (имя таблицы);
- `chunkNumber` — номер порции dataSet в рамках дельты;
- `isLastChunk` — флаг последней порции dataSet;
- `requestId` — идентификатор порции изменений (дельты).

В результате успешной загрузки при проверке статуса `requestId` (пример запроса по Endpoint'у `/status` представлен в Раздел 5.1.18.1) на запрос Витрина данных вернёт JSON-ответ, содержащий статус-сообщение об успешной операции.

Пример успешной загрузки:

```
{
    "requestId": "f3947645-88c8-4044-bd8b-de273f8a8461",
    "statusCode": "SUCCESS",
    "statusMessage": "Порция получена."
}
```

где:

– `requestId` — идентификатор порции изменений (дельты);
– `statusCode` — статус код результата запроса (SUCCESS - запрос выполнен успешно);
– `statusMessage` — описание статусного сообщения.

Если загрузка прервалась ошибкой, то при проверке статуса `requestId` (пример запроса по Endpoint'у `/status` представлен в Раздел 5.1.18.1) на запрос Витрина данных вернёт JSON-ответ с описанием ошибки.

Пример загрузки, прерванной ошибкой:

```
{
    "requestId": "aef2f195-b037-4aaa-b171-f2746511e7e2",
    "dataSets": [
        "stock"
    ],
    "inDeltaFlag": true,
    "statusCode": "ERROR",
    "statusMessage": "Произошла ошибка"
    "errors": [
        {
            "dataSet": "stock",
            "errorType": "INSERT",
            "message": "Ошибка вставки в таблицы: stock"
        }
    ]
}
```

где:

– `requestId` — идентификатор порции изменений (дельты);
– `dataSets` — массив имен набора данных (имен таблиц где была допущена ошибка);
– `inDeltaFlag` = true — загрузка согласованных данных производилась через endpoint /partOfDetla;
– `status` — статус код результата запроса (ERROR – внутренняя ошибка);
– `statusMessage` — описание статусного сообщения;
– `errors` — массив, ошибки загрузки или парсинга входящих данных;
– `dataSet` — название таблицы где допущена ошибка;
– `errorType` — тип ошибки;
– `message` — описание ошибки.

23. Примечание:
24. Возможна ситуация, когда после падения ETL приходит запрос с `requestId`, который был до падения, в данном случае Витрина данных возвращает ошибку со статусом NOT_FOUND. Необходимо снова направить запрос по Endpoint'у `/newDelta` с новым `requestId` и начать процесс загрузки заново.

4.1.17.3.1.3 Загрузка / удаление несогласованных данных (Endpoint – data)

Для загрузки несогласованных данных поддерживается возможность накапливания данных, аналогично загрузке согласованных данных, описанной в Раздел 4.1.17.3.1.2.

Несогласованные данные – могут быть вставлены в разных дельтах и будут доступны потребителю постепенно по мере загрузки. Этот способ подходит для первоначальной загрузки, когда еще нет потребителей.

Вставка в Витрину данных выполнится после накопления порции или по флагу `isLast`, который используется для последней порции данных. Флаг `isLast` подаёт сигнал для завершения формирования дельты, для того чтобы выполнить вставку накопленных данных и закрыть транзакцию.

Пример запроса:

```
curl -X POST "http://<ip-studio>:8088/api/v1/secure/<organization_ogrn>/<datamart_mnemonic>/<installation_name>/<installation_id>/data" -H "Authorization: Bearer <access_token>" -F upload=@"./product.avro" -F dataSetName=product -F isLast=false -F requestId=a6212a7d-4526-4e2d-89a7-9828f380c91d
```

где:

- `upload` — загружаемый avro-файл (пример avro-файла с данными представлен в Раздел 4.1.17.2.1);
- `dataSetName` — имя набора данных (имя таблицы);
- `isLast` — флаг последней порции данных (сигнал для завершения формирования дельты, для того чтобы выполнить вставку накопленных данных и закрыть транзакцию.);
- `requestId` — идентификатор порции изменений (дельты).

В результате успешной операции при проверке статуса запроса по Endpoint'у `/status` (пример запроса описан в Раздел 5.1.18.1) Витрина данных вернёт JSON-ответ, содержащий статус-сообщение:

```
{
    "requestId": "6B29FC40-CA47-1067-B31D-00DD010662DA",
    "statusCode": "SUCCESS",
    "statusMessage": "Порция получена."
}
```

где:

- `requestId` — идентификатор порции изменений (дельты);
- `statusCode` — статус код результата запроса (SUCCESS - запрос выполнен успешно);
- `statusMessage` — описание статусного сообщения.

Если загрузка прервалась ошибкой, то при проверке `requestId` (пример запроса по Endpoint'у `/status` представлен в Раздел 5.1.18.1) Витрина данных вернёт JSON-ответ с описанием ошибки.

Пример JSON-ответа:

```
{
    "requestId": "6B29FC40-CA47-1067-B31D-00DD010662DA",
    "statusCode": "NOT_FOUND",
    "statusMessage": "Не найдена дельта с requestId = 6B29FC40-CA47-1067-B31D-00DD010662DA"
}
```

где:

- `requestId` — идентификатор порции изменений (дельты);
- `statusCode` — статус код результата запроса (NOT_FOUND - данные по requestId были утеряны в результате остановки сервиса, необходимо зарегистрировать новую дельту и снова загрузить данные);
- `statusMessage` — описание статусного сообщения.

4.1.17.3.1.4 Описание возвращаемых кодов

Таблица 4.5 Описание возвращаемых кодов

Наименование	Код	Описание
EMPTY_ATTACHMENT	400	Нет файла вложения
ERROR	500	Внутренняя ошибка
NOT_FOUND	400	Данные не найдены, либо были утеряны в результате остановки сервиса
PROCESSING	400	Идет обработка данных
REQUESTID_ALREADY_EXIST	400	`requestId` уже зарегистрирован
SUCCESS	200	Успешное выполнение
UNREGISTERED_DATASETNAME	400	Незарегистрированный набор данных
WRONG_ENDPOINT	400	`requestId` зарегистрирован для другого Endpoint'а

4.1.17.4 Проверка статусной информации по загрузке

4.1.17.5 Проверка статусной информации по загрузке / удалению данных (Endpoint – status)

В данном разделе производится проверка статусной информации из сервисных таблиц по `requestId`.

Пример запроса:

```
Curl -X GET "http://<ip-
studio>:8088/api/v1/secure/<organization_ogrn>/<datamart_mnemonic>/<installation_name>/
<installation_id>/status/<requestId>" -H "Authorization: Bearer <access_token>"
```

где:

– `requestId` — UUID идентификатор порции изменений (дельты).

Пример ответа на такой запрос представлен ниже.

```
{
    "requestId": "13f2475e-f3dc-4c9e-b2f6-3a98320261f1",
    "inDeltaFlag": false,
    "dataSets": [
        "stock"
    ],
    "status": "ERROR",
    "statusMessage": "Произошла ошибка",
    "errors": [
        {
            "dataSet": "stock",
            "errorType": "PARCING",
            "message": "Неверно указан тип поля count_pieces: LONG. Ожидается: INTEGER"
        },
        {
            "dataSet": "stock",
            "errorType": "PARCING",
            "message": "Неверно указан тип поля product_id: LONG. Ожидается: INTEGER"
        }
    ]
}
```

где:

– `requestId` — UUID идентификатор порции изменений (дельты);

– `inDeltaFlag = false` — загрузка несогласованных данных производилась через

endpoint /data;

- `dataSets` — массив имен набора данных (имен таблиц где была допущена ошибка);
- `status` — статус код результата запроса (NOT_FOUND, PROCESSING, ERROR, SUCCESS);
- `statusMessage` — описание статусного сообщения;
- `errors` — массив, ошибки загрузки или парсинга входящих данных;
- `dataSet` — название таблицы где допущена ошибка;
- `errorType` — тип ошибки;
- `message` — описание ошибки.

4.1.17.6 Работа с вложениями через S3

4.1.17.6.1 Работа с вложениями через S3

4.1.17.6.1.1 Загрузка данных в хранилище (Endpoint – uploadAttachment)

Перед загрузкой источнику данных необходимо сгенерировать UUID (идентификатор для новой порции данных) и вставить его в запрос с Endpoint'ом `/uploadAttachment`. При совпадении имен вложений в хранилище, вложение перезаписывается. Пример запроса на загрузку вложения в хранилище представлен ниже:

```
curl -X POST "http://<ip-studio>:8088/api/v1/secure/<organization_ogrn>/<datamart_mnemonic>/<installation_name>/<installation_id>/uploadAttachment" -H "Authorization: Bearer <access_token>" -F upload=@"document.pdf" -F requestId=13f2475e-f3dc-4c9e-b2f6-3a98320261f1 -F name=Doc_1
```

где:

- `upload` — путь до загружаемого файла-вложения;
- `requestId` — UUID идентификатор запроса;
- `name` — уникальное имя вложения.

После успешной загрузки при проверке по Endpoint'у `/status` (пример запроса описан в Раздел 5.1.18.1) Витрина данных вернёт JSON-ответ, содержащий статус-сообщение:

```
{
    "requestId": "13f2475e-f3dc-4c9e-b2f6-3a98320261f1",
    "statusCode": "UPDATED",
    "statusMessage": "Файл Doc_1 успешно обновлен."
}
```

где:

- `requestId` — UUID идентификатор запроса;
- `statusCode` — статус код результата запроса (SUCCESS - запрос выполнен успешно; UPDATED – данные обновлены);
- `statusMessage` — описание статусного сообщения.

Пример неуспешной загрузки после проверки по endpoint'у /status (пример запроса описан в Раздел 5.1.18.1) представлен ниже:

```
{
    "requestId": "13f2475e-f3dc-4c9e-b2f6-3a98320261f1",
    "statusCode": "ERROR",
    "statusMessage": "Произошла ошибка"
}
```

где:

- `requestId` — UUID идентификатор запроса;

– statusCode — статус код результата запроса (ERROR – запрос завершился ошибкой);
– statusMessage — описание статусного сообщения.

4.1.17.6.1.2 Удаление данных из хранилища (Endpoint – deleteAttachment)

Для того чтобы удалить вложения из хранилища S3 необходимо направить следующий запрос:

```
curl -X DELETE "http://<ip-studio>:8088/api/v1/secure/<organization_ogrn>/<datamart_mnemonic>/<installation_name>/<installation_id>/deleteAttachment/Doc_1/requestId/<requestId>" -H "Authorization: Bearer <access_token>"
```

где:

– requestId — UUID идентификатор запроса.

В результате успешного удаления Витрина данных вернёт JSON-ответ, содержащий статус-сообщение:

```
{
    "requestId": "13f2475e-f3dc-4c9e-b2f6-3a98320261f1",
    "statusCode": "SUCCESS",
    "statusMessage": "Файл Doc_1 успешно удален."
}
```

где:

– requestId — UUID идентификатор запроса;
– statusCode — статус код результата запроса (SUCCESS - запрос выполнен успешно);
– statusMessage — описание статусного сообщения.

Если удаление завершилось ошибкой, то Витрина данных вернёт JSON-ответ с кодом ошибки:

```
{
    "requestId": "13f2475e-f3dc-4c9e-b2f6-3a98320261f1",
    "statusCode": "NOT_FOUND",
    "statusMessage": "Файл Doc_1 не найден."
}
```

где:

– requestId — UUID идентификатор запроса;
– statusCode — статус код результата запроса (NOT_FOUND);
– statusMessage — описание статусного сообщения.

4.1.17.6.1.3 Описание возвращаемых кодов

Таблица 4.6 Описание возвращаемых кодов

Наименование	Код	Описание
EMPTY_ATTACHMENT	400	нет файла вложения
ERROR	500	Внутренняя ошибка
SUCCESS	200	Успешное выполнение
UPDATED	200	данные обновлены

4.1.17.7 Маппинг данных

4.1.17.7.1 Маппинг данных (Endpoint – generateMapping)

В данном разделе описывается генерация файла маппинга. Endpoint предназначен для первичной настройки, а также перенастройки сервиса в случае изменения модели данных Витрины.

Файл маппинга генерируется источником данных в формате **Kotlin-script**. В файле описана модель данных Витрины данных в виде структуры объектов Kotlin (`table`, `column`). Объекты `table` описывают таблицы, каждый из них содержит имя таблицы и список колонок в том порядке, в котором они созданы в Витрине. Объекты `column` описывают колонки, каждый из них содержит имя колонки, тип данных, признак обязательности (nullable), признак первичного ключа.

Файл используется сервисом для описания модели данных и валидации входящих данных. Выполняются следующие проверки:

- проверяется соответствие состава полей входящей avro-структуры составу полей, описанных в файле маппинга;
- проверяется соответствие порядка полей входящей avro-структуры порядку полей, описанных в файле маппинга;
- проверяется соответствие типов данных полей входящей avro-структуры типам полей, описанных в файле маппинга. Для полей с установленным признаком обязательности (nullable = false) выполняется проверка на null.

При вызове Endpoint'а `/generateMapping` сервис генерирует файл на основе информации о модели, полученной из развернутой Витрины данных. Файл складывается сервисом на диск, а также возвращается в ответе на вызов.

Пример запроса на генерацию маппинга представлен ниже:

```
curl -X GET "http://<ip-
studio>:8088/api/v1/secure/<organization_ogrn>/<datamart_mnemonic>/<installation_name>/
<installation_id>/generateMapping/aef2f195-0001-4aaa-b171-f2746511e889" -H
"Authorization: Bearer <access_token>"
```

Результат Витрина данных вернёт в формате Kotlin-script:

```
import ru.supercode.mapping.common.ColumnType.*
import ru.supercode.mapping.mapper.dsl.mappingAvro
mappingAvro {
    table("product") {
        column("id", INTEGER) { nullable = false; primary = true; }
        column("name", STRING) { nullable = false; }
    }
    table("stock") {
        column("product_id", INTEGER) { nullable = false; primary = true; }
        column("count_pieces", INTEGER) { nullable = false; }
    }
}
```

4.1.17.8 Валидация данных

4.1.17.8.1 Валидация данных

Валидация порции данных производится в момент обработки и вставки.

25. Примечание:
26. Помимо валидации данных осуществляется валидация параметров запроса. Во всех Endpoint'ах `requestId` должен быть в формате

UUID.

В случае ошибок при валидации результат будет возвращен при вызове Endpoint'а `/status`.

Ошибки, возникающие в процессе обработки Endpoint'а /newDelta:

- отклоняются запросы, которые получены в момент обработки порции данных (`statusCode: PROCESSED`);
- если пустой параметр `dataSetName`;
- прислан запрос с уже зарегистрированным `requestId` и `statusCode` данного `requestId` не равен `NOT_FOUND` или `WAIT_DATA`.

Ошибки, возникающие в процессе обработки Endpoint'а /partOfDelta:

- прислан запрос с незарегистрированным `requestId`;
- прислан запрос с уже зарегистрированным `requestId` и `statusCode` данного `requestId` не равен `NOT_FOUND` или `WAIT_DATA`;
- прислан запрос с `requestId` зарегистрированным для Endpoint'а `/data`;
- прислан запрос с параметром `dataSetName`, который не был зарегистрирован в Endpoint'е `/newDelta`;
- нет файла вложения в параметре `upload`.

Ошибки, возникающие в процессе обработки Endpoint'а `/data`:

- отклоняются запросы, которые получены в момент обработки порции данных (`statusCode: PROCESSED`);
- прислан запрос с незарегистрированным `requestId`;
- прислан запрос с уже зарегистрированным `requestId` и `statusCode` данного `requestId` не равен `NOT_FOUND` или `WAIT_DATA`;
- прислан запрос с `requestId` зарегистрированным для Endpoint'а `/partOfDelta`;
- нет файла вложения в параметре `upload`.

Ошибки, возникающие в процессе обработки Endpoint'а /uploadAttachment:

- нет файла вложения в параметре `upload`.

Ошибки, возникающие в процессе обработки Endpoint'а /generateMapping:

- не созданы логические таблицы в схеме.

4.1.18 Установка утилиты Back Manager

4.1.18.1 Установка модуля

Общий процесс установки состоит из следующих действий:

1. Настроить конфигурацию модуля.
2. Создать на сервере директорию для загрузки файлов модуля.
3. Загрузить файлы модуля в созданную директорию.
4. Запустить модуль.
5. Проверить установку модуля.

4.1.18.1.1 Настройка конфигурации

Настройка конфигурации выполняется путем редактирования параметров файла `application.yml`, пример которого и возможные настройки конфигурации модуля см. в разделе см. Конфигурация утилиты Backup manager (application.yml) Руководства администратора ПО «Витрина данных НСУД».

4.1.18.1.2 Добавление папки для загрузки файлов модуля

Создайте на сервере папку, в которую будут загружены файлы модуля, например, `/opt/backup-manager`.

В случае, если ранее была установлена старая версия утилиты, сделайте его резервную копию.

4.1.18.1.3 Загрузка файлов на сервер

Для загрузки файла на сервер выполните команду:

```
scp file.jar user_name@IP:/opt/backup-manager
```

где,

- `file.jar` - название jar-файла модуля;
- `user_name` - имя пользователя, например, `sudo` или `root`;
- `IP` - адрес сервера;
- `/opt/backup-manager` - директория на сервере, в которую будет загружен файл.

4.1.19 Установка Apache Airflow

4.1.19.1 Подготовка конфигурации

1. Клонировать репозиторий Apache Airflow из официального репозитория

```
git clone https://github.com/apache/airflow.git
```

2. Перейти в папку **airflow**
3. Создать в папке файл `Dockerfile.custom` со следующим содержимым

```
FROM ${BASE_AIRFLOW_IMAGE}
SHELL ["/bin/bash", "-o", "pipefail", "-e", "-u", "-x", "-c"]
USER 0
# Install Java
RUN mkdir -pv /usr/share/man/man1 ¥¥
&& mkdir -pv /usr/share/man/man7 ¥¥
&& curl -fsSL
https://adoptopenjdk.jfrog.io/adoptopenjdk/api/gpg/key/public ¥|
apt-key add - ¥¥
&& echo 'deb https://adoptopenjdk.jfrog.io/adoptopenjdk/deb/ buster
main' > ¥¥
/etc/apt/sources.list.d/adoptopenjdk.list ¥¥
&& apt-get update ¥¥
&& apt-get install --no-install-recommends -y ¥¥
adoptopenjdk-8-hotspot-jre ¥¥
&& apt-get autoremove -yqq --purge ¥¥
&& apt-get clean ¥¥
&& rm -rf /var/lib/apt/lists/¥*
# Установка JDR 1.8
ENV JAVA_HOME=/usr/lib/jvm/adoptopenjdk-8-hotspot-jre-amd64
USER ${AIRFLOW_UID}
# Установка компонента apache.spark
RUN pip install --upgrade pip ¥¥
&& pip install --no-cache-dir --user 'apache-airflow[apache.spark]'
```

4.1.19.2 Установка Apache Airflow

Установка **Apache Airflow** выполняется с помощью **Docker**. Установка **Docker** описана в разделе Установка Arenadata Cluster Manager (ADCM).

1. Создайте новый образ в *Docker*, в той же директории, где расположен файл

Dockerfile.custom:

```
docker build . ¥¥
--build-arg BASE_AIRFLOW_IMAGE="apache/airflow:2.0.1-python3.8" ¥¥
-f Dockerfile.java8 ¥¥
-t airflow:2.0.1-python3.8-custom
```

2. Скачать файл docker-compose.yaml с официального сайта Airflow https://Airflow.apache.org/docs/apache-airflow/2.0.1/docker-compose.yaml
3. Установить docker-compose.yaml через SSH-консоль технологического пользователя.
   – Выдать права на исполнение файла

```
sudo chmod +x /usr/local/bin/docker-compose
```

   – Запустить sudo

```
ln -s /usr/local/bin/docker-compose /usr/bin/docker-compose
```

   – Проверить корректность установки

```
$ docker-compose --version
docker-compose version 1.28.6, build 1110ad01
```

1. Перейти в директорию с файлом docker-compose.yml:

```
cd ~/direct
```

5. Инициализировать компоненты

```
docker-compose up airflow-init
```

6. Запустить сервисы с новым образом

```
docker-compose up
AIRFLOW_IMAGE_NAME=apache/airflow:2.0.1-python3.8-custom
```

4.1.20 Установка Apache Spark

Установка **Docker** описана в разделе Установка Arenadata Cluster Manager (ADCM).

4.1.20.1 Подготовка конфигурации

1. Клонировать репозиторий **Apache Airflow** из официального репозитория:

```
git clone https://github.com/big-data-europe/docker-spark.git
```

2. Перейти в директорию docker-spark:

```
cd ~/direct
```

3. Выполнить скрипт ./build.sh или вручную последовательно запустить следующие команды
   – сборка базового образа

```
docker build -t bde2020/spark-base:3.1.1-hadoop3.2
```

   – сборка образа мастер

```
docker build -t bde2020/spark-master:3.1.1-hadoop3.2
```

   – сборка образа воркера

```
docker build -t bde2020/spark-worker:3.1.1-hadoop3.2
```

4. Проверить наличие собранных образов, выполнив команду docker:

```
REPOSITORY TAG IMAGE ID CREATED SIZE
bde2020/spark-worker 3.1.1-hadoop3.2 05c349b4646f 4 minutes ago 460MB
bde2020/spark-master 3.1.1-hadoop3.2 7918c5357d6d 4 minutes ago 460MB
bde2020/spark-base 3.1.1-hadoop3.2 5430434220d2 4 minutes ago 460MB
```

4.1.20.2 Установка

1. Перейти в директорию *docker-spark*, где располагается файл `docker-compose.yml`:

```
cd ~/direct
```

2. Запустить Apache Spark командой:

```
docker-compose up
```

3. Проверить командой:

```
docker ps
```

В списке запущенных образов должны присутствовать `spark-worker` и `spark-master`.

4.1.21 Установка Apache Hadoop

Установка **Docker** описана в разделе Установка Arenadata Cluster Manager (ADCM).

4.1.21.1 Подготовка конфигурации

1. Клонировать репозиторий **Apache Hadoop** из официального репозитория

```
git clone https://github.com/big-data-europe/docker-hadoop.git
```

Перейти в папку **docker-hadoop**

```
cd ~/direct
```

2. Выполнить скрипт `make build` или вручную последовательно запустить следующие команды:
   - сборка базового образа

```
docker build -t bde2020/hadoop-base:master ./base
```

   - сборка остальных образов

```
docker build -t bde2020/hadoop-namenode:master ./namenode
docker build -t bde2020/hadoop-datanode:master ./datanode
docker build -t bde2020/hadoop-resourcemanager:master
./resourcemanager
docker build -t bde2020/hadoop-nodemanager:master ./nodemanager
docker build -t bde2020/hadoop-historyserver:master
./historyserver
docker build -t bde2020/hadoop-submit:master ./submit
```

3. Проверить наличие собранных образов:
   Выполнить команду docker:

```
REPOSITORY TAG IMAGE ID CREATED SIZE
bde2020/hadoop-submit master 665a424edc23 9 minutes ago 1.37GB
bde2020/hadoop-historyserver master ff53bf6835e2 9 minutes ago 1.37GB
bde2020/hadoop-nodemanager master c48c47bc840f 9 minutes ago 1.37GB
bde2020/hadoop-resourcemanager master 74fc55d664d2 9 minutes ago 1.37GB
bde2020/hadoop-datanode master f69c6460a292 9 minutes ago 1.37GB
bde2020/hadoop-namenode master 7a8250da8510 9 minutes ago 1.37GB
bde2020/hadoop-base master aeb6500ab4b5 10 minutes ago 1.37GB
```

4.1.21.2 Процесс установки

1. Перейти в директорию **docker-hadoop**, где располагается файл `docker-compose.yml`

```
cd ~/direct
```

2. Поднять Apache Hadoop командой

```
docker-compose up
```

3. Проверить командой

```
docker ps
```

В списке запущенных образов должны присутствовать `hadoop-resourcemanager`, `hadoop-historyserver`, `hadoop-datanode`, `hadoop-nodemanager`, `hadoop-namenode`.

4.1.22 Установка Tarantool (Vinyl)

Установка **Docker** описана в разделе «Установка Arenadata Cluster Manager (ADCM).

4.1.22.1 Подготовка конфигурации

1. Клонировать репозиторий Tarantool из официального репозитория

```
git clone https://github.com/tarantool/docker.git
```

2. Выполнить сборку согласно инструкции: https://github.com/tarantool/docker#how-to-push-an-image-for-maintainers
3. Перейти в клонированную директорию **docker**

```
cd ~/direct
```

4. Выполнить сборку образа *docker* для Tarantool

```
export TAG=2
export OS=alpine DIST=3.9 VER=2.8.0
PORT=5200 make -f .gitlab.mk build
```

5. Проверить наличие собранных образов командой docker images

```
REPOSITORY TAG IMAGE ID CREATED SIZE
tarantool/tarantool 2 27f80564ecce 2 minutes ago 298MB
```

6. Создать в клонированной директории *docker* файл `docker-compose.yml` со следующим содержимым:

```
version: "3"
services:
tarantool1:
image: tarantool/tarantool:2
container_name: tarantool1
restart: always
networks:
- tarantool_network
ports:
- "3301:3301"
volumes:
- tarantool1_volume:/opt/tarantool
environment:
# https://github.com/tarantool/docker#environment-variables
TARANTOOL_REPLICATION: "tarantool1,tarantool2"
tarantool2:
  image: tarantool/tarantool:2
```

```
  container_name: tarantool2
  restart: always
  networks:
  - tarantool_network
  ports:
  - "3302:3301"
  volumes:
  - tarantool2_volume:/opt/tarantool
  environment:
    TARANTOOL_REPLICATION: "tarantool1,tarantool2"
    volumes:
    tarantool1_volume:
    tarantool2_volume:
    networks:
    tarantool_network:
    driver: bridge
```

4.1.22.2 Процесс установки

1. Перейти в директорию `/docker`, где располагается файл `docker-compose.yml`

```
cd ~/direct
```

2. Пример запуска мастер-мастер репликации:

```
https://github.com/tarantool/docker#start-a-master-master-replica-set
```

1. Поднять Tarantool командой

```
docker-compose up
```

2. Проверить командой

```
docker ps
```

В списке запущенных образов должны присутствовать `tarantool1` и `tarantool2`.

4.1.23 Установка CSV-Uploader

4.1.23.1 Процесс установки CSV-uploader

Действия по установке выполняются через SSH консоль технологического пользователя.

Общий процесс установки состоит из следующих действий:

1. Настроить конфигурацию модуля.
2. Создать на сервере директорию для загрузки файлов модуля.
3. Загрузить файлы модуля в созданную директорию.
4. Запустить модуль.
5. Проверить установку модуля.

4.1.23.1.1 Настройка конфигурации

Настройка конфигурации выполняется путем редактирования параметров файла конфигурации `application.yml`.

Пример файла `application.yml` и возможные настройки конфигурации модуля см. в разделе Конфигурация CSV-uploader (application.yml) Руководства администратора ПО «Витрина данных НСУД».

4.1.23.1.2 Загрузка JAR-файла на сервер

Для загрузки файла на сервер выполните команду:

```
scp file.jar user_name@IP:/home/dir
```

где,

- `file.jar` - название JAR-файла;
- `user_name` - имя пользователя, например, `sudo` или `root`;
- `IP` - адрес сервера;
- `/home/dir` - директория на сервере, в которую будет загружен файл.

4.1.24 Установка REST-адаптера

Установка сервисов и необходимых сервисных баз данных.

> **Внимание:**
>
> Установка данных сервисов выполняется после выполнения действий, описанных в разделе Предварительные действия.

Действия по установке выполняются на сервере **ADCM** через SSH-консоль технологического пользователя.

4.1.24.1 Установка docker-образов

Установка Docker описана в разделе Установка Arenadata Cluster Manager (ADCM).

Далее необходимо установить образ REST-адаптера, при этом файл образа `.tar` должен быть загружен через Docker из дистрибутива программы.

Пример команды для установки:

```
docker image load -i <image_file>
```

4.1.24.2 Подготовка конфигурации

Для подготовки скриптов необходимо выполнить следующие действия:

1. Отредактировать переменные в файле `application.yml` для рабочей среды.
   Отредактированный файл необходимо поместить в директорию `config`:
   - указать корректный путь до целевой БД. Переменная `jdbc_url`.

   Например:

```
jdbc:{drv_name}://{IP_Host}:{Port}/{db_name}
```

   , где:
   - `drv_name` - имя драйвера , берется из описания драйвера базы
   - `IP_Host` - ip или имя хоста с базой;
   - `Port` - порт, на котором будет слушать сервис. По умолчанию 8080;
   - `db_name` - имя базы данных.
   - указать `driver-class-name` - имя класса , берется из описания драйвера базы;
   - указать переменную `file_path` - наименование файла с описанием запросов, по умолчанию `sample.yaml`;
   - указать переменные `execquery` - сопоставление operationId из файла с описанием запросов с файлом обработчиком, по умолчанию `sample.peb`.

4.1.24.3 Процесс установки

Для установки необходимо выполнить следующие действия:

1. Перейти в директорию с файлом `docker-compose.yml`:

```
cd ~/direct
```

2. Поднять сервис конечных точек командой:

```
docker-compose up -d
```

3. Проверить установку следующей командой:

```
docker ps
```

В списке запущенных образов должен присутствовать `rest-adapter`.

4.1.25 Установка Counter-provider

4.1.25.1 Процесс установки

Действия по установке выполняются через SSH консоль технологического пользователя.

Общий процесс установки состоит из следующих действий:

1. Настроить конфигурацию модуля.
2. Создать на сервере директорию для загрузки файлов модуля.
3. Загрузить файлы модуля в созданную директорию.
4. Запустить модуль.
5. Проверить установку модуля.

4.1.25.1.1 Настройка конфигурации

Настройка конфигурации выполняется путем редактирования параметров файла конфигурации `application.yml`.

Пример файла `application.yml` и возможные настройки конфигурации модуля см. в разделе Конфигурация модуля Counter-Provider (application.yml) Руководства администратора ПО «Витрина данных НСУД».

4.1.25.1.2 Загрузка JAR-файла на сервер

Для загрузки файла на сервер выполните команду

```
scp file.jar user_name@IP:/home/dir
```

где,

- `file.jar` - название JAR-файла;
- `user_name` - имя пользователя, например, `sudo` или `root`;
- `IP` - адрес сервера;
- `/home/dir` - директория на сервере, в которую будет загружен файл.

4.1.26 Установка коннектора Kafka-Postgres

1. Из полученного дистрибутива ПО скопировать и загрузить в папку **kafka-postgres-connector** файлы:
   - `kafka-postgres-writer-0.3.0.jar`,
   - `kafka-postgres-reader-0.3.0.jar`,
   - `kafka-postgres-avro-0.3.0.jar`.
2. Скопировать конфигурационные файлы **KAFKA-POSTGRES-WRITER** `application.yaml` и **KAFKA-POSTGRES-READER** `application.yaml` в папку **kafka-postgres-connector/config**.

   Конфигурационный файл **KAFKA-POSTGRES-WRITER** `application.yaml`:

```
logging:
level:
    ru.datamart.kafka: ${LOG_LEVEL:DEBUG}
    org.apache.kafka: ${KAFKA_LOG_LEVEL:INFO}
```

```
http:
port: ${SERVER_PORT:8096}

vertx:
pools:
    eventLoopPoolSize: ${VERTX_EVENT_LOOP_SIZE:12}
    workersPoolSize: ${VERTX_WORKERS_POOL_SIZE:32}
verticle:
    query:
    instances: ${QUERY_VERTICLE_INSTANCES:12}
    insert:
    poolSize: ${INSERT_WORKER_POOL_SIZE:32}
    insertPeriodMs: ${INSERT_PERIOD_MS:1000}
    batchSize: ${INSERT_BATCH_SIZE:500}
    consumer:
    poolSize: ${KAFKA_CONSUMER_WORKER_POOL_SIZE:32}
    maxFetchSize: ${KAFKA_CONSUMER_MAX_FETCH_SIZE:10000}
    commit:
    poolSize: ${KAFKA_COMMIT_WORKER_POOL_SIZE:1}
    commitPeriodMs: ${KAFKA_COMMIT_WORKER_COMMIT_PERIOD_MS:1000}

client:
kafka:
    consumer:
    checkingTimeoutMs: ${KAFKA_CHECKING_TIMEOUT_MS:10000}
    responseTimeoutMs: ${KAFKA_RESPONSE_TIMEOUT_MS:10000}
    consumerSize: ${KAFKA_CONSUMER_SIZE:10}
    closeConsumersTimeout: ${KAFKA_CLOSE_CONSUMER_TIMEOUT:15000}
    property:
        bootstrap.servers: ${KAFKA_BOOTSTRAP_SERVERS:kafka.host:9092}
        group.id: ${KAFKA_CONSUMER_GROUP_ID:postgres-query-execution}
        auto.offset.reset: ${KAFKA_AUTO_OFFSET_RESET:earliest}
        enable.auto.commit: ${KAFKA_AUTO_COMMIT:false}
        auto.commit.interval.ms: ${KAFKA_AUTO_INTERVAL_MS:1000}

datasource:
postgres:
    database: ${POSTGRES_DB_NAME:test}
    user: ${POSTGRES_USERNAME:dtm}
    password: ${POSTGRES_PASS:dtm}
    hosts: ${POSTGRES_HOSTS:localhost:5432}
    poolSize: ${POSTGRES_POOLSIZE:10}
    preparedStatementsCacheMaxSize: ${POSTGRES_CACHE_MAX_SIZE:256}
    preparedStatementsCacheSqlLimit: ${POSTGRES_CACHE_SQL_LIMIT:2048}
    preparedStatementsCache: ${POSTGRES_CACHE:true}
```

Конфигурационный файл **KAFKA-POSTGRES-READER** application.yaml:

```
logging:
level:
    ru.datamart.kafka: ${LOG_LEVEL:DEBUG}
    org.apache.kafka: ${KAFKA_LOG_LEVEL:INFO}

http:
port: ${SERVER_PORT:8094}

vertx:
pools:
    eventLoopPoolSize: ${VERTX_EVENT_LOOP_SIZE:12}
    workersPoolSize: ${VERTX_WORKERS_POOL_SIZE:32}
verticle:
```

```
    query:
    instances: ${QUERY_VERTICLE_INSTANCES:12}

datasource:
postgres:
    database: ${POSTGRES_DB_NAME:test}
    user: ${POSTGRES_USERNAME:dtm}
    password: ${POSTGRES_PASS:dtm}
    hosts: ${POSTGRES_HOSTS:localhost:5432}
    poolSize: ${POSTGRES_POOLSIZE:10}
    preparedStatementsCacheMaxSize: ${POSTGRES_CACHE_MAX_SIZE:256}
    preparedStatementsCacheSqlLimit: ${POSTGRES_CACHE_SQL_LIMIT:2048}
    preparedStatementsCache: ${POSTGRES_CACHE:true}
    fetchSize: ${POSTGRES_FETCH_SIZE:1000}

kafka:
client:
    property:
    key.serializer: org.apache.kafka.common.serialization.ByteArraySerializer
    value.serializer: org.apache.kafka.common.serialization.ByteArraySerializer
```

4.1.27 Установка Arenadata Cluster Manager (ADCM)

> Внимание:
>
> При условии установки CentOS 7.9

ADCM – сервер, с которого будет централизованно производиться установка почти всех компонентов программы. Данный компонент обязателен для установки.

ADCM поставляется в docker-контейнерах, поэтому чтобы установить ADCM на сервере должен быть установлен Docker или совместимое программное обеспечение для работы с контейнерами. Текущая версия программного обеспечения несовместима с SELinux.

Подробная инструкция по установке **ADCM** приведена в официальной документации разработчика: https://docs.arenadata.io/adcm/user/install.html.

Установка **ADCM** осуществляется пользователем согласно следующим этапам:

1. Выключить `selinux`. Для этого в файле `/etc/selinux/config` установите значение `SELINUX=disabled` и перезапустить сервер командой `reboot now`.
2. Установите Docker для этого используйте следующие команды

```
yum-config-manager --add-repo
https://download.docker.com/linux/centos/docker-ce.repo
yum install -y containerd.io
yum install -y docker-ce-3:18.09.1-3.el7
systemctl enable docker
systemctl start docker
```

3. Установить **ADCM** в **Docker**. Для этого используйте следующие команды

```
docker pull arenadata/adcm:latest
docker create --name adcm -p 8000:8000 -v /opt/adcm:/adcm/data
arenadata/adcm:latest
```

4. Запустите **ADCM** командой

```
docker start adcm
```

5. Убедитесь, что **ADCM** установлен корректно, для этого перейдите в браузере по адресу `http://<ip_of_your_server>`.

