FindFace Security

Выпуск 4.1.1

NtechLab

июн. 26, 2023

Содержание

FindFace Security — это система биометрической видеоидентификации, предназначенная для автоматизации основной деятельности сотрудников служб безопасности и гостеприимства. В основе FindFace Security лежит FindFace Enterprise Server, передовая технология распознавания лиц на базе искусственного интеллекта. FindFace Security представляет собой готовый к использованию продукт, который может использоваться в таких областях, как транспорт, розничная торговля, банковское обслуживание, индустрия развлечений, спортивные мероприятия, организация мероприятий, сервисы знакомств, видеонаблюдение, общественная и корпоративная безопасность и др.

FindFace Security распознает лица нежелательных персон и VIP-гостей и оповещает сотрудников служб безопасности и гостеприимства об их приходе. FindFace Security также распознает такие атрибуты лица, как пол, возраст, эмоции, наличие очков и бороды, и отображает данную информацию для каждого обнаруженного на видеоизображении лица.

Интегрированная в FindFace Security антиспуфинговая система гарантирует, что перед камерой находится живой человек, и исключает возможность мошенничества с использованием фотографии лица на бумаге или экране мобильного устройства.

Раннее распознавание прихода нежелательных персон и VIP-гостей позволяет решать следующие задачи:

- снижение операционных потерь от мошеннических действий;
- снижение репутационных потерь и предотвращение конфликтных ситуаций;
- повышение качества обслуживания клиентов, в частности VIP-гостей;
- предотвращение потенциально опасных ситуаций, угрожающих жизни и здоровью людей.

FindFace Security поддерживает интеграцию сторонних решений через *HTTP API*, *вебхуки* и *плагины*, так что вы с легкостью сможете усовершенствовать свою текущую систему или приложение, добавив в них функционал распознавания лиц.

Полный список возможностей:

- Архитектура на основе AI.
- Быстрая и надежная биометрическая идентификация в реальном времени по базам данных досье.
- Повышенная производительность и отказоустойчивость в высоконагруженных системах с большим количеством подключенных камер и клиентов.
- Поддержка как видеопотоков с камер, так и архивных файлов.
- Быстрое создание базы данных досье.
- Распределение базы данных досье между несколькими серверами с синхронизацией и репликацией.
- Настройка содержимого досье.
- Поддержка дедупликации.
- Верификация лиц.
- AI-распознавание пола, возраста, эмоций, очков, бороды и других атрибутов лица.
- AI-детектор живых лиц (Liveness).
- Видеонаблюдение.
- Поиск по базам данных.
- Расширенный набор поисковых фильтров.

- Расширенное управление пользователями.
- Аутентификация по паролю или сертификату.
- CPU- и GPU-ускорение на ваш выбор.
- Удобный консольный инсталлятор и дружественный интерфейс.
- Развертывание на одном или нескольких серверах.
- Возможность лицензирования в открытых и закрытых системах.
- Интеграция через HTTP API, вебхуки и плагины на python.
- Партнерские интеграции с популярными системами.
- Мобильное приложение.

Данное руководство будет для вас наиболее полезным, если вы:

- администратор FindFace Security,
- сотрудник службы безопасности,
- сотрудник службы гостеприимства,
- специалист, обслуживающий систему и ее комплекс технических средств,
- специалист по системной интеграции, который собирается интегрировать технологию распознавания лиц в свою систему.

Глава 1

Руководство администратора

1.1 Архитектура

Хотя взаимодействие с FindFace Security происходит в основном через веб-интерфейс, не забудьте уделить немного времени изучению архитектуры программного комплекса. Эти знания необходимы для развертывания, интеграции, обслуживания и устранения проблем при работе FindFace Security.

