

# VisionLabs LUNA Access

Руководство администратора

2.17.0

## Содержание

<b>1</b>	<b>Глоссарий</b>	<b>4</b>
<b>2</b>	<b>Введение</b>	<b>5</b>
<b>3</b>	<b>Системные требования</b>	<b>6</b>
<b>4</b>	<b>Архитектура Access</b>	<b>7</b>
4.1	Общая схема работы . . . . .	7
4.2	Диаграммы последовательности работы Access . . . . .	8
4.2.1	Взаимодействие внутренних компонентов Access . . . . .	8
4.2.2	Создание компонента . . . . .	10
4.2.3	Автоматический мониторинг . . . . .	12
<b>5</b>	<b>Лицензирование</b>	<b>13</b>
<b>6</b>	<b>Установка Access</b>	<b>14</b>
6.1	Подготовка к установке . . . . .	14
6.2	Установка Docker и Docker Compose . . . . .	14
6.3	Подготовка окружения и распаковка дистрибутива . . . . .	14
6.4	Настройка Access . . . . .	15
6.5	Запуск Access . . . . .	18
<b>7</b>	<b>Управление учетными записями</b>	<b>20</b>
7.1	Добавление учетной записи . . . . .	20
7.2	Просмотр списка учетных записей . . . . .	20
7.3	Удаление учетной записи . . . . .	20
<b>8</b>	<b>Скрипты</b>	<b>22</b>
8.1	Скрипт загрузки образов . . . . .	22
<b>9</b>	<b>Обновление Access</b>	<b>23</b>
<b>10</b>	<b>Удаление Access</b>	<b>25</b>
<b>11</b>	<b>Система логирования</b>	<b>26</b>
11.1	Инструкция по сбору логов . . . . .	26
11.1.1	Файл настроек . . . . .	26
11.1.2	Логи контейнеров . . . . .	27
11.1.3	Файл .env . . . . .	28
11.1.4	Информация о рабочем окружении . . . . .	28
11.1.5	Скриншоты UI . . . . .	29

11.2	Отслеживание проходов с помощью trace_id . . . . .	29
11.2.1	Отслеживание проходов через события Luna Platform . . . . .	30
11.2.2	Как найти trace_id по ФИО персоны . . . . .	31
11.2.3	Поиск события в Luna Platform по trace_id . . . . .	32

## 1. Глоссарий

Термин	Определение
Docker	Платформа для разработки, доставки и запуска контейнерных приложений. Позволяет создавать контейнеры, автоматизировать их запуск и развертывание, управляет жизненным циклом. Позволяет запускать множество контейнеров на одной хост-машине
Docker Compose	Позволяет разворачивать и настраивать несколько контейнеров одновременно
Идентификация	Процедура определения субъекта биометрических данных путем сравнения биометрических признаков, полученных от субъекта биометрических данных, со всеми эталонными биометрическими признаками (контрольными шаблонами), хранящимися в биометрической базе данных
Контейнер	Способ упаковать приложение и все его зависимости в единый образ. Этот образ запускается в изолированной среде, не влияющей на основную операционную систему. Контейнеры позволяют отделить приложение от инфраструктуры
Образ	Неизменяемый образ, из которого разворачивается контейнер. Набор файлов, необходимых для запуска и работы приложения на другом хосте
Программное обеспечение (ПО)	Программа или множество программ, используемых для управления компьютером
Система контроля управления доступом (СКУД)	Совокупность программно-аппаратных технических средств, направленных на контроль входа и выхода в помещение с целью обеспечения безопасности и регулирования посещения определенного объекта. Например, турникеты на входе в банки/офисные здания

## 2. Введение

Документ описывает процесс установки и настройки сервиса VisionLabs LUNA Access версии 2.17.0 (далее — Access), а также содержит аппаратные и программные требования к ПО.

Процесс настройки и установки необходимо выполнять под учетной записью суперпользователя (с root правами).

### 3. Системные требования

Требования к программному обеспечению (Таблица 1).

**Таблица 1.** Требования к программному обеспечению

Ресурс	Рекомендовано
Операционная система (ОС)	ОС, поддерживающие Docker (CentOS, RedOS и др.)
Docker	v.20 и новее
Docker-compose	v.1.21 и новее
unzip	v.6.0-47

Требования к аппаратному обеспечению рабочей станции (Таблица 2).

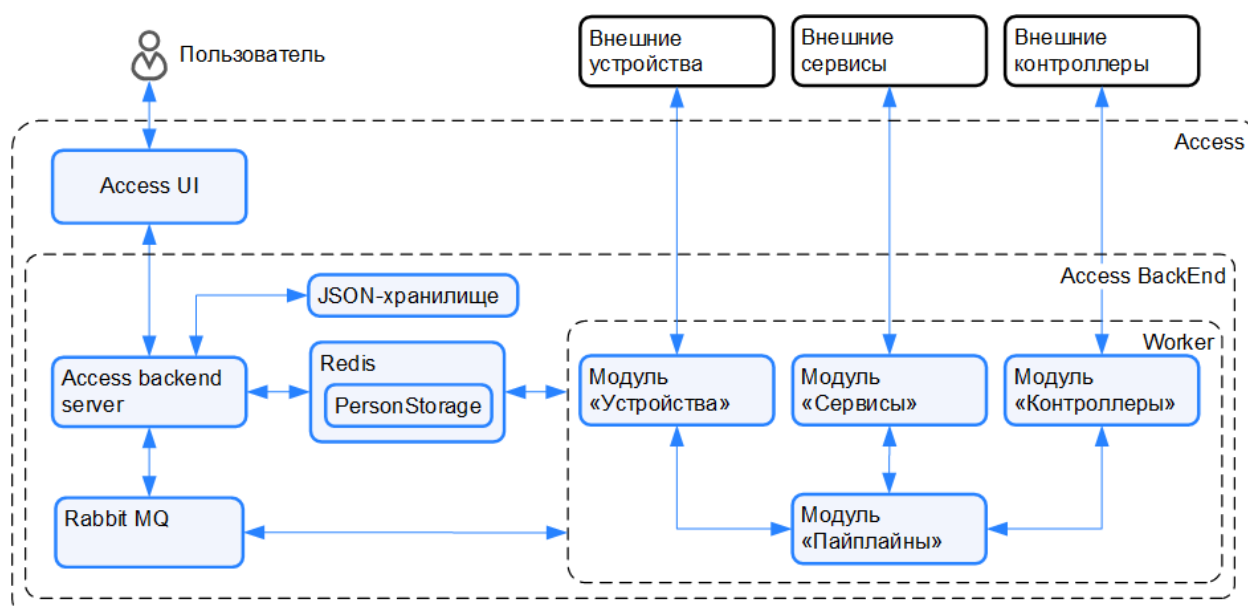
**Таблица 2.** Требования к аппаратному обеспечению

Ресурс	Рекомендовано
Процессор (CPU)	Intel/AMD x64 2,0 hHz
Оперативная память (RAM)	4 ГБ и выше
Свободное место на диске (HDD/SSD)	20 ГБ и выше

## 4. Архитектура Access

### 4.1. Общая схема работы

Схема работы Access и описание представлены ниже (Рисунок 1) и (Таблица 3).