   В случае корректной установки откроется окно «Авторизация» (см. Рисунок - 4.2).

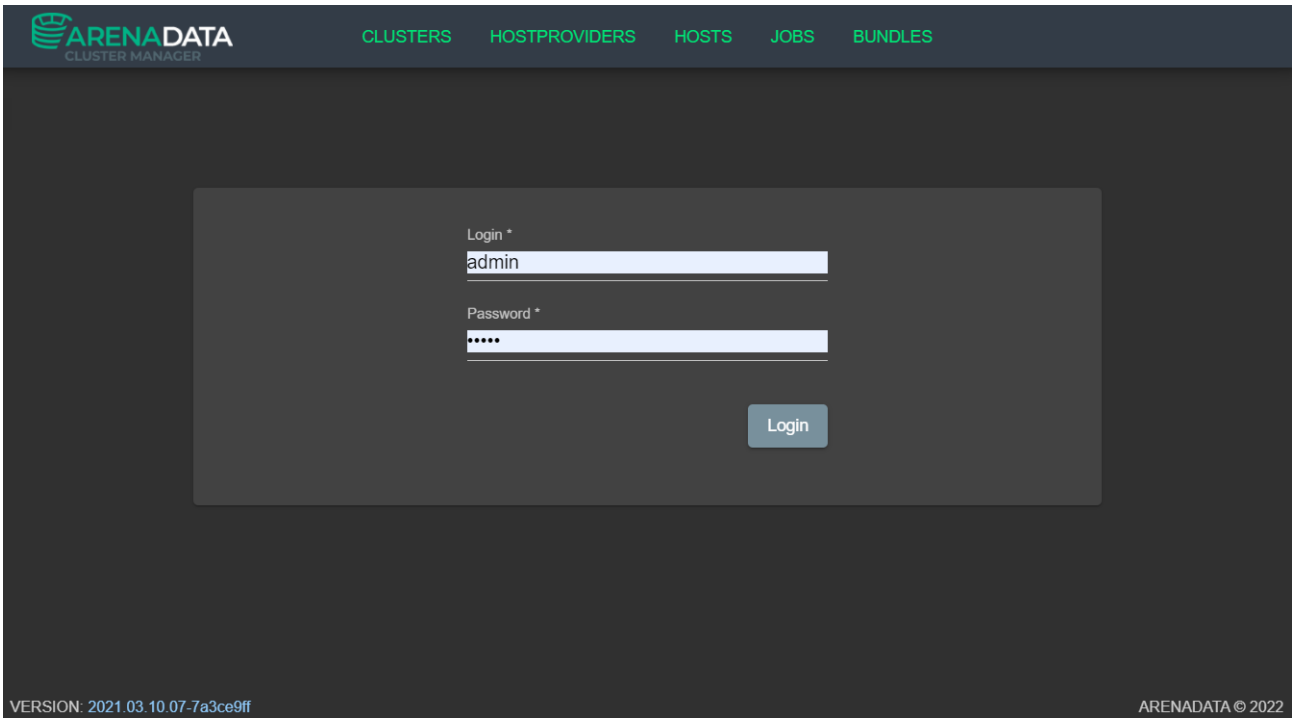

Рисунок - 4.2 Окно «Авторизация»

Для авторизации используйте следующие данные:

- пользователь - `admin`;
- пароль - `admin`.

После завершения процедуры авторизации откроется главное окно программы Arenadata Cluster Manager (см. Рисунок - 4.3).

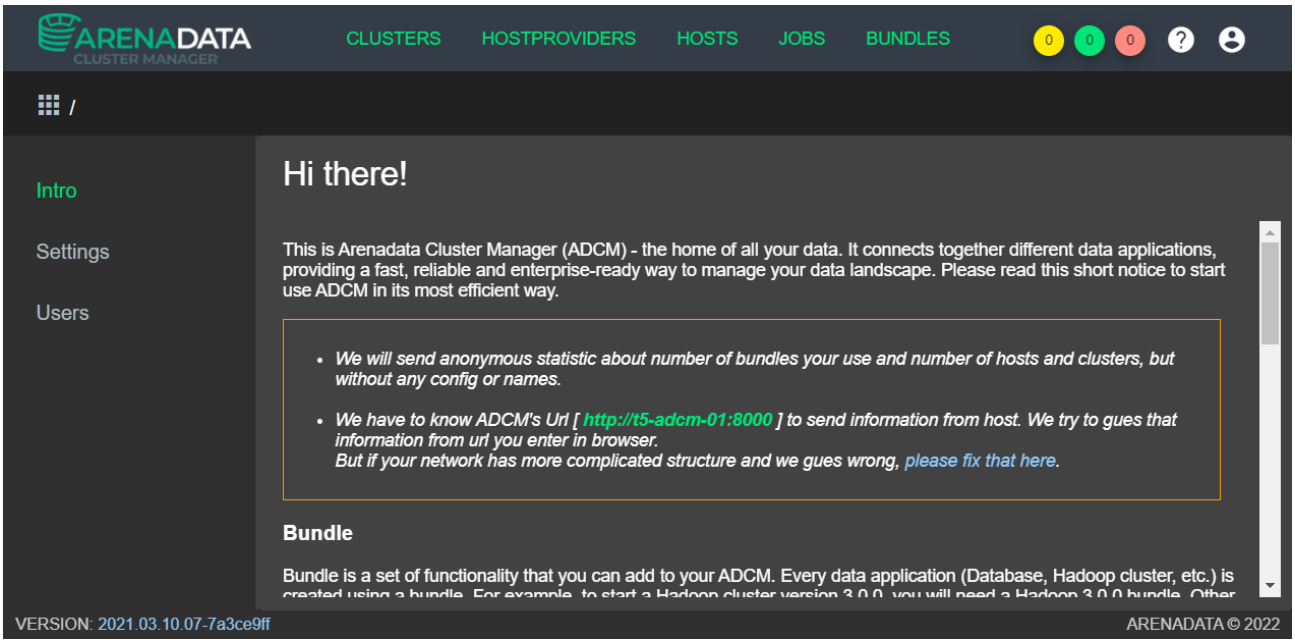

Рисунок - 4.3 Главное окно программы «Arenadata Cluster Manager»

4.1.28 Установка Arenadata Streaming (ADS)

Внимание:
При условии установки CentOS 7.9

ADS устанавливается на кластере с помощью **ADCM** из загруженного установочного пакета (бандла).

Скачать установочный пакет (версия: `ads_v1.6.0.0-1`) можно с сайта разработчика: https://store.arenadata.io/#products/arenadata_streaming.

Подробная инструкция по установке **ADS** приведена в официальной документации разработчика:

- https://docs.arenadata.io/DTM/Install/ADS.html
- https://docs.arenadata.io/ads/Install/index.html.

В данной инструкции описывается установка кластера Kafka и импорт уже установленного кластера **Zookeeper**.

1. Для создания кластера необходимо иметь предварительно загруженный бандл (версия: ads_v1.6.0.0-1). В графическом интерфейсе **ADCM** перейти в раздел *CLUSTERS -> Create cluster*. Создать кластер (см. Рисунок - 4.4).

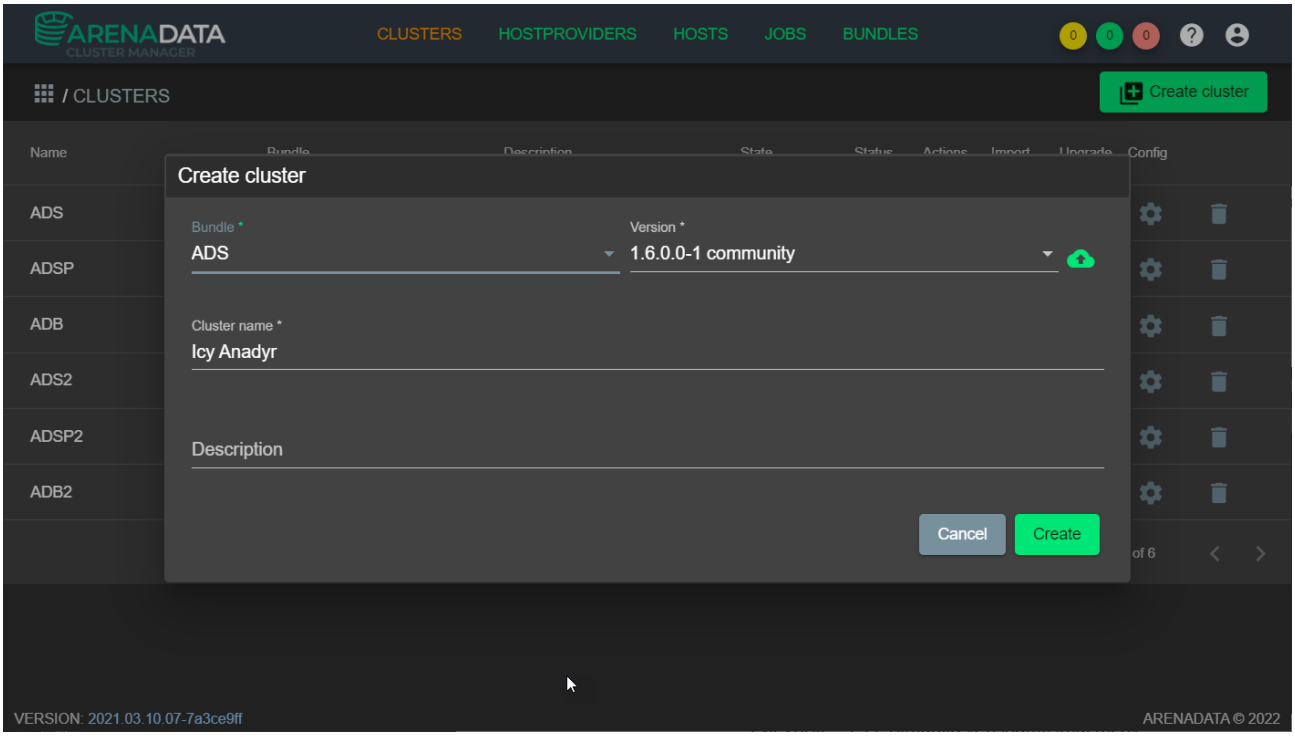

Рисунок - 4.4 Создание кластера ADS через графический интерфейс ADCM

2. В разделе **Services** графического интерфейса ADCM необходимо добавить сервисы в созданный кластер (см. Рисунок - 4.5). Версии сервисов могут отличаться от указанных на рисунке.

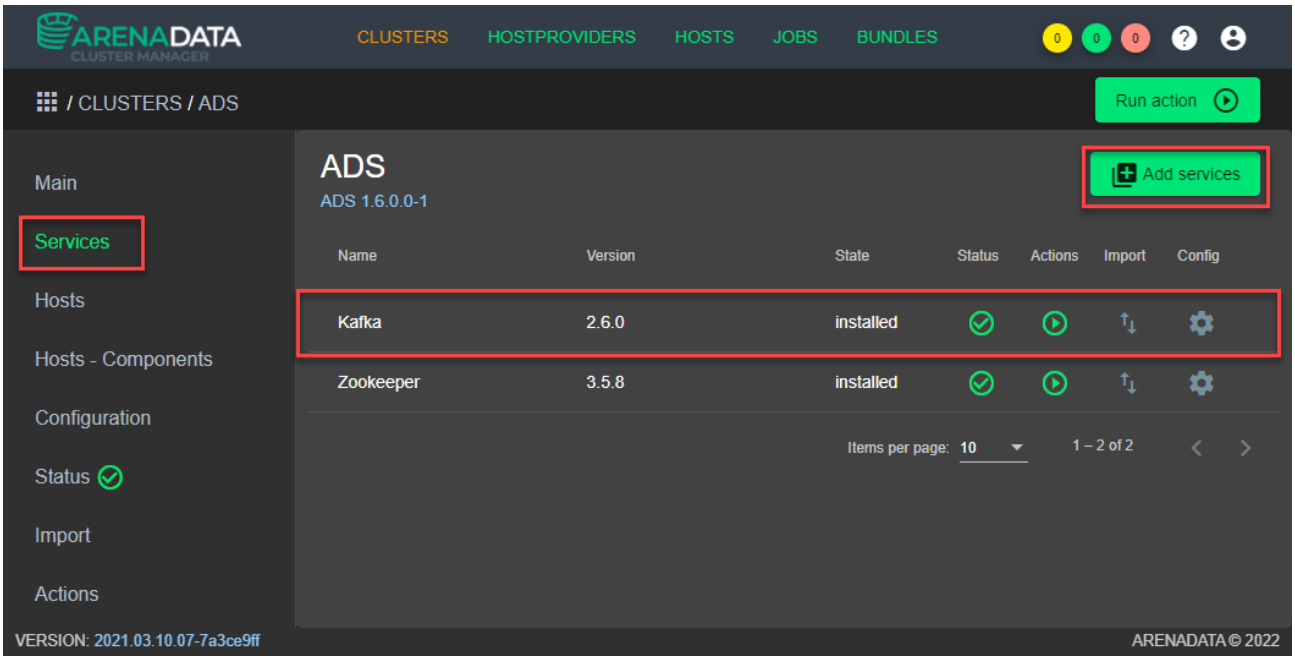

Рисунок - 4.5 Добавление через графический интерфейс ADCM сервисов в созданный кластер

3. В случае, если **Kafka** и **Zookeeper** разворачивается в одном кластере, необходимо добавить сервис **Zookeeper**.
4. В разделе **Hosts** графического интерфейса ADCM указать серверы созданного кластера, на которых будет развёрнуто ПО **ADS** (см. Рисунок - 4.6).

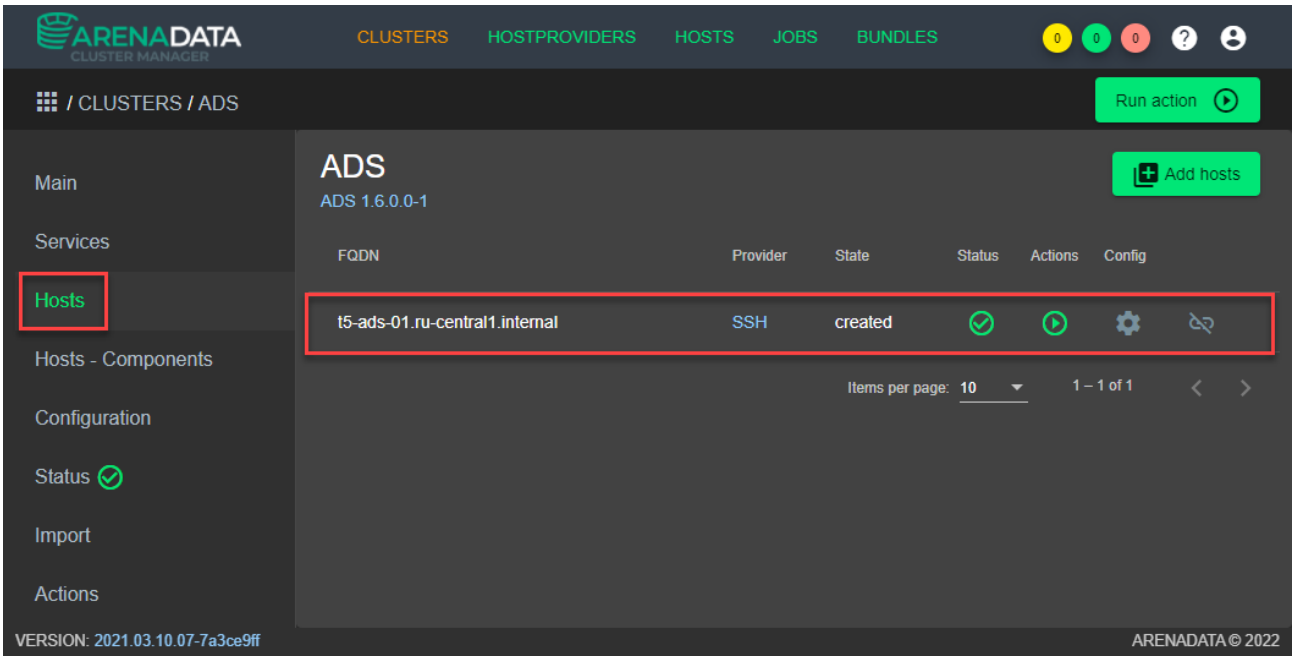

Рисунок - 4.6 Выбор через графический интерфейс ADCM серверов для развёртывания кластера ADS

5. В разделе *Hosts* - *Components* указать три узла (ноды) **Kafka** (см Рисунок - 4.7).

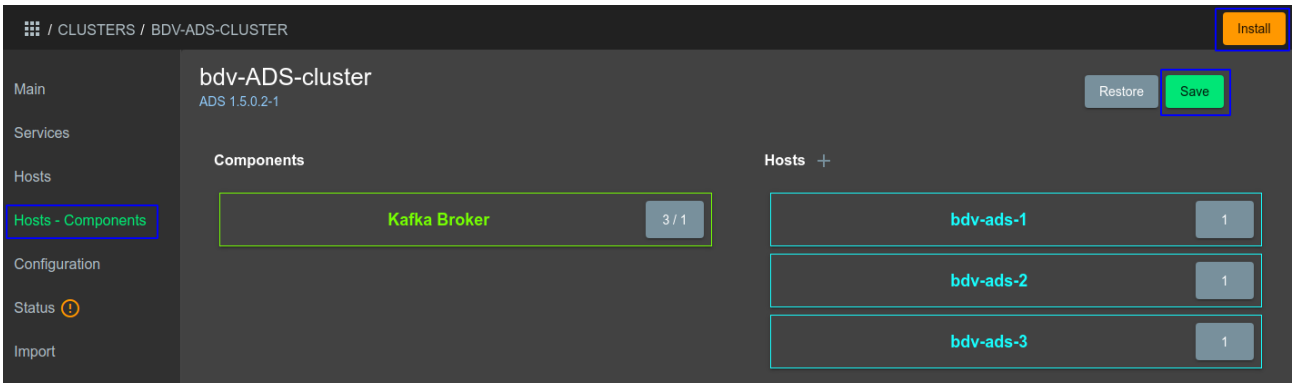

Рисунок - 4.7 Выбор через графический интерфейс ADCM узлов Kafka

6. Указать кластер *Zookeeper* в разделе **Import**, в случае если **Zookeeper** находится на другом кластере (см Рисунок - 4.8).

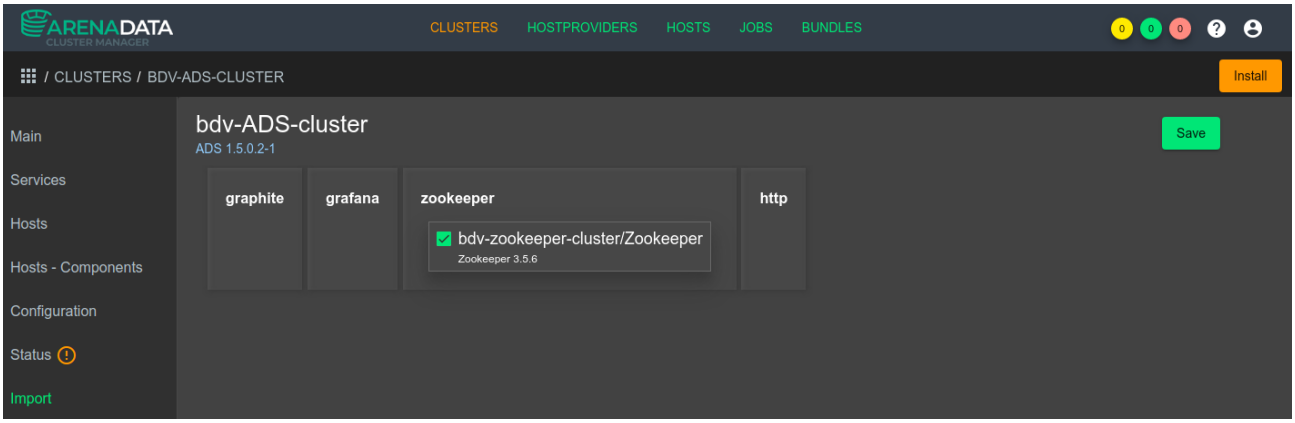

Рисунок - 4.8 Выбор через графический интерфейс ADCM кластера *Zookeeper* для импорта

7. Нажмите кнопку **Install**, чтобы запустить процесс установки. В окне «Дополнительные параметры» нажмите кнопку **Run**, чтобы выполнить установку (см Рисунок - 4.9).

Рисунок - 4.9 Запуск через графический интерфейс ADCM процесса установки

8. Проверьте, что установка завершена, для этого:
   – Проверьте лог-файлы *Zookeeper* по относительному пути:

```
/var/log/zookeeper/
```

   – Проверьте лог-файл *Kafka* по относительному пути:

```
/var/log/kafka/
```

   – Проверьте настройки подключения к кластеру **Zookeeper** с помощью сторонней

утилиты **KafkaTool** (см Рисунок - 4.10).

– Проверьте корректность разрешения имен серверов для автоматического определения bootstrap-серверов **Kafka**. При отсутствии DNS-севера добавьте в локальный `hosts` адреса **Kafka**.
– Определите путь `chroot path`, для этого выполните на любом сервере **Zookeeper** команду:

```
/usr/lib/zookeeper/bin/zkCli.sh
```

Далее, выполните команду:

```
ls /arenadata/cluster
```

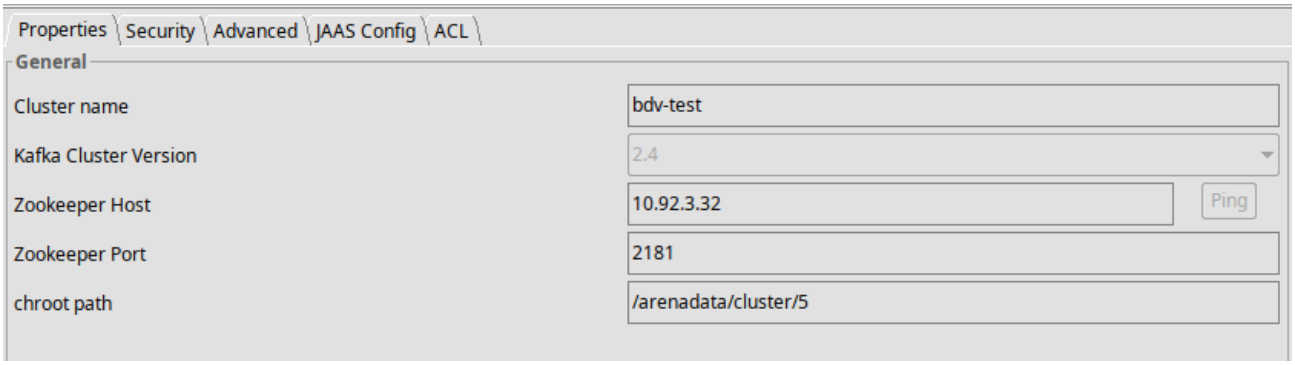

Рисунок - 4.10 Проверка через графический интерфейс ADCM настроек подключения к кластеру Zookeeper

При проверке, в данном руководстве, использовался графический интерфейс приложения Offset Explorer 2.1 (https://www.kafkatool.com/download.html).

4.1.29 Установка компонента сбора данных запросов и ответов Витрины данных

Компонент сбора данных запросов и ответов Витрины данных реализован с целью проведения бизнес-мониторинга ИЭП процессов обработки запросов Типовым ПО «Витрина данных», как в целом, так и в части функционирования отдельных витрин для последующей передачи данных в СЦЛ.

4.1.29.1 Процесс установки

Общий процесс установки состоит из следующих действий:

1. Настройка логирования модулей.
2. Установка и настройка Vector.
3. Установка и настройка HaProxy.
4. Установка и настройка fluentbit.
5. Установка ClickHouse.

4.1.29.1.1 Настройка логирования модулей

На стороне модулей **ПОДД-адаптер**, **Модуль MPPR**, **BLOB-адаптер** и **Сервис формирования документов** необходимо настроить формирование логов в формате JSON.

Для этого необходимо в файле `logback.xml` включить `net.logstash.logback.encoder.LogstashEncoder`.

Пример logback.xml:

```xml
<?xml version="1.0" encoding="UTF-8"?>
    <configuration>
    <appender name="FILE" class="ch.qos.logback.core.rolling.RollingFileAppender">
        <file>logs/application.log</file>
        <rollingPolicy class="ch.qos.logback.core.rolling.TimeBasedRollingPolicy">
            <!-- daily rollover -->
            <fileNamePattern>logs/application.%d{yyyy-MM-dd}.log</fileNamePattern>
            <!-- keep 30 days' worth of history capped at 3GB total size -->
            <maxHistory>30</maxHistory>
            <totalSizeCap>3GB</totalSizeCap>
        </rollingPolicy>
        <encoder class="net.logstash.logback.encoder.LogstashEncoder" />
    </appender>

    <root level="INFO">
        <appender-ref ref="FILE" />
    </root>
</configuration>
```

Подробная информация об encoder

4.1.29.1.2 Установка и настройка Vector

Установка производится по официальной документации Vector

Настройка Vector:

Пример настройки source:

```
json_source:
    type: fluent
    address: 0.0.0.0:24226
```

Пример фильтрации сообщений, имеющих флаг `scl`:

```
scl_tags_filter:
type: filter
inputs:
  - json_source
condition:
  type: "vrl"
  source: |-
    exists(.tags) && includes(array!(.tags), "TYPE_SCL")
```

Пример парсинга scl-сообщений:

```
scl_message_remap:
    type: remap
    inputs:
    - scl_tags_filter
    source: |-
    . = parse_json!(.message)
```

Пример отправки scl-сообщений в **Kafka**:

```
podd_agent_sink:
    type: kafka
    inputs:
      - scl_message_remap
    bootstrap_servers: kafka:9092
    topic: "<префикс>.scl.signal"
    acknowledgements: true
    compression: "gzip"
    encoding:
      codec: json
    healthcheck: true
```

4.1.29.1.3 Установка и настройка HaProxy

Установка производится по официальной документации HaProxy

Для настройки HaProxy в секции `backend` нужно перечислить список установленных инстансов **Vector**.

Пример файла `haproxy.cfg`:

```
global
    log 127.0.0.1 local2

    chroot /var/lib/haproxy
    pidfile /var/run/haproxy.pid
    maxconn 4000
    user haproxy
    group haproxy
    daemon

    stats socket /var/lib/haproxy/stats

defaults
    mode tcp
    log global
    retries 3

    maxconn 3000

listen stats
    bind                0.0.0.0:1936
    mode                http
    stats               enable
    stats               uri /

frontend services
    bind 0.0.0.0:24226
    default_backend services
    mode tcp

backend services
    balance roundrobin
    mode tcp
    server vector01 vector-01:24226
    server vector02 vector-02:24226
```

4.1.29.1.4 Установка и настройка FluentBit

Установка производится по официальной документации FluentBit.

Далее необходимо настроить FluentBit на чтение файлов с логами приложений.

Пример файла конфигурации `fluent-bit.conf`:

```
[SERVICE]
    flush        5
    daemon       off
    log_level    info
    parsers_file parsers.conf
[INPUT]
    name tail
    path <путь до лог файла приложения>
    tag *
    parser json
[OUTPUT]
    name forward
    match *
    host haproxy
    port 24226
```

Пример файла `parsers.conf`:

```
[PARSER]
    Name        json
    Format      json
```

На этом настройка fluentbit завершена.

4.1.29.1.4.1 Включение / выключение отправки сообщений в СЦЛ

Отправка логов в СЦЛ осуществляется автоматически после корректной настройки компонента.

Для выключения отправки логов можно закомментировать блок `podd_agent_sink` отправки сообщений в Kafka в настройках Vector.

4.1.29.1.5 Установка и настройка ClickHouse

Установка производится по официальной документации ClickHouse

Пример задания конфигурационных настроек:

```
clickhouse_default_config:
clickhouse:
   logger:
      level: trace
      log: /var/log/clickhouse-server/clickhouse-server.log
      errorlog: /var/log/clickhouse-server/clickhouse-server.err.log
      size: 1000M
      count: 10
   http_port: 8123
   tcp_port: 9000
   listen_host: 0.0.0.0
   max_connections: 4096
   keep_alive_timeout: 3
   user_directories:
      users_xml:
      path: users.xml
      local_directory:
      path: "{{ clickhouse_root_data_folder }}/access/"
   path: "{{ clickhouse_root_data_folder | add_slash }}"
```

4.2 Установка ПО конфигурации лайт

> **Внимание:**
>
> Перед установкой программы необходимо обязательно выполнить предварительные действия по настройке (см. раздел Предварительные действия).

Состав компонентов дистрибутива и номер версии программы приведены в разделе Состав компонентов в дистрибутиве документа «Техническое описание программы ПО «Витрина данных НСУД»».

Требования к серверному оборудованию, телекоммуникационному оборудованию и каналам связи приведены в Раздел 1 документа «Техническое описание программы ПО «Витрина данных НСУД»».

4.2.1 Настройка конфигурационного файла

Чтобы запустить процесс установки программы с помощью Ansible, необходимо настроить конфигурационный файл. Для этого выполните следующие действия:

1. Переименовать файл `custom.example.yml` расположенный в папке *ansible/group_vars/* в `custom.yml`. Для этого выполните команду:

```
cp -n ansible/group_vars/custom.example.yml ansible/group_vars/custom.yml
```

2. В файле `custom.yml` указать корректные значения для следующих переменных:
   - `server_ip` - адреса сервера. Укажите IP-адрес сервера, на который будет установлена программа. Например:

```
server_ip: "172.16.10.59"
```

   - `server_user_name` - имя пользователя операционной системы. Укажите имя пользователя операционной системы сервера, под которым будет производиться установка программы (см. раздел Создание пользователя), например:

```
server_user_name: datamart
```

   - `podd_kafka_topic_prefix` - префикс перед именем топиков для ПОДД-агента. Например:

> **Внимание:**
>
> В качестве префикса рекомендуется использовать мнемонику витрины. После определения параметра префикса следует обязательно ставить символ `.` (точка)!
> Пример: `podd_kafka_topic_prefix: "prod_vitrina98."`

4.2.2 Установка программы

Для установки программы выполните команду:

```
docker-ansible-cmd ansible-playbook -i hosts install.yml
```

Начнется процесс установки программы (см. Рисунок - 4.11):

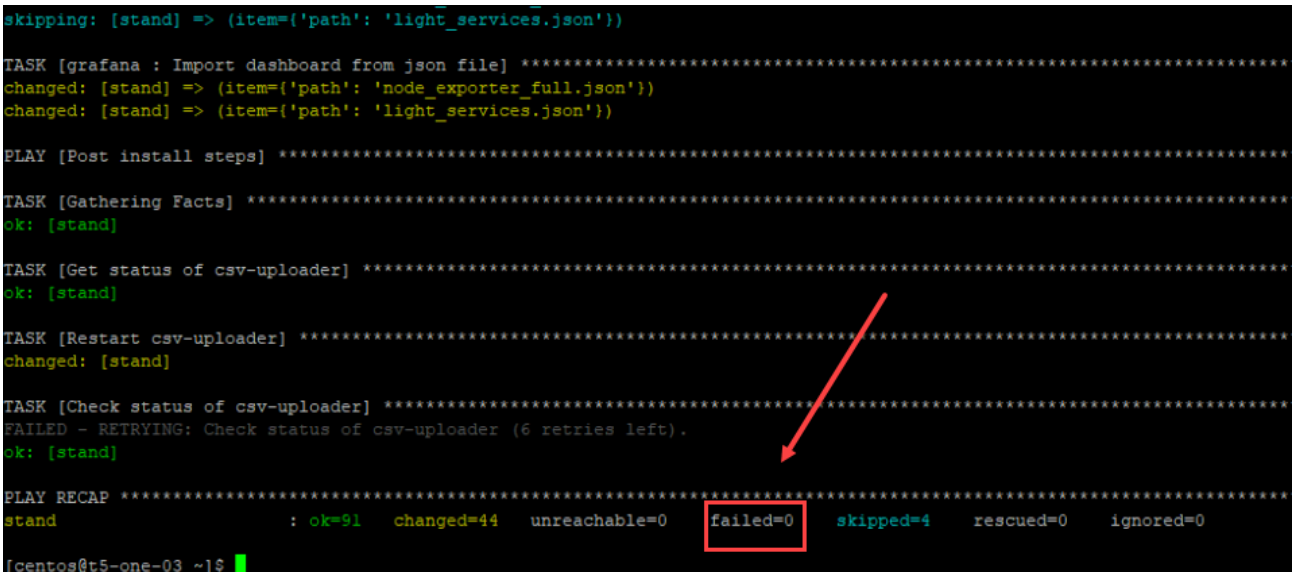

Рисунок - 4.11 Процесс установки

Установка программы завершена. При успешной установке параметр `failed` должен иметь значение - `0` (см. Рисунок - 4.11).

Это значит, что все компоненты программы установлены, а необходимые взаимосвязи между ними настроены корректно.

После установки программы следует провести ее проверку.

4.3 Установка системы мониторинга

Для мониторинга состояния работы Типового ПО «Витрина данных» используется связка Grafana + Prometheus.

Prometheus — система мониторинга, обладающая возможностями тонкой настройки метрик. Prometheus используется для отслеживания состояния работы компонентов системы на низком уровне.

Grafana — инструмент с открытым исходным кодом для визуализации данных из различных систем сбора статистики. Grafana используется для представления в графическом виде временных рядов и текстовых данных.

Для Grafana и Prometheus доступны установки как на Bare metal, так и под Docker.

27. Примечание:
28. Описание настроек системы мониторинга приведено в разделе Настройка сервиса мониторинга документа «Руководство администратора Типового ПО «Витрина данных»».

4.3.1 Установка Prometheus на Bare metal

4.3.1.1 Подготовка сервера

Перед установкой нужно настроить параметры сервера, необходимые для правильно работы системы.

1. Установить пакеты, нужные для работы:
   - **wget** - для загрузки файлов;
   - **tar** - для распаковки архивов.

В зависимости от системы, команды будут отличаться.

на CentOS

```
yum updateinfo
yum install wget tar
```

на Ubuntu

```
apt update
apt install wget tar
```

2. Проверить, что установлен нужный часовой пояс, описание настройки часового пояса приведено в разделе Выбор часового пояса.
3. На фаерволе, при его использовании, необходимо открыть порт TCP 9090 — http для сервера Prometheus.

   Используя IPtables открыть порт командой:

```
iptables -I INPUT -p tcp --match multiport --dports 9090 -j ACCEPT
```

Сохранить правила с помощью iptables-persistent:

на CentOS

```
service iptables save
```

на Ubuntu

```
apt install iptables-persistent
netfilter-persistent save
```

4. При использовании CentOS необходимо отключить SELinux, как указано в разделе Отключение SELinux (только для CentOS).

4.3.1.2 Установка Prometheus

Prometheus не устанавливается из репозитория и имеет не простой процесс установки.

Необходимо скачать исходник, создать пользователя, вручную скопировать нужные файлы, назначить права и создать юнит для автозапуска.

4.3.1.2.1 Загрузка

Для загрузки нужно перейти на официальную страницу загрузки, скопировать ссылку на пакет для Linux (желательно, использовать версию LTS) и загрузить пакет командой:

```
wget https://github.com/prometheus/prometheus/releases/download/v2.45.0/prometheus-
2.45.0.linux-amd64.tar.gz
```

29. Примечание:
30. Если система вернет ошибку, необходимо установить пакет wget.

4.3.1.2.2 Установка (копирование файлов)

После скачивания архив Prometheus, необходимо его распаковать и скопировать содержимое по разным каталогам.

1. Создать каталоги, в которые скопировать файлы для Prometheus командой:

```
mkdir /etc/prometheus /var/lib/prometheus
```

2. Распаковать архив командой:

```
tar -zxf prometheus-*.linux-amd64.tar.gz
```

3. Перейти в каталог с распакованными файлами:

```
cd prometheus-*.linux-amd64
```

4. Распределить файлы по каталогам:

```
cp prometheus promtool /usr/local/bin/
cp -r console_libraries consoles prometheus.yml /etc/prometheus
```

5. Выйти из каталога и удалить исходник:

```
cd .. && rm -rf prometheus-*.linux-amd64/ && rm -f prometheus-*.linux-amd64.tar.gz
```

4.3.1.2.3 Назначение прав

1. Создать пользователя, который будет запускать систему мониторинга:

```
useradd --no-create-home --shell /bin/false prometheus
```

31. Примечание:
32. Данная команда создает пользователя Prometheus без домашней директории и без возможности входа в консоль сервера.