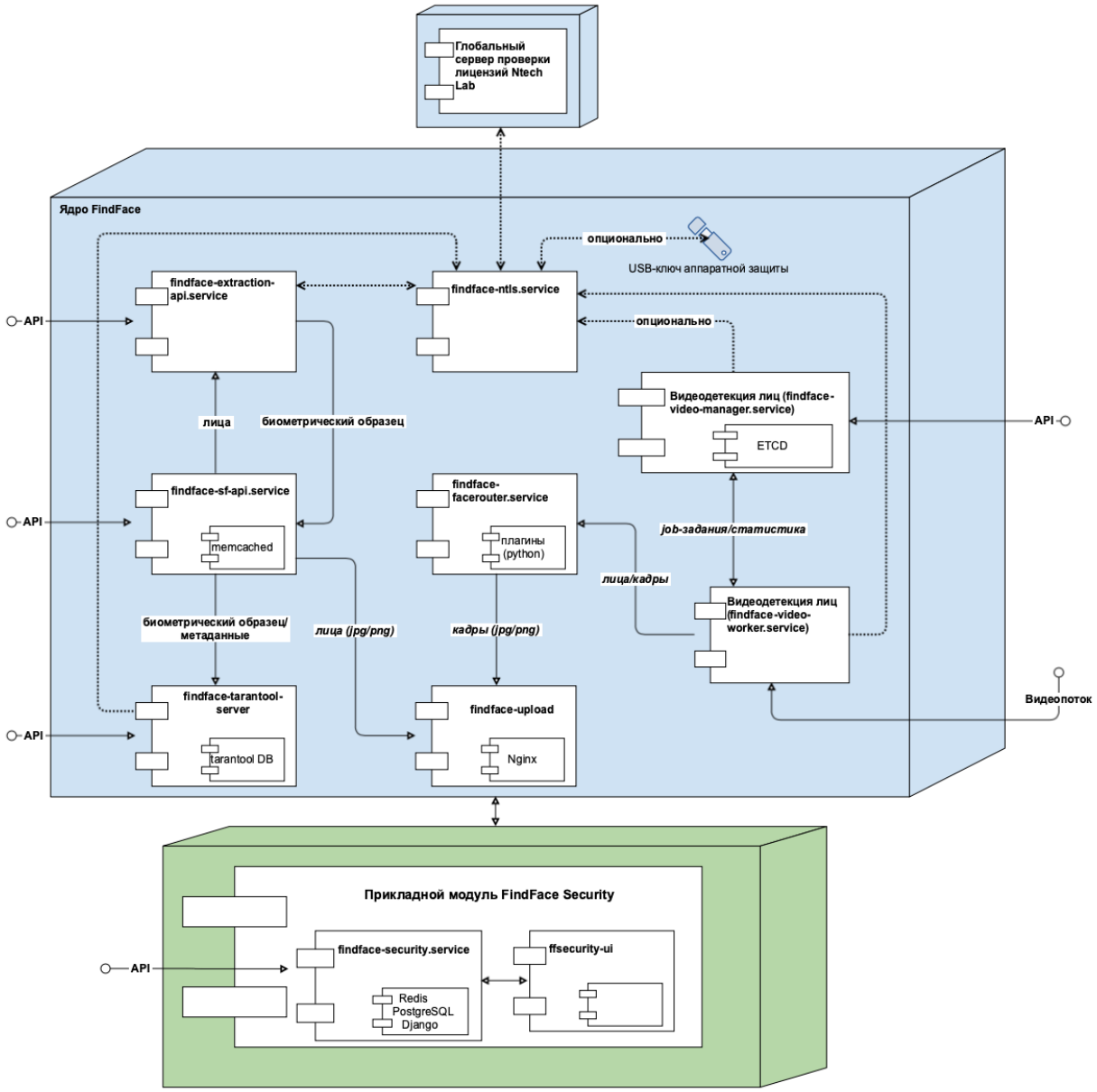
В этой главе:

1.1.1 Архитектурные элементы

FindFace Security состоит из следующих основных архитектурных элементов:

- Ядро FindFace, передовая технология распознавания лиц на базе искусственного интеллекта, которая может использоваться в качестве отдельного продукта FindFace Enterprise Server.
- FindFace Security, готовый прикладной модуль к FindFace Enterprise Server.

Схема архитектуры



Ядро FindFace

Ядро FindFace включает в себя следующие компоненты:

Компонент	Описание	Поставщик
findface-extraction-api	Сервис, использующийся для обнаружения лиц на статических изображениях и извлечения из лиц биометрических образцов (векторов признаков). Работа на базе нейронных сетей. Поддерживается CPU- или GPU-ускорение.	Собственная разработка Ntech Lab
findface-sf-api	Сервис, реализующий HTTP API обнаружения и распознания лиц.	
findface-tarantool-server	Сервис, обеспечивающий взаимодействие между сервисом `findface-sf-api` и биометрической базой данных (базой, в которой хранятся биометрические образцы) на основе Tarantool.	
findface-upload	Веб-сервер на базе NginX, используемый как хранилище исходных изображений, миниатюр лиц и нормализованных изображений лиц.	
findface-facerouter	Сервис, который используется для задания правил обработки обнаруженных на видео лиц. В FindFace Security функции `findface-facerouter` выполняются `findface-security` (см. *Прикладной модуль FindFace Security*). Однако в целях интеграции вы можете установить и настроить данный компонент (см. *Плагины*).	
findface-video-manager	Сервис, являющийся частью модуля видеодетекции лиц, через который осуществляется управление детекцией лиц на видео, а именно задаются настройки и список видеопотоков для обработки.	
findface-video-worker	Сервис, часть модуля видеодетекции лиц, который распознает лица на видео и отправляет их нормализованные изображения, соответствующие видеокадры и метаданные (такие как ID камеры и время обнаружения лица) в сервис `findface-facerouter` для дальнейшей обработки в соответствие с заданными правилами. Поддерживается CPU- и GPU-ускорение.	
findface-ntls	Локальный сервер лицензий, который проверяет подлинность *лицензии* FindFace Security, взаимодействуя с глобальным сервером лицензий NtechLab. Для закрытых систем поддерживается работа с аппаратными лицензионными ключами. Поддерживается лицензирование через прокси-сервер.	
Tarantool	Стороннее программное обеспечение, на основе которого реализована биометрическая база данных, хранящая извлеченные биометрические образцы (векторы признаков) и события обнаружения лиц. Системные данные, досье, пользовательские аккаунты и настройки камер хранятся в PostgreSQL (часть прикладного модуля FindFace Security).	Tarantool
etcd	Стороннее программное обеспечение, реализующее распределенное хранилище ключей для компонента `findface-video-manager`. Используется в качестве координационной службы в распределенной системе, обеспечивая отказоустойчивость видеодетектора лиц.	etcd
NginX	Стороннее программное обеспечение, которое реализует веб-интерфейсы системы.	nginx
memcached	Стороннее программное обеспечение, реализующее сервис кэширования данных в оперативной памяти на основе хеш-таблицы. Используется для временного хранения извлеченных биометрических образцов перед их записью в базу данных Tarantool.	memcached

Прикладной модуль FindFace Security

Прикладной модуль FindFace Security включает в себя следующие компоненты:

Компонент	Описание	Поставщик
findface-security	Компонент, обеспечивающий доступ конечного пользователя к функциям ядра FindFace. Обеспечивает взаимодействие между ядром FindFace и веб-интерфейсом, функционирование системы в целом, HTTP и веб-сокет, обновление биометрической базы данных, отправку уведомлений о событиях, объединение событий в эпизоды, вебхуки. Включает в себя следующие внутренние сервисы: обновление списков для мониторинга, уведомление о неприятных событиях, обновление вебхуков, проверка лицензии, управление эпизодами событий.	Собственная разработка Ntech Lab
ffsecurity-ui	Основной веб-интерфейс, использующийся для взаимодействия с FindFace Security. Позволяет работать с событиями идентификации лиц, искать лица, управлять камерами, пользователями, досье и списками наблюдения.	
PostgreSQL	Стороннее программное обеспечение, реализующее основную базу данных системы. В данной базе хранятся детализированные досье персон с разбиением по категориям (спискам наблюдения), а также данные внутреннего характера, такие как профили пользователей FindFace Security, настройки видеокамер и пр. Биометрические данные лиц и события идентификации лиц хранятся в Tarantool (часть ядра FindFace).	PostgreSQL
Redis	Стороннее программное обеспечение, которое реализует брокер сообщений внутри `findface-security`.	Redis
Django	Стороннее программное обеспечение, реализующее веб-фреймворк для веб-интерфейса FindFace Security.	Django

См.также:

Подробно о компонентах

1.1.2 Развертывание на одиночном сервере или в кластере

Вы можете развернуть FindFace Security как на одиночном сервере, так и в кластерной среде. При выборе последнего варианта доступны следующие схемы развертывания:

- Центральный сервер FindFace Security, взаимодействующий с несколькими дополнительными серверами для обработки видео (с одним установленным компонентом `findface-video-worker`).
- Полностью распределенная архитектура FindFace Security. Может понадобиться балансировка нагрузки.

Подробнее см. раздел *Типичная установка в кластере*.

1.1.3 Аппаратное ускорение на CPU и GPU

Сервисы `findface-extraction-api` и `findface-video-worker` могут использовать как CPU-, так и GPU-ускорение. Нужный тип ускорения выбирается во время установки из консольного *инсталлятора*.