**Рис. 1:** Схема работы Access

**Таблица 3.** Описание схемы Access.

Компонент	Описание
Пользователи	Пользователь Access, занимающийся настройкой компонентов и отслеживанием их работы.
Access	Совокупность программных средств управления, позволяющих реализовать совместную работу продуктов VisionLabs и различных систем контроля и управления доступом (СКУД).
Access UI	Графический интерфейс Access.
Access Backend	Backend Access, отвечающий за работу с внешними компонентами, взаимодействием с UI.
Access backend server	Набор библиотек, объединяющий модули Access и UI.
RabbitMQ	Брокер сообщений, обеспечивающий передачу событий между Access backend server и Worker.
Redis	Система хранения данных в виде структур для обеспечения работоспособности Access.

Компонент	Описание
PersonStorage	Локальное хранилище, хранящее связь между персонами СКУД и их биометрическими данными. Используется для синхронизации данных между системами контроля доступа и биометрическими системами (LP, КБС) или внешними сервисами.
JSON-хранилище	Файл <code>settings.json</code> для хранения настроек интеграций и учетных записей пользователей.
Worker	Сервис, обеспечивающий выполнение задач модулей в фоновом режиме.
Модуль «Пайплайн»	Модуль Access, обеспечивающий основную логику и взаимодействие модулей.
Модуль «Устройства»	Модуль Access, содержащий библиотеки для подключения внешних устройств к Access.
Модуль «Сервисы»	Модуль Access, содержащий библиотеки для подключения внешних сервисов к Access.
Модуль «Контроллеры»	Модуль Access, содержащий библиотеки для работы с внешними контроллерами и преобразователями.
Внешние устройства	Подключаемые камеры, терминалы и тепловизоры, передающие видеопоток для дальнейшей обработки. Полный список доступных устройств см. в Руководстве пользователя.
Внешние сервисы	СКУД, КБС или продукты VisionLabs, которые могут быть использованы в интеграции. Полный список доступных СКУД и продуктов VisionLabs см. в Руководстве пользователя.
Внешние контроллеры	Внешние контроллеры (например, преобразователи для контроллеров СКУД), используемые в интеграциях.

## 4.2. Диаграммы последовательности работы Access

### 4.2.1. Взаимодействие внутренних компонентов Access

Взаимодействие и описание компонентов Access на примере типовой интеграции источник видеосигнала + LUNA PLATFORM 5 (LP5) + Sigur (Рисунок 2) и (Таблица 4).



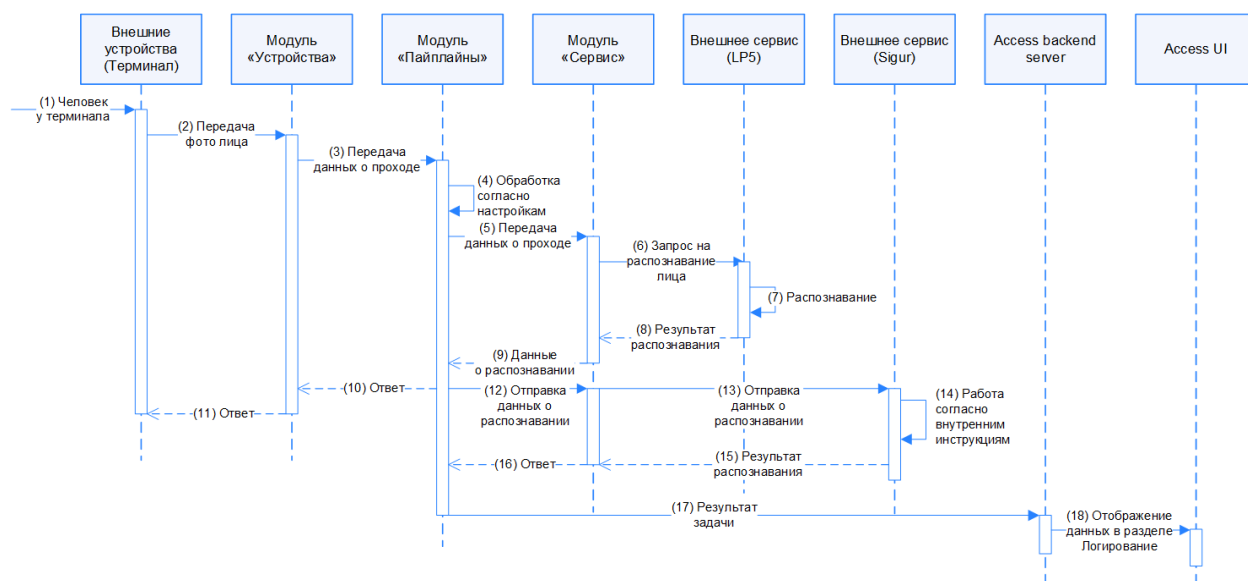


Рис. 2: Взаимодействие компонентов

Таблица 4. Описание диаграммы последовательности

Шаг	Описание
(1)	Человек подошел к турникету с целью прохода, лицо человека зафиксировал терминал.
(2)	Терминал передал фото человека в модуль «Устройства»
(3)	Модуль устройства передает в модуль Пайплайны данные о лице для дальнейшей обработки.
(4)	Модуль Пайплайн обрабатывает фото согласно внутренним настройкам.
(5)	Модуль Пайплайн передает данные о проходе в модуль Сервис для дальнейшей обработки.
(6)	Модуль Сервис отправляет запрос во внешний сервис (LP5) на идентификацию лица.
(7)	Внешние сервис (LP5) производит идентификацию.
(8)	Внешние сервис (LP5) возвращает ответ о результате распознавания и идентификации лица.
(9)	Модуль Сервисы передает в модуль Пайплайн данные о распознавании.
(10)	Модуль Пайплайны возвращает ответ в Модуль устройство о результате распознавания.

Шаг	Описание
(11)	Модуль Устройства возвращает данные о имени человека у терминала для отображения имени и сообщения об успешной идентификации на экране терминала
(12)	Модуль Пайплайны передает данные о распознавании в модуль Сервисы.
(13)	Модуль Сервисы передает данные о распознавании во Внешние сервис (Sigur)
(14)	Внешний сервис (Sigur) выполняет обработку результата распознавания согласно внутренним инструкциям. В случае успешного распознавания отправляет сигнал на открытие турникета.
(15)	Внешний сервис (Sigur) возвращает результат обработки в Модуль Сервисы.
(16)	Модуль Сервисы передает в модуль Пайплайн данные об обработке.
(17)	Модуль Пайплайны передает данные в Access backend server с помощью RabbitMQ.
(18)	Access backend server отправляет в Access UI данные для отображения результата прохода в разделе Логирование.

#### 4.2.2. Создание компонента

Схема создания компонента в Access UI и описание (Рисунок 3) и (Таблица 5).