2. Задать владельца для каталогов, которые были созданы ранее:

```
chown -R prometheus:prometheus /etc/prometheus /var/lib/prometheus
```

3. Задать владельца для скопированных файлов:

```
chown prometheus:prometheus /usr/local/bin/{prometheus,promtool}
```

4.3.1.2.4 Запуск и проверка

1. Запусить Prometheus от одноименного пользователя командой:

```
sudo -u prometheus /usr/local/bin/prometheus --config.file
/etc/prometheus/prometheus.yml --storage.tsdb.path /var/lib/prometheus/ --
web.console.templates=/etc/prometheus/consoles --
web.console.libraries=/etc/prometheus/console_libraries
```

Система выведет в консоль лог запуска с сообщением в конце:

```
level=info ts=2019-08-07T07:39:06.849Z caller=main.go:621 msg="Server is ready to
receive web requests."
```

2. В браузере набрать адрес: `http://<IP-адрес сервера>:9090` — в случае успешной установки загрузится консоль Prometheus.

4.3.1.2.5 Автозапуск

Для настройки автоматического старта Prometheus нужно создать новый юнит в systemd.

1. Открыть консоль сервера, прервать работу Prometheus с помощью комбинации Ctrl + C и создать файл `prometheus.service`:

```
vi /etc/systemd/system/prometheus.service

[Unit]
Description=Prometheus Service
Documentation=https://prometheus.io/docs/introduction/overview/
After=network.target

[Service]
User=prometheus
```

```
Group=prometheus
Type=simple
ExecStart=/usr/local/bin/prometheus ¥
--config.file /etc/prometheus/prometheus.yml ¥
--storage.tsdb.path /var/lib/prometheus/ ¥
--web.console.templates=/etc/prometheus/consoles ¥
--web.console.libraries=/etc/prometheus/console_libraries
ExecReload=/bin/kill -HUP $MAINPID
Restart=on-failure

[Install]
WantedBy=multi-user.target
```

2. Разрешить автозапуск:

```
systemctl enable prometheus
```

3. Запустить службу:

```
systemctl start prometheus
```

4. Проверить корректность запуска:

```
systemctl status prometheus
```

4.3.2 Установка Grafana на Bare metal

Установка на CentOS

1. Создать файл конфигурации репозитория:

```
vi /etc/yum.repos.d/grafana.repo

[grafana]
name=grafana
baseurl=https://packages.grafana.com/oss/rpm
repo_gpgcheck=1
enabled=1
gpgcheck=1
gpgkey=https://packages.grafana.com/gpg.key
sslverify=1
sslcacert=/etc/pki/tls/certs/ca-bundle.crt
```

2. Установить Grafana командой:

```
yum install grafana
```

3. На вопросы системы, задаваемые в процессе установки нужно будет выбирать ответ: **Y**.

Установка на Ubuntu

1. Добавить репозиторий командой:

```
add-apt-repository "deb https://packages.grafana.com/oss/deb stable main"
```

2. Установить ключ для проверки подлинности репозитория Grafana:

```
wget -q -O - https://packages.grafana.com/gpg.key | apt-key add -
```

3. Обновить список портов:

```
apt update
```

4. Установить Grafana:

```
apt install grafana
```

5. На вопросы системы, задаваемые в процессе установки нужно будет выбирать ответ: **Y**.

4.3.2.1 Настройка брэндмауэра

По умолчанию, Grafana работает на порту 3000.

Для возможности подключиться к серверу нужно открыть данный порт командой:

```
iptables -A INPUT -p tcp --dport 3000 -j ACCEPT
```

Сохранить правила с помощью iptables-persistent:

на CentOS

```
service iptables save
```

на Ubuntu

```
netfilter-persistent save
```

33. Примечание:
34. Если при вводе второй команды система выдаст ошибку, нужно установить пакет командой `apt install iptables-persistent`.

4.3.2.2 Запуск Grafana

Разрешить автозапуск командой:

```
systemctl enable grafana-server
```

Запуск сервиса:

```
systemctl start grafana-server
```

4.3.3 Установка Prometheus и Grafana в Docker

Для этого варианта установки необходимо установить Docker.

Описание установки Docker приведено в разделе Установка Docker.

Для того чтобы установить Prometheus и Grafana необходимо создать файл `docker-compose.yml` со следующим содержимым:

```
version: '3.3'

networks:
  monitoring:
    driver: bridge

volumes:
  prometheus_data: {}

services:
  grafana:
    image: grafana/grafana-enterprise
    container_name: grafana
    restart: unless-stopped
    ports:
      - 3000:3000
  prometheus:
    image: prom/prometheus:latest
    container_name: prometheus
    restart: unless-stopped
    volumes:
      - ./prometheus.yml:/etc/prometheus/prometheus.yml
```

```
    - prometheus_data:/prometheus
  command:
    - '--config.file=/etc/prometheus/prometheus.yml'
    - '--storage.tsdb.path=/prometheus'
    - '--web.console.libraries=/etc/prometheus/console_libraries'
    - '--web.console.templates=/etc/prometheus/consoles'
    - '--web.enable-lifecycle'
  ports:
    - 9090:9090
  networks:
    - monitoring
```

В примере выше указан путь к корневой папке с которой будет запущен docker-compose файл.

Для Prometheus необходимо указать папку в которой будет располагаться файл конфигурации `prometheus.yml`. Пример файла приведен в разделе Настройка конфигурационного файла Prometheus.

После этого нужно вернуться на уровень выше, где находится файл `docker-compose.yml` и выполнить установку командой:

```
docker-compose up -d
```

После установки Prometheus будет доступен по адресу http://ip:9090

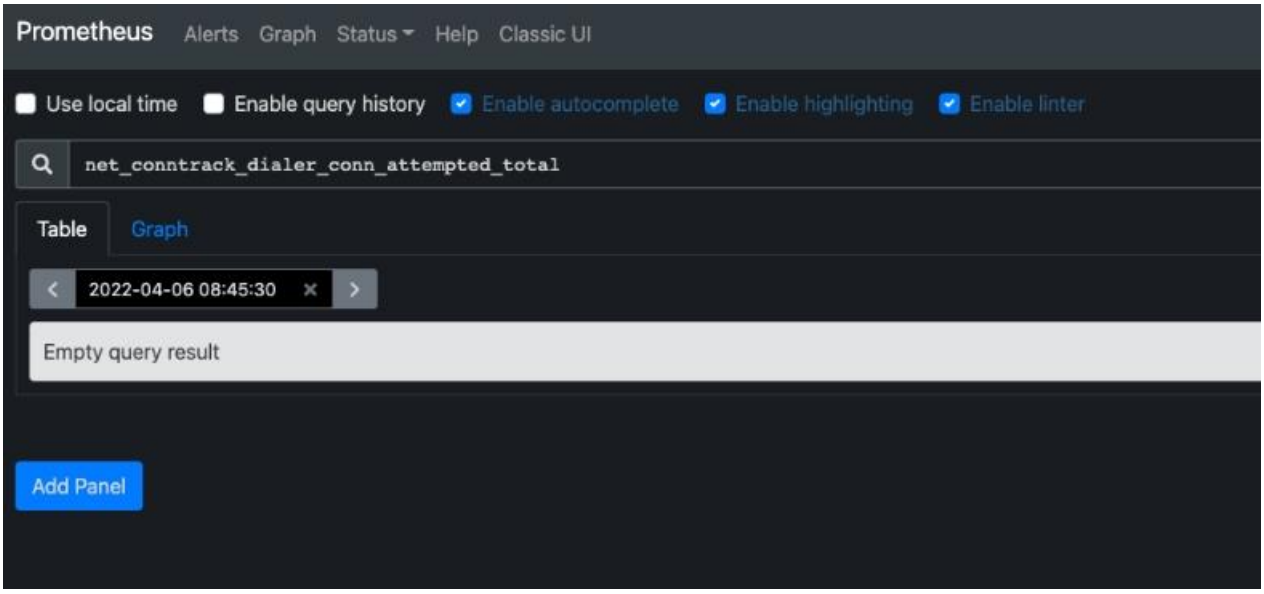

Рисунок - 4.12 Доступ Prometheus

Веб интерфейс Grafana будет доступен по адресу http://ip:3000.

Для авторизации необходимо ввести логин `viewer` и пароль `viewer`.

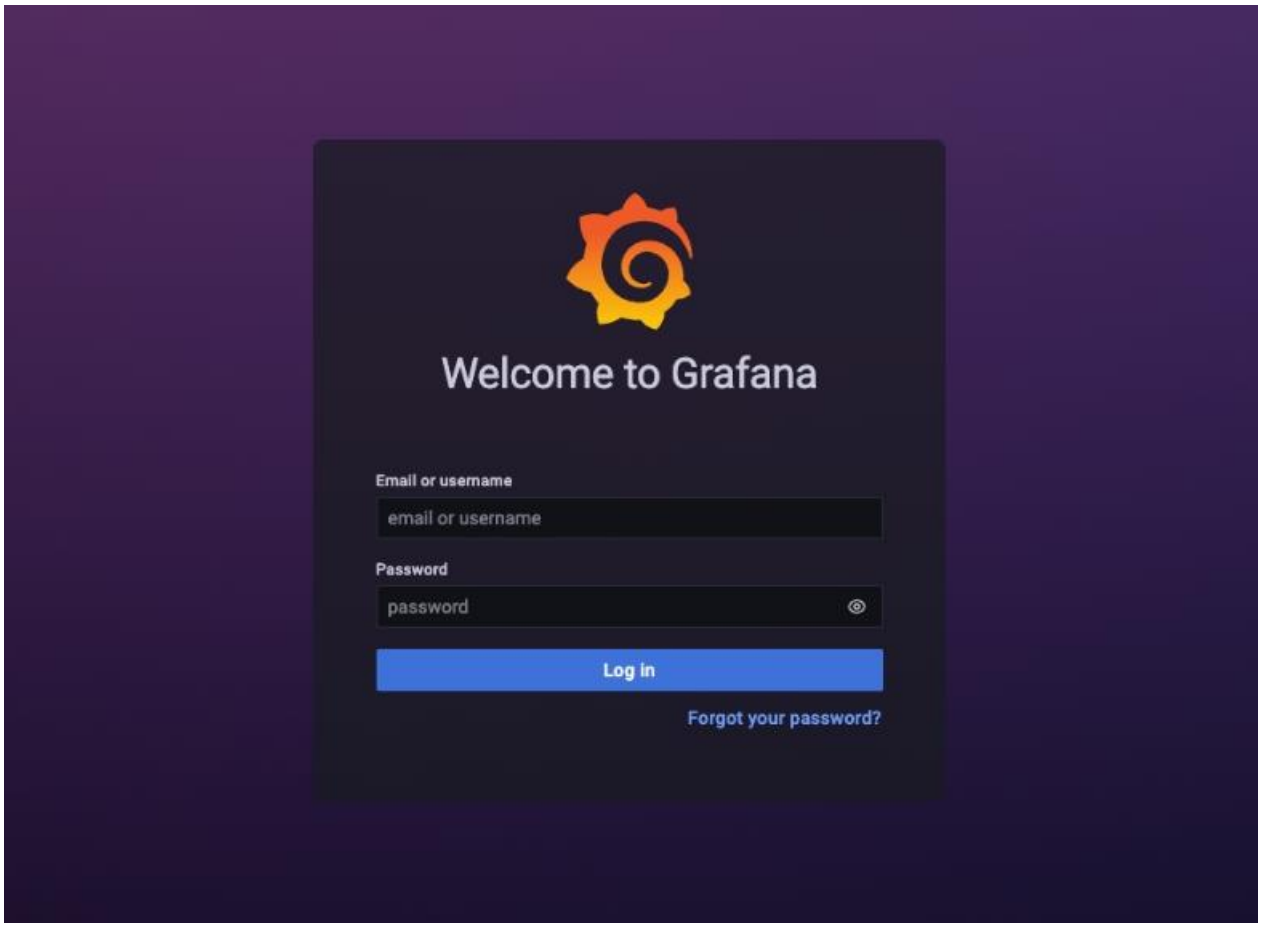

Рисунок - 4.13 Авторизация в Grafana

5 ПРОВЕРКА ПРОГРАММЫ

5.1 Проверка ПО конфигурации Стандарт

Процесс установки программы описан в разделе Установка ПО конфигурации Стандарт.

5.1.1 Проверка Arenadata Cluster Manager (ADCM)

Внимание:
Проверка Arenadata Cluster Manager (ADCM) осуществляется только при условии установки CentOS 7.9

Arenadata Cluster Manager (ADCM), при успешной установке, должен быть доступен по адресу `http://<ip_adress_of_server>:8000`.

Для авторизации используйте следующие данные:

- логин: `admin`;
- пароль: `admin`.

5.1.2 Проверка Arenadata Streaming (ADS)

Внимание:
Проверка Arenadata Streaming (ADS) осуществляется только при условии установки CentOS 7.9

5.1.2.1 Проверка сервиса Zookeeper

Проверка сервиса **Zookeeper** осуществляется через графический пользовательский интерфейс Arenadata Cluster Manager (ADCM). Чтобы выполнить проверку, выполните следующие действия:

1. Выберите кластер **ADS**, для этого откройте вкладку **Cluster-ADB**.
2. На вкладке **Services** для сервиса **Zookeeper**, в поле **Actions** нажмите кнопку **Run action** и выберите **Check**.

5.1.2.2 Проверка сервиса Apache Kafka

Проверка сервиса **Apache Kafka** осуществляется через графический пользовательский интерфейс Arenadata Cluster Manager (ADCM). Чтобы выполнить проверку, выполните следующие действия:

1. Выберите кластер ADS, для этого откройте вкладку **Cluster-ADB**.
2. На вкладке **Services** для сервиса Apache Kafka, в поле **Actions** нажмите кнопку **Run action** и выберите **Check**.

5.1.3 Проверка ProStore

Проверка ПО **ProStore** осуществляется путём отправки SQL-запросов к **ProStore** через клиентское JDBC-подключение и сопоставления ожидаемого эталонного и полученного результатов.

Проверка осуществляется согласно следующим этапам:

1. Создать Витрину в **ProStore** с помощью SQL-запроса:

```
CREATE DATABASE <имя несуществующей логической базы>, например,
CREATE DATABASE testdb;
```

2. Создать таблицу в **ProStore** со всеми типами колонок с помощью SQL-запроса:

```
CREATE TABLE <имя логической базы из п.1>.all_types (

- id int not null,
- double_col double,
- float_col float,
- char_col varchar(36),
- boolean_col boolean,
- int_col int not null,
- bigint_col bigint,
- date_col date,
- timestamp_col timestamp,
- primary key (id)
- )
- distributed by (id)
```

3. Проверить существование и структуру созданной таблицы в **ProStore** с помощью SQL-запросов:

```
select ¥* from <имя логической базы из п.1>.all_types
DATASOURCE_TYPE='ADG'

select ¥* from <имя логической базы из п.1>.all_types
DATASOURCE_TYPE='ADQM'

select ¥* from <имя логической базы из п.1>.all_types
DATASOURCE_TYPE='ADB'
```

4. Удалить таблицу со всеми типами колонок из **ProStore** с помощью SQL-запроса:

```
DROP TABLE <имя логической базы из п.1>.all_types
```

5. Удалить Витрину с помощью SQL-запроса:

```
DROP DATABASE <имя логической базы из п.1>.
```

> Внимание:
>
> Наличие сообщений об ошибках, а также отличие получаемых состояний **ProStore** на различных этапах проверки от ожидаемых состояний является индикатором неуспешного прохождения проверки.

5.1.4 Проверка СМЭВ QL Сервера

5.1.4.1 Проверки и валидации

Валидации запускаются либо на все объекты данного типа (указать `all`), либо только на указанные, в том числе через запятую.

Доступность источников проверяется командой:

```
./smevql test source <all | source-name>
```

Валидность моделей проверяется командой:

```
./smevql test model <all | model-name>
```

5.1.5 Проверка СМЭВ3-адаптера

5.1.5.1 Проверка модуля

Для проверки модуля **СМЭВ3-адаптер** необходимо выполнить запрос к сервису:

```
curl -s IP:Port/metrics | grep '^liveness '
```

где,

- `IP` - адрес сервера.
- `Port` - адрес сервера.
- `liveness` - параметр проверки работоспособности модуля.

Например:

```
curl -s http://172.16.10.67:9837/metrics | grep '^liveness '
```

Пример успешного ответа:

```
liveness 1.0
```

Ответ `1` означает корректную работу модуля.

5.1.6 Проверка ПОДД-адаптера - Модуля исполнения запросов

5.1.6.1 Проверка модуля

Для проверки модуля **ПОДД-адаптер - Модуль исполнения запросов** необходимо выполнить запрос к сервису:

```
curl -s IP:Port/metrics | grep '^liveness '
```

где,

- `IP` - адрес сервера.
- `Port` - адрес сервера.
- `liveness` - параметр проверки работоспособности модуля.

Например:

```
curl -s http://172.16.10.67:9837/metrics | grep '^liveness '
```

Пример успешного ответа

```
liveness 1.0
```

Ответ `1` означает корректную работу модуля.

5.1.7 Проверка ПОДД-адаптер – Модуля MPPR

5.1.7.1 Проверка модуля

Для проверки **ПОДД-адаптера - Модуль MPPR** необходимо выполнить запрос к сервису:

```
curl -s IP:Port/metrics | grep '^liveness '
```

где,

- `IP` - адрес сервера.
- `Port` - адрес сервера.
- `liveness` - параметр проверки работоспособности модуля.

Например:

```
curl -s http://172.16.10.67:9837/metrics | grep '^liveness '
```

Пример успешного ответа

```
liveness 1.0
```

Ответ `1` означает корректную работу модуля.

5.1.8 Проверка ПОДД-адаптера - Модуля MPPW

5.1.8.1 Проверка модуля

Для проверки модуля **ПОДД-адаптер - Модуль MPPW** необходимо выполнить запрос к сервису:

```
curl -s IP:Port/metrics | grep '^liveness '
```

где,

- `IP` - адрес сервера.
- `Port` - адрес сервера.
- `liveness` - параметр проверки работоспособности модуля.

Например:

```
curl -s http://172.16.10.67:9837/metrics | grep '^liveness '
```

Пример успешного ответа

```
liveness 1.0
```

Ответ `1` означает корректную работу модуля.

5.1.9 Проверка ПОДД-адаптера – Модуля импорта данных табличных параметров

5.1.9.1 Проверка модуля

Для проверки модуля **ПОДД-адаптер – Модуль импорта данных табличных параметров** необходимо выполнить запрос к сервису:

```
curl -s IP:Port/metrics | grep '^liveness '
```

где,

- `IP` - адрес сервера.
- `Port` - адрес сервера.
- `liveness` - параметр проверки работоспособности модуля.

Например:

```
curl -s http://172.16.10.67:9837/metrics | grep '^liveness '
```

Пример успешного ответа

```
liveness 1.0
```

Ответ `1` означает корректную работу модуля.

5.1.10 Проверка ПОДД-адаптера – Модуля группировки данных табличных параметров

5.1.10.1 Проверка модуля

Для проверки модуля **ПОДД-адаптер – Модуль группировки данных табличных параметров** необходимо выполнить запрос к сервису:

```
curl -s IP:Port/metrics | grep '^liveness '
```

где,

– IP - адрес сервера.
– Port - адрес сервера.
– liveness - параметр проверки работоспособности модуля.

Например:

```
curl -s http://172.16.10.67:9837/metrics | grep '^liveness '
```

Пример успешного ответа

```
liveness 1.0
```

Ответ 1 означает корректную работу модуля.

5.1.11 Проверка ПОДД-адаптера – Wrapper

5.1.11.1 Проверка модуля

Для проверки модуля **ПОДД-адаптер - Wrapper** необходимо выполнить запрос к сервису:

```
curl -s IP:Port/metrics | grep '^liveness '
```

где,

– IP - адрес сервера.
– Port - адрес сервера.
– liveness - параметр проверки работоспособности модуля.

Например:

```
curl -s http://172.16.10.67:9837/metrics | grep '^liveness '
```

Пример успешного ответа:

```
liveness 1.0
```

Ответ 1 означает корректную работу модуля.

5.1.12 Проверка модуля группировки чанков репликации

5.1.12.1 Проверка модуля

Для проверки **Модуля группировки чанков репликации** необходимо выполнить запрос к сервису:

```
curl -s IP:Port/metrics | grep '^liveness '
```

где,

– IP - адрес сервера.
– Port - адрес сервера.
– liveness - параметр проверки работоспособности модуля.

Например:

```
curl -s http://172.16.10.67:9837/metrics | grep '^liveness '
```

Пример успешного ответа:

```
liveness 1.0
```

Ответ 1 означает корректную работу модуля.

5.1.13 Проверка DATA-uploder – Модуля исполнения асинхронных заданий

5.1.13.1 Проверка модуля

Для проверки модуля **DATA-Uploader** необходимо выполнить запрос к сервису:

```
curl -s IP:Port/metrics | grep '^liveness '
```

где,

- `IP` - адрес сервера.
- `Port` - адрес сервера.
- `liveness` - параметр проверки работоспособности модуля.

Например:

```
curl -s http://172.16.10.67:9837/metrics | grep '^liveness '
```

Пример успешного ответа:

```
liveness 1.0
```

Ответ `1` означает корректную работу модуля.

5.1.14 Проверка REST-uploader – Модуля асинхронной загрузки данных из сторонних источников

5.1.14.1 Проверка модуля

Для проверки модуля **REST-uploader** необходимо выполнить запрос к сервису:

```
curl -s IP:Port/metrics | grep '^liveness '
```

где,

- `IP` - адрес сервера.
- `Port` - адрес сервера.
- `liveness` - параметр проверки работоспособности модуля.

Например:

```
curl -s http://172.16.10.67:9837/metrics | grep '^liveness '
```

Пример успешного ответа:

```
liveness 1.0
```

Ответ `1` означает корректную работу модуля.

5.1.15 Проверка ПОДД-адаптера – Модуля подписки

5.1.15.1 Проверка модуля

Для проверки модуля **ПОДД-адаптер - Модуль подписки** необходимо выполнить запрос к сервису:

```
curl -s IP:Port/metrics | grep '^liveness '
```

где,

- `IP` - адрес сервера.
- `Port` - адрес сервера.
- `liveness` - параметр проверки работоспособности модуля.

Например:

```
curl -s http://172.16.10.67:9837/metrics | grep '^liveness '
```

Пример успешного ответа

```
liveness 1.0
```

Ответ `1` означает корректную работу модуля.

5.1.16 Проверка BLOB-адаптера

5.1.16.1 Проверка модуля

Для проверки модуля **Blob-адаптер** необходимо выполнить запрос к сервису:

```
curl -s IP:Port/metrics | grep '^liveness '
```

где,

- `IP` - адрес сервера.
- `Port` - адрес сервера.
- `liveness` - параметр проверки работоспособности модуля.

Например

```
curl -s http://172.16.10.67:9837/metrics | grep '^liveness '
```

Пример успешного ответа:

```
liveness 1.0
```

Ответ `1` означает корректную работу модуля.

5.1.17 Проверка Сервиса формирования документов

5.1.17.1 Проверка модуля

Для проверки модуля **Сервис Формирования документов** необходимо выполнить запрос к сервису:

```
curl -s IP:Port/metrics | grep '^liveness '
```

где,

- `IP` - адрес сервера.
- `Port` - адрес сервера.
- `liveness` - параметр проверки работоспособности модуля.

Например:

```
curl -s http://172.16.10.67:9837/metrics | grep '^liveness '
```

Пример успешного ответа:

```
liveness 1.0
```

Ответ `1` означает корректную работу модуля.

5.1.18 Проверка ETL

5.1.18.1 Проверка статусной информации по загрузке / удалению данных (Endpoint – status)

В данном разделе производится проверка статусной информации из сервисных таблиц

по `requestId`.

Пример запроса:

```
Curl -X GET "http://<ip-studio>:8088/api/v1/secure/<organization_ogrn>/<datamart_mnemonic>/<installation_name>/<installation_id>/status/<requestId>" -H "Authorization: Bearer <access_token>"
```

где:

– `requestId` — UUID идентификатор порции изменений (дельты).

Пример ответа на такой запрос представлен ниже.

```
{
    "requestId": "13f2475e-f3dc-4c9e-b2f6-3a98320261f1",
    "inDeltaFlag": false,
    "dataSets": [
        "stock"
    ],
    "status": "ERROR",
    "statusMessage": "Произошла ошибка",
    "errors": [
        {
            "dataSet": "stock",
            "errorType": "PARCING",
            "message": "Неверно указан тип поля count_pieces: LONG. Ожидается: INTEGER"
        },
        {
            "dataSet": "stock",
            "errorType": "PARCING",
            "message": "Неверно указан тип поля product_id: LONG. Ожидается: INTEGER"
        }
    ]
}
```

где:

– `requestId` — UUID идентификатор порции изменений (дельты);
– `inDeltaFlag = false` — загрузка несогласованных данных производилась через endpoint /data;
– `dataSets` — массив имен набора данных (имен таблиц где была допущена ошибка);
– `status` — статус код результата запроса (NOT_FOUND, PROCESSING, ERROR, SUCCESS);
– `statusMessage` — описание статусного сообщения;
– `errors` — массив, ошибки загрузки или парсинга входящих данных;
– `dataSet` — название таблицы где допущена ошибка;
– `errorType` — тип ошибки;
– `message` — описание ошибки.

5.1.18.2 Проверка Apache Airflow

Проверка сервиса **Apache Airflow** осуществляется через графический пользовательский интерфейс, в случае успешной установки, он должен быть доступен по адресу `http://localhost:8080`.

Для авторизации используйте следующие данные:

– логин: `airflow`;
– пароль: `airflow`.

Также, можно проверить удалённое подключение с помощью http-запроса, для этого

выполните следующую команду

```
ENDPOINT_URL="http://localhost:8080/"
curl -X GET ¥¥
--user "airflow:airflow" ¥¥
"${ENDPOINT_URL}/api/v1/pools"
```

5.1.18.3 Проверка Apache Spark

Проверка сервиса **Apache Spark** осуществляется через графический пользовательский интерфейс, в случае успешной установки, он должен быть доступен по адресу:

Веб-сервер **Spark Master** доступен по адресу

```
http://<ваш ip-адрес>:8080 например http://localhost:8080/
```

Веб-сервер **Spark Worker 1** доступен по адресу

```
http://<ваш ip-адрес>:8081 например http://localhost:8081/
```

5.1.18.4 Проверка Apache Hadoop

Проверить корректность работы **Apache Hadoop** можно командой:

```
make wordcount
```

или последовательно выполнить следующие команды:

```
docker build -t hadoop-wordcount ./submit

docker run --network docker-hadoop_default --env-file hadoop.env
bde2020/hadoop-base:master hdfs dfs -mkdir -p /input/

docker run --network docker-hadoop_default --env-file hadoop.env
bde2020/hadoop-base:master hdfs dfs -copyFromLocal -f
/opt/hadoop-3.2.1/README.txt /input/

docker run --network docker-hadoop_default --env-file hadoop.env
hadoop-wordcount

docker run --network docker-hadoop_default --env-file hadoop.env
bde2020/hadoop-base:master hdfs dfs -cat /output/¥*

docker run --network docker-hadoop_default --env-file hadoop.env
bde2020/hadoop-base:master hdfs dfs -rm -r /output

docker run --network docker-hadoop_default --env-file hadoop.env
bde2020/hadoop-base:master hdfs dfs -rm -r /input
```

После запуска **Apache Hadoop** можно зайти в следующие WEB–интерфейсы:

```
Namenode:
http://<dockerhadoop_IP_address>:9870/dfshealth.html#tab-overview

History server: http://<dockerhadoop_IP_address>:8188/applicationhistory

Datanode: http://<dockerhadoop_IP_address>:9864/

Nodemanager: http://<dockerhadoop_IP_address>:8042/node

Resource manager: http://<dockerhadoop_IP_address>:8088/
```

5.1.18.5 Проверка Tarantool(Vinyl)

Проверка СУБД **Tarantool** осуществляется согласно документации на СУБД Tarantool:

https://www.tarantool.io/en/doc/latest/book/monitoring/ .

5.1.19 Проверка Backup manager

5.1.19.1 Проверка модуля

Для проверки **Backup manager** необходимо выполнить запрос к сервису:

```
curl -s IP:Port/metrics | grep '^liveness '
```

где,

- IP - адрес сервера.
- Port - адрес сервера.
- liveness - параметр проверки работоспособности модуля.

Например

```
curl -s http://172.16.10.67:9837/metrics | grep '^liveness '
```

Пример успешного ответа

```
liveness 1.0
```

Ответ 1 означает корректную работу модуля.

5.1.20 Проверка REST-адаптер

Проверить удалённое подключение с помощью HTTP-запроса:

```
curl localhost:8080
```

В случае успешной установки ответ будет следующим:

```
{"timestamp":"2021-03-15T10:22:57.325+0000","status":404,"error":"Not
Found","message":"","path":"/"}%
```

5.1.21 Проверка Counter-provider - Сервиса генерации уникального номера

5.1.21.1 Проверка модуля

Для проверки **Сервиса генерации уникального номера** необходимо выполнить запрос к сервису:

```
curl -s IP:Port/metrics | grep '^liveness '
```

где,

- IP - адрес сервера.
- Port - адрес сервера.
- liveness - параметр проверки работоспособности модуля.

Например

```
curl -s http://172.16.10.67:9837/metrics | grep '^liveness '
```

Пример успешного ответа

```
liveness 1.0
```

Ответ 1 означает корректную работу модуля.

5.2 Проверка ПО конфигурации Лайт

Процесс установки программы описан в разделе Установка ПО конфигурации лайт.

Для проверки установки программы следует выполнить следующие действия:

1. Открыть в браузере web-интерфейс **Portainer** для управления docker-контейнерами по адресу `IP:9000`, где
   - `IP` - адрес сервера.
   - `9000` - порт сервера.
2. Введите логин и пароль администратора Portainer. По умолчанию - `admin`/ `LongPassword` (см. Рисунок - 5.1).

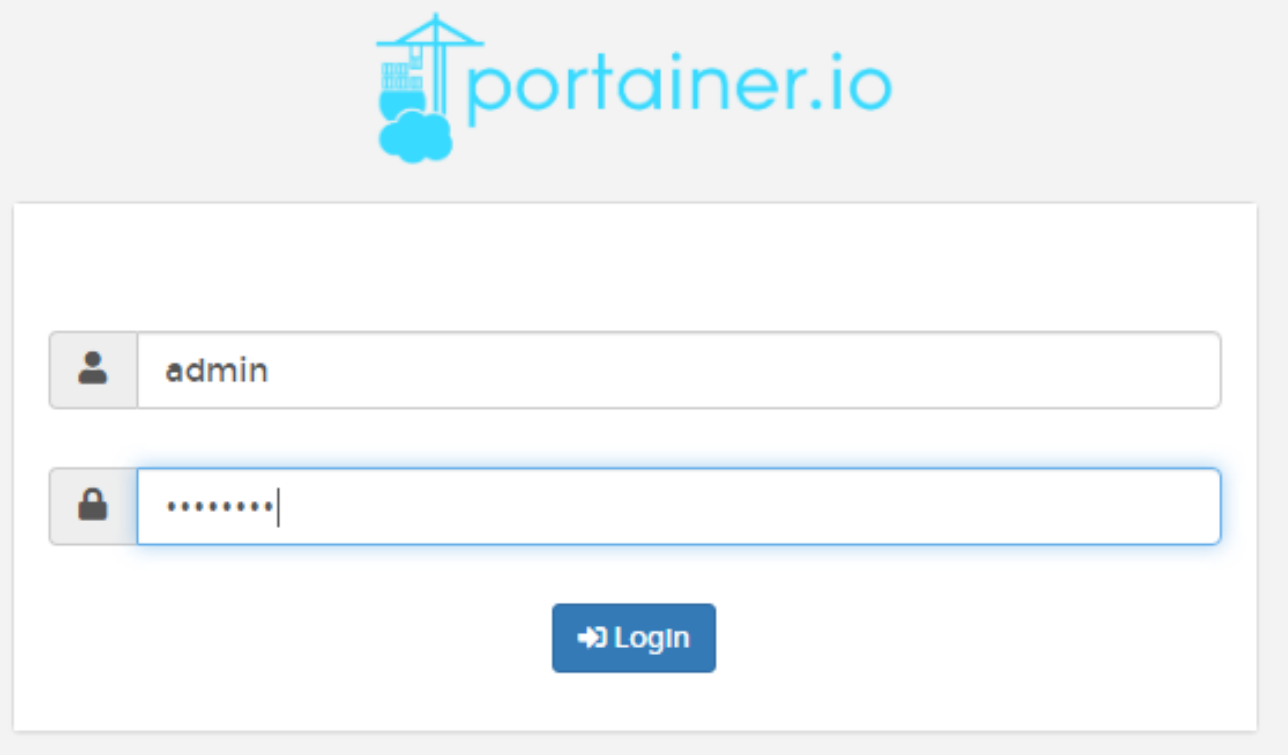

Рисунок - 5.1 Авторизация в Portainer

3. Чтобы определить и автоматически настроить локальную среду нажмите значок **Get Started** (см. Рисунок - 5.2).

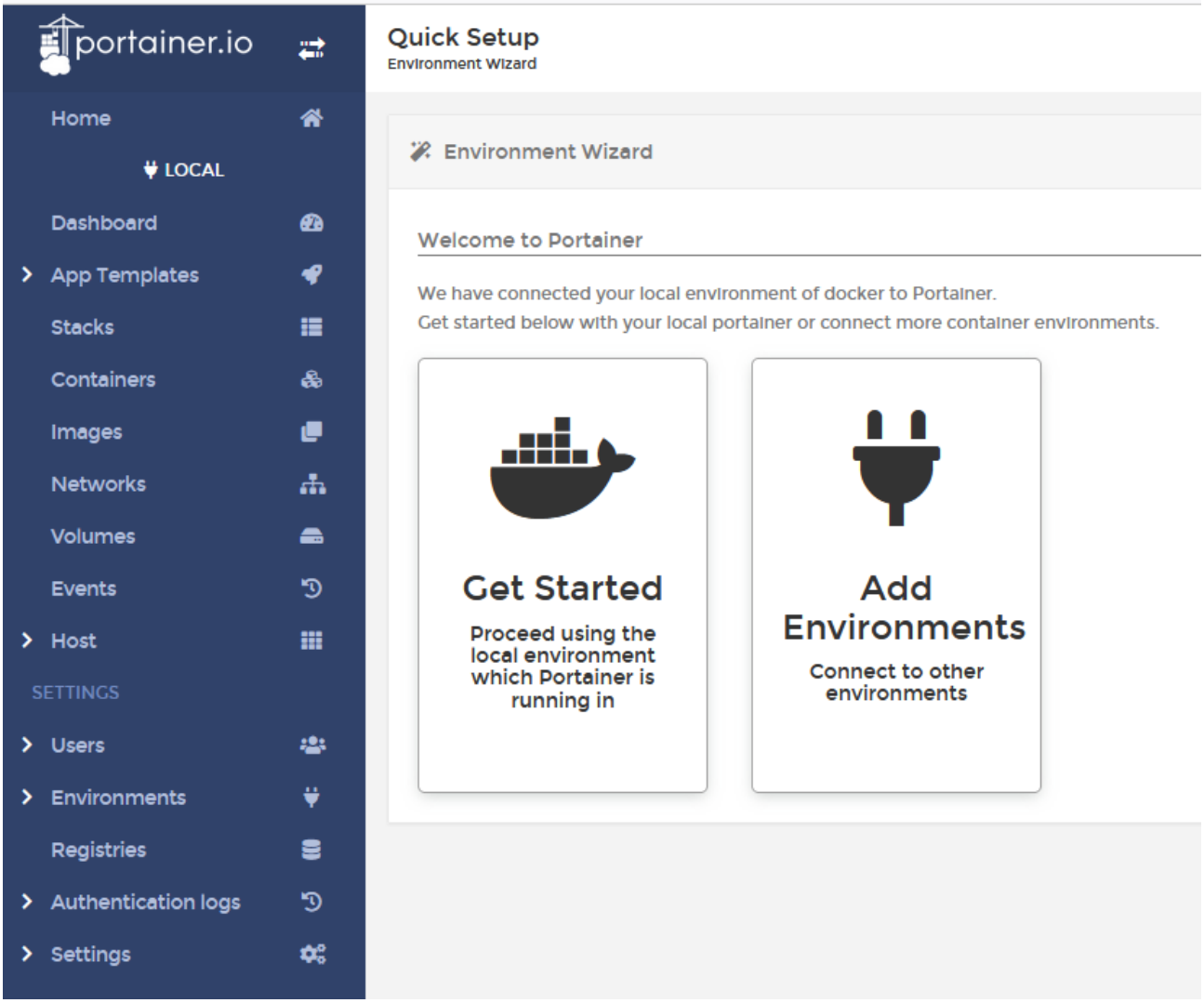

Рисунок - 5.2 Окно «Quick Setup»

4. На главной странице нажмите ссылку **local** (см. Рисунок - 5.3).