Если установка FindFace Security выполняется из *apt-репозитория*, на CPU-сервере нужно развернуть пакеты `findface-extraction-api` и/или `findface-video-worker-cpu`, а на GPU-сервере пакеты `findface-extraction-api-gpu` и/или `findface-video-worker-gpu`.

Важно: Для выбора конфигурации оборудования см. *Системные требования*.

Важно: Если разрешение используемой камеры превышает 1280x720 пикселей, настоятельно рекомендуется использовать пакет с ускорением на GPU `findface-video-worker-gpu`.

Примечание: *Liveness-детектор* на CPU работает гораздо медленнее, чем на GPU.

1.2 Системные требования

Для расчета характеристик серверов FindFace Security используйте приведенные ниже системные требования.

Совет: Сначала обязательно ознакомьтесь с *архитектурой* FindFace Security.

В этой главе:

1.2.1 Базовая конфигурация

Важно: Если разрешение используемой камеры превышает 1280x720 пикселей, настоятельно рекомендуется использовать пакет с ускорением на GPU `findface-video-worker-gpu`.

	Минимальная	Рекомендуемая
CPU	Intel Core i5 CPU с 4-мя физическими ядрами 2.8 ГГц	Intel Xeon E5v3 с 6-ю физическими ядрами, лучший или похожий процессор
	На собственные нужды FindFace Security требуется 2 ядра HT > 2.5 ГГц. Характеристики также зависит от количества камер. Для одной камеры `720p@25FPS` требуется 2 ядра >2.5 ГГц. Поддержка AVX2	
GPU (опционально)	Nvidia Geforce® GTX 980 4 Гб	Nvidia Geforce® GTX 1080+ с 8+ Гб RAM
	Поддерживаемые серии: GeForce (Maxwell, Pascal, Turing и выше), Tesla (Maxwell, Pascal, Volta v100, Turing и выше)	
RAM	10 Гб	16+ Гб
	На собственные нужды FindFace Security требуется 8 Гб. Потребление памяти также зависит от количества камер. Для одной камеры `720p@25FPS` требуется 2 ГБ RAM	
HDD	16 Гб	16+ Гб
	На собственные нужды операционной системы и FindFace Security требуется 15 ГБ. Суммарный объем определяется в зависимости от требуемой глубины архива событий в базе данных и в логе из расчета 1.5 Мб на 1 событие	
Операционная система	Ubuntu 16.04, только x64	

Совет: Для более точного подбора конфигурации свяжитесь с нашими техническими экспертами по адресу support@ntechlab.com.

1.3 Принцип лицензирования

FindFace Security лицензируется по следующим критериям:

1. Количество извлеченных биометрических образцов и биометрических образцов в мониторинге (т. е. в списках наблюдения). В ходе работы FindFace Security биометрические образцы извлекаются из лиц, обнаруженных на видео, и из фотографий, загружаемых в досье. В целом схема лицензирования выглядит следующим образом:
   - События: 1 событие распознавания лица на видео = 1 лицо в лицензии.
   - Досье: 1 фотография в досье = 2 лица в лицензии (извлечение биометрического образца + мониторинг лица).
2. Количество камер в системе.
3. Количество экземпляров моделей, используемых `findface-extraction-api`.
4. Распознавание атрибутов лица: пол/возраст/эмоции/очки/борода.
5. Распознавание живых лиц в реальном времени (Liveness).
6. Интеграции с партнерами.

Вы можете выбрать между онлайн-лицензированием и лицензированием в закрытой сети:

- Для онлайн-лицензирования необходимо стабильное интернет-соединение. После отключения от интернета система продолжит работать в автономном режиме в течение 1 часа.
- Для лицензирования в закрытой сети необходимо наличие USB-порта на физическом сервере с компонентом `findface-ntls` (сервер лицензирования в составе *ядра FindFace*) для подключения предоставляемого USB-ключа аппаратной защиты.

Для обеспечения функционирования системы достаточно одного экземпляра `findface-ntls`. Если по какой-либо причине ваша система нуждается в большем количестве серверов лицензирования, заблаговременно сообщите об этом своему менеджеру Ntech Lab, чтобы предотвратить блокировку системы.