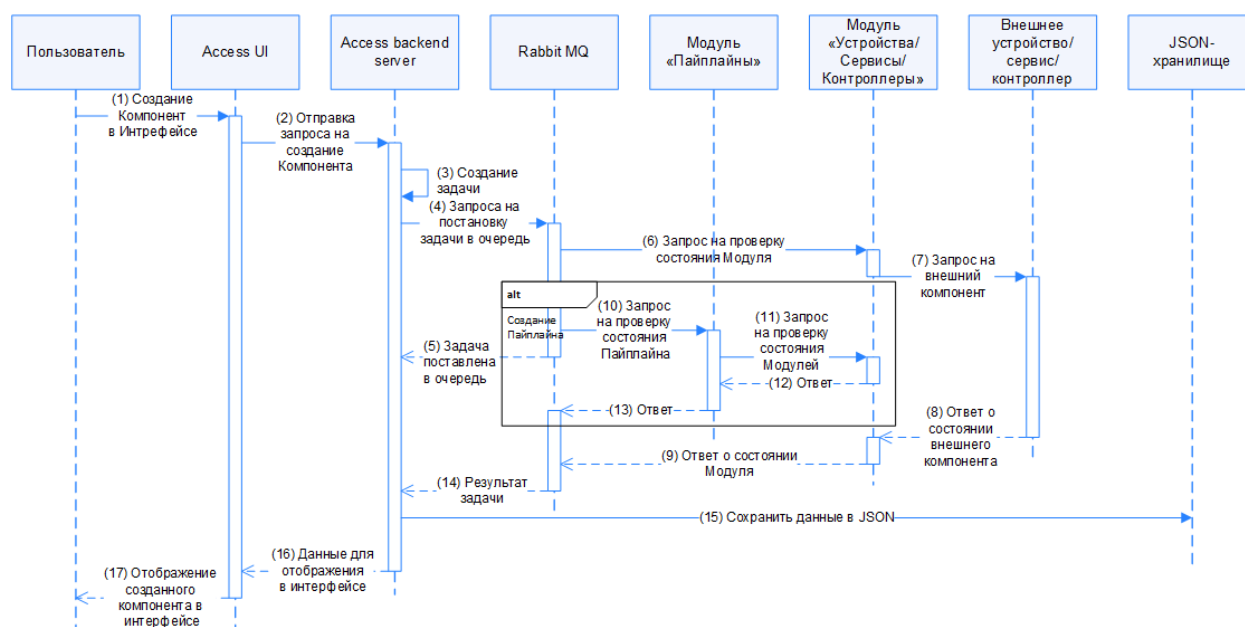


Рис. 3: Взаимодействие компонентов

**Таблица 5.** Описание диаграммы последовательности

Шаг	Описание
(1)	Пользователь заполняет данные нового объекта (устройства, сервиса, пайплайна или контроллера) в Access UI.
(2)	Access UI отправляет запрос на Access backend server на создание компонента.
(3)	Access backend server создает компонента и формирует задачу на проверку состояния подключенного компонента.
(4)	Access backend server отправляет задачу в RabbitMQ для постановки задачи в очередь.
(5)	RabbitMQ возвращает ответ о постановки задачи в очередь.
(6)	RabbitMQ отправляет запрос на проверку состояния компонента в соответствующий модуль (Устройства/Сервисы/Контроллеры), указанного в настройках компонента.
(7)	Модуль Устройства/Сервисы/Контроллеры перенаправляет запрос на проверку состояния во внешние устройство/сервис/контроллер.
(8)	Внешние устройство/сервис/контроллер возвращает ответ о состоянии активности.
(9)	Модуль Устройства/Сервисы/Контроллеры возвращает ответ о состоянии внешнего устройство/сервис/контроллер.
(10)	(Alt создание пайплайна) RabbitMQ отправляет запрос в Модуль Пайплайн на проверку состояния компонентов, указанных в настройках пайплайна.
(11)	(Alt создание пайплайна) Модуль Пайплайн перенаправляет запрос на проверку состояния в модули Устройства/Сервисы/Контроллеры, которые указаны в настройках пайплайна.
(12)	(Alt создание пайплайна) Модули Устройства/Сервисы/Контроллеры возвращают ответ о доступности компонентов.
(13)	(Alt создание пайплайна) Модуль Пайплайны возвращает ответ о состоянии.
(14)	RabbitMQ возвращает результат задачи в Access backend server.
(15)	Access backend server сохраняет данные в JSON-хранилище.
(16)	Access backend server отправляет в Access UI данные для отображения результата создания компонента.
(17)	Access UI отображает созданные компонент с активным статусом для пользователя.

### 4.2.3. Автоматический мониторинг

Схема автоматического мониторинга состояния компонентов и описание (Рисунок 4) и (Таблица 6).

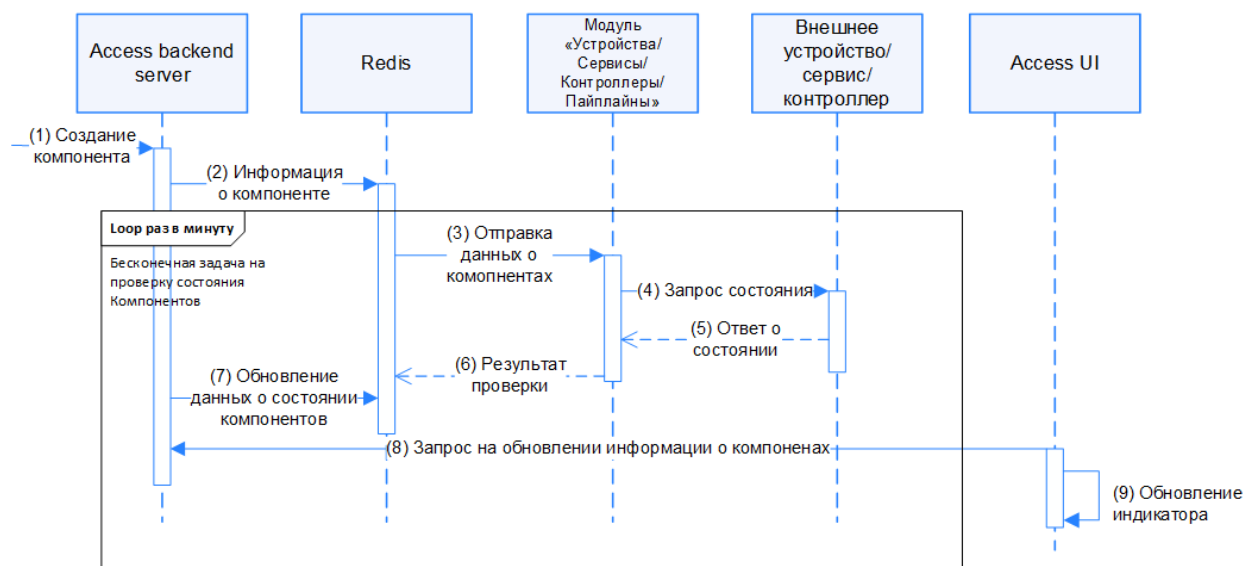


Рис. 4: Взаимодействие компонентов

Таблица 6. Описание диаграммы последовательности

Шаг	Описание
(1)	Входной точкой для этого процесса является создание любого компонента.
(2)	Access backend server передает данные о созданном компоненте в Redis.
(3)	Redis передает данные о созданном компоненте в Worker, который каждую минуту отправляет запрос в подключенные Модули для проверки состояния внешних устройств.
(4)	Модули перенаправляют запрос на проверку подключения к внешним компонентам.
(5)	Внешние компоненты возвращают ответ о состоянии подключения, если ответ на запрос не был получен, то модуль считает подключенное устройство отключенным.
(6)	Модуль записывает данные о состоянии компонента в Redis.
(7)	Access backend server получает данные о состоянии компонентов из Redis.
(8)	Access UI запрашивает статус активности компонента у Access backend server
(9)	Access UI обновляет индикатор статуса активности компонента.