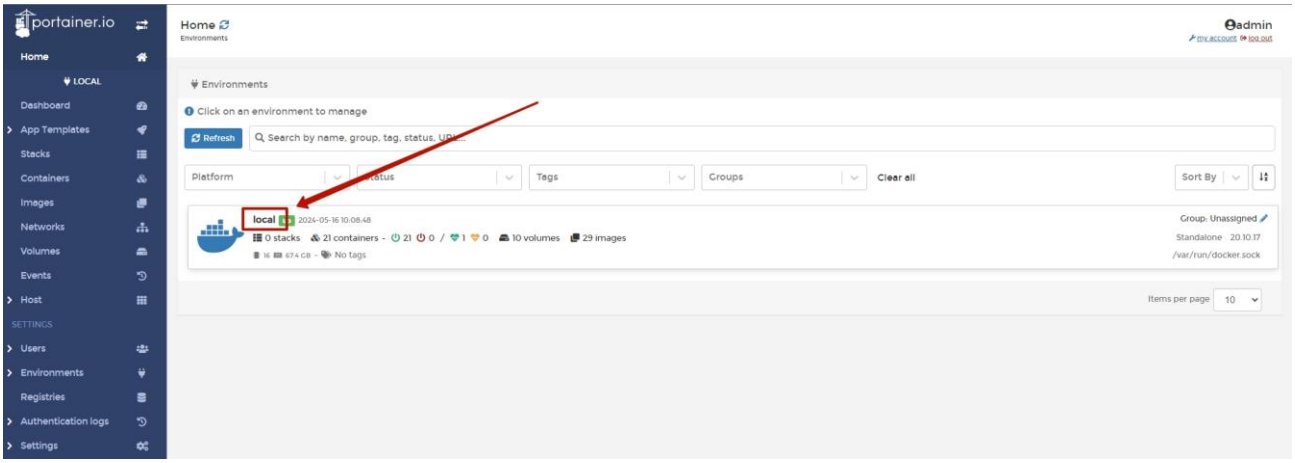

Рисунок - 5.3 Окно «Home»

5. Нажмите значок **Containers** (см. Рисунок - 5.4).

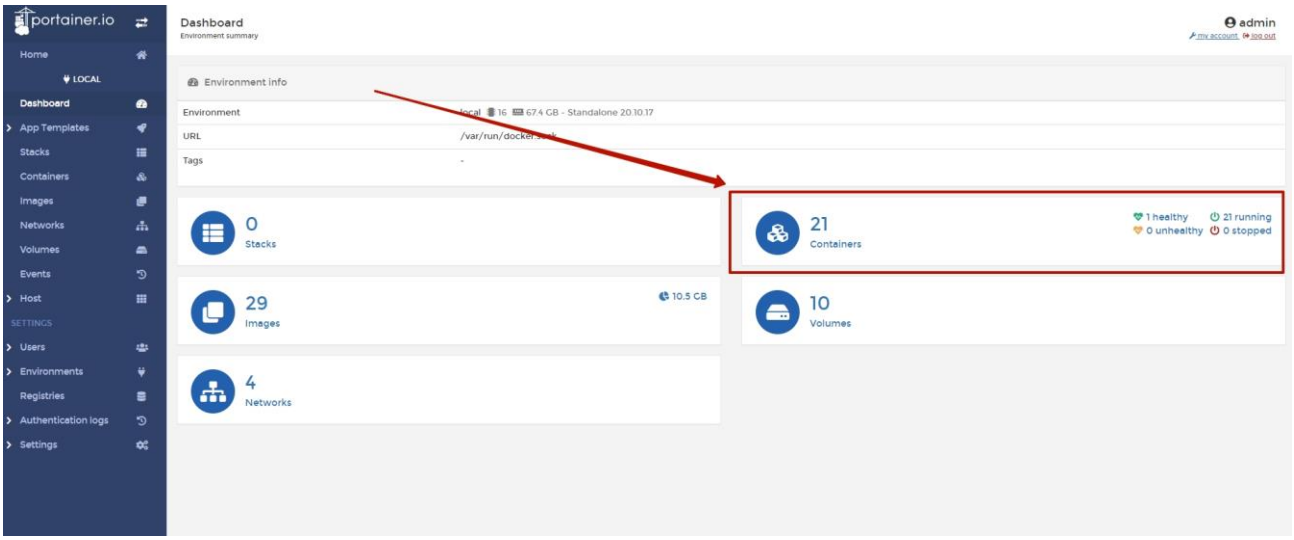

Рисунок - 5.4 Окно «Containers»

6. В разделе **Container list** сверить наличие компонентов со списком компонентов дистрибутива программы, приведенных в Состав компонентов в дистрибутиве документа «Техническое описание программы ПО «Витрина данных НСУД»».
7. Подключиться к программе по SSH (например, через Putty). Выполнить запрос, согласно приведенному примеру ниже.

Пример запроса

```
curl -X POST -H "Content-Type: application/json" -d '{"requestId": "797de19a-54e2-4c9c-af6e-a9ee312230b5","datamartMnemonic": "base01","sql": "CHECK_VERSIONS()"}' http://0.0.0.0:9090/query/execute
```

Пример успешного ответа (см. image9_link).

Рисунок - 5.5 Проверка подключения к базе данных Prostore

8. Подключиться к Grafana.

Для этого следует перейти по адресу: http://<имя сервера>:3000 и выполнить авторизацию (указаны значения по умолчанию):

– логин: admin;
– пароль: admin.

Проверить, что показатели Healthcheck Leaviness и Readiness (дашборд Lite) работают (индикатор зеленого цвета).

9. Просмотреть лог-файл установки (ansible/ansible.log). Лог-файл не должен содержать записей с ошибками установки.

6 ОБНОВЛЕНИЕ ПРОГРАММЫ

6.1 Обновление ПО конфигурации Стандарт

Для обновления версии ПО нужно обновить файлы компонентов, полученные из дистрибутива новой версии программы.

Все настройки программы (адреса серверов, порты и т.д.) останутся без изменений.

6.2 Обновление ПО конфигурации Лайт

Внимание:
Перед началом обновления выполните резервное копирование!

6.2.1 Обновление с версии 1.0.0 до версии 1.0.1

Внимание:
Данное обновление предусмотрено только для перехода с версии программы 1.0.0 до версии 1.0.1.

Чтобы обновить программу необходимо выполнить следующие действия (подробная инструкция по обновлению ниже):

1. Выполнить резервное копирование.
2. Скопировать архив с обновлением программы на сервер.
3. Распаковать архив с обновлением.
4. Запустить процесс обновления.
5. Проверить, что обновление прошло успешно.

6.2.1.1 Резервное копирование

Внимание:
Действия этого раздела необходимо выполнять под созданной учетной записью **datamart** (см. раздел Создание пользователя).

Для того чтобы выполнить резервное копирование создайте архив текущей версии с указанием актуальной даты копирования.

Например:

```
tar -czvf dtm-lite-2022-01-17.tgz ansible/ images/ jdbc/ rpms/
```

6.2.1.2 Копирование архива с обновлением программы на сервер

Для загрузки на сервер архива с файлами обновления программы используйте SFTP-клиент (например, WinSCP или Filezilla).

Для авторизации используйте логин и пароль учетной записи администратора **datamart** созданной при установке ОС (см. раздел Создание пользователя).

1. Подключитесь по SSH к серверу (см. раздел Подключение к серверу через SSH-клиент PuTTY), используя логин и пароль учетной записи администратора.
2. Загрузите файл с архивом в домашнюю директорию пользователя **datamart** командой:

```
mv ~/dtm-lite-1.0.1.tgz /home/datamart/
```

где,

– `dtm-lite-1.0.1.tgz` - название архива программы.

– `datamart` - имя пользователя.

6.2.1.3 Распаковка архива с обновлением

> **Внимание:**
>
> Перед тем как выполнить разархивирование рекомендуется просмотреть/скопировать значения переменных (IP-адрес сервера, префикс и т.д.), которые использовались в предыдущей версии программы в файле `ansible/group_vars/all/main.yml`. После распаковки архива с обновлением программы данные из этого файла будут удалены.

Чтобы распаковать архив, выполните команду:

```
tar -xzvf dtm-lite-1.0.1.tgz
```

6.2.1.4 Процесс обновления программы

Чтобы запустить процесс обновления программы с помощью Ansible, необходимо выполнить следующие действия:

1. Переименовать файл `custom.example.yml` расположенный в папке `ansible/group_vars/` в `custom.yml`. Для этого выполните команду:

```
cp -n ansible/group_vars/custom.example.yml ansible/group_vars/custom.yml
```

2. В файле `custom.yml` указать корректные значения для следующих переменных:
   – `server_ip` - адреса сервера. Укажите IP-адрес сервера, на который будет установлено обновление программы. Например:

```
server_ip: "172.16.10.59"
```

   – `server_user_name` - имя пользователя операционной системы. Укажите имя пользователя операционной системы сервера, под которым устанавливается обновление (см. раздел Создание пользователя), например:

```
server_user_name: datamart
```

   – `podd_kafka_topic_prefix` - префикс перед именем топиков для ПОДД-агента (указывается опционально). Например:

```
podd_kafka_topic_prefix: "user_prefix."
```

> **Внимание:**
>
> В качестве префикса рекомендуется использовать мнемонику витрины. После определения параметра префикса следует обязательно ставить символ `.` (точка)! Пример: `podd_kafka_topic_prefix: "prod_vitrina98."`

Чтобы запустить процесс обновления программы выполните команду:

```
docker-ansible-cmd ansible-playbook -i hosts install.yml
```

Начнется процесс обновления программы (см. рис. ниже):

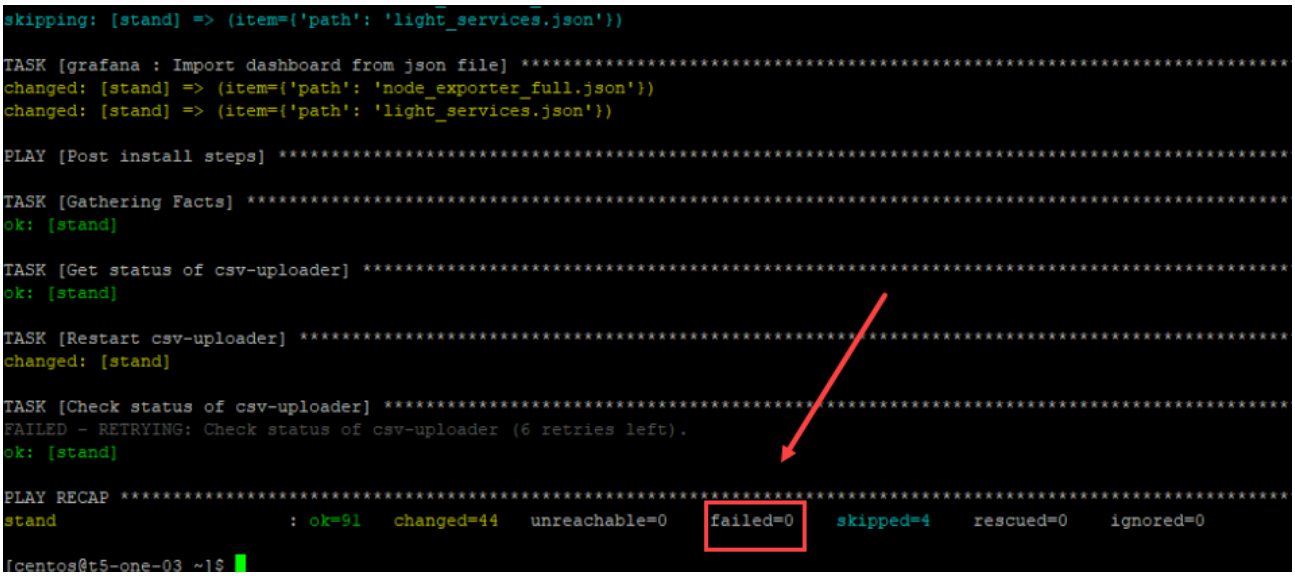

Рисунок - 6.1 Обновление программы

Установка обновления завершена.

После завершения обновления необходимо убедиться, что обновление программы прошло успешно.

6.2.1.5 Проверка обновления программы до версии 1.0.1

Для проверки обновления программы необходимо выполнить следующие действия:

1. Убедиться, что после установки обновления нет ошибок в работе программы.
2. Убедиться, что параметр `failed` (см. рис. выше) после установки обновления имеет значение - `0`. Это значит, что все необходимые компоненты программы были обновлены, а необходимые взаимосвязи между ними настроены корректно.

3. Открыть *Portainer* и проверить, что версии компонентов соответствуют указанным ниже:

```
csv-uploader:1.0.12
podd-adapter:5.0.7
query-execution:5.2.2
```

6.2.2 Обновление с версии 1.0.1 до версии выше

Чтобы обновить программу необходимо выполнить следующие действия:

1. Выполнить резервное копирование.
2. Скопировать архив с обновлением программы на сервер.
3. Распаковать архив с обновлением.
4. Запустить процесс обновления.
5. Проверить, что обновление прошло успешно.

6.2.2.1 Резервное копирование

Внимание:
Действия этого раздела необходимо выполнять под созданной учетной записью **datamart** (см. раздел Создание пользователя).

Для того чтобы выполнить резервное копирование создайте архив текущей версии с указанием актуальной даты копирования.

Например:

```
tar -czvf dtm-lite-2022-01-17.tgz ansible/ images/ jdbc/ rpms/
```

6.2.2.2 Копирование архива с обновлением программы на сервер

Для загрузки на сервер архива с файлами обновления программы используйте SFTP-клиент (например, WinSCP или Filezilla).

Для авторизации используйте логин и пароль учетной записи администратора **datamart** созданной при установке ОС (см. раздел Создание пользователя).

1. Подключитесь по SSH к серверу (см. раздел Подключение к серверу через SSH-клиент PuTTY), используя логин и пароль учетной записи администратора.
2. Загрузите файл с архивом в домашнюю директорию пользователя **datamart** командой:

   mv ~/dtm-lite-<номер версии>.tgz /home/datamart/

   где,

   – `dtm-lite-<номер версии>.tgz` - название архива программы.
   – `datamart` - имя пользователя.

6.2.2.3 Распаковка архива с обновлением

Чтобы распаковать архив, выполните команду:

```
tar -xzvf dtm-lite-<номер версии>.tgz
```

6.2.2.4 Процесс обновления программы

Чтобы запустить процесс обновления программы с помощью Ansible, необходимо выполните команду:

```
docker-ansible-cmd ansible-playbook -i hosts install.yml
```

Начнется процесс обновления программы (см. рис. ниже):

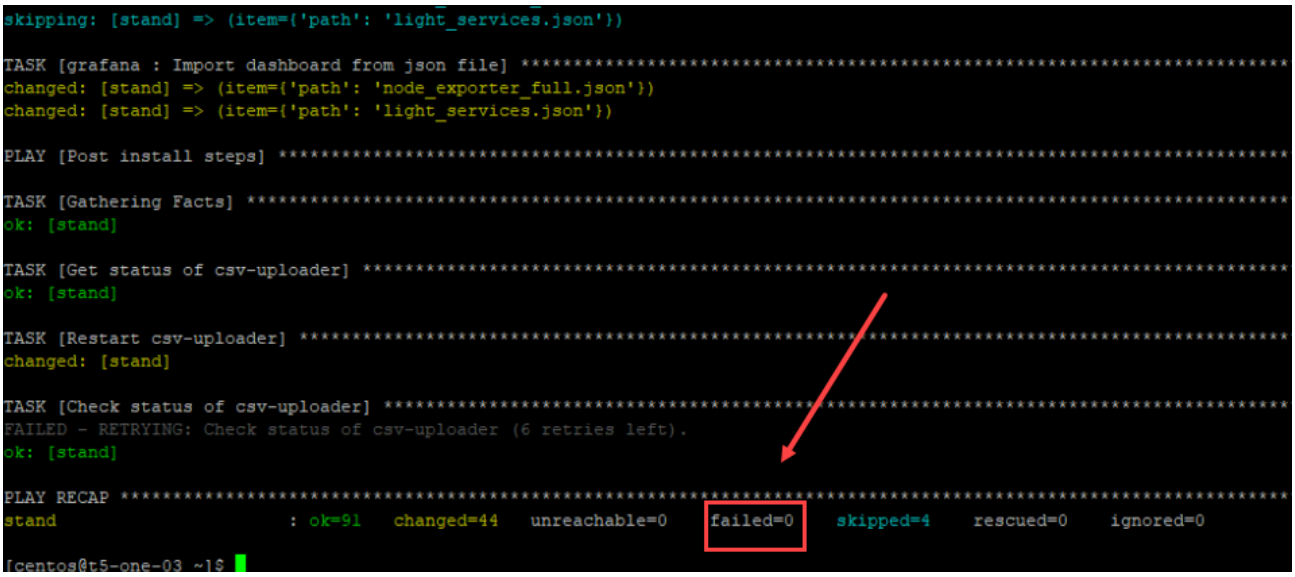

Рисунок - 6.2 Обновление программы

Установка обновления завершена.

После завершения обновления необходимо убедиться, что обновление программы прошло успешно.

6.2.2.5 Проверка обновления программы

Для проверки обновления программы необходимо выполнить следующие действия:

1. Убедиться, что после установки обновления нет ошибок в работе программы.
2. Убедиться, что параметр `failed` (см. рис. выше) после установки обновления имеет значение - `0`. Это значит, что все необходимые компоненты программы были обновлены, а необходимые взаимосвязи между ними настроены корректно.
3. Открыть *Portainer* и проверить, что версии компонентов соответствуют указанным компонентам для конфигурации Лайт.

ПРИЛОЖЕНИЕ 1. НАСТРОЙКА FIREWALL (IPTABLES)

Утилита **iptables** - это межсетевой экран для операционных систем Linux.

Настройка iptables производится в командной строке, с помощью правил iptables можно разрешать или блокировать прохождение трафика.

Для выполнения настройки межсетевого экрана необходимо создать конфигурационный файл `iptables.conf` в папке `etc/`:

```
/etc/iptables.conf
```

Далее, необходимо скопировать в файл следующие настройки:

```
*filter
:INPUT ACCEPT [0:0]
:FORWARD DROP [0:0]
:OUTPUT ACCEPT [0:0]
:FILTERS - [0:0]
:DOCKER-USER - [0:0]

-F INPUT
-F DOCKER-USER
-F FILTERS

-A INPUT -i lo -j ACCEPT
-A INPUT -p icmp --icmp-type any -j ACCEPT
-A INPUT -j FILTERS

-A DOCKER-USER -o docker0 -j FILTERS

-A FILTERS -m state --state ESTABLISHED,RELATED -j ACCEPT
-A FILTERS -m state --state NEW -m tcp -p tcp --dport 22 -j ACCEPT
-A FILTERS -m state --state NEW -m tcp -p tcp --dport 3000 -j ACCEPT
-A FILTERS -m state --state NEW -m tcp -p tcp --dport 8080 -j ACCEPT
-A FILTERS -m state --state NEW -m tcp -p tcp --dport 9000 -j ACCEPT
-A FILTERS -j REJECT --reject-with icmp-host-prohibited

COMMIT
```

1. Выполнить команду:

```
iptables-restore -n /etc/iptables.conf
```

2. Создать файл:

```
/etc/systemd/system/iptables.service
```

3. Сохранить в файл:

```
[Unit]
Description=Restore iptables firewall rules
Before=network-pre.target

[Service]
Type=oneshot
ExecStart=/sbin/iptables-restore -n /etc/iptables.conf

[Install]
WantedBy=multi-user.target
```

4. Включить iptables, для этого выполнить команду:

```
sudo systemctl enable --now iptables
```

или выполнить следующие две команды:

```
sudo systemctl enable iptables
sudo systemctl start iptables
```

После обновления правил в файле `/etc/iptables.conf`, выполнить следующую команду:

```
sudo systemctl restart iptables
```

ПРИЛОЖЕНИЕ 2. ПРОСМОТР ВЫПОЛНЕНИЯ ЗАГРУЗКИ ДАННЫХ В ПРОГРАММУ

В данном приложении описан порядок проверки загрузки данных в программу, а также необходимые настройки для подключения к базе данных программы.

1 Настройка подключения к базе данных

Для подключения к базе данных программы необходимо выполнить следующие действия:

- получить у системного программиста учетные записи для доступа к базе данных (название БД, логин и пароль пользователя БД);
- установить и настроить программу DBeaver (менеджер баз данных);
- установить и настроить JDBC-драйвер для работы с базой данных программы;
- выполнить проверку подключения к базе данных.

1.1 Установка программы DBeaver

DBeaver — это бесплатное программное обеспечение с открытым исходным кодом для управления базами данных (БД). Для взаимодействия с реляционными БД в программе используется программный интерфейс JDBC (через JDBC-драйвер).

С помощью программы DBeaver оператор может выполнить следующие действия:

- настроить доступ и подключится к БД;
- проверить работоспособность БД.

Для установки DBeaver скачайте дистрибутив программы с официального сайта https://dbeaver.io/download/.

Внимание:
Необходимо выбрать дистрибутив программы для операционной системы, которая установлена на вашем компьютере.

1.1.1 Установка DBeaver для ОС Linux

(В данном разделе описан процесс установки DBeaver, в операционную систему Linux Ubuntu, версия 20.04).

1. В строке поиска *Ubuntu Software* введите название программы *DBeaver* (см. search_DBeaver).

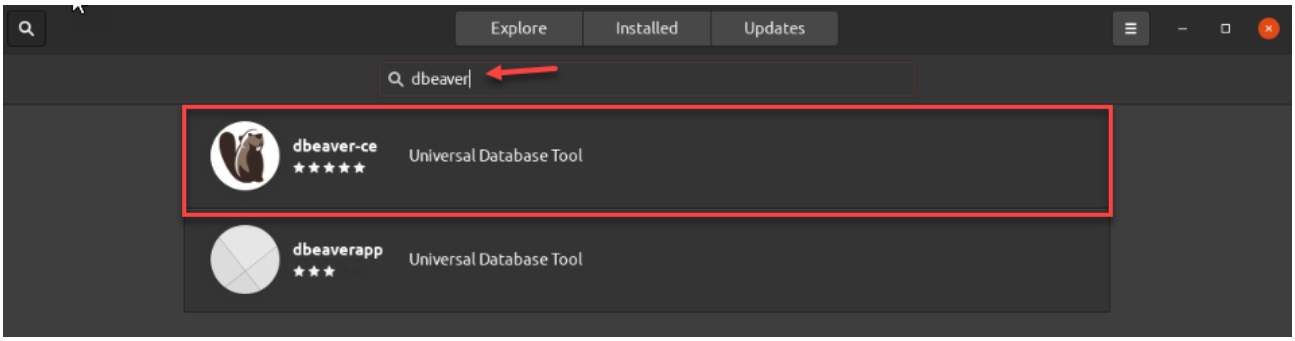

Рисунок - 8.1 Поиск DBeaver

2. В предложенном списке (см.рис. выше) выберите программу DBeaver.

3. Запустите установку программы, для этого нажмите кнопку *Install* (см. `install_DBeaver`).

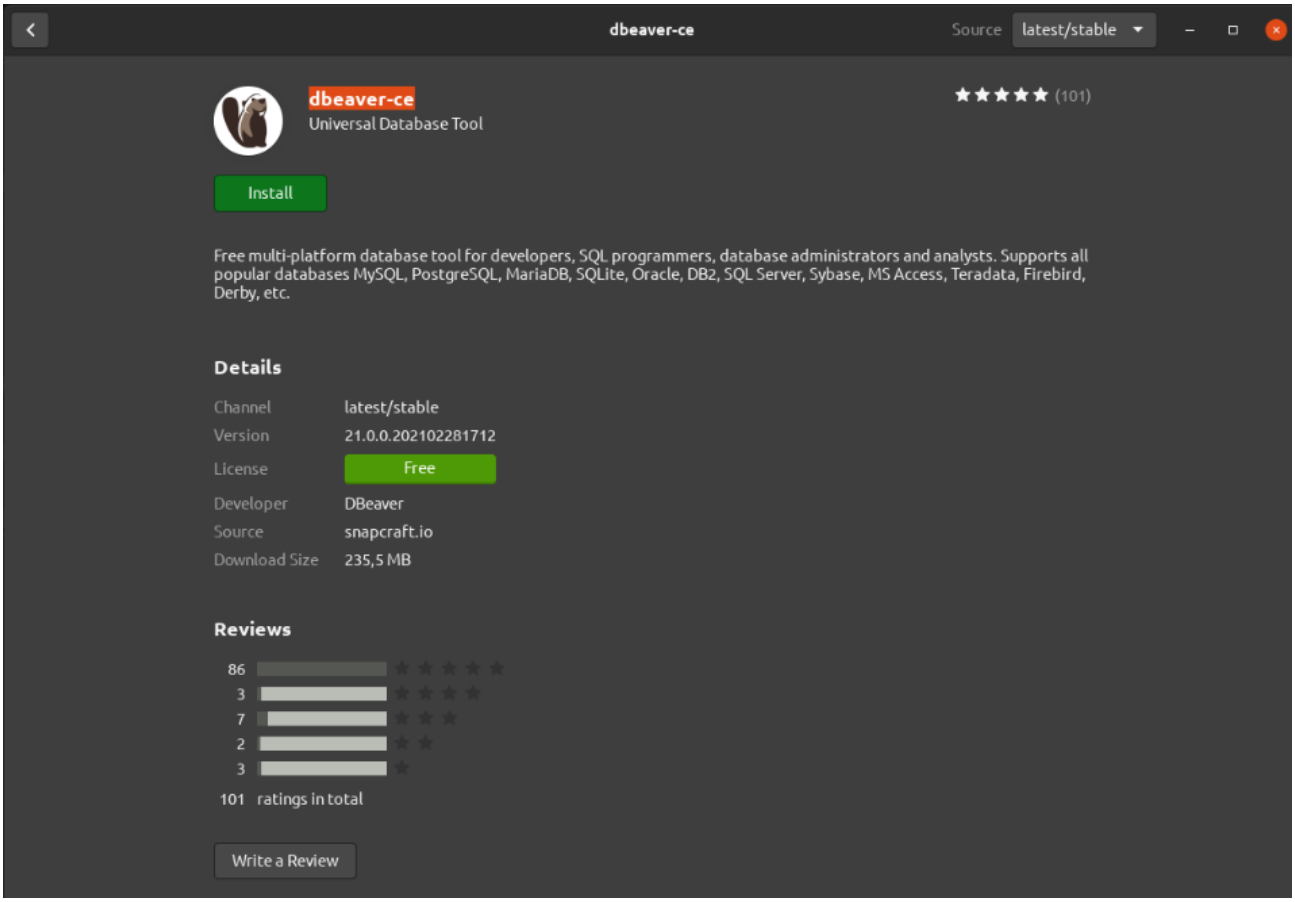

Рисунок - 8.2 Установка DBeaver

4. Дождитесь окончания процесса копирования файлов (см. `file_copy`).

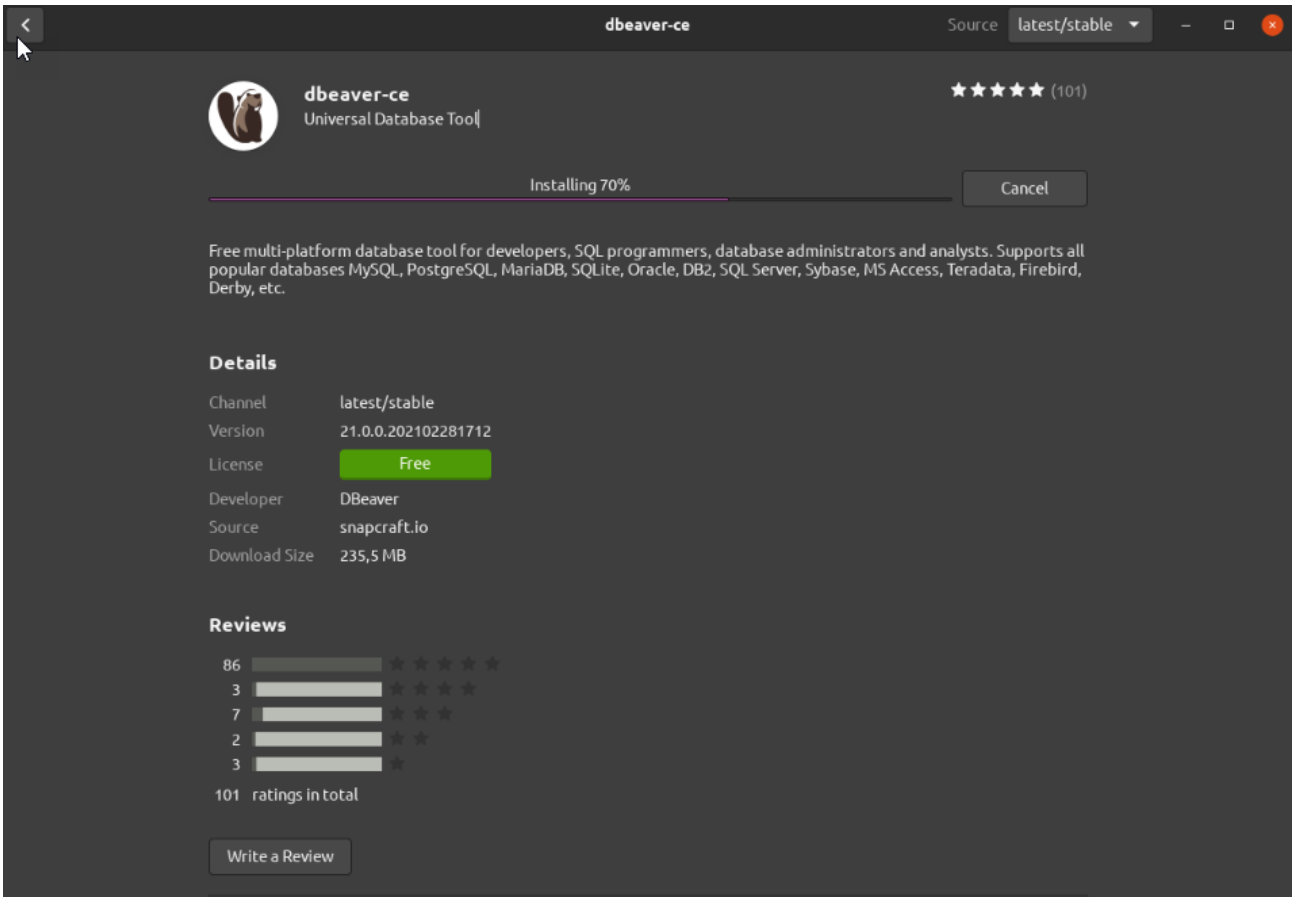

Рисунок - 8.3 Процесс копирования файлов программы

5. После завершения копирования файлов на экране монитора отобразится окно с сообщением об успешной установке программы (см. `install_ready`).

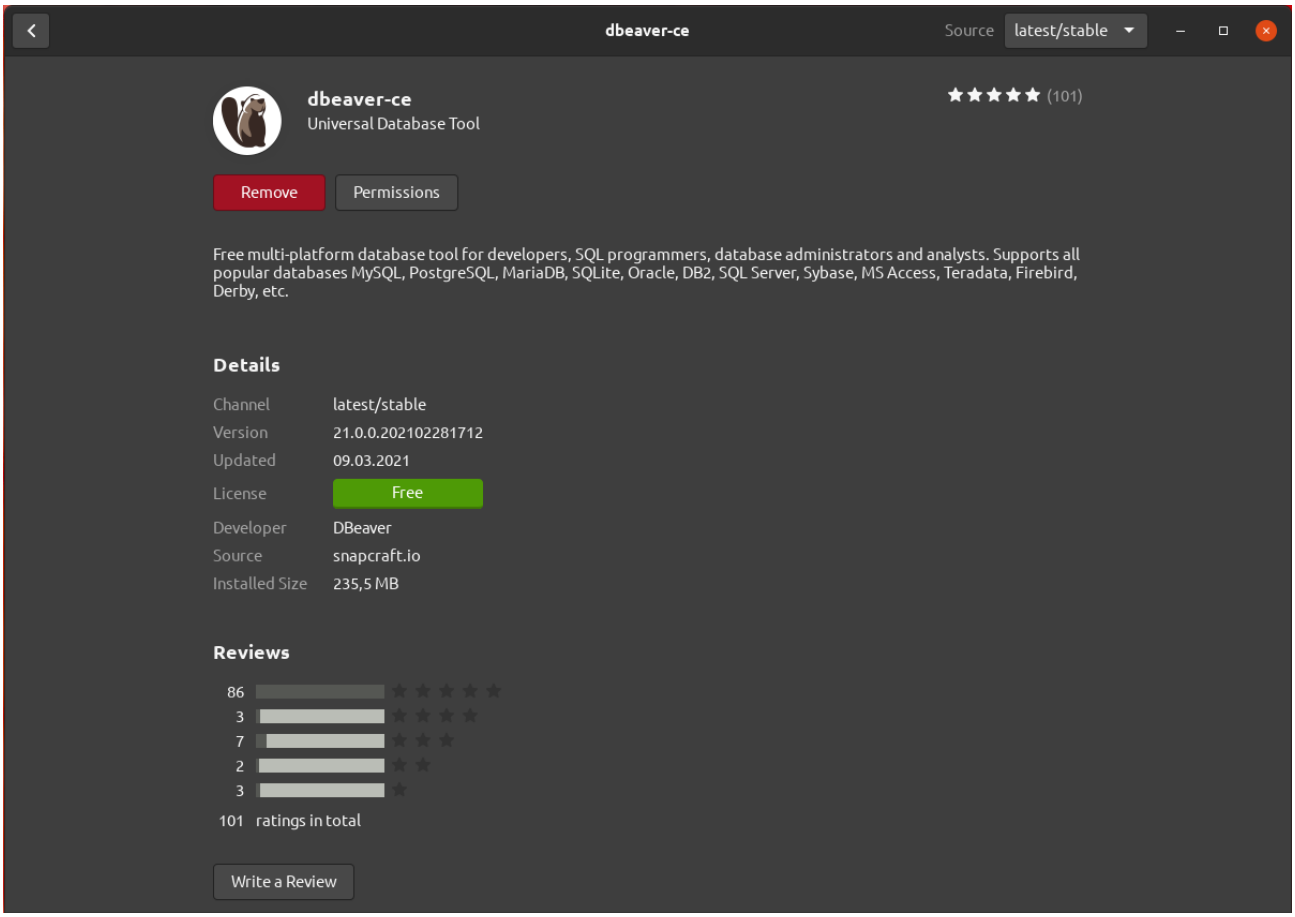

Рисунок - 8.4 Завершение установки программы

1.1.2 Установка DBeaver для ОС Windows

1. Запустите установочный файл от имени администратора (см. `install_admin`).