См.также:

Лицензирование

1.4 Развертывание FindFace Security

Для вашего удобства мы предлагаем несколько вариантов развертывания FindFace Security:

- Развертывание из консольного инсталлятора
- Пошаговое развертывание из apt-репозитория

1.4.1 Развертывание из консольного инсталлятора

Для развертывания FindFace Security используется консольный инсталлятор.

Совет: Перед тем как приступить к развертыванию, обязательно ознакомьтесь с *системными требованиями*.

Важно: Для успешного функционирования системы после установки из инсталлятора IP-адрес сервера должен быть статическим. Для того чтобы сделать IP-адрес статическим, откройте файл `etc/network/interfaces` и измените текущую запись для основного сетевого интерфейса так, как показано в примере ниже. Не забудьте заменить адреса в примере на актуальные с учетом настроек сети.

```
sudo vi /etc/network/interfaces

iface eth0 inet static
address 192.168.112.144
netmask 255.255.255.0
gateway 192.168.112.254
dns-nameservers 192.168.112.254
```

Перезапустите сетевые интерфейсы.

```
sudo service networking restart
```

С осторожностью редактируйте файл `etc/network/interfaces`. Перед тем как приступить к редактированию, ознакомьтесь с инструкцией по настройке сетей Ubuntu.

Для развертывания FindFace Security из инсталлятора выполните следующие действия:

1. Загрузите файл инсталлятора `findface-security-and-server-4.1.1.run`.
2. Поместите файл `.run` в любой каталог на сервере установки (например, `/home/username`).
3. Из данного каталога сделайте файл `.run` исполняемым.

   ```
   chmod +x findface-security-and-server-4.1.1.run
   ```

4. Запустите файл `.run`.

   ```
   sudo ./findface-security-and-server-4.1.1.run
   ```

   Инсталлятор задаст вам несколько вопросов, после чего проверит, соответствует ли сервер системным требованиям. Вопросы следующие:

   1. Устанавливаемый продукт: FindFace Security.
   2. Тип установки:
      - `1`: установить FindFace Security на одиночном физическом сервере.
      - `2`: установить FindFace Security в качестве центрального сервера и настроить его на взаимодействие с дополнительными удаленными серверами `findface-video-worker`.

        Совет: Для отдельной установки `findface-video-worker` см. *Дополнительное развертывание findface-video-worker на удаленных серверах*.

- 3: установить только apt-репозиторий для *пошагового развертывания* в будущем.

  Важно: При данном типе установке модели нейронных сетей, необходимые для функционирования `findface-extraction-api`, не устанавливаются. Обязательно установите их *вручную* на серверах с `findface-extraction-api`.

- 4: полностью настраиваемая установка (установка нужных пакетов).

  Важно: Обязательно *установите* модели нейронных сетей на серверах с `findface-extraction-api`.

3. Тип пакета `findface-video-worker`: CPU или GPU.
4. Тип пакета `findface-extraction-api`: CPU или GPU.

Ответы на вопросы будут сохранены в файл `/tmp/<findface-installer-*>.json`. Вы можете отредактировать его и использовать для установки FindFace Security на других серверах, не отвечая повторно на вопросы инсталлятора.

При выборе установки одиночного сервера FindFace Security, его компоненты будут автоматически установлены, настроены и запущены в соответствии со следующей конфигурацией:

Важно: В случае чистой установки инсталлятор автоматически настроит `findface-extraction-api` на использование нейронной сети `grapefruit_480`. В противном случае вам будет предложено сделать выбор между `grapefruit_480` и предыдущей моделью. Категорически не рекомендуется использовать инсталлятор для обновления системы. См. инструкции в разделе *Обновление FindFace Security до 4.1.x*.

Сервис	Конфигурация
postgresql-9.5	Устанавливается и запускается.
redis-server	Устанавливается и запускается.
etcd	Устанавливается и запускается.
memcached	Устанавливается и запускается.
nginx	Устанавливается и запускается.
django	Устанавливается и запускается как веб-фреймворк для веб-интерфейса FindFace Security.
findface-ntls	Устанавливается и запускается.
findface-tarantool-server	Устанавливается и запускается. Количество экземпляров (шардов) рассчитывается по формуле: `N = max (min (mem_mb // 2000, cpu_cores), 1)`, т.е. оно равно размеру оперативной памяти в Мб, разделенному на 2000, или количеству физических ядер процессора (но не менее 1 шарда).
findface-extraction-api	Устанавливается и запускается.
findface-sf-api	Устанавливается и запускается.
findface-upload	Устанавливается.
findface-video-manager	Устанавливается и запускается (CPU/GPU-ускорение).
findface-video-worker-*	Устанавливается и запускается.
findface-data-*	Модели нейронных сетей для распознавания лиц и их атрибутов (пол, возраст, эмоции, очки, борода, и пр.). Устанавливаются.
findface-gpudetector-data/	Данные gpu-детектора NTechLab. Устанавливается.
python3-ntech.ffsecurity-client	Библиотека клиента Python, используемая в API FindFace Security. Устанавливается.
findface-security	Устанавливается и запускается.
jq	Устанавливается. Используется для структурирования API-ответов от FindFace Security в формате JSON.