## 5. Лицензирование

Для работы Access не требуется лицензия.

Лицензированию могут быть подвержены внешние системы и сервисы, которые используются в интеграции. Лицензия в этом случае приобретается отдельно у правообладателя.

## 6. Установка Access

Данный раздел описывает установку и использование Docker и Docker Compose для развертывания Access.

Docker и Docker Compose не входят в дистрибутив Access.

### 6.1. Подготовка к установке

Создайте главную директорию, где в дальнейшем будут все версии продукта и перейдите в нее:

```
sudo mkdir /var/lib/vl-access-2/  
sudo chown $(whoami) /var/lib/vl-access-2/  
cd /var/lib/vl-access-2/
```

### 6.2. Установка Docker и Docker Compose

Используйте официальную инструкцию для установки Docker Engine и Docker Compose для используемой ОС.

Перед запуском Access необходимо убедиться что Docker запущен и активен.

1. Запустите Docker:

```
systemctl start docker  
systemctl enable docker
```

2. Проверьте Docker:

```
systemctl status docker
```

Ответ должен содержать статус Active (running).

### 6.3. Подготовка окружения и распаковка дистрибутива

Дистрибутив представляет собой архив вида «vl-access-2-v2.17.0».

Ссылку для скачивания дистрибутива необходимо запросить у представителя VisionLabs.

Архив содержит компоненты, необходимые для установки и эксплуатации Access.

Архив не включает зависимости, которые входят в стандартную поставку ОС и могут быть загружены из открытых источников.

Запуск Access осуществляется из Docker образа.

Для запуска необходимо выполнить действия:

1. Переместите скачанный архив vl-access-2-v2.17.0.zip в директорию /var/lib/vl-access-2.
2. Установите архиватор unzip, если он не установлен. В команде указана версия 6.0-47, более поздние версии утилиты не тестировались:

```
yum install unzip-6.0-47.el8_10
```

3. Распакуйте файлы дистрибутива:

```
unzip vl-access-2-v2.17.0.zip -d vl-access-2-v2.17.0
```

4. Директория содержит:

- директорию docs с документацией в формате pdf/html;
- README\_FOR\_ENGINEERS.md файл с описанием быстрого запуска;
- .env файл конфигурации;
- конфигурацию docker-compose.yml;
- конфигурацию conf.yml;
- архив изменений CHANGELOG.md;
- архив с образом vl-access-2-images-v2.17.0.tar.gz (при наличии).

5. Перейдите в директорию:

```
cd vl-access-2-v2.17.0
```

При работе с дистрибутивом без образа необходимо скачать образ при запуске, см. раздел [Запуск](#)

## 6.4. Настройка Access

Произведите настройку с помощью файла .env (Таблица 7):

```
nano .env
```

**Таблица 7.** Описание параметров env

Параметр	Описание	Значения по умолчанию
<b>Параметры FastAPI и Worker</b>		
DEBUG	Режим отладки Access - вывод в логах ОС и в интерфейсе информации типа Debug о работе Access <ul style="list-style-type: none"> <li>• 1 – отладка.</li> <li>• 0 – без отладки.</li> </ul>	0
VL_ACCESS_TAG	Тэг Access, берется из внутренних настроек. Не рекомендуется изменять этот параметр.	2.17.0
LOG_DB_HOST	Имя хоста для хранения логов.	log-storage
LOG_DB_PORT	Порт для подключения к log-storage.	27017
LOG_DB_NAME	Имя БД логов.	logs
LOG_DB_USER	Имя пользователя БД логов.	username
LOG_DB_PASSWORD	Пароль пользователя БД логов.	password
C_FORCE_ROOT	Принудительный запуск Celery от имени root пользователя.	true
WORKER_CONCURRENCY	Максимальное кол-во процессов у компонента Worker, которые могут обрабатываться параллельно. Задается исходя от нагрузки на систему.	16
WORKERS_AMOUNT	Количество экземпляров worker (необходим для увеличения количества обрабатываемых событий, используется на высоконагруженных объектах).	1
<b>Параметры Redis</b>		
REDIS_HOST	Имя хоста Redis.	redis
REDIS_PORT	Порт, на котором развернут Redis.	6379



REDIS_- DB_BASE	Номер основной базы данных Redis.	0
REDIS_- DB_- PERSONS	Номер базы данных Redis, хранящей информацию о людях.	1
REDIS_- DB_- CELERY_- BEAT	Номер базы данных Redis для хранения информации о периодических задачах сервиса worker-beat.	2
<b>Параметры Rabbit</b>		
RABBITMQ	Название брокера очередей сообщений. Access поддерживает только RabbitMQ.	rabbitmq
RABBITMQ_ DEFAULT_- USER	Логин пользователя для подключения к RabbitMQ внешних систем	guest
RABBITMQ_- DEFAULT_- PASS	Пароль для подключения к RabbitMQ внешних систем	guest
RABBITMQ_ PROTOCOL	Тип протокола RabbitMQ. Поддерживается только AMQP.	amqp
RABBITMQ_- URL	Адрес для подключения к RabbitMQ	<pre> \${ RABBITMQ_PROTOCOL }://\${ RABBITMQ_DEFAULT_USER }: \${ RABBITMQ_DEFAULT_PASS }@\${RABBITMQ }:5672/ </pre>

CELERY_	Адрес для подключения к брокеру Celery	\${
BROKER_		RABBITMQ_PROTOCOL
URL		}://\${
		RABBITMQ_DEFAULT_USER
		}:
		\${
		RABBITMQ_DEFAULT_PASS
		}@\${RABBITMQ
		}:5672/

#### Параметры подключения FrontEnd

BACKEND_	Хост сервиса fastapi. Указать IP адрес, если frontend и	fastapi
HOST	fastapi запущены на разных машинах	
BACKEND_	Порт для подключения к backend Access.	9091
PORT		

## 6.5. Запуск Access

1. Импортируйте образ:

1.1. При наличии образа в архиве

```
docker load -i vl-access-2-images-v2.17.0.tar.gz
```

1.2. Без образа:

```
docker login dockerhub.visionlabs.ru
docker-compose pull
docker logout dockerhub.visionlabs.ru
```

2. Добавьте symlink в директорию `/var/lib/vl-access-2/`, которая ссылается на последнюю версию продукта:

```
ln -s /var/lib/vl-access-2-v2.17.0 /var/lib/vl-access-2/current
```

3. Запустите Access:

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

Контейнеры Access поставляются с предустановленными утилитами, необходимыми для работы с образом.

4. Проверьте доступность Access по адресу: `http://<IP_address>:9092/`.

Для доступа к интерфейсу Access создайте администратора (см. раздел [Управление учетными записями](#)).