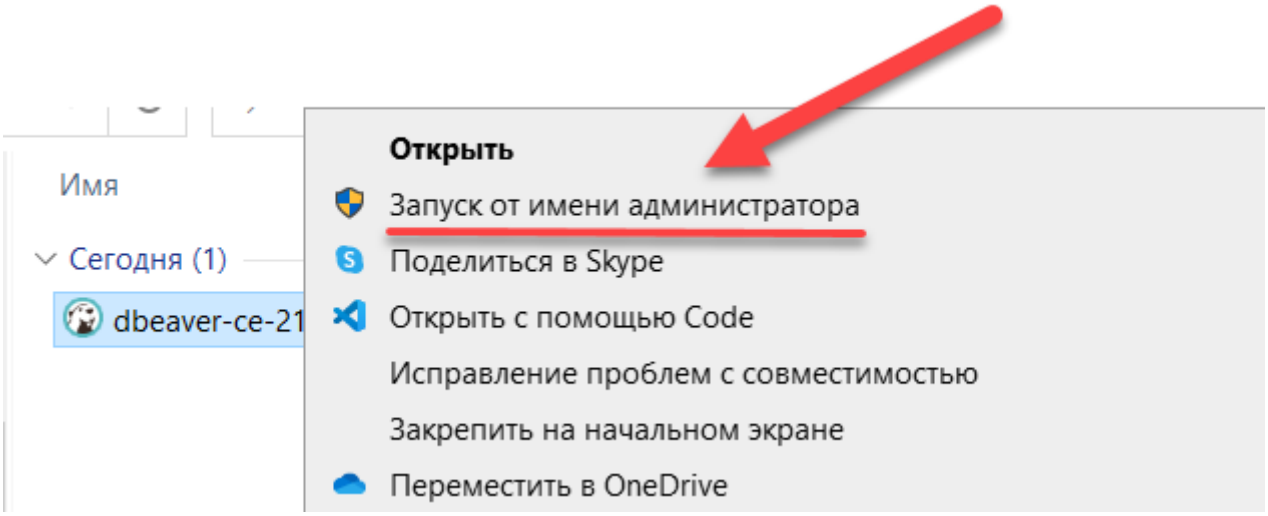

Рисунок - 8.5 Запуск установочного файла от имени администратора

2. В открывшемся окне (см. `lang`) выберите язык установки:

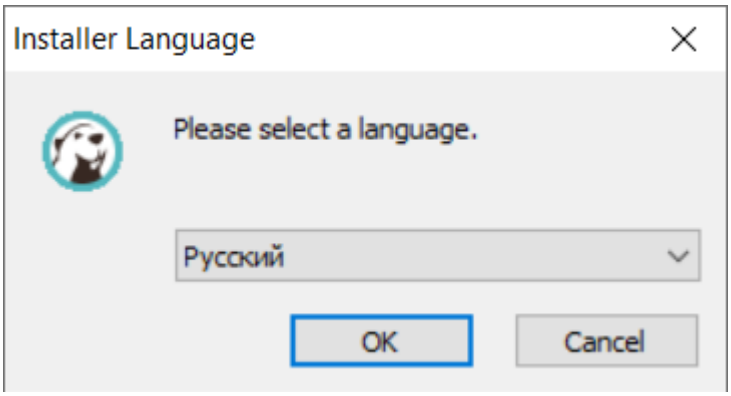

Рисунок - 8.6 Выбор языка установки

3. В окне «Мастер установки DBeaver Community» нажмите кнопку **Далее** (см. master_install).

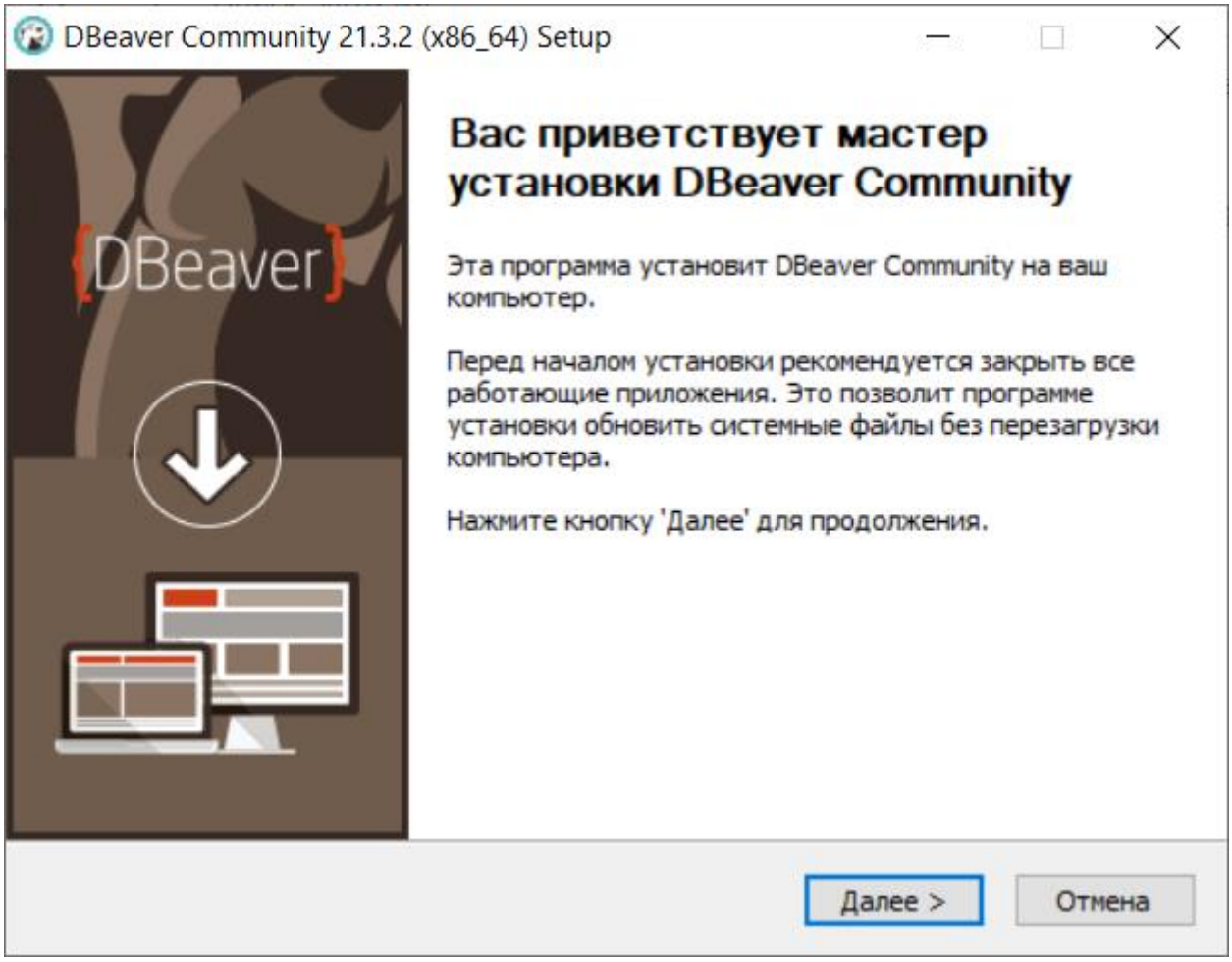

Рисунок - 8.7 Окно «Мастер установки DBeaver Community»

4. В открывшемся окне «Лицензионное соглашение» нажмите кнопку **Принимаю** (см. lisence).

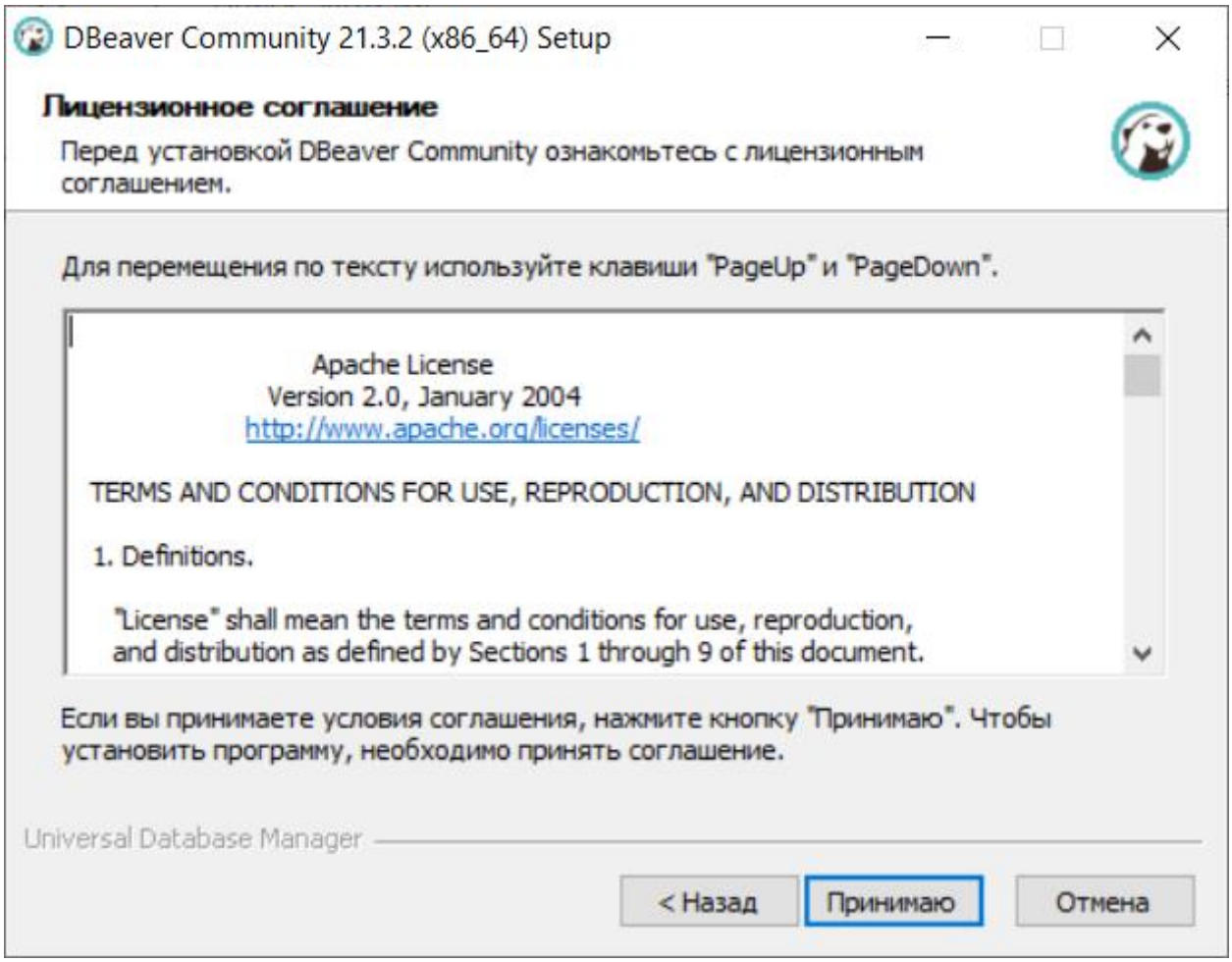

Рисунок - 8.8 Лицензионное соглашение

5. В окне «Выбор пользователя» выберите пользователей компьютера, которым будет доступна программа и нажмите кнопку **Далее** (см. `select_user_lite`).

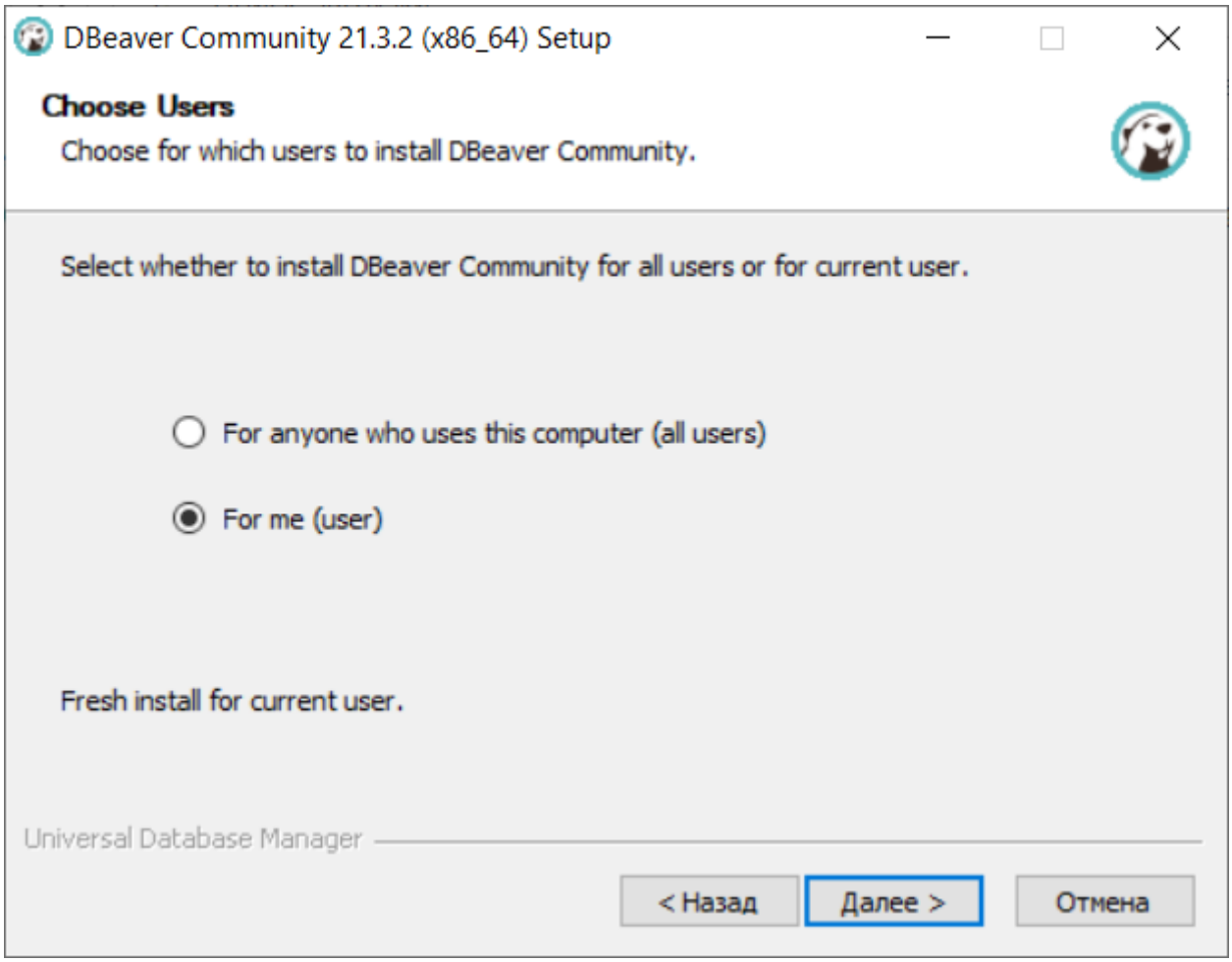

Рисунок - 8.9 Выбор пользователя

6. В окне «Компоненты устанавливаемой программы» выберите компоненты программы, которые требуется установить (см. `select_component`) и нажмите кнопку **Далее**.

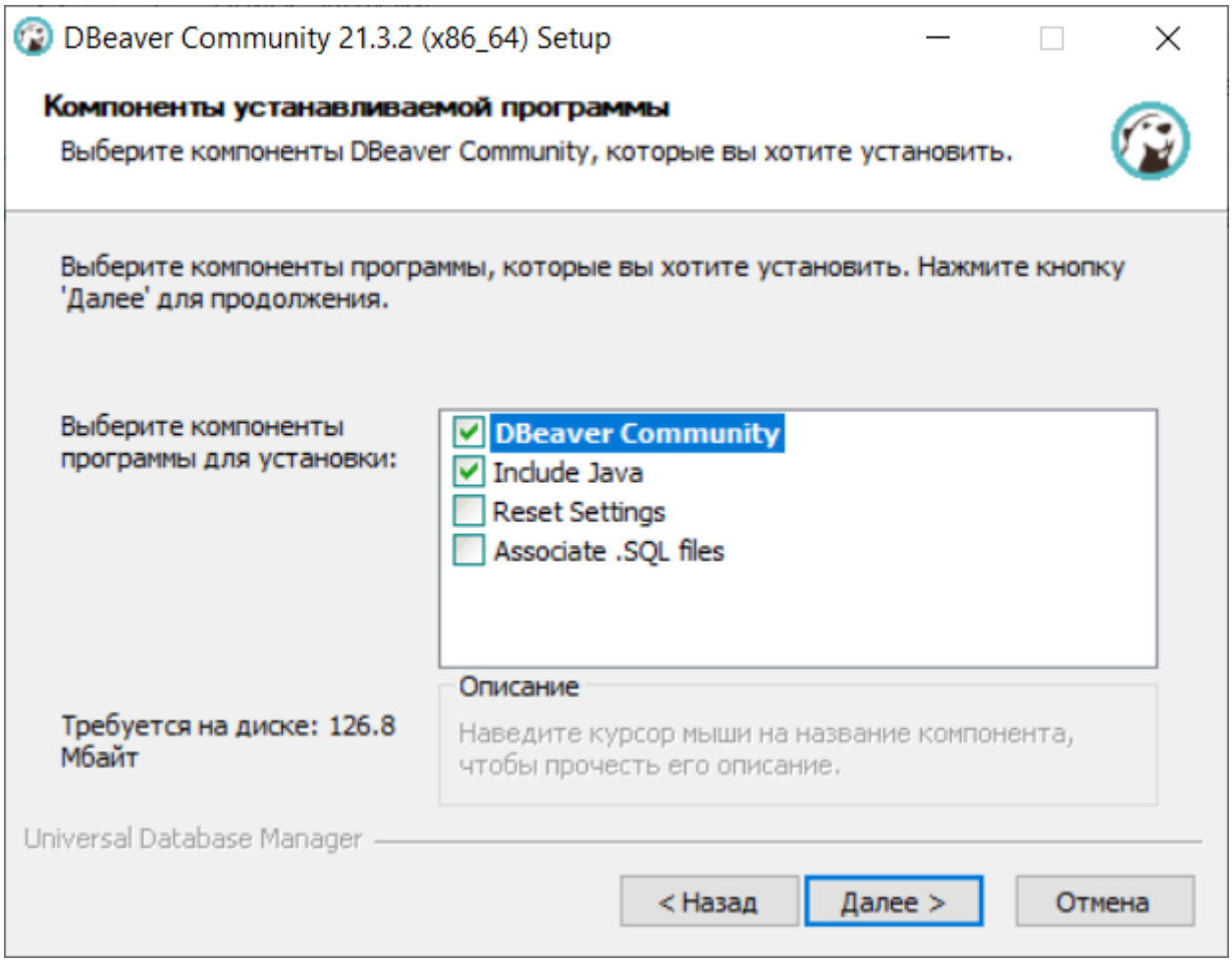

Рисунок - 8.10 Выбор компонентов программы для установки

7. В открывшемся окне «Выбор папки установки» вы можете выбрать папку установки или оставить путь к папке установки по умолчанию. Чтобы изменить папку установки нажмите кнопку **Обзор** и выберите требуемую папку. Для продолжения установки нажмите кнопку **Далее** (см. `select_folder`).

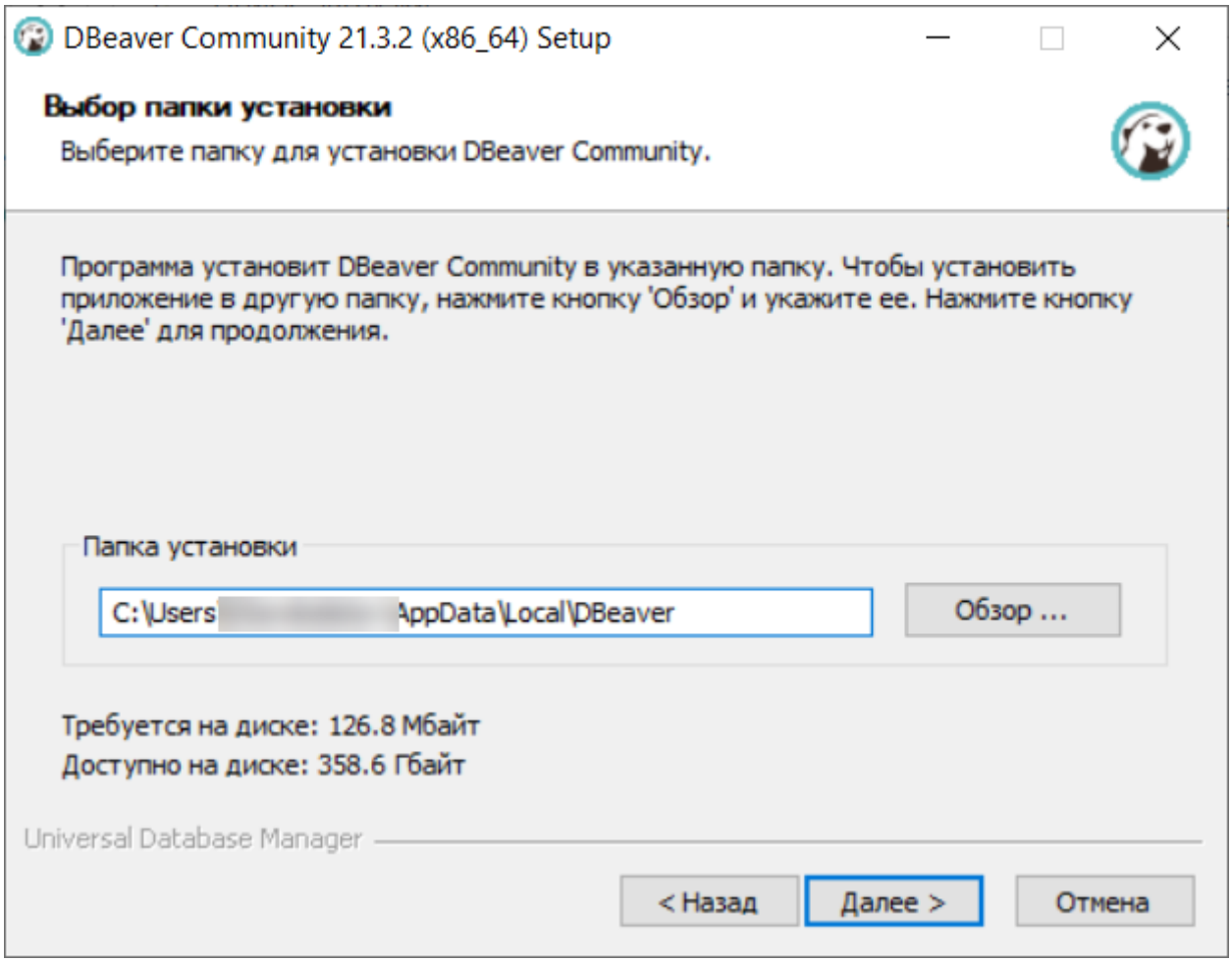

Рисунок - 8.11 Выбор папки установки

8. В окне «Папка в меню «Пуск» выберите папку, в которую будет помещен ярлык программы или установите маркер в поле «Не создавать ярлыки», в этом случае ярлык программы не будет создан (см. `select_folder_start`). Нажмите кнопку **Установить** для продолжения процесса установки.

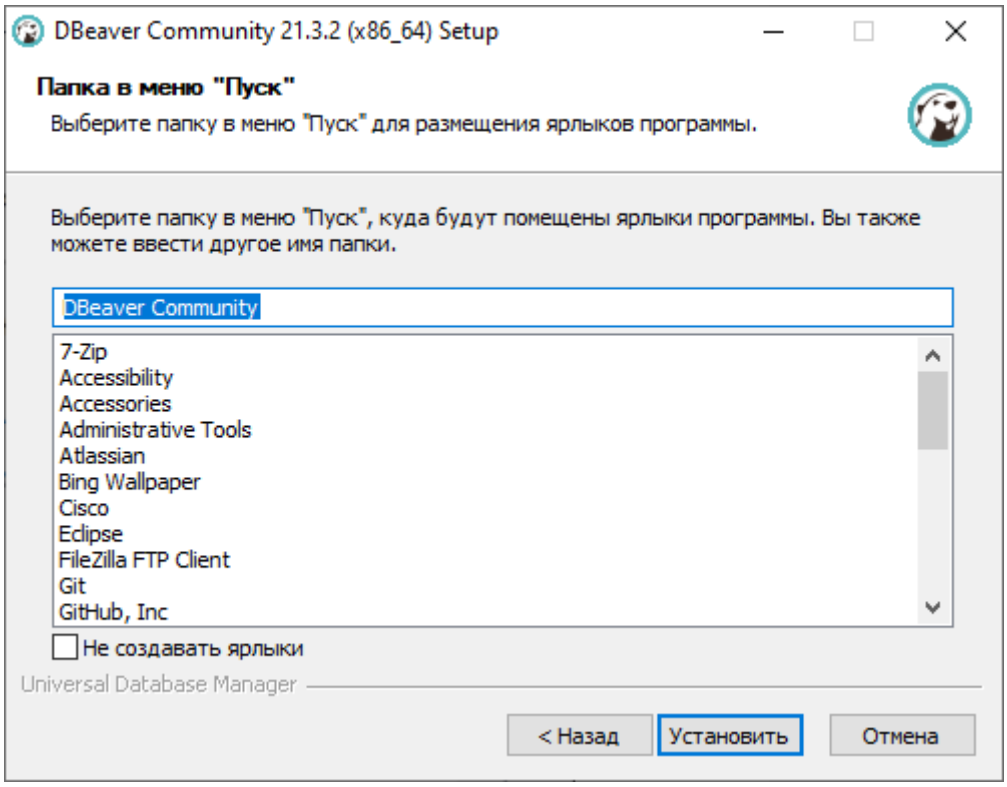

Рисунок - 8.12 Выбор папки в меню «Пуск»

9. В окне «Копирование файлов» будет отображен процесс копирования установочных файлов программы (см. `soft_copy`).

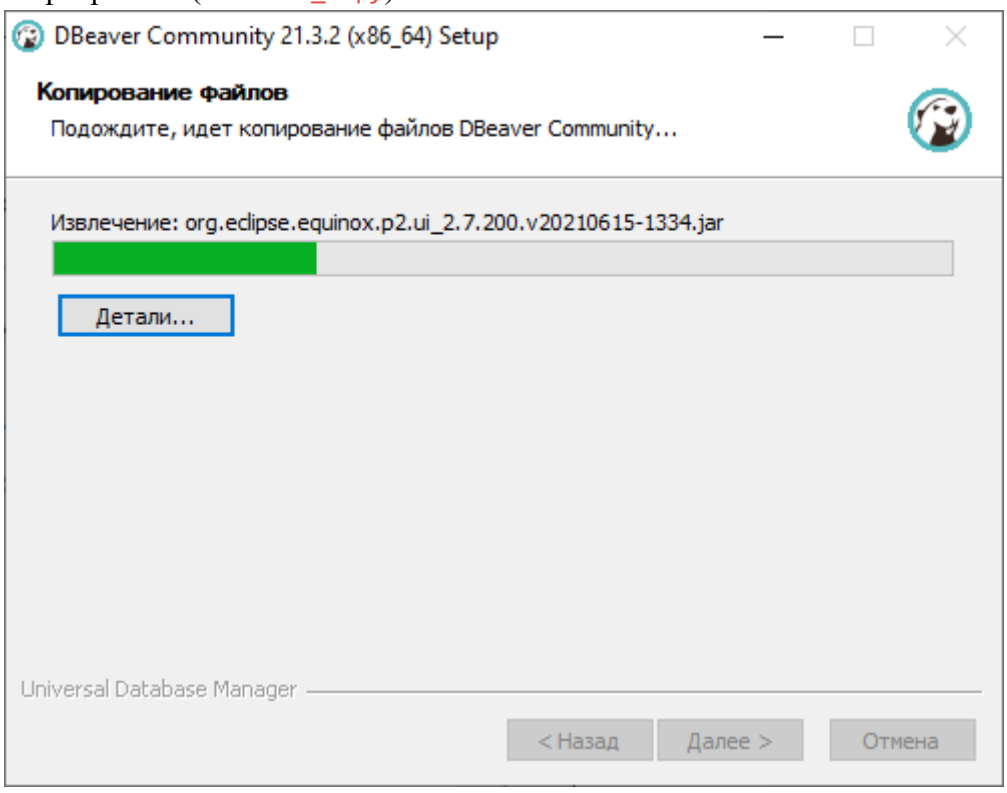

Рисунок - 8.13 Копирование файлов программы

10. Дождитесь окончания процесса копирования файлов. После завершения процесса установки на экране монитора отобразится окно «Завершение работы мастера установки» (см. `master_end`). Нажмите кнопку **Готово**.

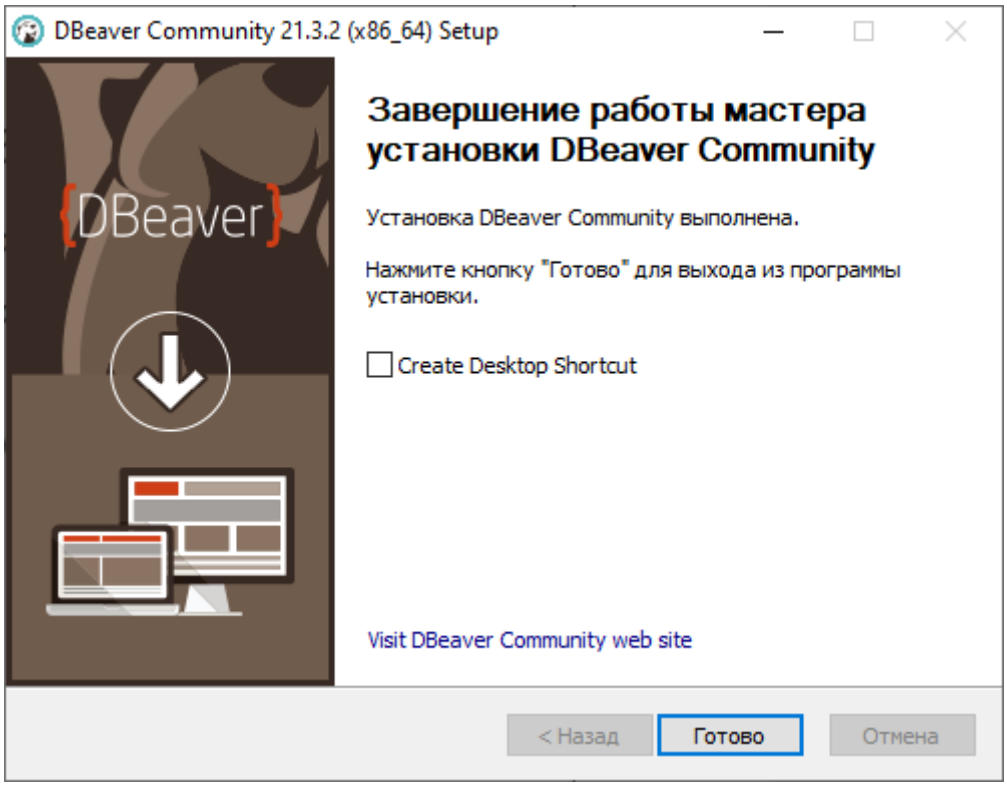

Рисунок - 8.14 Завершение работы мастера установки

1.2 Установка и настройка JDBC-драйвера

Для дальнейшей работы с программой оператору необходимо настроить подключение к базам данных программы. Для этого необходимо установить и настроить JDBC-драйвер.

Дистрибутив с актуальной версией JDBC-драйвер находится в установочном пакете с программой.

1.2.1 Установка и настройка JDBC-драйвера для ОС Windows

Чтобы установить JDBC-драйвер и настроить подключение к базам данных в программе Dbeaver, работающей под операционной системой Windows, выполните следующие действия:

1. Откройте программу Dbeaver.
2. В главном меню программы выберите «Базы данных» и нажмите пункт **Управление драйверами** (см. `drivers`).

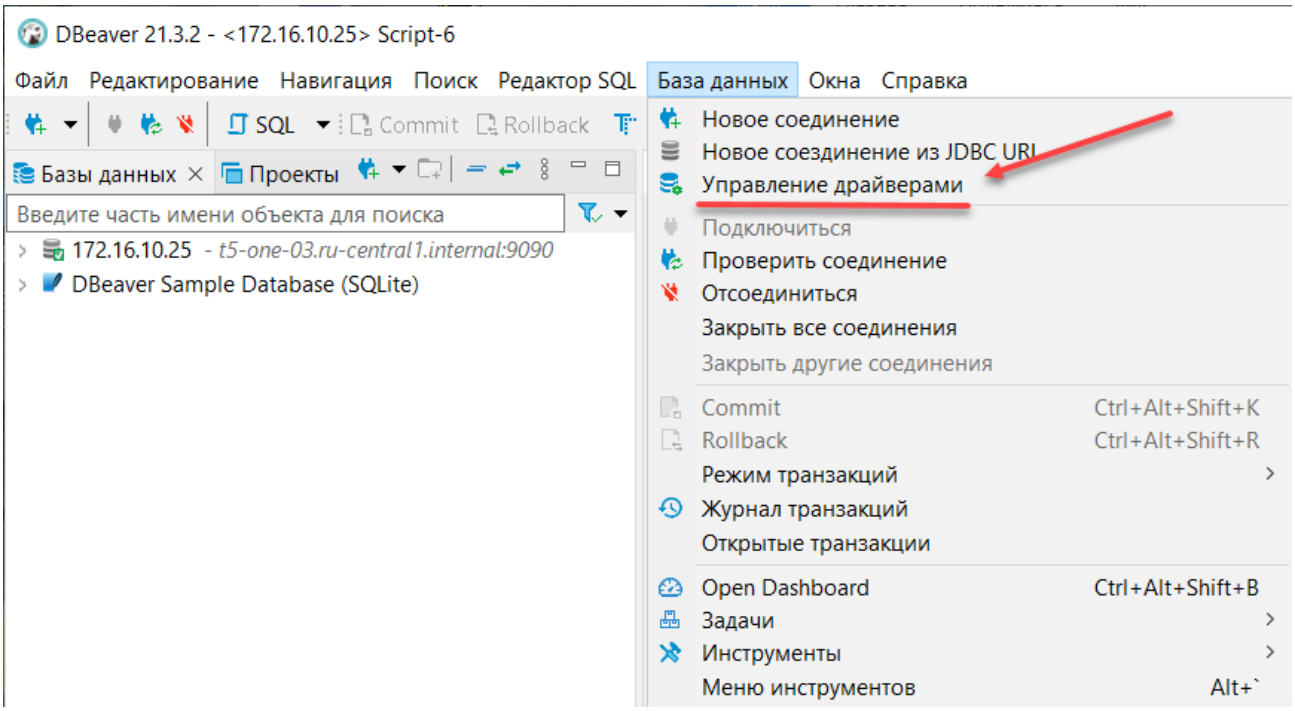

Рисунок - 8.15 Управление драйверами

3. В открывшемся окне «Менеджер драйверов» нажмите кнопку **Новый** (см. `driver_manager`).

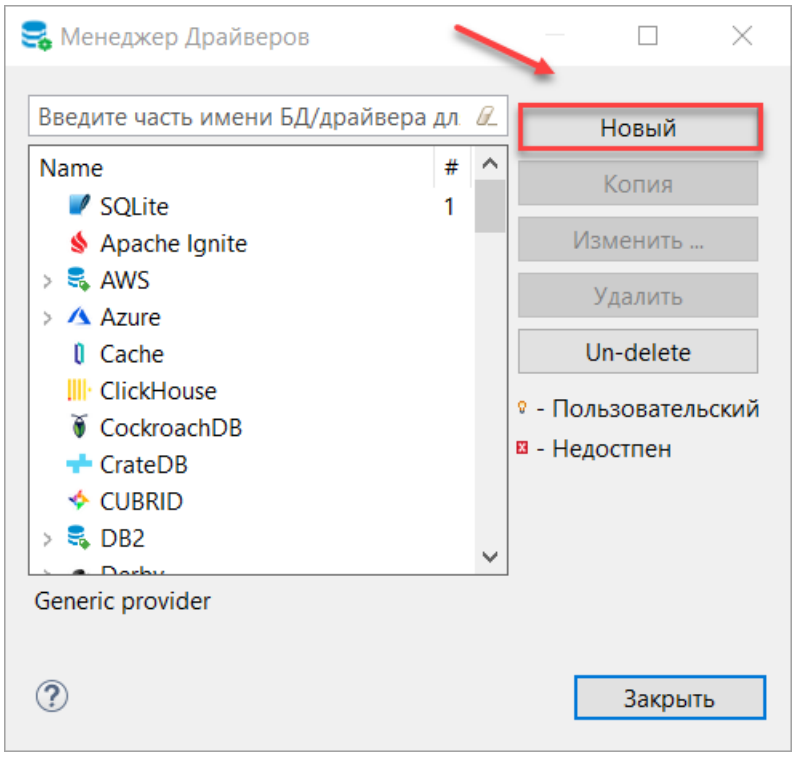

Рисунок - 8.16 Окно «Менеджер драйверов»

4. В открывшемся окне «Создать драйвер» (см. `driver_create`) заполните следующую информацию:
   - Имя драйвера: `DtmDriver`;
   - Имя класса: `ru.datamart.prostore.jdbc.Driver`;
   - Шаблон URL: `jdbc:prostore://{host}:{port}`.