По завершении установки в консоль будет выведена информация, необходимая для использования FindFace Security:

Совет: Обязательно сохраните эти данные: они вам понадобятся.

```
################################################################################
#                          Installation is complete                            #
################################################################################
- upload your license to http://172.20.77.17/#/license/
```

(continues on next page)

(продолжение с предыдущей страницы)

```
- user interface: http://172.20.77.17/
  superuser:      admin
  password:       admin
  documentation:  http://172.20.77.17/doc/
```

5. Загрузите файл лицензии через основной веб-интерфейс `http://<IP_адрес_сервера>/#/license`. Для доступа в веб-интерфейс используйте логин и пароль администратора, выведенные в консоли.

Примечание: IP-адрес сервера в ссылках на веб-интерфейсы FindFace имеет вид `127.0.0.1` или <IP_адрес_в_сети>, в зависимости от того, принадлежит ли сервер к сети.

Важно: Не передавайте данные `superuser` (Супер Администратора) третьим лицам. Для администрирования системы создайте назначаемого администратора. Отличие назначаемого администратора от Супер Администратора в том, что последний не может лишиться прав администратора даже при смене роли.

---

6. Для того чтобы автоматически установить FindFace Security на других серверах, не отвечая на вопросы инсталлятора, используйте файл `/tmp/<findface-installer-*>.json`. Запустите инсталлятор следующей командой:

```
sudo ./findface-security-and-server-4.1.1.run -f /tmp/<findface-installer-*>.json
```

Совет: Пример данного файла можно посмотреть в разделе *Файл с параметрами установки*.

1.4.2 Пошаговое развертывание из apt-репозитория

Данный раздел содержит подробную информацию о пошаговом развертывании компонентов FindFace Security. Выполните приведенные ниже инструкции, придерживаясь заданного порядка.

В этом разделе:

Установка apt-репозитория

Прежде всего установите apt-репозиторий FindFace следующим образом:

1. Загрузите файл инсталлятора `findface-security-and-server-4.1.1.run`.
2. Поместите файл `.run` в любой каталог на сервере установки (например, `/home/username`).
3. Из данного каталога сделайте файл `.run` исполняемым.

```
chmod +x findface-security-and-server-4.1.1.run
```

4. Запустите файл `.run`.

```
sudo ./findface-security-and-server-4.1.1.run
```

Инсталлятор задаст вам несколько вопросов, после чего проверит, соответствует ли сервер системным требованиям. Вопросы следующие:

1. Устанавливаемый продукт: `FindFace Security`.
2. Тип установки: `repo: Don't install anything, just set up the APT repository`.
3. Устанавливаемые модели нейронных сетей (при необходимости). Для того чтобы выбрать модели, сначала снимите выделение, введя в командной строке `-*`, затем введите через пробел порядковые номера нужных моделей, например: `1 3`. Введите `done` для сохранения выбора и перехода к следующему шагу.

Важно: Должна быть установлена как минимум одна модель для биометрии лица.

После этого apt-репозиторий FindFace будет автоматически установлен.

Установка необходимого стороннего ПО

Для работы FindFace Security необходимо стороннее программное обеспечение PostgreSQL, Redis, etcd и memcached. Выполните следующие действия:

1. Установите пакеты с указанным сторонним ПО следующим образом:

```
sudo apt update
sudo apt install -y postgresql-9.5 redis-server etcd memcached
```

2. Откройте файл конфигурации `memcached`. Установите максимальный размер памяти в мегабайтах, используемый для хранения элементов: `-m 512`. Установите максимальный размер элемента: `-I 16m`. Если один или оба этих параметра отсутствуют, добавьте их в файл.

```
sudo vi /etc/memcached.conf

-m 512
-I 16m
```

3. Добавьте сервисы стороннего ПО в автозагрузку Ubuntu и запустите их:

```
sudo systemctl enable postgresql@9.5-main.service redis-server etcd.service memcached.service
sudo systemctl start postgresql@9.5-main.service redis-server etcd.service memcached.service
```

Обеспечение лицензирования

См.также:

Принцип лицензирования

Вы получаете файл лицензии вместе с установочными пакетами FindFace Security. Для лицензирования в закрытой сети вам также будет предоставлен ключ аппаратной защиты.