## 7. Управление учетными записями

### 7.1. Добавление учетной записи

В Access поддерживается создание учетной записи с ролью Администратора. Более подробно о ролях и доступах см. в Руководстве пользователя раздел Роли в сервисе.

1. Запустите скрипт создания администратора:

```
docker-compose exec fastapi python backend/manage.py createadmin
```

2. Следуйте указаниям консоли:

```
(venv) [root@localhost vl-access-2]# docker-compose exec fastapi python
backend/manage.py createadmin
Admin creation:
Enter your login: admin
Enter your password:
Confirm your password
Admin user created successfully
```

Логин администратора должен быть уникальным

3. Проверьте корректность создания администратора – войдите в созданную учетную запись `http://<IP_address>:9092/`.

### 7.2. Просмотр списка учетных записей

1. Выполните команду просмотра списка созданных администраторов:

```
docker-compose exec fastapi python backend/manage.py listadmin
```

В консоли отобразится список администраторов:

```
(venv) [root@localhost vl-access-2]# docker-compose exec fastapi python
backend/manage.py listadmin
admin
admin2
```

### 7.3. Удаление учетной записи

Получите информацию о логине необходимого аккаунта для удаления с помощью команды [просмотра](#).

1. Выполните команду для удаления администратора:

```
docker-compose exec fastapi python backend/manage.py deleteadmin
```

2. Введите логин администратора.

В ответе на запрос консоль выдаст сообщение об успешном удалении:

```
(venv) [root@localhost vl-access-2]# docker-compose exec fastapi python
backend/manage.py deleteadmin
Admin deletion:
Enter your login: admin2
Admin was deleted successfully.
```

## 8. Скрипты

### 8.1. Скрипт загрузки образов

Скрипт `scripts/save_images.sh` позволяет выгрузить docker образы по путям, указанным в `docker-compose.yml` в формат `tar.gz` (Таблица 8).

Выгрузка может понадобиться для сохранения образов на носитель и перенос на объект при отсутствии/плохом интернет-соединении.

Запуск скрипта:


```
./save_images.sh -f <path_to/docker-compose.yml> -d <local_path_to/
docker_images> -u <dockerhub_username> -p <dockerhub_password>
```

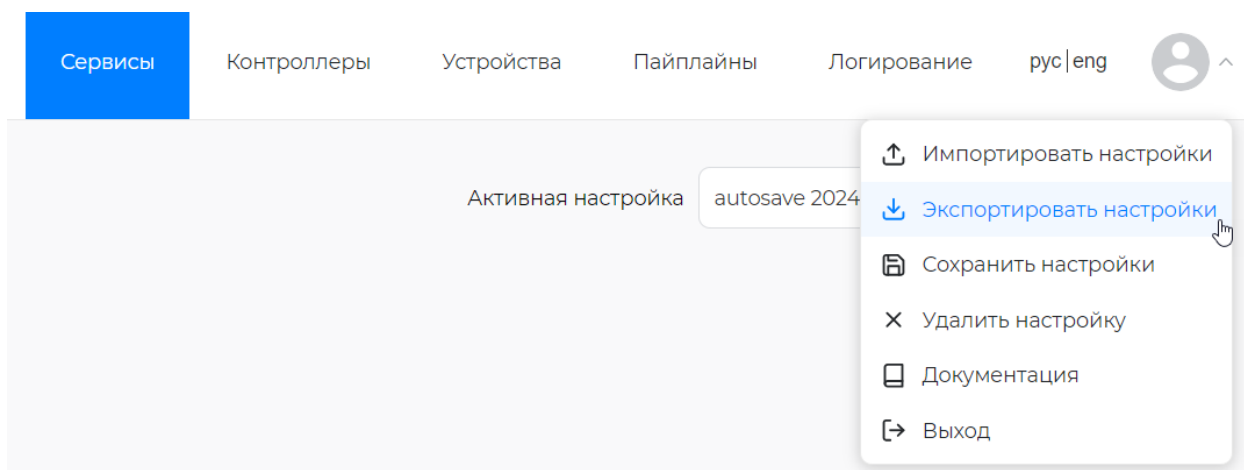
**Таблица 8.** Доступные команды:

Ключ	Описание
-h	Справка по командам
-f	Путь до и имя <code>docker-compose.yml</code> . Обязательный
-d	Путь до и директория сохранения образа. Обязательный
-o	Имя образа.
-u	Логин для подключения к <code>dockerhub.visionlabs.ru</code> . Обязательный
-p	Пароль для подключения к <code>dockerhub.visionlabs.ru</code> . Обязательный


Итоговый архив необходимо поместить в директорию Access и произвести запуск по [инструкции](#).

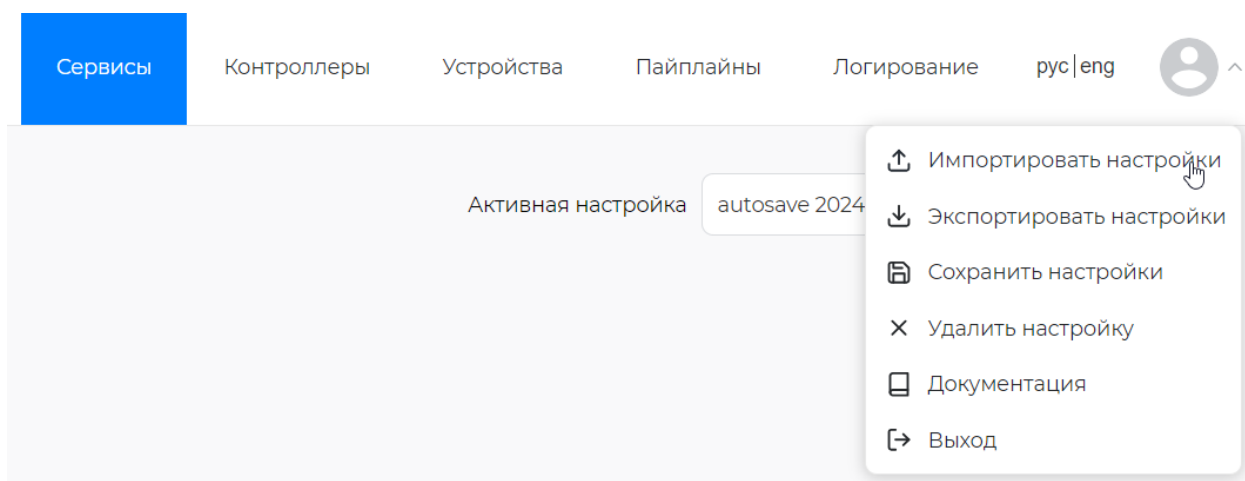
## 9. Обновление Access

1. Экспортируйте файл настроек старой версии продукта, для этого нажмите на стрелку  справа от аватара пользователя и кнопку **Экспортировать настройки** (Рисунок 5).