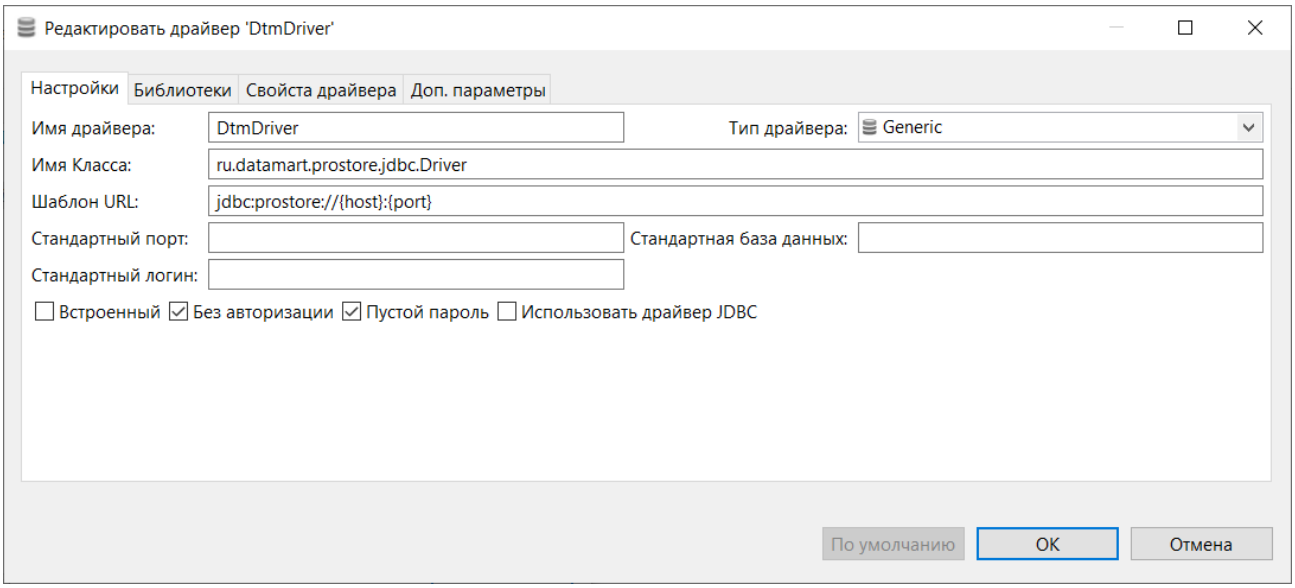

Рисунок - 8.17 Окно «Создать драйвер»

5. Установите маркер в поле «Без авторизации» и «Пустой пароль».
6. Перейдите на вкладку **Библиотека** (см. `library`), нажмите кнопку **Добавить файл** и укажите путь к jar-файлу JDBC-драйвера.

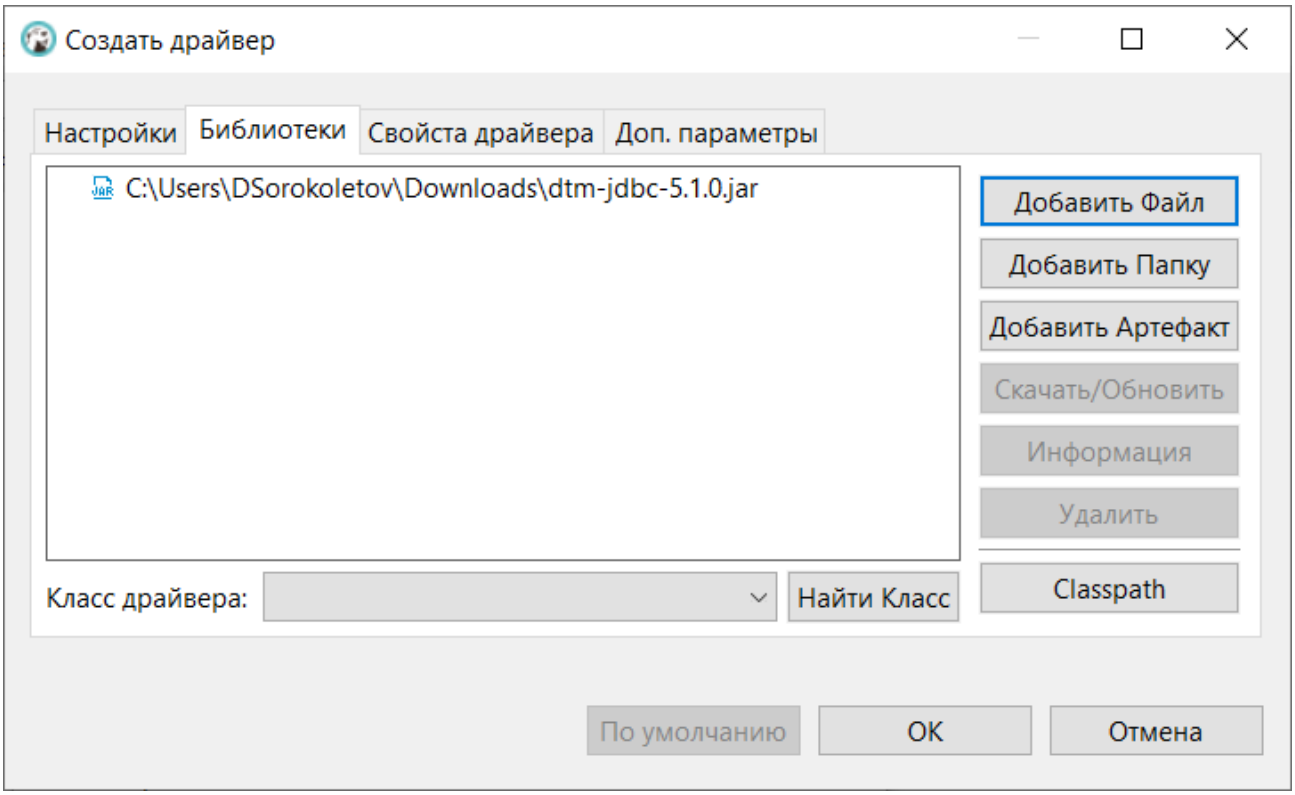

Рисунок - 8.18 Вкладка «Библиотека»

7. Нажмите кнопку «Ок».
8. Проверьте, что драйвер был добавлен в программу. Для этого откройте окно «Менеджер драйверов» (*База данных* > *Управление драйверами*) и в поисковой строке введите название драйвера – `DtmDriver` (см. `dtmDriver`).

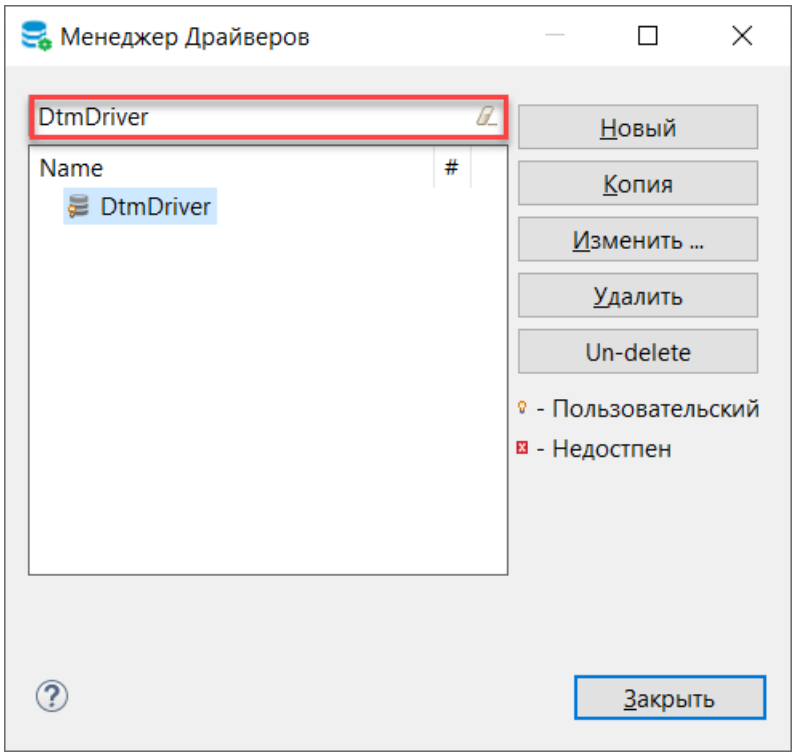

Рисунок - 8.19 Поиск DtmDriver в Менеджере драйверов

1.2.2 Подключение к базе данных

Для подключения к базам данных через JDBC-драйвер, выполните следующие действия:

1. Откройте Dbeaver.
2. В главном меню программы выберите пункт *База данных* > *Новое соединение*.
3. В окне «Создать соединение» в поисковой строке введите `dtmdriver` (см. `new_connect_lite`).

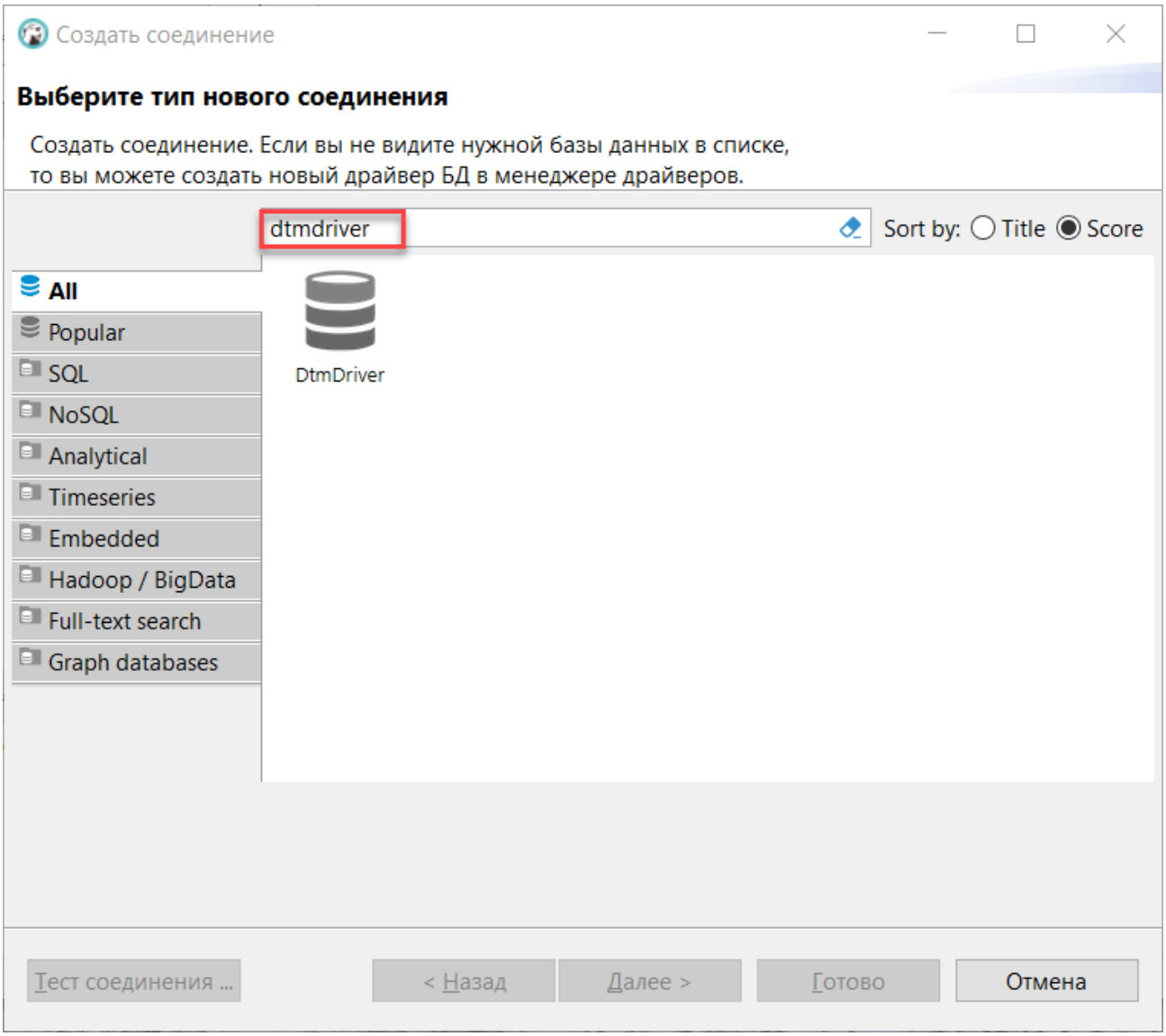

Рисунок - 8.20 Установка нового соединения с базой данных

4. Выберите DtmDriver и нажмите кнопку **Далее**

В окне «Настройка соединения» (см. `set_connect`) заполните следующие поля:

- Хост - DNS-name или IP адрес сервера базы данных. Например, `172.16.0.17`.
- Порт - порт для сервера базы данных, например, `9090`.

В результате выполненных действий, в поле `JDBC URL` будет отображено следующее значение `jdbc:prostore://172.16.0.17:9090` (см. рис. ниже).

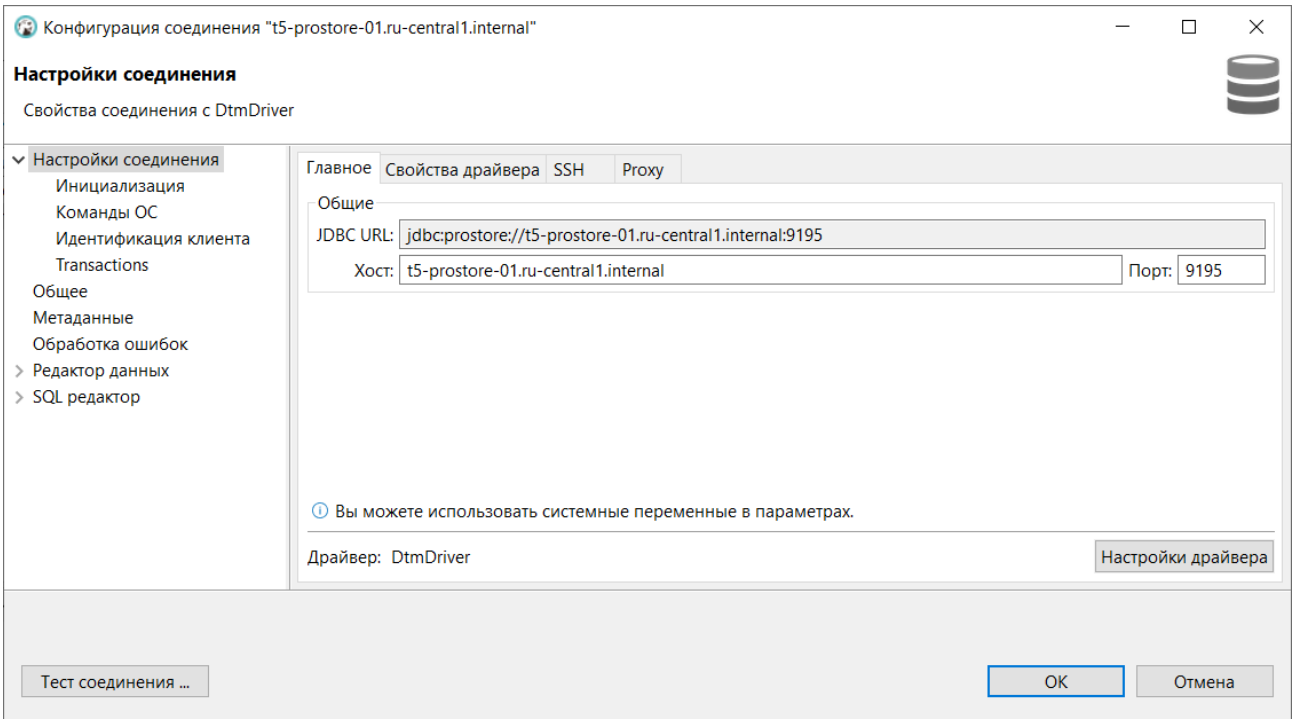

Рисунок - 8.21 Настройка соединения

5. Нажмите кнопку **Тест соединения** для проверки подключения.

В случае успешного подключение отобразится сообщение о корректном подключении (см. success_connect).

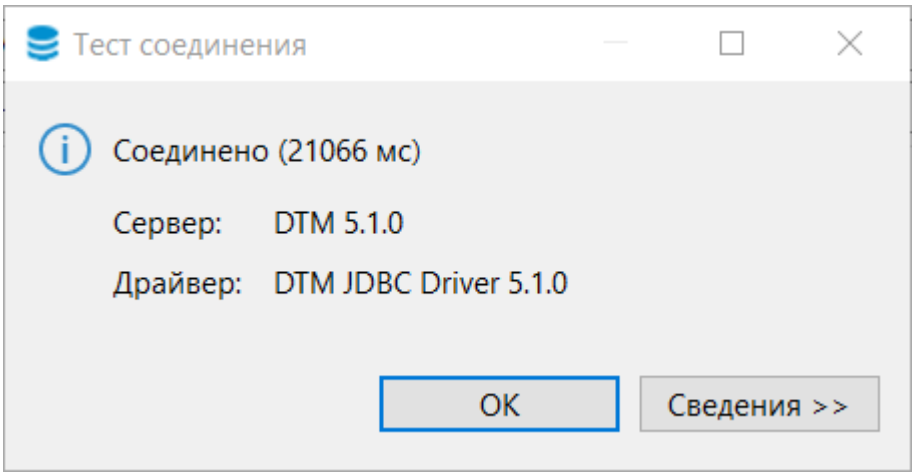

Рисунок - 8.22 Сообщение об успешном подключении к БД

6. Закройте окно проверки соединения, для этого нажмите кнопку **ОК**.
7. В окне «Настройка базового соединения» нажмите кнопку **Готово**.

1.2.3 Установка и настройка драйвера JDBC-драйвер для ОС Linux

(В данном разделе описан процесс установки драйвера в Dbeaver, работающий под управлением операционной системы Linux Ubuntu, версия 20.04).

Чтобы установить драйвер и настроить подключение к базам данных, выполните следующие действия:

1. Откройте программу Dbeaver.
2. В главном меню программы выберите «Database» и нажмите пункт **Driver Manager** (см. driver_manager_linux).

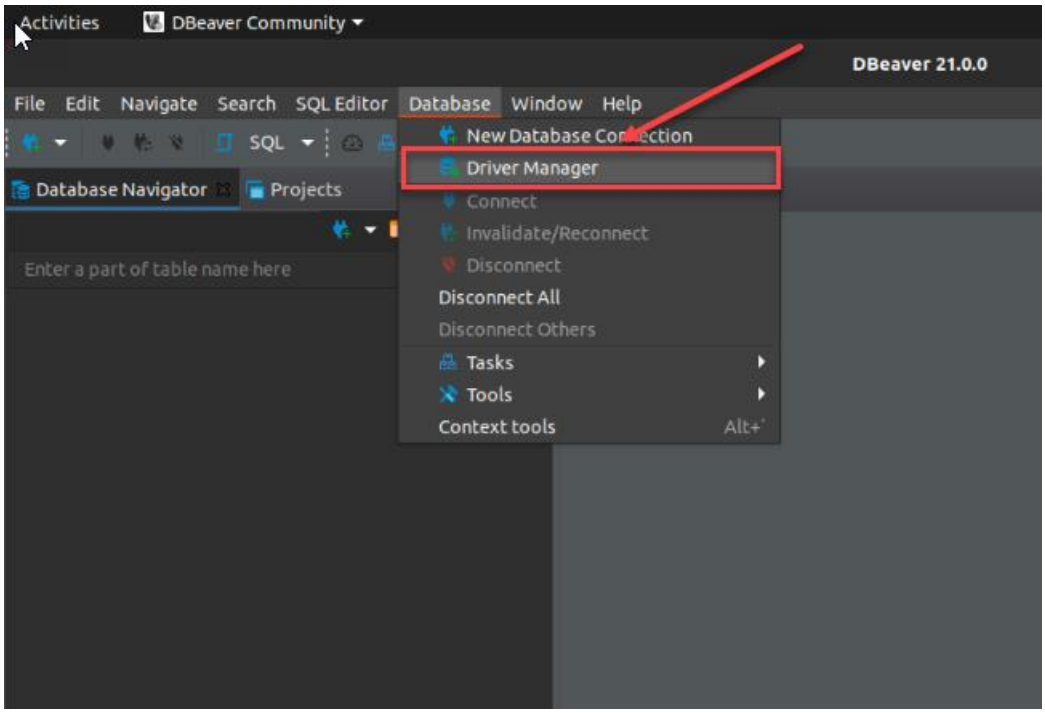

Рисунок - 8.23 Управление драйверами

3. В открывшемся окне «Driver Manager» нажмите кнопку **New** (см. `new_driver`).

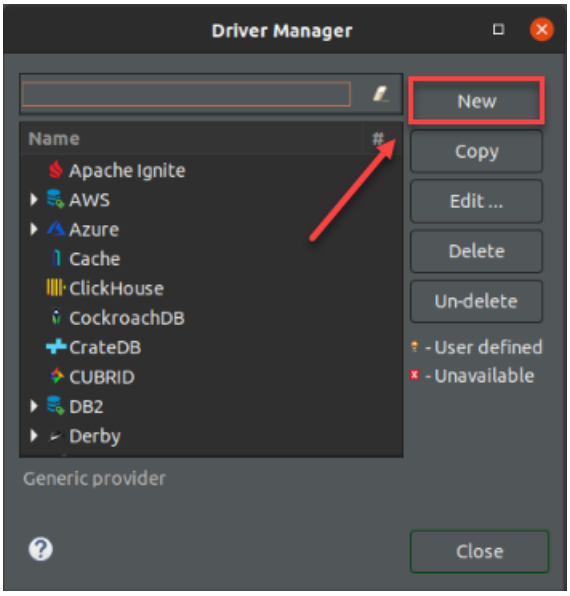

Рисунок - 8.24 Окно «Driver Manager»

4. В открывшемся окне «Create new driver» (см. `new_driver_create`) заполните следующую информацию:
   - Driver Name: `DtmDriver`;
   - Class Name: `ru.datamart.prostore.jdbc.Driver`;
   - URL Template: `jdbc:prostore://{host}:{port}`.

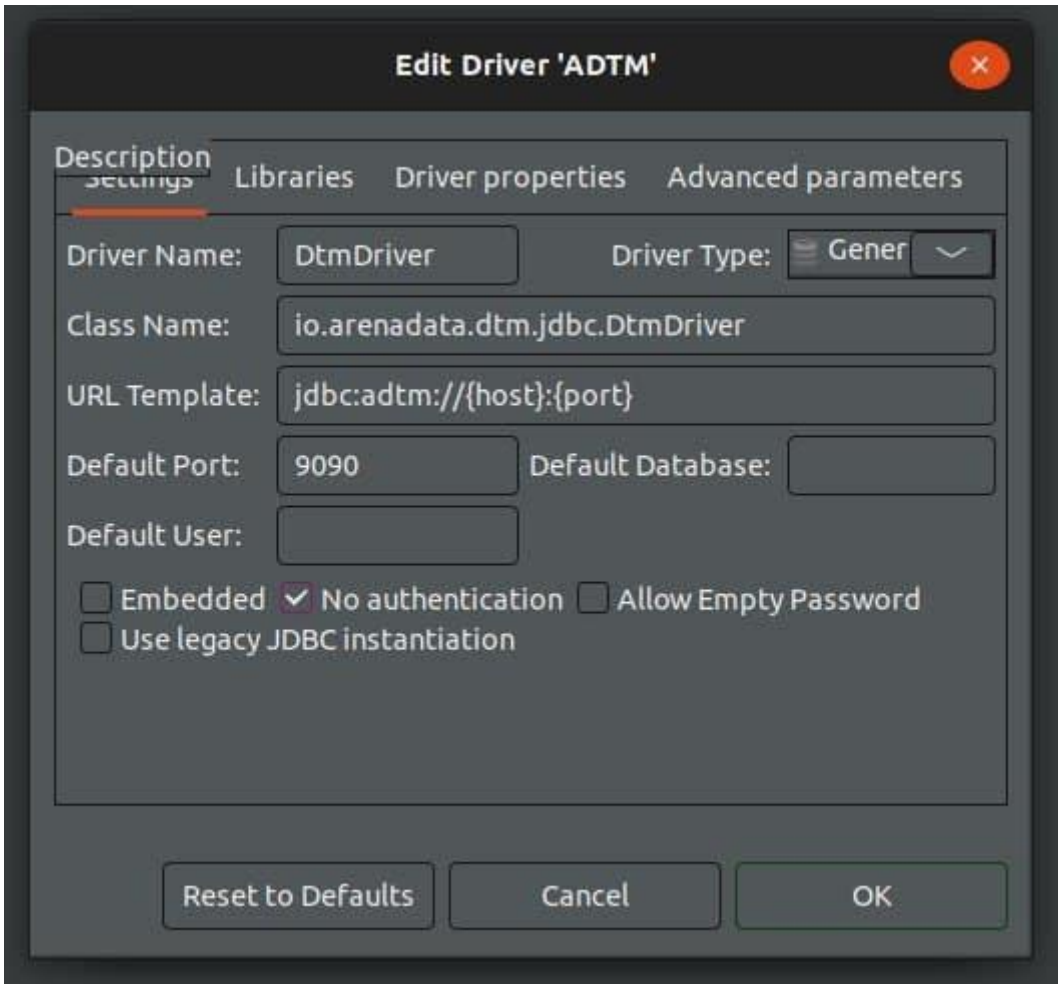

Рисунок - 8.25 Окно «Create new driver»

5. Установите маркер в поле «No authentication» и «Allow Empty Password».
6. Во вкладке «Libraries» укажите путь к jar-файлу с JDBC-драйвер.
7. Нажмите кнопку **Ок**.
8. Проверьте, что драйвер был добавлен в программу. Для этого в окне «Driver Manager» в поисковой строке введите название драйвера – DtmDriver (см. `driver_search`).

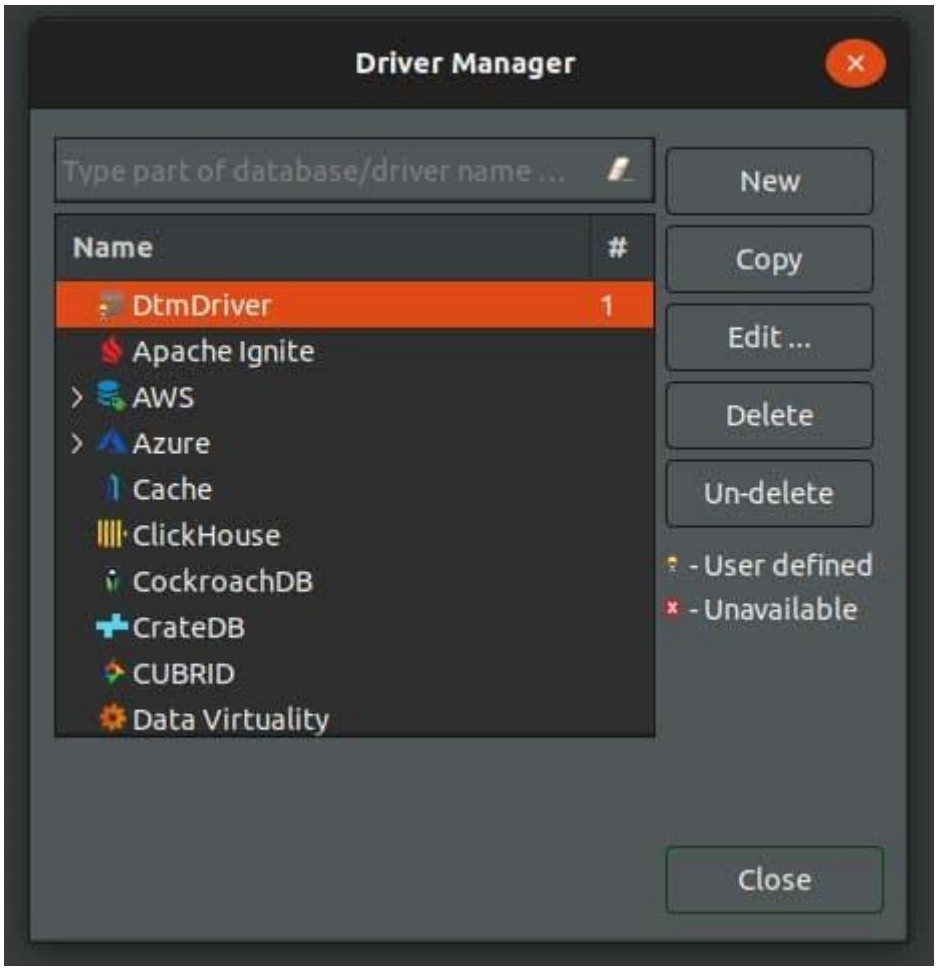

Рисунок - 8.26 Окно «Поиск DtmDriver в Driver Manager»

1.2.4 Подключение к базе данных

Для подключения к базам данных через JDBC-драйвер, выполните следующие действия:

1. Откройте программу Dbeaver.
2. В главном меню программы выберите пункт *Database* > *Connect to a Database*.
3. В окне «Connect to a Database» в поисковой строке введите `dtmdriver` (см. `driver_search_new`).

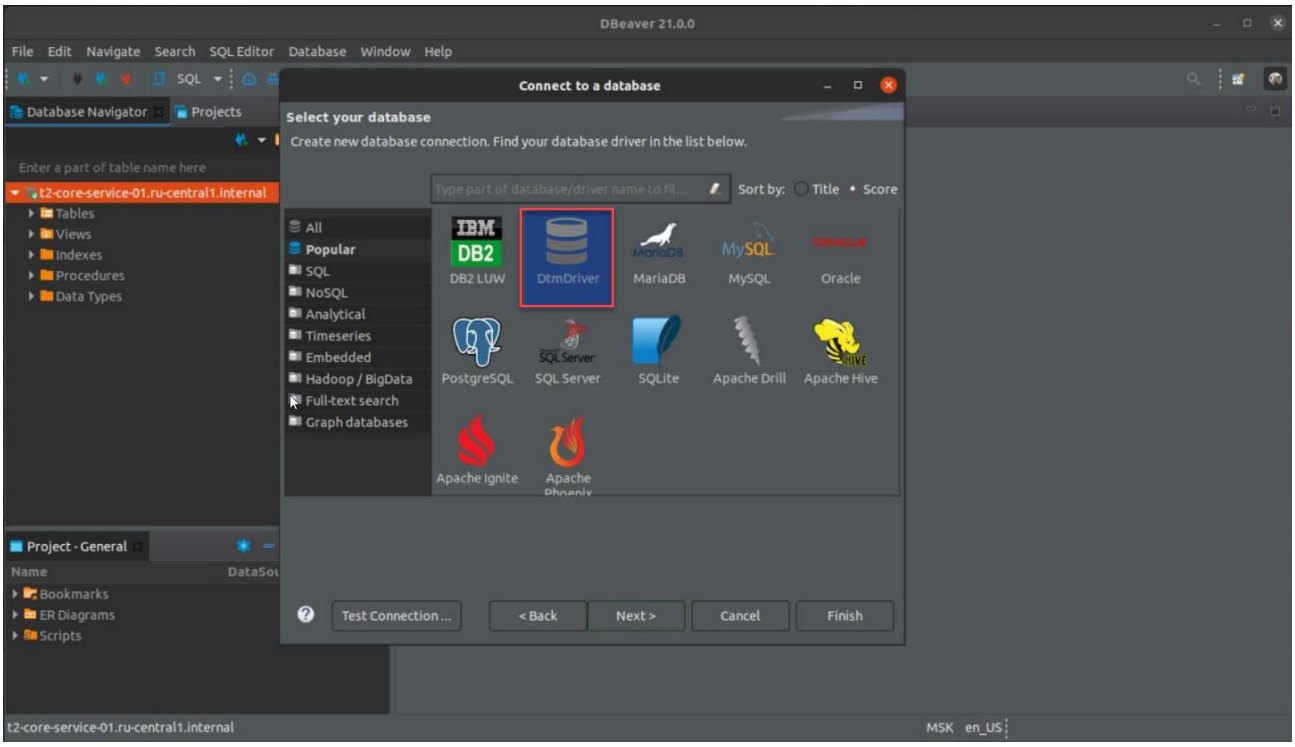

Рисунок - 8.27 Окно Поиск DtmDriver при создании нового подключения

4. Выберите DtmDriver (см. `driver_select`).

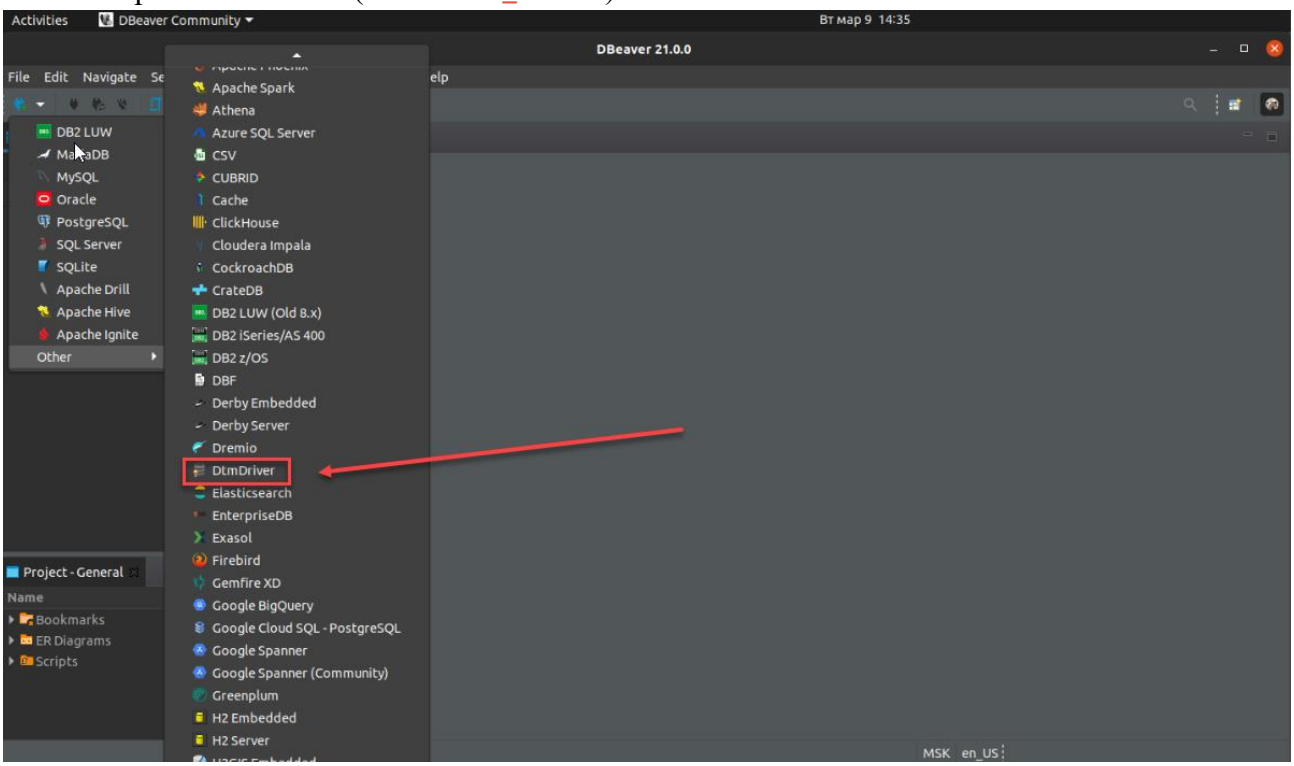

Рисунок - 8.28 Установка нового соединения с базой данных

5. В окне «Connection Settings» (см. `connection_set`) заполните следующие поля:
   - Хост - DNS-name или IP адрес сервера базы данных. Например, `172.16.0.17`.
   - Порт - порт для сервера базы данных, например, `9090`.