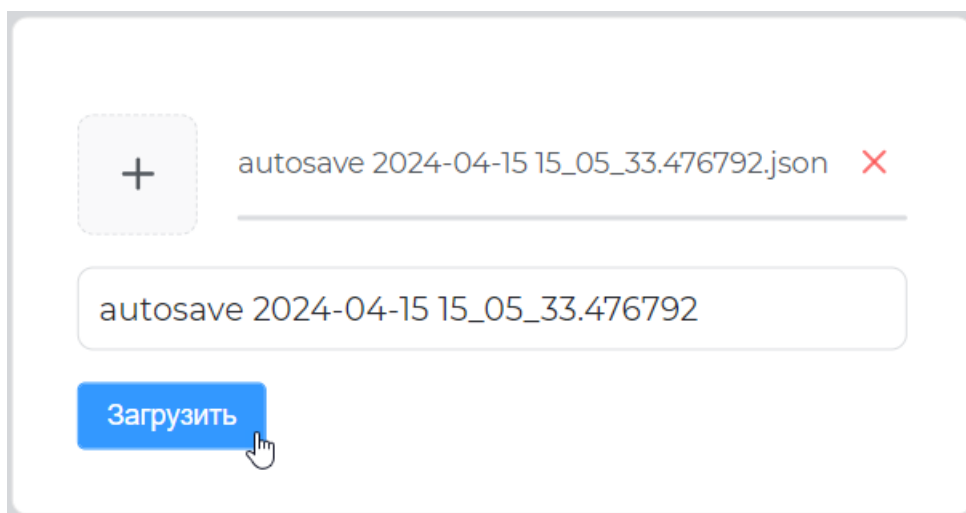


**Рис. 5:** Экспортировать настройки

- Загрузится файл со всеми настройками компонентов в формате json.
2. Сохраните конфигурационный файл .env старой версии, расположенный по пути `/var/lib/vl-access-2/vl-access-2-v2.*.*/.env`, где `v2.*.*` - номер версии продукта.
  3. Для обновления Access необходимо выполнить [Удаление](#) → [Распаковку новой версии дистрибутива](#) → [Настройку параметров env](#) (перенесите необходимые значения переменных из сохраненного конфигурационного файла .env старой версии в новый файл) и повторную [Установку и запуск Access](#).
  4. После установки новой версии перейдите в UI Access, нажмите на стрелку  справа от аватара пользователя и кнопку **Импортировать настройки** (Рисунок 6).

**Рис. 6:** Импортировать настройки

В открывшемся окне выберете ранее сохраненный json файл, введите название настройки и нажмите кнопку **Загрузить** (Рисунок 7).

**Рис. 7:** Загрузка файла настройки

При попытке импортировать файл настроек прежней версии Access в новую могут возникнуть ошибки обратной совместимости компонентов. В случае возникновения проблем обратитесь в техническую поддержку VisionLabs.



## 10. Удаление Access

Удаление Access выполняется путем остановки и удаления контейнеров Docker.

Выполните следующие действия:

1. Перейдите в директорию с файлом `docker-compose.yml`. Остановите запущенные контейнеры:

```
docker-compose down
```

2. Удалите директорию установленного Сервиса:

```
rm -rf vl-access-2-v2.*.*
```

## 11. Система логирования

### 11.1. Инструкция по сбору логов

Сбор логов необходим для:

- Предоставления информации технической поддержки VisionLabs для составления тикета на поиск проблемы;
- Самостоятельного поиска ошибок.


Для получения полной информации о нештатной ситуации при работе Access необходимо подготовить и передать информацию представителю VisionLabs о:

- [Файле настроек](#);
- [Логах контейнеров](#);
- [Файле .env](#);
- [Информацию о рабочем окружении](#);
- [Скриншоты UI](#) (только в случаях ошибок UI).

#### 11.1.1. Файл настроек

Файл настроек – JSON файл, который содержит информацию об используемых компонентах в Access.

Файл настроек становится доступен для экспорта после создания в Access компонента.

1. После создания любого из 4-х типов компонентов необходимо нажать на  справа от аватара пользователя и нажать кнопку «Экспортировать настройки» (Рисунок 8).

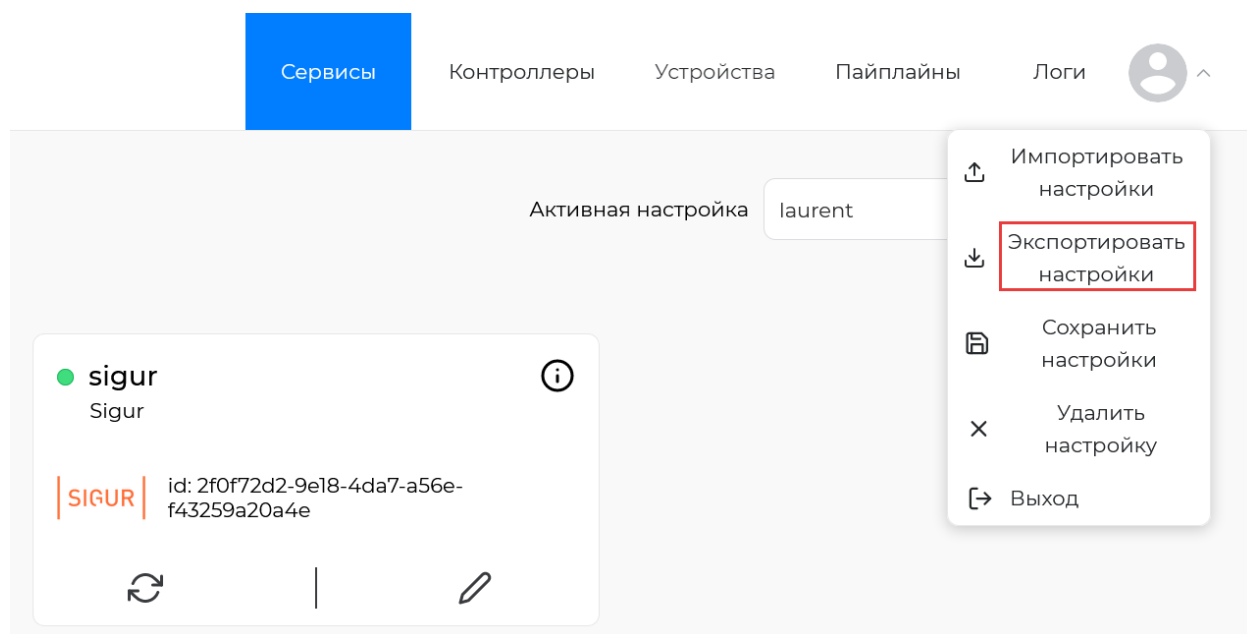


Рис. 8: Экспорт настроек

При этом произойдет загрузка JSON файла с именем vl-access\_setting.json.

2. Найдите JSON на локальной машине.

Для Linux-систем по умолчанию /home/<username>/Downloads.

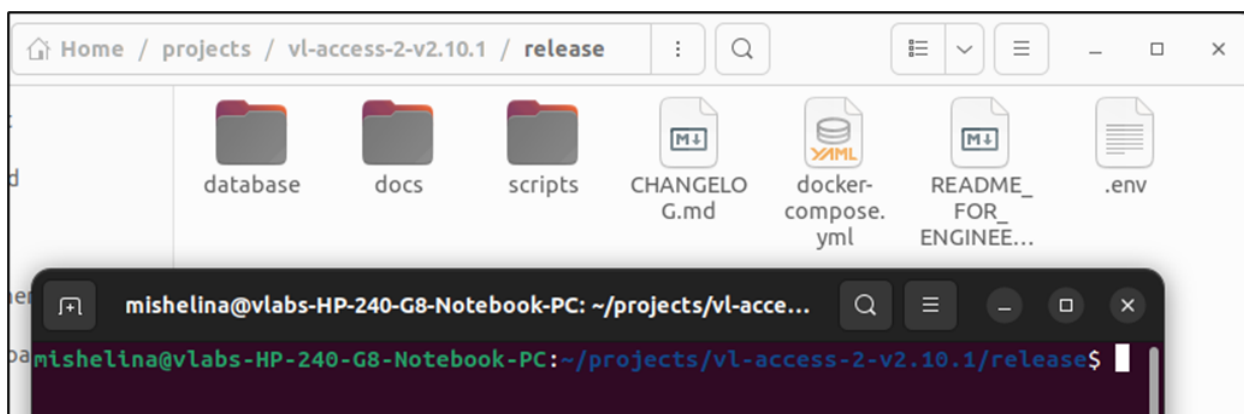
3. Переименуйте файл настроек в зависимости от основных используемых сервисов и устройств, например, bolid+gate+fast.json.

### 11.1.2. Логи контейнеров

Файлы логов контейнеров содержат всю информацию о работе Access от момента запуска (docker compose up) до создания логов, при условии активности контейнеров.

1. Откройте в консоли директорию Access.

Для самопроверки удостоверьтесь, что в этой директории есть /db и docker-compose.yml (Рисунок 9).

**Рис. 9:** Корректная директория

2. Выполните команды для записи логов контейнеров worker и fastapi:

В названии лог файла должны быть указаны названия основных компонентов.

```
docker-compose logs worker &> worker_<components_names, bolid+gate+fast>.log
docker-compose logs fastapi &> fastapi_<components_names, bolid+gate+fast>.log
```

3. Проверьте наличие созданных файлов .log в той же директории.

### 11.1.3. Файл .env

Файл .env находится в корневой папке дистрибутива (там же где docker-compose.yml), но может не отображаться в UI по умолчанию.

1. Найдите файл .env для передачи представителю VisionLabs.

### 11.1.4. Информация о рабочем окружении

Сверьте аппаратные и программные свойства рабочей машины с минимальными (см. [Требования](#)).

Для проверки требований выполните команды просмотра:

1. Версия ОС локальной машины:

```
hostnamectl
```

2. Версия docker/docker-compose:

```
docker --version
docker-compose --version
```

3. Информация о аппаратных характеристиках:

```
cat /proc/cpuinfo | grep "model name"
```

4. Объем свободной оперативной памяти:

```
free -h
```

5. Объем свободного дискового пространства:

```
df -h
```

#### 11.1.5. Скриншоты UI

Скриншоты необходимы только для поиска проблемы с UI:

- неадекватно отображается статус `is_alive` компонента,
- отображается компонент, которого не должно быть (удалили, но он «вернулся»),
- появилась нечитаемая ошибка,
- в списке в LUNA Clementine дубли - на скриншот должны попасть даты создания лиц в списке,
- есть расхождения с событиями в LUNA Clementine, неверный тег события,
- прочее проблемы с UI.

#### 11.2. Отслеживание проходов с помощью `trace_id`

В Access для каждого прохода через СКУД формируется уникальный идентификатор — `trace_id`. Данный идентификатор связывает все события, относящиеся к одному проходу конкретного лица: от момента детекции до предоставления доступа через турникет.

Использование `trace_id` позволяет быстро и точно отслеживать весь процесс прохода, анализировать задержки на разных этапах и выявлять возможные проблемы.

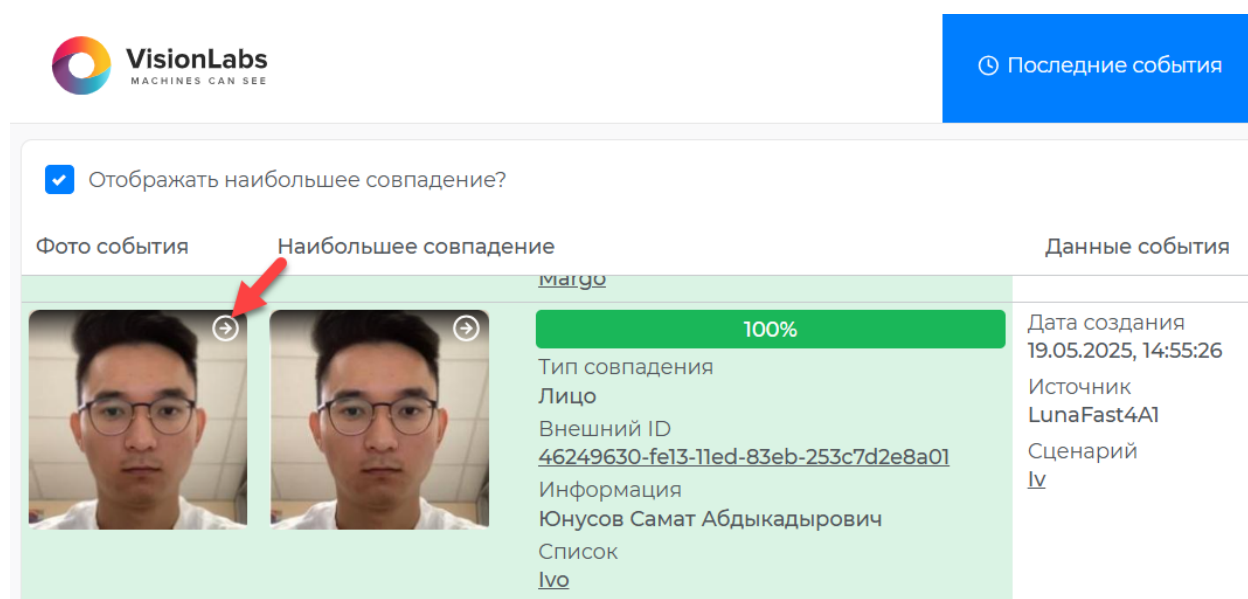
**С помощью `trace_id` можно отследить:**

- Время получения и обработки событий в системе;
- Информацию о результате распознавания: ФИО, номер карты, идентификаторы в СКУД и биометрической системе;
- Время выполнения ключевых операций: детекция, распознавание, отправка данных на терминал и контроллер;

- Другие технические детали, связанные с обработкой прохода.

### 11.2.1. Отслеживание проходов через события Luna Platform



1. Откройте интерфейс **Luna CLEMENTINE / Luna Platform** и перейдите в раздел **Последние события**.
2. Найдите самое раннее успешное событие распознавания.
3. Перейдите в карточку этого события, нажав на стрелку в правом верхнем углу возле фотографии (Рисунок 10).



**VisionLabs**  
MACHINES CAN SEE

Последние события

☒ Отображать наибольшее совпадение?

Фото события	Наибольшее совпадение	Данные события
	 <div> <p>100%</p> <p>Тип совпадения Лицо</p> <p>Внешний ID <a href="#">46249630-fe13-11ed-83eb-253c7d2e8a01</a></p> <p>Информация Юнусов Самат Абдыкадырович</p> <p>Список <a href="#">Ivo</a></p> </div>	<p>Дата создания 19.05.2025, 14:55:26</p> <p>Источник LunaFast4A1</p> <p>Сценарий <a href="#">lv</a></p>

**Рис. 10:** Событие успешного распознавания

4. Скопируйте восьмизначный `trace_id` из поля **Теги** (Рисунок 11).