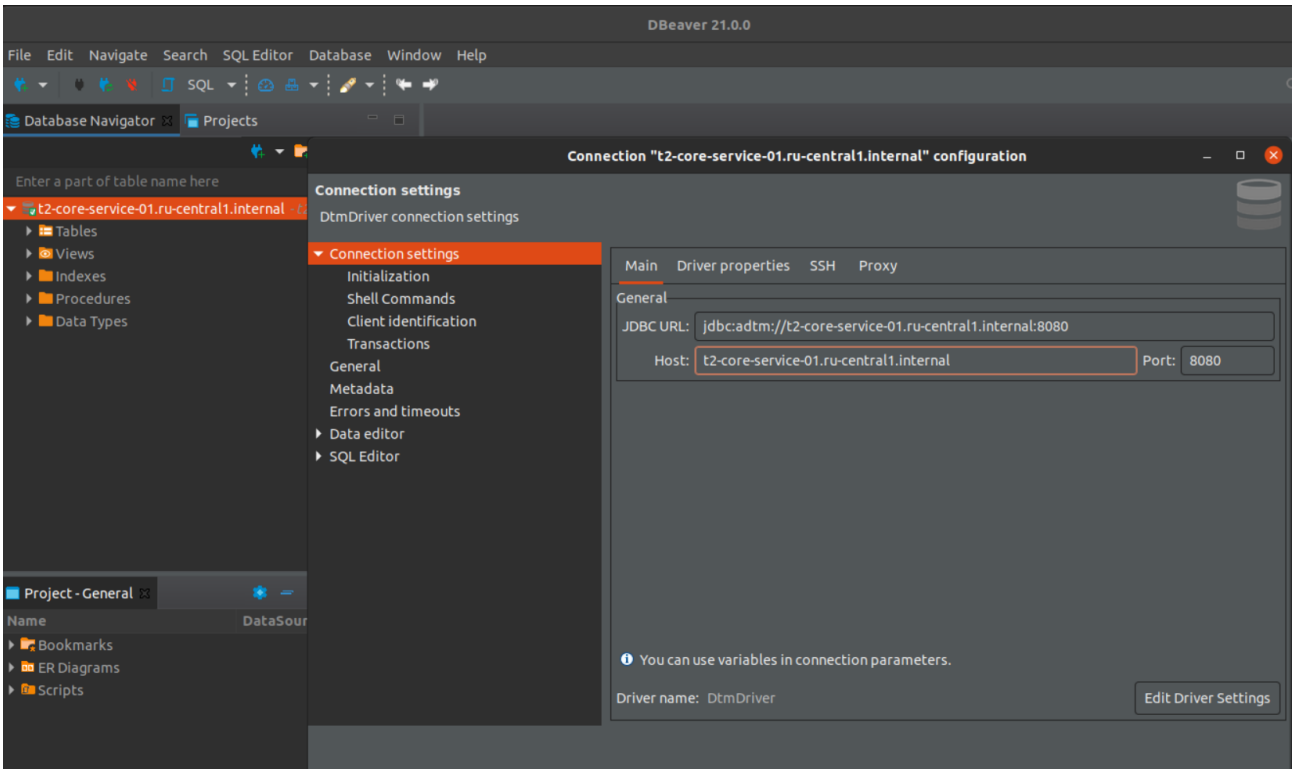

Рисунок - 8.29 Настройка базового подключения

6. Нажмите кнопку «Connection test» для проверки подключения.
7. В случае успешного подключение отобразится сообщение об успешном подключении (см. `connection_test`).

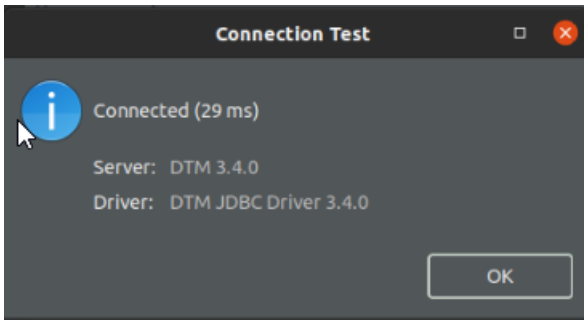

Рисунок - 8.30 Сообщение об успешном подключении к БД

8. Нажмите кнопку **Ok**

1.3 Проверка загрузки данных в БД

Данная проверка описывает возможность загрузки данных и проверку того, что данные были загружены. Для проверки загрузки данных следует выполнить следующие действия:

- создать в программе тестовую базу данных `test_upload_data` с тремя таблицами: `Passenger`, `Ticket` и `Trip`;
- проверить, что записи в таблицах отсутствуют;
- загрузить через *CSV-uploader* в таблицы `Ticket` и `Trip` тестовые данные;
- проверить, что данные в таблицы `Ticket` и `Trip` были загружены, а в таблице `Passenger` записи отсутствуют.

Внимание:

Проверку загрузки данных следует проводить на тестовом стенде.

Внимание:

В случае использования VPN следует проверить его настройки и подключение.

1.4 Создание тестовой БД

1. Откройте программу **Dbeaver**, установите подключение к БД программы и проверьте, что логическая БД `test_upload_data` отсутствует, для этого выполните следующий sql-запрос:

```
SELECT *
FROM INFORMATION_SCHEMA.schemata
WHERE schema_name = UPPER('test_upload_data');
```

2. Убедитесь, что БД отсутствует (см. `test_upload_data`).

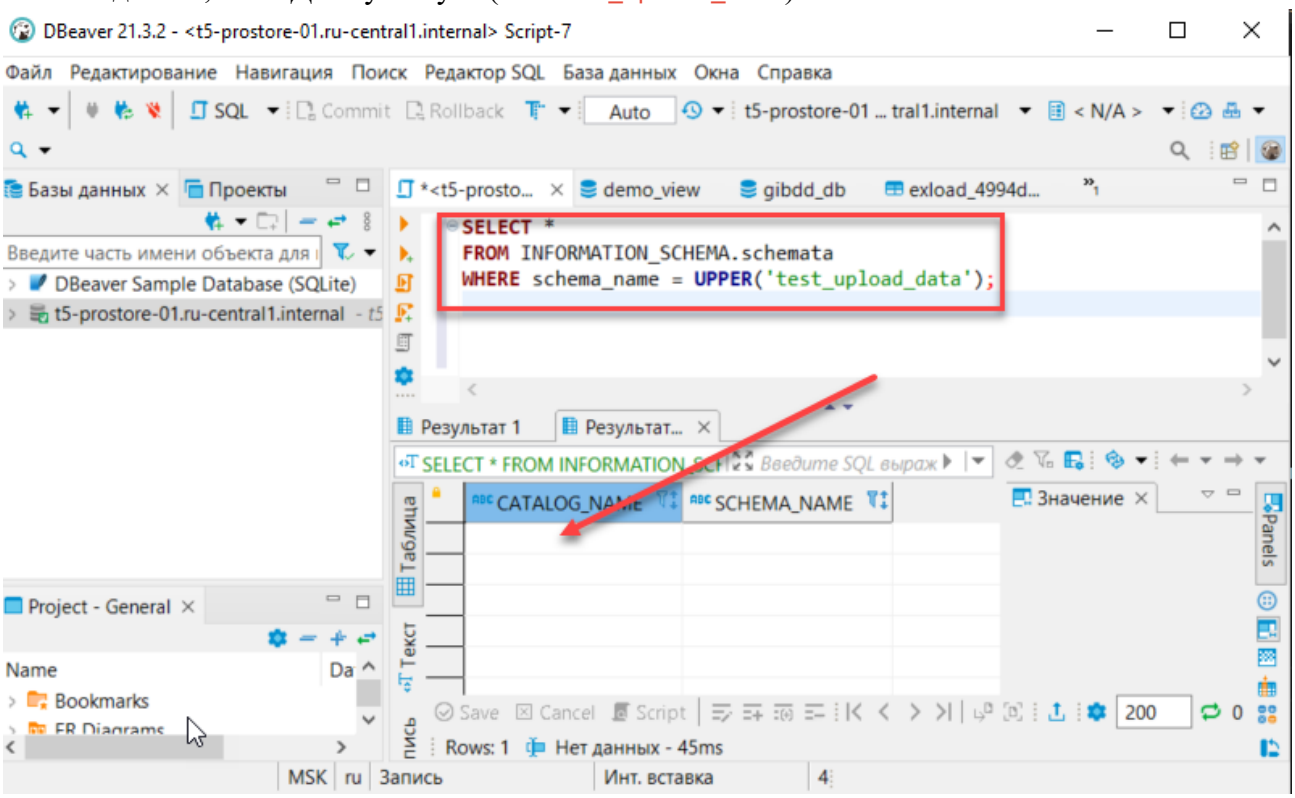

Рисунок - 8.31 Проверка БД *test_upload_data*: База данных отсутствует

3. Подготовьте XML-файл (`MetadataRequest_test_upload_data.xml`) со следующим содержанием:

```
<?xml version='1.0' encoding='utf-8'?>
<ns:PODDMetadataRequest
        xmlns:ns="urn://x-artefacts-podd-gosuslugi-local/metadata/datamart/2/1.6.0"
        xmlns:ns1="urn://x-artefacts-podd-gosuslugi-local/metadata/types/1.4.0">
    <ns:requestId>00000000-0000-0000-0000-000000000001</ns:requestId>
    <ns:metadata>
        <ns1:datamart>
            <ns1:id>1806436d-437a-400d-b32e-aa15c1a2d4bc</ns1:id>
            <ns1:mnemonic>test_upload_data</ns1:mnemonic>
            <ns1:description>test_upload_data</ns1:description>
```

```
        <ns1:tenantId>c52f062e-af97-4a44-a33f-d1a94024d0cf</ns1:tenantId>
        <ns1:version>
            <ns1:major>1</ns1:major>
            <ns1:minor>0</ns1:minor>
        </ns1:version>
        <ns1:supportedFrom>2021-01-01T00:00:00</ns1:supportedFrom>
        <ns1:datamartClass>
            <ns1:id>4c4ff97b-938b-4db6-9f4d-ae21046e4d20</ns1:id>
            <ns1:mnemonic>Passenger</ns1:mnemonic>
            <ns1:description>Passenger</ns1:description>
            <ns1:classAttribute>
                <ns1:id>6fe29bdb-7db1-405a-a05c-b49c541c92bd</ns1:id>
                <ns1:mnemonic>Code</ns1:mnemonic>
                <ns1:description>Code</ns1:description>
                <ns1:type>LONG</ns1:type>
            </ns1:classAttribute>
            <ns1:classAttribute>
                <ns1:id>e590e7b3-b611-4891-bbd1-a5e256105e73</ns1:id>
                <ns1:mnemonic>Id</ns1:mnemonic>
                <ns1:description>Id</ns1:description>
                <ns1:type>STRING</ns1:type>
            </ns1:classAttribute>
            <ns1:classAttribute>
                <ns1:id>c97a8102-6ad0-4dbd-934d-c82b83a4d83f </ns1:id>
                <ns1:mnemonic>FirstName</ns1:mnemonic>
                <ns1:description>FirstName</ns1:description>
                <ns1:type>STRING</ns1:type>
    <ns1:length>36</ns1:length>
            </ns1:classAttribute>
            <ns1:classAttribute>
                <ns1:id>d2312bfb-7ec0-4c95-9026-0f6dea48c5d9</ns1:id>
                <ns1:mnemonic>MiddleName</ns1:mnemonic>
                <ns1:description>MiddleName</ns1:description>
                <ns1:type>STRING</ns1:type>
            </ns1:classAttribute>
            <ns1:classAttribute>
                <ns1:id>7b63db89-bd0e-4c92-8bc0-e609175937b9</ns1:id>
                <ns1:mnemonic>LastName</ns1:mnemonic>
                <ns1:description>LastName</ns1:description>
                <ns1:type>STRING</ns1:type>
            </ns1:classAttribute>
            <ns1:classAttribute>
                <ns1:id>8f3e7f95-f66b-4d4a-b2eb-55a3e6134c3e</ns1:id>
                <ns1:mnemonic>Birthday</ns1:mnemonic>
                <ns1:description>Birthday</ns1:description>
                <ns1:type>DATE</ns1:type>
            </ns1:classAttribute>
            <ns1:classAttribute>
                <ns1:id>e3658240-b405-4838-99af-d32cd063c463</ns1:id>
                <ns1:mnemonic>Passport</ns1:mnemonic>
                <ns1:description>Passport</ns1:description>
                <ns1:type>STRING</ns1:type>
            </ns1:classAttribute>
            <ns1:primaryKey>
                <ns1:id>6fe29bdb-7db1-405a-a05c-b49c541c92bd</ns1:id>
                <ns1:mnemonic>Code</ns1:mnemonic>
                <ns1:description>Code</ns1:description>
                <ns1:type>
                    <ns1:id>00000000-0000-0000-0000-000000000001</ns1:id>
                    <ns1:value>LONG</ns1:value>
                </ns1:type>
            </ns1:primaryKey>
```

```
        </ns1:datamartClass>
        <ns1:datamartClass>
            <ns1:id>cafe41db-3878-4796-ba60-cbd54f042c63</ns1:id>
            <ns1:mnemonic>Ticket</ns1:mnemonic>
            <ns1:description>Ticket</ns1:description>
            <ns1:classAttribute>
                <ns1:id>bc90563b-168a-4faa-9394-7b7390dd0d92</ns1:id>
                <ns1:mnemonic>Id</ns1:mnemonic>
                <ns1:description>Id</ns1:description>
                <ns1:type>STRING</ns1:type>
            </ns1:classAttribute>
            <ns1:classAttribute>
                <ns1:id>ac93618f-752b-44d5-a77c-23a3c9eb069b</ns1:id>
                <ns1:mnemonic>PassengerId</ns1:mnemonic>
                <ns1:description>PassengerId</ns1:description>
                <ns1:type>STRING</ns1:type>
            </ns1:classAttribute>
            <ns1:classAttribute>
                <ns1:id>51355519-2d59-426e-b199-9589930acaaa</ns1:id>
                <ns1:mnemonic>TripId</ns1:mnemonic>
                <ns1:description>TripId</ns1:description>
                <ns1:type>STRING</ns1:type>
            </ns1:classAttribute>
            <ns1:classAttribute>
                <ns1:id>fe92c245-929e-4684-b9c9-22bda6939c09</ns1:id>
                <ns1:mnemonic>Number</ns1:mnemonic>
                <ns1:description>Number</ns1:description>
                <ns1:type>LONG</ns1:type>
            </ns1:classAttribute>
            <ns1:classAttribute>
                <ns1:id>4a32ded4-c970-4874-b0b1-2e3eed8b6483</ns1:id>
                <ns1:mnemonic>ByCard</ns1:mnemonic>
                <ns1:description>ByCard</ns1:description>
                <ns1:type>BOOLEAN</ns1:type>
            </ns1:classAttribute>
            <ns1:classAttribute>
                <ns1:id>35f59c80-fcc3-483c-9cd3-dc3afb606d66</ns1:id>
                <ns1:mnemonic>Price</ns1:mnemonic>
                <ns1:description>Price</ns1:description>
                <ns1:type>DOUBLE</ns1:type>
            </ns1:classAttribute>
            <ns1:classAttribute>
                <ns1:id>8b46ff55-6853-458c-851d-6e1666da918b</ns1:id>
                <ns1:mnemonic>Sold</ns1:mnemonic>
                <ns1:description>Sold</ns1:description>
                <ns1:type>TIMESTAMP</ns1:type>
            </ns1:classAttribute>
            <ns1:primaryKey>
                <ns1:id>fe92c245-929e-4684-b9c9-22bda6939c09</ns1:id>
                <ns1:mnemonic>Number</ns1:mnemonic>
                <ns1:description>Number</ns1:description>
                <ns1:type>
                    <ns1:id>00000000-0000-0000-0000-000000000001</ns1:id>
                    <ns1:value>LONG</ns1:value>
                </ns1:type>
            </ns1:primaryKey>
        </ns1:datamartClass>
        <ns1:datamartClass>
            <ns1:id>76268090-60ee-4960-8268-1b91f4186e87</ns1:id>
            <ns1:mnemonic>Trip</ns1:mnemonic>
            <ns1:description>Trip</ns1:description>
            <ns1:classAttribute>
```

```
                <ns1:id>bd173e24-ea7e-4869-9d43-9f57f5b0a82f</ns1:id>
                <ns1:mnemonic>Id</ns1:mnemonic>
                <ns1:description>Id</ns1:description>
                <ns1:type>STRING</ns1:type>
            </ns1:classAttribute>
            <ns1:classAttribute>
                <ns1:id>1ed32816-8bdb-4d35-9f66-8c08df13ad28</ns1:id>
                <ns1:mnemonic>Number</ns1:mnemonic>
                <ns1:description>Number</ns1:description>
                <ns1:type>INTEGER</ns1:type>
            </ns1:classAttribute>
            <ns1:classAttribute>
                <ns1:id>78f587fa-b53e-4912-b631-0c4a249d20b6</ns1:id>
                <ns1:mnemonic>Duration</ns1:mnemonic>
                <ns1:description>Duration</ns1:description>
                <ns1:type>STRING</ns1:type>
            </ns1:classAttribute>
            <ns1:classAttribute>
                <ns1:id>1750c564-20a7-4e07-988a-b382227123e4</ns1:id>
                <ns1:mnemonic>Length</ns1:mnemonic>
                <ns1:description>Length</ns1:description>
                <ns1:type>FLOAT</ns1:type>
            </ns1:classAttribute>
            <ns1:primaryKey>
                <ns1:id>1ed32816-8bdb-4d35-9f66-8c08df13ad28</ns1:id>
                <ns1:mnemonic>Number</ns1:mnemonic>
                <ns1:description>Number</ns1:description>
                <ns1:type>
                    <ns1:id>00000000-0000-0000-0000-000000000002</ns1:id>
                    <ns1:value>INTEGER</ns1:value>
                </ns1:type>
            </ns1:primaryKey>
        </ns1:datamartClass>
    </ns1:datamart>
</ns:metadata>
</ns:PODDMetadataRequest>
```

3. Откройте web-интерфейс *CSV-uploader*.
4. Во вкладке «Загрузка структуры», в поле выбора файла `XML` для загрузки в Витрину выберите XML-файл (`MetadataRequest_test_upload_data.xml`) созданный в п.2.
5. Нажмите кнопку «Загрузить» (см. `xml_load`).

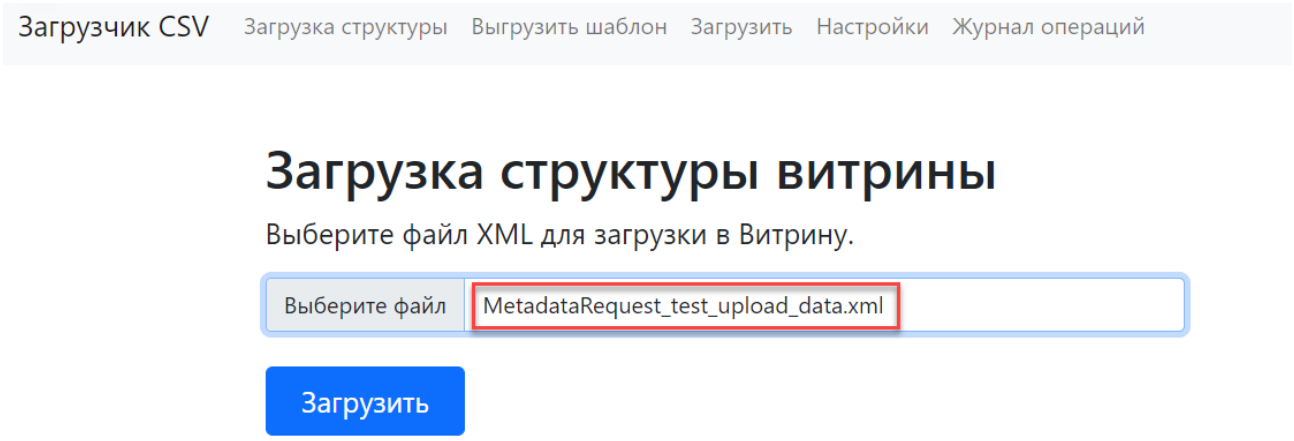

Рисунок - 8.32 Загрузка XML-файла со структурой витрины

6. В случае успешной загрузки, отобразится сообщение «Структура витрины успешно

создана» (см. structure_create).

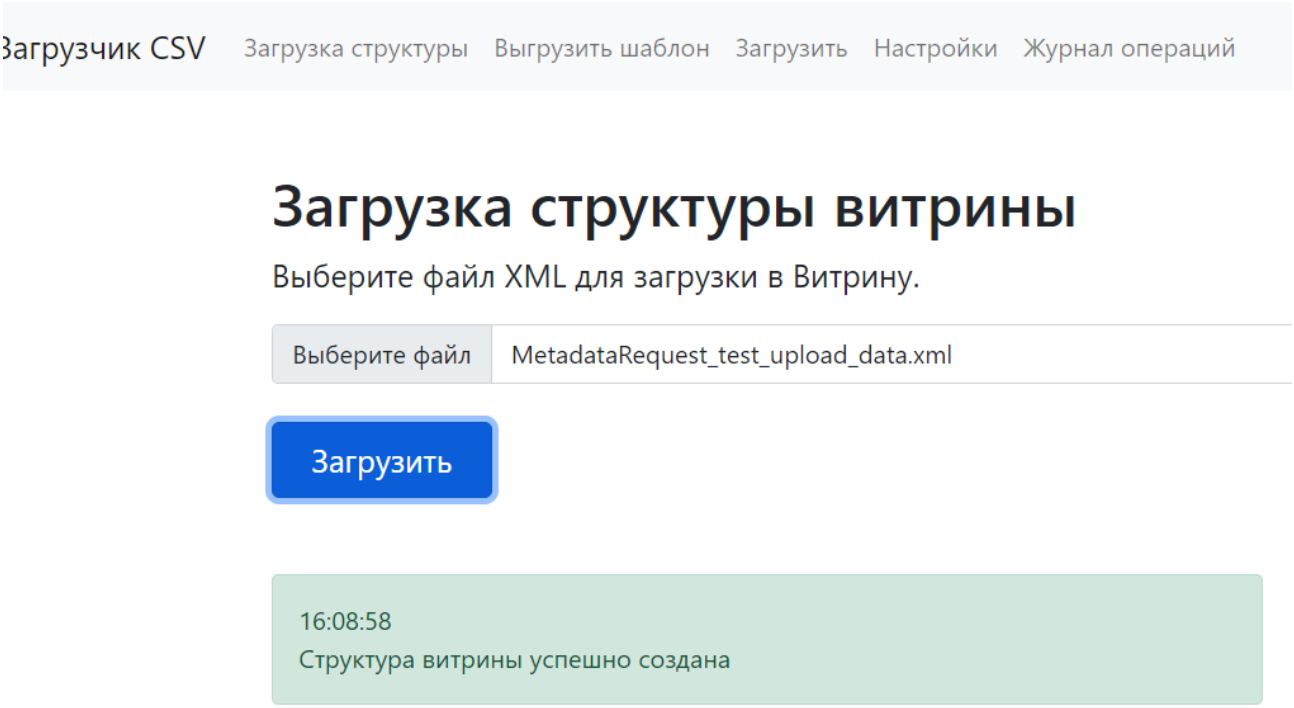

Рисунок - 8.33 Сообщение «Структура витрины успешно создана»

7. Откройте программу **Dbeaver**, установите подключение к БД программы и проверьте, что логическая БД test_upload_data создана, для этого выполните запрос:

```
SELECT *
FROM INFORMATION_SCHEMA.schemata
WHERE schema_name = UPPER('test_upload_data');
```

SQL-запрос вернул одну строку, в которой поле **SCHEMA_NAME** имеет значение TEST_UPLOAD_DATA (см. test_upload_data_4).

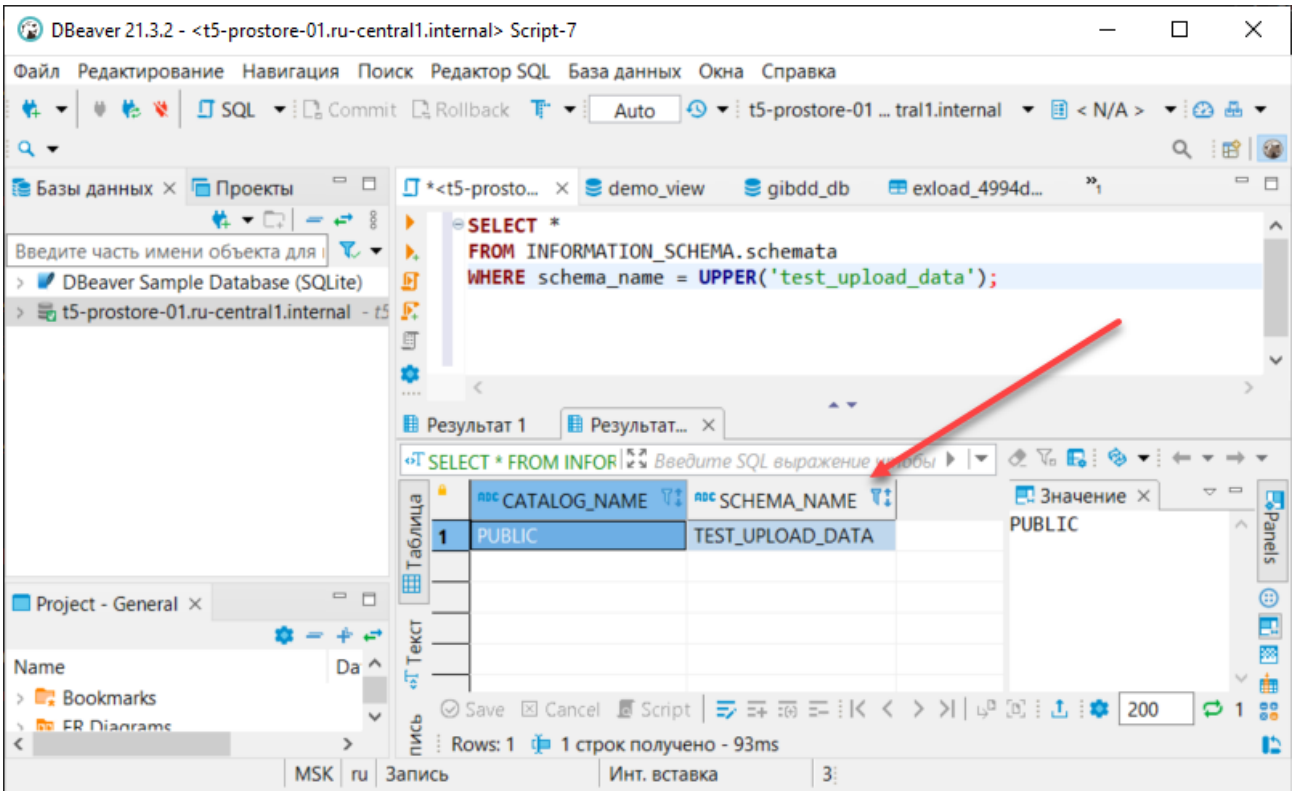

Рисунок - 8.34 Проверка БД *test_upload_data*: База данных создана

8. Убедитесь, что в созданной БД существуют таблицы: `Passenger`, `Ticket` и `Trip`. Для этого выполните запрос к БД:

```
SELECT *
FROM INFORMATION_SCHEMA.tables
WHERE table_schema = UPPER('test_upload_data');
```

SQL-запрос вернул записи о созданных логических таблицах (см. `test_upload_data_5`).

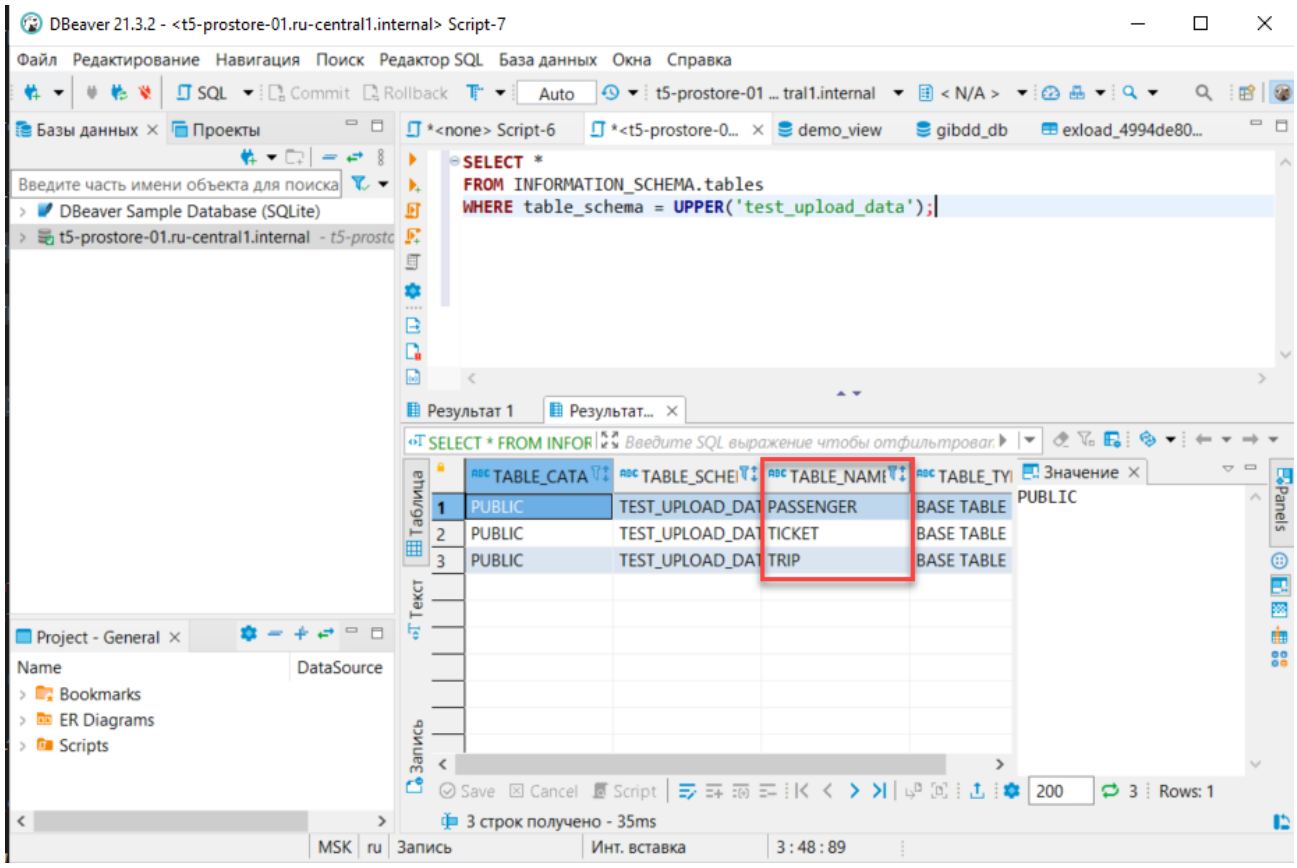

Рисунок - 8.35 Проверка таблиц БД

9. Проверьте количество записей в таблицах: `Passenger`, `Ticket` и `Trip`. Для этого выполните запрос к БД:

```
SELECT COUNT(*) FROM test_upload_data.passenger;
SELECT COUNT(*) FROM test_upload_data.ticket;
SELECT COUNT(*) FROM test_upload_data.trip;
```

SQL-запрос вернул нулевое значении (см. `test_upload_data_6`). Это значит, что записи в таблицах отсутствуют.

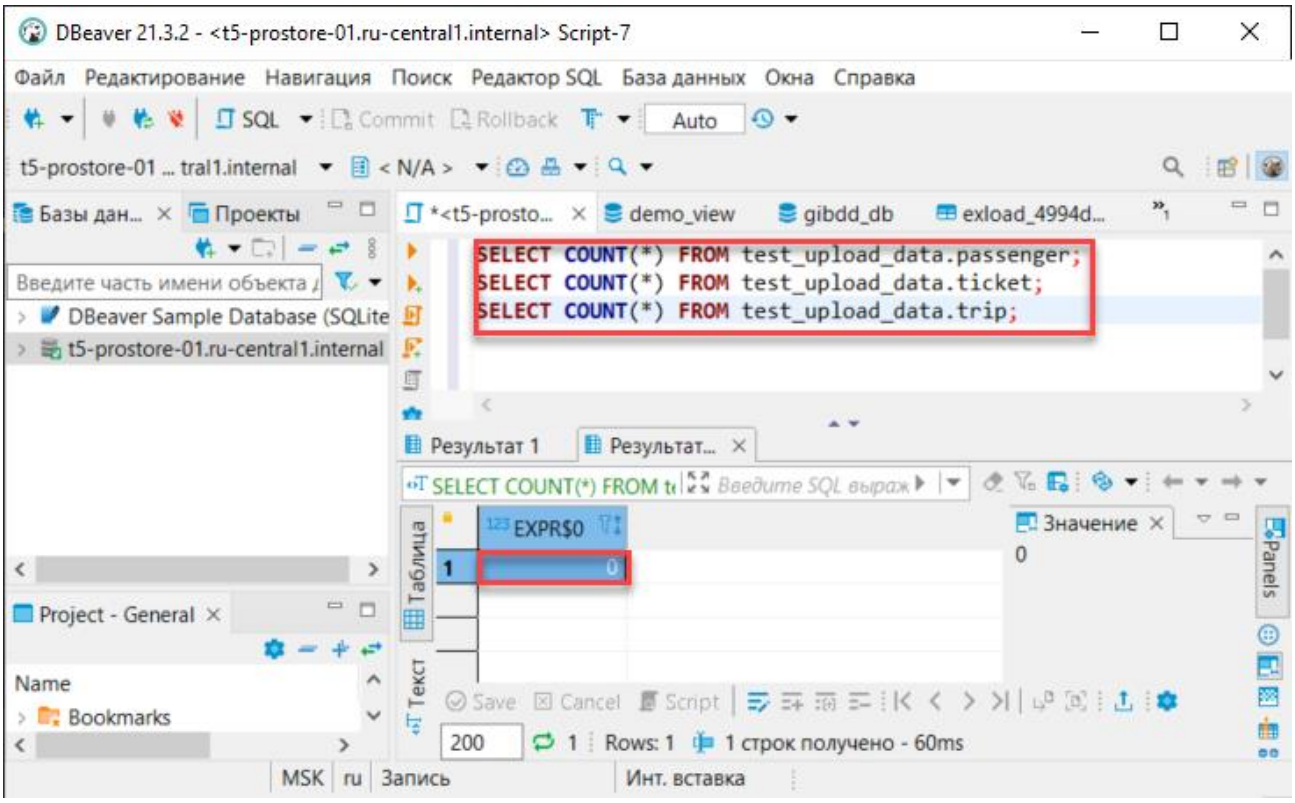

Рисунок - 8.36 Проверка таблиц БД

1.5 Загрузка данных

1. Скачайте CSV-файлы (`tickets.csv`, `trips.csv`).
2. Откройте web-интерфейс *CSV-uploader* и перейдите на вкладку **Загрузка**.
3. Выберите режим «Вставка».
4. Установите переключатель в режим **Автоматическое определение таблицы**.
5. В поле **CSV** выберите CSV-файлы (`tickets.csv` и `trips.csv`) для загрузки в БД (см. `test_upload_data_7`).