## Детали события

### Информация о событии

Дата создания	19.05.2025, 14:55:26
Дополнительная информация	ID события 
Сценарий	<a href="#">lv</a>
Источник	LunaFast4AI
Теги	<b>35e74eda</b> MBP
Liveness	Живое лицо

Рис. 11: Расположение trace\_id в карточке события

- Убедитесь, что в LUNA Access 2 включён режим отладки (debug), и в логе worker присутствуют сообщения уровня DEBUG.
- Найдите все логи, относящиеся к этому проходу, выполнив команду:

```
docker-compose logs fastapi worker | grep "<trace_id>"
```

Пример (Рисунок 12).

```
vlabs@livery:~/vl-access-2$ docker-compose logs fastapi worker | grep 35e74eda
WARN[0000] The "VL_ACCESS_TAG" variable is not set. Defaulting to a blank string.
WARN[0000] The "VL_ACCESS_TAG" variable is not set. Defaulting to a blank string.
vl_access_2 fastapi | [2025-05-19 12:55:26,427: DEBUG/EventManager] Event added to queue events: <FaceDetectionEvent: image_bytes='...' image_base64='...' trace_id='35e74eda' name='35e74eda-ccb4-4189-ada7-c119efd00303' source='LunaFast4AI' liveness=True>
vl_access_2 worker | [2025-05-19 12:55:26,536: DEBUG/MatchByPhoto 56356c1c-1401-42a2-9d0a-782cfc288b72] <trace 35e74eda>: Got event <FaceDetectionEvent: LunaFast4AI | 35e74eda-ccb4-4189-ada7-c119efd00303>.
vl_access_2 worker | [2025-05-19 12:55:26,978: DEBUG/Luna 772d7b8f-829a-483e-9b96-29e85b183501] <trace 35e74eda>: Image is successfully send to Luna. Response: [{'event_id': '6435f6be-3d8c-42e5-adaa-3f93064b8a77', 'url': '/6/events/6435f6be-3d8c-42e5-adaa-3f93064b8a77', 'detections': [{'filename': 'raw_image', 's
```

Рис. 12: Пример лога

### 11.2.2. Как найти trace\_id по ФИО персоны

- Убедитесь, что включён режим отладки (debug) для LUNA Access 2 и в worker есть DEBUG-логи.
- Найдите нужный лог по ФИО:

```
docker-compose logs worker | grep -i "<name>"
```

- В найденном логе будет указан trace\_id в формате trace <trace\_id> (Рисунок 13).

```
vlabs@ivery:~/vl-access-2$ docker-compose logs worker | grep -i "Юнусов"
WARN[0000] The "VL_ACCESS_TAG" variable is not set. Defaulting to a blank string.
WARN[0000] The "VL_ACCESS_TAG" variable is not set. Defaulting to a blank string.
vl_access_2 worker | [2025-05-19 12:55:01.559; INFO/Strazh b468d271-b58b-49ba-94bb-8137d5f2c8d8] Successfully updated face <LunaFace id: 46249630-fe13-11ed-8
3eb-253c7d2e8a01 name: Юнусов Самат Абдыкадырович> in list Luna
vl_access_2 worker | [2025-05-19 12:55:01.560; INFO/Strazh b468d271-b58b-49ba-94bb-8137d5f2c8d8] Saved or updated person in storage: <PacsPerson: 46249630-fe
13-11ed-83eb-253c7d2e8a01 | Юнусов Самат Абдыкадырович>
vl_access_2 worker | [2025-05-19 12:55:26.978; DEBUG/Luna 772d7b8f-829a-483e-9b96-29e85-100000000000] <trace_35e74eda> Image is successfully send to Luna. Respons
e: [{'event_id': '6435f6be-3d8c-42e5-adaa-3f93064b8a77', 'url': '/6/events/6435f6be-3d8c-42e5-adaa-3f93064b8a77', 'detections': [{'filename': 'raw_image', 'sa
```

Рис. 13: Отображение trace\_id в логе

- Используйте данный trace\_id для поиска всех связанных логов:

```
docker-compose logs fastapi worker | grep "<trace_id>"
```

### 11.2.3. Поиск события в Luna Platform по trace\_id

Порядок действий для отображения события по trace\_id в UI (Рисунок 14).

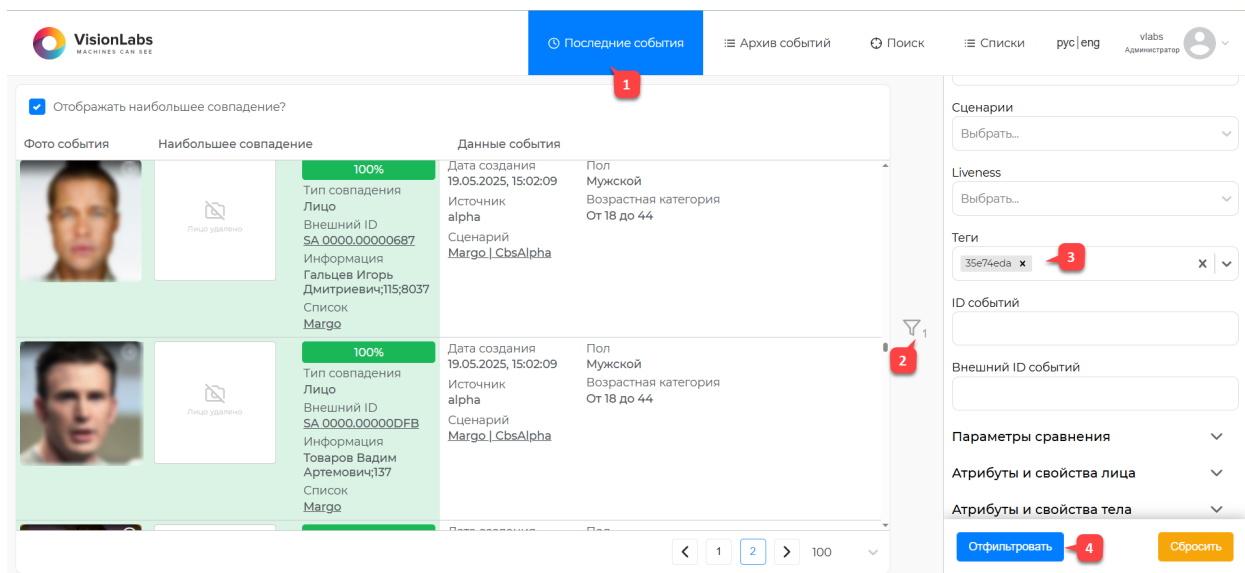


Рис. 14: Отображение trace\_id в логе

- В **Luna CLEMENTINE / Luna Platform** перейдите в раздел **Последние события**.
- Нажмите на значок фильтра справа.
- Введите значение trace\_id в поле **Теги**:
- Нажмите кнопку **Отфильтровать**.