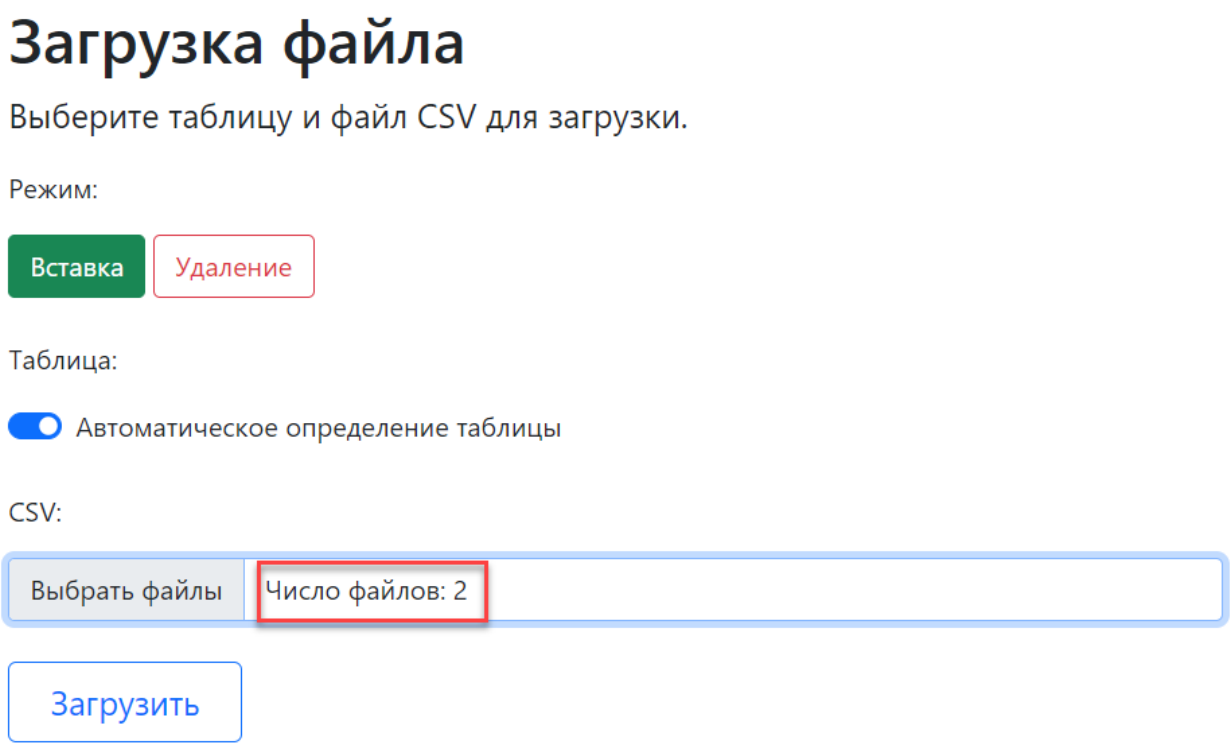

Рисунок - 8.37 Загрузка данных из CSV-файла

6. Отобразится сообщение **Отправка запроса на загрузку…** (см. `test_upload_data_8`).

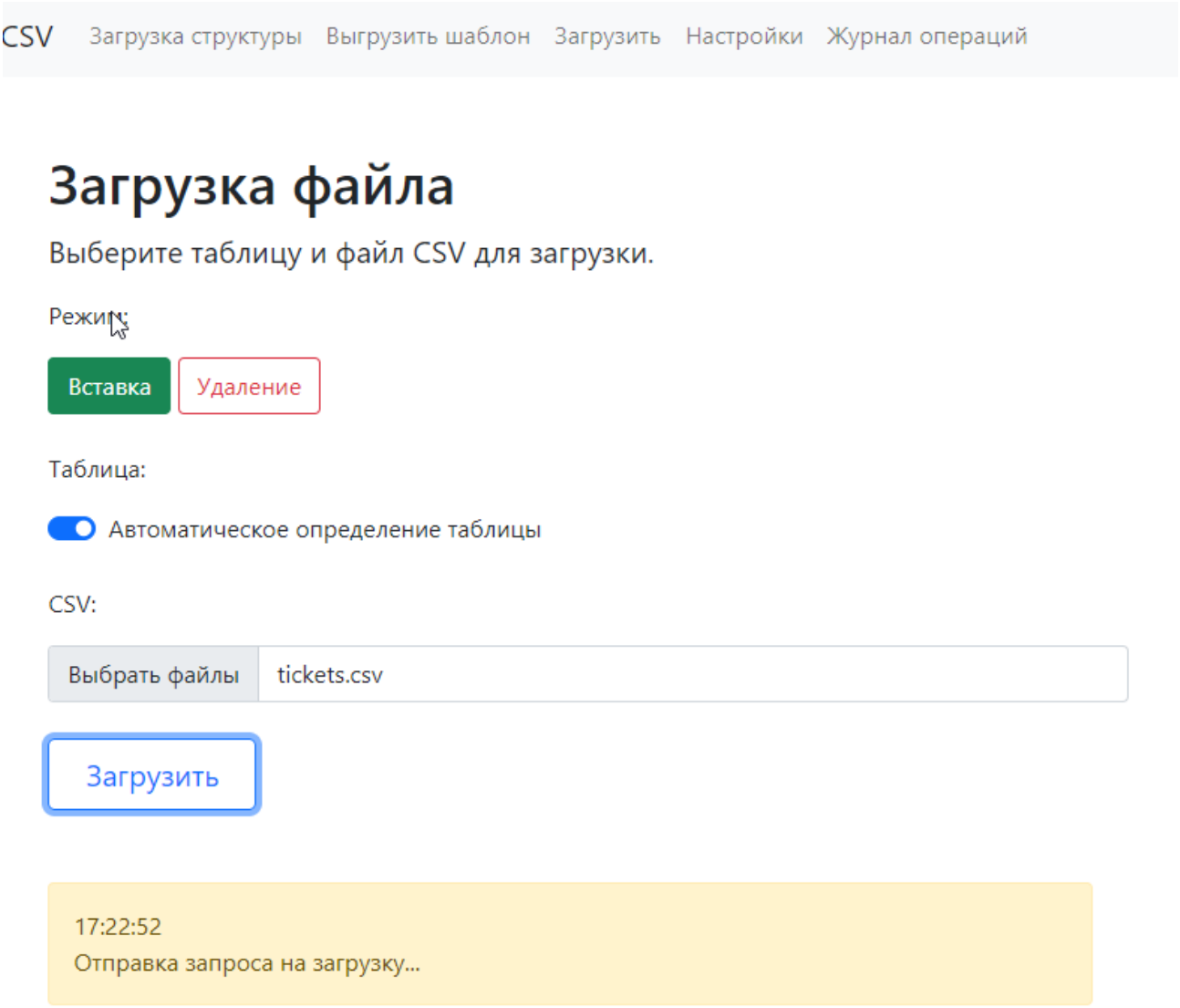

Рисунок - 8.38 Отправка запроса на загрузку

В случае успешной загрузки файла отобразится сообщение **Операция выполнена успешно** (см. `test_upload_data_9`).

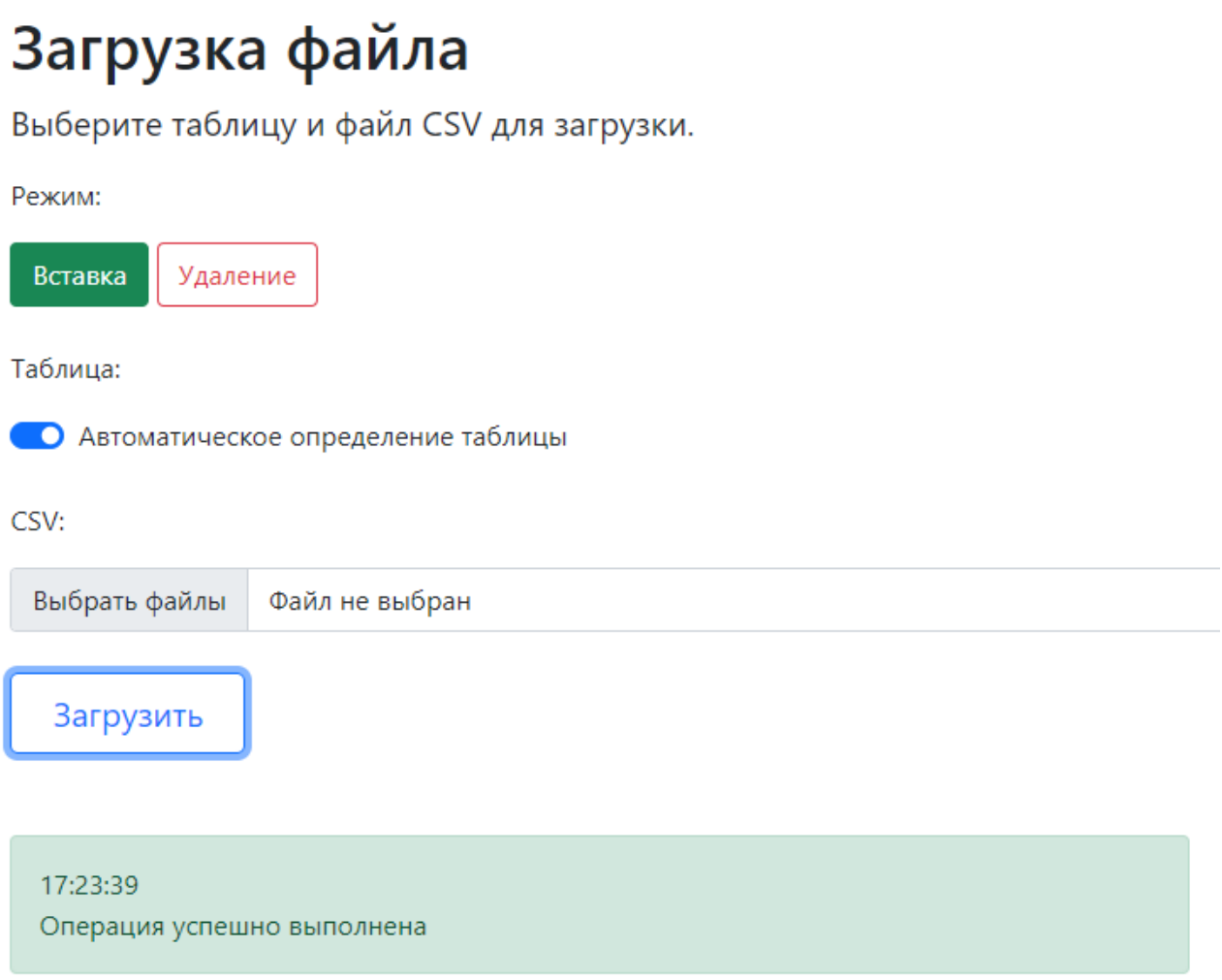

Рисунок - 8.39 Сообщение «Операция выполнена успешно»

7. Выполните проверку загрузки данных. Для этого последовательно выполните следующие запросы к БД:

Проверка таблицы `passenger`

```
SELECT COUNT(*) FROM test_upload_data.passenger;
```

SQL-запрос вернул нулевое значении т.к загрузки данных в эту таблицу не было (см. `test_upload_data_10`).

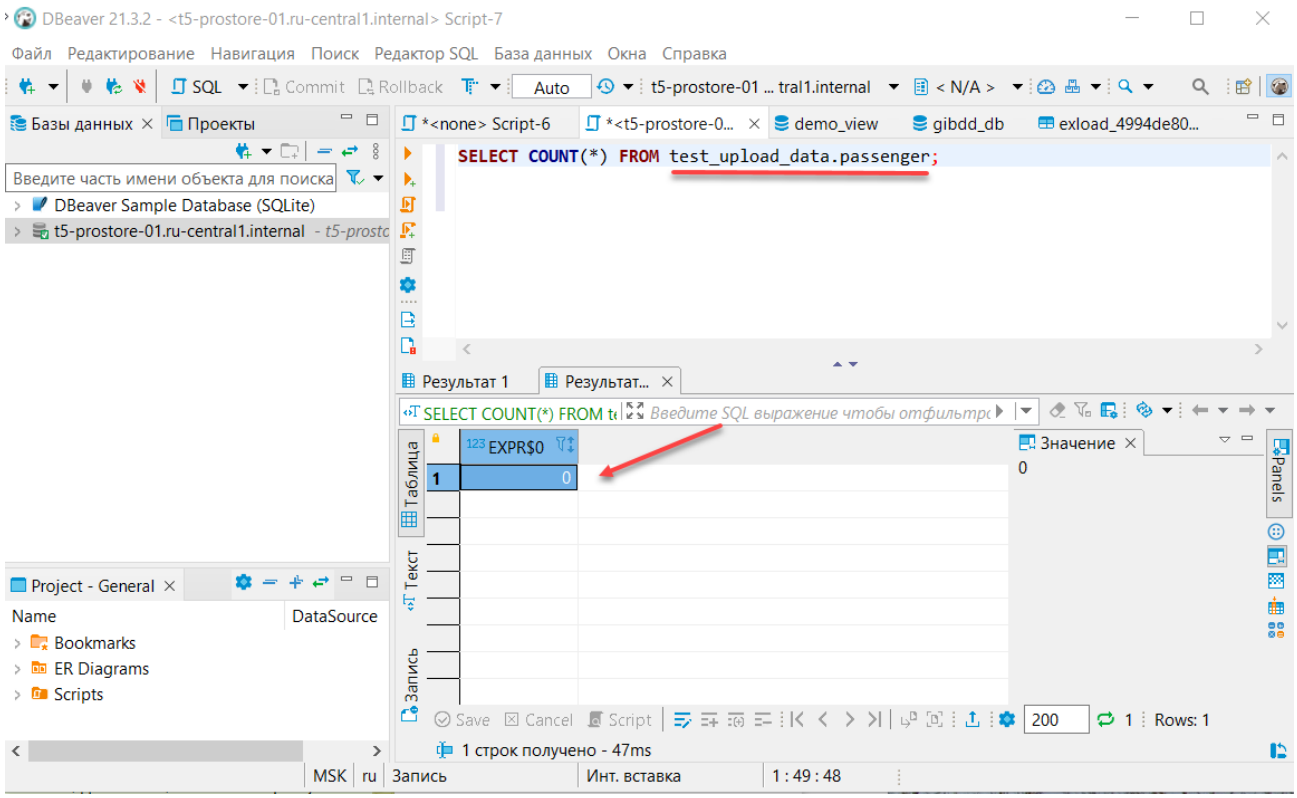

Рисунок - 8.40 Проверка таблицы *passenger*

Проверка таблицы `ticket`

```
SELECT COUNT(*) FROM test_upload_data.ticket;
```

SQL-запрос вернул значение `5000` (см. `test_upload_data_11`).

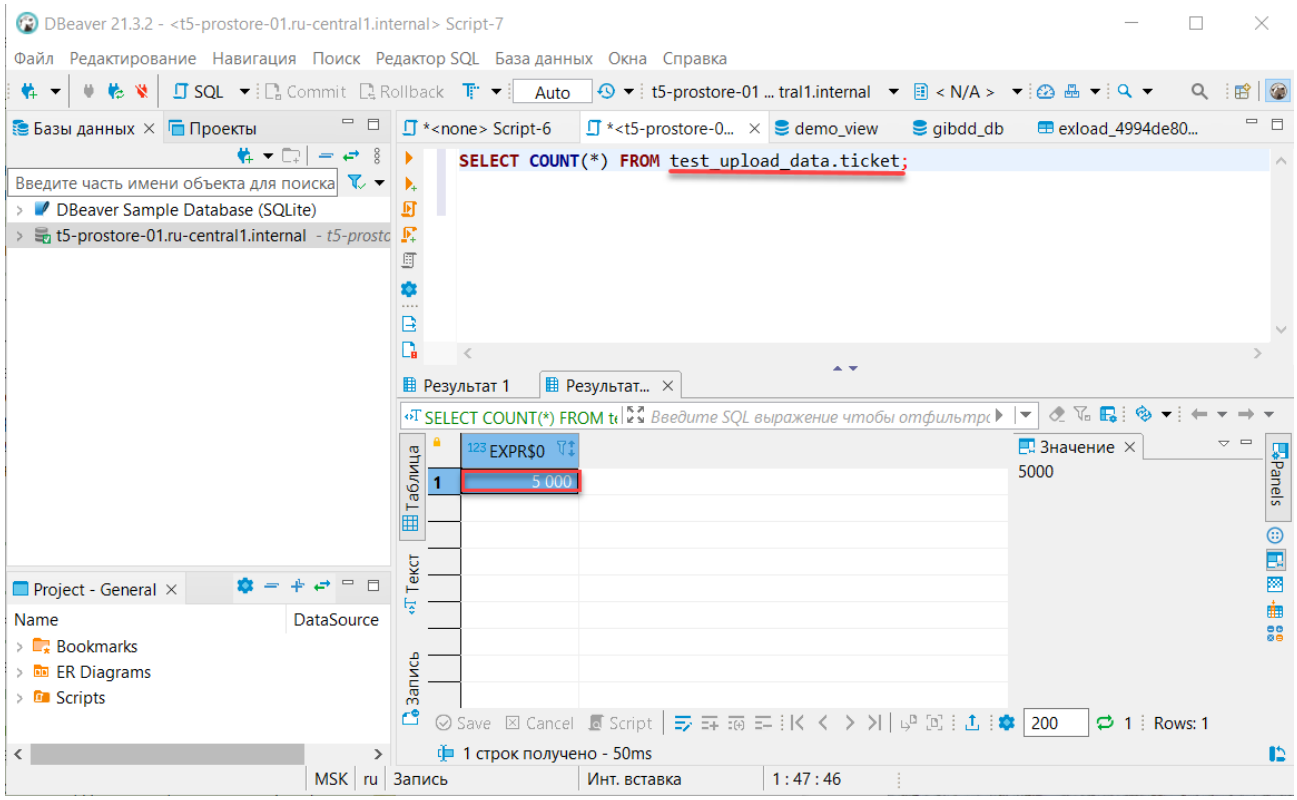

Рисунок - 8.41 Проверка таблицы *ticket*

Проверка таблицы `trip`

```
SELECT COUNT(*) FROM test_upload_data.trip;
```

SQL-запрос вернул значение `500` (см. `test_upload_data_12`).

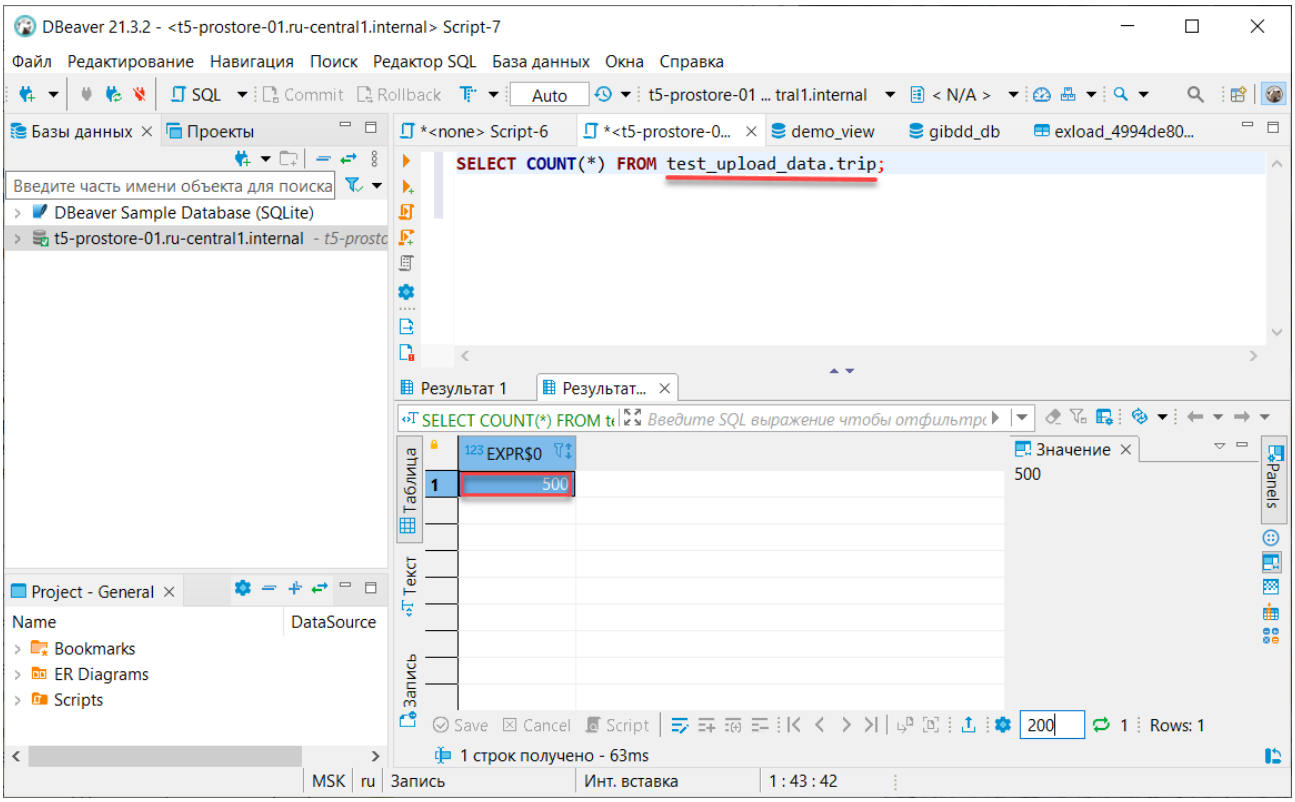

Рисунок - 8.42 Проверка таблицы *trip*

С дополнительной информацией по работе с программой **Dbeaver** (настройка, выполнение запросов, просмотр таблиц и т.д.) можно ознакомиться на официальном сайте разработчика программы https://dbeaver.io/.

ТЕРМИНЫ И ОПРЕДЕЛЕНИЯ

ADCM

Arenadata Cluster Manager (ADCM) - Универсальный оркестратор гибридного ландшафта. Он позволяет быстро устанавливать, настраивать все data-сервисы компании и управлять ими. Наиболее ярко преимущества ADCM раскрываются при работе с гетерогенной инфраструктурой, при которой появляется возможность размещать data-сервисы на различных типах инфраструктур: в облаке, on-premise или в качестве PaaS-сервисов.

ADS

Arenadata Streaming (ADS) - Масштабируемая отказоустойчивая система для потоковой обработки данных в режиме реального времени на базе Apache Kafka и Apache Nifi.

Airflow

открытое программное обеспечение для создания, выполнения, мониторинга и оркестровки потоков операций по обработке данных.

Apache

Организация-фонд, способствующая развитию проектов программного обеспечения Apache.

Apache Airflow

Платформа для программного создания, планирования и мониторинга рабочих процессов.

Apache Avro

Линейно-ориентированный (строчный) формат передачи наборов данных, используемый в качестве платформы сериализации, разрабатываемый в рамках фонда Apache.

Apache Hadoop

Свободно распространяемый набор утилит, библиотек и фреймворк для разработки и выполнения распределённых программ, работающих на кластерах из сотен и тысяч узлов.

Apache Kafka

Распределённый программный брокер сообщений, проект с открытым исходным кодом, разрабатываемый в рамках фонда Apache.

Apache Spark

Фреймворк с открытым исходным кодом для реализации распределённой обработки неструктурированных и слабоструктурированных данных.

API

Application programming interface (англ.) - Программный интерфейс приложения, описание сервисов взаимодействия компьютерной программы с другими программами.

BLOB-адаптер

Информационно-технологический компонент Витрины, обеспечивающий чтение бинарных файлов из **Хранилища BLOB-объектов ведомства**.

ClickHouse

Колоночная аналитическая СУБД с открытым кодом, которая позволяет выполнять аналитические запросы в режиме реального времени на структурированных больших данных, разрабатывается компанией Яндекс.

Counter-Provider

Сервис генерации уникального номера.

CSV

Comma-Separated Values (англ.) - текстовый формат, предназначенный для представления табличных данных.

CSV-extractor

Специализированное программное обеспечение, которое извлекает данные из csv-файлов в собственную БД-хранилища сервиса **Tarantool**.

CSV-Uploader

Программный модуль Витрины данных, который предназначен для загрузки csv-файлов в Витрину данных.

DAG

Файл, содержащий блок данных.

DATA-uploader

Модуль исполнения асинхронных заданий.

DBeaver

Клиентское приложение для управления базами данных (БД), которое использует программный интерфейс **JDBC** для взаимодействия с реляционными БД через драйвер **JDBC**.

DDL

Data definition language (англ.) - семейство компьютерных языков, используемых в компьютерных программах для описания структуры баз данных.

DNS

Domain Name System «система доменных имён» - компьютерная распределённая система для получения информации о доменах. Чаще всего используется для получения IP-адреса по имени хоста (компьютера или устройства), получения информации о маршрутизации почты и/или обслуживающих узлах для протоколов в домене.

Docker

Программное обеспечение для автоматизации развёртывания и управления приложениями в средах с поддержкой контейнеризации, контейнеризатор приложений.

Docker Compose

Платформа контейнеризации, предназначена для конфигурирования многоконтейнерных приложений. В Docker Compose можно управлять несколькими контейнерами **Docker**.

Endpoint

Шлюз (в переводе с англ. — конечная точка), который соединяет серверные процессы приложения с внешним интерфейсом. Простыми словами, это адрес, на который отправляются сообщения (работает с API).

ETL

Extract, transform, load (англ.) - решение, используемое при выгрузке данных из различных источников ведомств и дальнейшего хранения их в Витрине **ProStore** для чтения, использования и взаимодействия с другими ведомствами.

FileZilla

FTP-клиент.

Grafana

Веб-приложение для аналитики и интерактивной визуализации показателей мониторинга с открытым исходным кодом.

Greenplum

Массово-параллельная СУБД для хранилищ данных на основе PostgreSQL.

HikariCP

Hikari Connection Pool.

HTTP

HyperText Transfer Protocol (англ.) - протокол прикладного уровня передачи данных, в настоящий момент используется для передачи произвольных данных.

IAM

Сервисы управления идентификацией и контролем доступа (Identity&AccessManagement).

JDBC

Java DataBase connectivity (англ.) - платформенно-независимый промышленный стандарт взаимодействия Java-приложений с различными СУБД.

JDBC-драйвер

Библиотека классов, реализующая стандарт JDBC и подключения к источнику данных с использованием специализированного протокола, поддерживаемого источником данных.

JDBC-extractor

Специализированное программное обеспечение, которое извлекает данные из jdbc-источника (ведомства) в собственную БД-хранилища сервиса (**Tarantool**).

JSON

JavaScript Object Notation - Общий формат для представления значений и объектов в соответствии со стандартом RFC 4627.

Kafka-loader

Специализированное программное обеспечение, которое загружает данные, извлеченные и приведенные в соответствие логической структуре данных Витрины, собственно в Витрину.

Loki

Приложение для агрегирования log-файлов, используется совместно с **Prometheus**.

MD5

128-битный алгоритм хеширования. Предназначен для создания «отпечатков» или дайджестов сообщения произвольной длины и последующей проверки их подлинности.

MPP

Массово-параллельная архитектура (*англ. massive parallel processing*, MPP, также «массивно-параллельная архитектура»).

NTP

Network Time Protocol — сетевой протокол для синхронизации внутренних часов компьютера с использованием сетей с переменной латентностью.

OpenAPI

The OpenAPI Specification (англ.) – Формализованная спецификация и экосистема множества инструментов, предоставляющая интерфейс между front-end системами, кодом библиотек низкого уровня и коммерческими решениями в виде API.

ProStore

Интеграционная система, обеспечивающая единый интерфейс к хранилищу разнородных данных. Определяет структуры данных, запись и чтение данных Витрины. Позволяет работать со входящими в состав хранилища СУБД одинаковым образом, используя единый синтаксис запросов SQL и единую логическую схему данных.

Prostore

Ядро интеграционной системы ProStore, сервис исполнения запросов.

Prometheus

Программное приложение, используемое для мониторинга событий и оповещения, которое записывает метрики в реальном времени в базу данных временных рядов, построенную с использованием модели HTTP-запроса, с гибкими запросами и оповещениями в режиме реального времени.

Proxy API

Проксирование запросов через Datamart Studio к инсталляциям приложений Витрин данных.

PSQL

Терминальный клиент для работы с PostgreSQL.

PuTTY

Свободно распространяемый клиент для различных протоколов удалённого доступа, включая SSH, Telnet, rlogin.

PXF

Фреймворк, позволяющий **ADB** (Greenplum) параллельно обмениваться данными со сторонними системами.

REST

Representational state transfer (англ.) – архитектурный стиль взаимодействия компонентов распределенного приложения в сети.

REST-адаптер

Сервис, реализующий публикацию конечных точек API для обработки запросов с использованием спецификации OpenAPI версии 3. Используется для сохранения обратной совместимости получения данных из ведомства по REST.

REST API

Набор правил, по которым различные программы могут взаимодействовать между собой и обмениваться данными с помощью протокола HTTP.

REST-Uploader

Модуль асинхронной загрузки данных из сторонних источников.

SOAP

(от англ. Simple Object Access Protocol — простой протокол доступа к объектам) — протокол обмена структурированными сообщениями в распределённой вычислительной среде.

SQL

Structured query language (англ.) – язык структурированных запросов. Декларативный язык программирования, применяемый для создания, модификации и управления данными в реляционной базе данных.

SQL-запрос

Запрос к Витрине данных Поставщика. Произвольный или регламентированный запрос к данным, сформулированный на языке SQL.

SSH

Secure Shell (англ.) – «безопасная оболочка». Сетевой протокол прикладного уровня, позволяющий производить удалённое управление операционной системой и туннелирование TCP-соединений.

Tarantool

Платформа in-memory вычислений с гибкой схемой данных для создания высоконагруженных приложений. Включает в себя базу данных и сервер приложений на Lua.

UDP

Протокол передачи данных. С UDP компьютерные приложения могут посылать сообщения другим хостам по IP-сети без необходимости предварительного сообщения для установки специальных каналов передачи или путей данных.

URI

Унифицированный идентификатор ресурса. URI — последовательность символов, идентифицирующая абстрактный или физический ресурс.

UUID

Стандарт идентификации, используемый в создании программного обеспечения, стандартизированный Open Software Foundation как часть DCE — среды распределённых вычислений. Основное назначение UUID — это позволить распределённым системам уникально идентифицировать информацию без центра координации.

Vert.x

Библиотека для разработки асинхронных приложений, основанная на событиях.

VipNet

программное обеспечение (далее - ПО) для защиты сетевого трафика на рабочих местах пользователей.

XML

eXtensibe Markup Language (англ.) – универсальный текстовый формат для хранения и передачи структурированных данных.

XML-extractor

Специализированное программное обеспечение, для копирования данных из xml-файлов в собственную

БД-хранилища сервиса (**Tarantool**).

ZooKeeper

Сервер с открытым исходным кодом для высоконадежной распределенной координации облачных приложений.

Агент СМЭВ4 (Агент)

Типовое программное обеспечение, устанавливаемое в контуре ИС УВ и обеспечивающее сопряжение Витрин данных и ИС УВ с Ядром ПОДД СМЭВ.

База данных

Совокупность данных, хранимых в соответствии со схемой данных, манипулирование которыми выполняют в соответствии с правилами средств моделирования данных.

(Большой) Двоичный объект (BLOB / БЛОБ)

Тип данных, значение которого представляет собой массив байт, размер которого существенно превышает размер базовых скалярных типов (int, float, double, date)

Брокер сообщений

Архитектурный паттерн в распределённых системах; приложение, которое преобразует сообщение по одному протоколу от приложения-источника в сообщение протокола приложения-приёмника, тем самым выступая между ними посредником.

Витрина данных

Комплекс программных и технических средств в составе информационно-телекоммуникационной инфраструктуры Участника взаимодействия, обеспечивающий хранение и предоставление данных другим Участникам взаимодействия с использованием ПОДД СМЭВ.

Вид сведения СМЭВ (ВС)

Комплекс документальных и программных компонентов, зарегистрированный в СМЭВ 3.x, обеспечивающий взаимодействие ИС ведомств в определённом формате и по определённым правилам.

ГОСТ

Нормативно-правовой документ, в соответствии требованиями которого производится стандартизация производственных процессов.

Дельта

Логически целостная совокупность изменений информации об объектах. Каждой дельте поставлено в соответствие целое число из монотонно возрастающей последовательности целых чисел начиная с 0, отражающее ее место в общей последовательности дельт и дата-время ее исполнения.

ЕИП

Единая информационная платформа.

ИС

Информационная система.

ИС УВ

Информационная система Участника взаимодействия.

КриптоПро

Разработанная одноименной компанией линейка криптографических утилит (вспомогательных программ) — так называемых криптопровайдеров. Они используются в других программах для генерации электронной подписи (ЭП), работы с сертификатами, организации структуры PKI и т.д.

ЛК УВ

Личный кабинет участника взаимодействия. Система, предназначенная для управления информационными системами и мониторинга информационных обменов в СМЭВ 3 и СМЭВ 4 участниками взаимодействия.

Логическая модель данных

Схема базы данных, выраженная в понятиях бизнес-требований.

Мнемоника Витрины

Уникальное строковое значение, определяющее модель данных Витрины.

Модель данных Витрины

Описание структуры Витрины (общая информация, перечень сущностей, атрибутный состав), загруженное в Ядро ПОДД СМЭВ.

Набор данных

Совокупность систематизированных данных (датасетов), представляющих собой базовый элемент для работы с данными.

НСУД

Национальная система управления данными.

ОГРН

Основной государственный регистрационный номер, присваивается юридическим лицам сразу же после регистрации в ФНС РФ.

Параметр запроса

Символическое имя, входящее в текст SQL-запроса и не содержащееся в Модели данных Витрины, в терминах которой сформулирован SQL-запрос.

ПО

Программное обеспечение.

ПОДД

Подсистемы обеспечения доступа к данным.

ПОДД-адаптер

Программно-технический продукт, обеспечивающий взаимодействие витрины и ПОДД СМЭВ.

ПОДД-адаптер - Модуль исполнения запросов

Логический модуль ПОДД-адаптера, предназначен для исполнения запросов ПОДД СМЭВ (через протокол коммуникации Агент ПОДД).

ПОДД-адаптер - Модуль MPPR

Логический модуль ПОДД-адаптера, предназначен для чтения данных в многопоточном режиме (massively parallel processing, **MPP**).

ПОДД-адаптер - Модуль MPPW

Логический модуль ПОДД-адаптера выполняет загрузку данных в многопоточном режиме.

Подписка (потребителя)

Предоставление права Потребителю данных ПОДД СМЭВ на информационный обмен с использованием Регламентированного запроса типа «Рассылка».

Поставщик данных

Участник взаимодействия, являющийся источником данных для других участников и использующий ПОДД СМЭВ для передачи данных.

Потребитель данных

Участник взаимодействия, получающий данные от Поставщиков данных для дальнейшей их обработки и использующий для передачи запросов и получения данных ПОДД СМЭВ.

Распределенный запрос

Регламентированный запрос, инициированный Потребителем, SQL-выражение которого содержит наборы данных из двух или более Витрин данных.

Регламентированный SQL-запрос (РЗ)

SQL-запрос, выраженный в терминах Модели данных, загруженной в ПОДД, и зарегистрированный в Ядре ПОДД под символической мнемоникой, используемой ИС Потребителя ПОДД для выполнения регламентированного запроса. Может иметь параметры, значения которых задаются Потребителем данных ПОДД при выполнении регламентированного запроса.

Реплика

СУБД, хранящая реплицируемые наборы данных, полученные от Поставщика данных.

Сервис Формирования документов

Модуль витрины, предназначенный для работы с формируемыми документами.

СМЭВ

Система межведомственного электронного взаимодействия.

СМЭВ 3

Единая система межведомственного электронного взаимодействия, функционирующая в соответствии с Методическими рекомендациям по работе со СМЭВ версии `3.x`.

СМЭВ3-адаптер

Информационно-технологический компонент СМЭВ, устанавливается на стороне Участника взаимодействия. СМЭВ3-адаптер обеспечивает информационное взаимодействие через единый электронный сервис единой системы межведомственного электронного взаимодействия (**СМЭВ**).

Сообщение

Сведения в виде законченного блока данных, передаваемые при функционировании информационной системы.

СУБД

Система управления базами данных.

Табличный параметр (запроса)

Параметр, значение которого представляет собой двумерный массив с именованными колонками и неупорядоченными строками. Формальный табличный параметр может использоваться в инструкциях `FROM`, `JOIN` как источник данных.

Токен

Ключ безопасности (Цифровой сертификат).

Участник взаимодействия

Орган или организация, участвующий в информационном обмене через СМЭВ.

ФЛК

Форматно-логический контроль загружаемых в Витрину данных.

Хранилище BLOB-объектов

Место для хранения BLOB-объектов (бинарных данных). Располагается на стороне ведомства и не является частью Витрины данных. Взаимодействие с Хранилищем BLOB-объектов осуществляется через **BLOB-адаптер**.

Хранилище S3 (объектное хранилище S3)

Хранилище бинарных объектов, позволяющее хранить файлы любого типа и объема. Доступ к хранилищу предоставляется через API.

Чанк

Фрагмент результирующих данных оптимального для передачи по сети размера.