Лицензирование FindFace Security обеспечивается следующим образом:

1. Разверните `findface-ntls`, сервер лицензий в составе ядра FindFace.

   Важно: Система на базе FindFace Security может включать в себя только один экземпляр `findface-ntls`.

   Совет: В файле конфигурации `findface-ntls` вы можете изменить папку для хранения файла лицензии и настроить удаленный доступ к веб-интерфейсу `findface-ntls`, используемому для управления лицензией. Подробнее см. *findface-ntls*.

   ```
   sudo apt update
   sudo apt install -y findface-ntls
   sudo systemctl enable findface-ntls.service && sudo systemctl start findface-ntls.service
   ```

2. Загрузите файл лицензии через веб-интерфейс `findface-ntls` одним из следующих способов:

   - Откройте веб-интерфейс `findface-ntls`: `http://<NTLS_IP_address>:3185/#/`. Загрузите файл лицензии.

     Совет: Впоследствии используйте основной веб-интерфейс FindFace Security, чтобы посмотреть информацию о лицензиях, обновить или продлить лицензию (*Настройки -> Лицензия*).

   - Непосредственно положите файл лицензии в предназначенную для этого папку (по умолчанию, `/ntech/license`, может быть изменена в файле конфигурации `/etc/findface-ntls.cfg`).

3. При лицензировании в закрытой системе вставьте USB-ключ аппаратной защиты в USB-порт.
4. Если лицензируемые компоненты установлены на удаленных серверах, впоследствии укажите IP-адрес сервера `findface-ntls` в их файлах конфигурации. Подробнее см. *findface-extraction-api*, *findface-tarantool-server*, *Видеодетекция лиц: findface-video-manager и findface-video-worker*.

См.также:

Просмотр и обновление лицензии

Развертывание основной базы данных

Основная база данных FindFace Security построена на PostgreSQL. Для того чтобы развернуть основную базу данных, выполните следующие действия:

1. В консоли PostgreSQL создайте пользователя `ntech` и базу данных `ffsecurity`.

   ```
   sudo -u postgres psql

   postgres=# CREATE ROLE ntech WITH LOGIN;

   postgres=# CREATE DATABASE ffsecurity WITH OWNER ntech ENCODING 'UTF-8' LC_COLLATE='en_US.UTF-
   ↪8' LC_CTYPE='en_US.UTF-8' TEMPLATE template0;
   ```

(continues on next page)

(продолжение с предыдущей страницы)

Совет: Для выхода из консоли PostgreSQL введите `\q` и нажмите `Enter`.

2. Разрешите авторизацию в PostgreSQL по UID клиента сокета. Перезапустите PostgreSQL.

```
echo 'local all ntech peer' | sudo tee -a /etc/postgresql/9.5/main/pg_hba.conf

sudo systemctl restart postgresql@9.5-main.service
```

Развертывание ядра FindFace

Для развертывания ядра FindFace выполните следующие действия:

Совет: Вы можете найти описание компонентов ядра FindFace и их параметров конфигурации в разделах *Архитектура* и *Подробно о компонентах*.

1. Установите компоненты ядра FindFace:

```
sudo apt update
sudo apt install -y findface-tarantool-server findface-extraction-api findface-sf-api␣
↪findface-upload findface-video-manager findface-video-worker-cpu
```

Примечание: Для того чтобы установить компонент `findface-extraction-api` с GPU-ускорением, вместо `findface-extraction-api` в команде введите `findface-extraction-api-gpu`.

Примечание: Для того чтобы установить компонент `findface-video-worker` с GPU-ускорением, вместо `findface-video-worker-cpu` в команде введите `findface-video-worker-gpu`. Если на физическом сервере установлено несколько видеокарт, см. *Использование нескольких видеокарт*.

Важно: Обязательно *установите* модели нейронных сетей на серверах с `findface-extraction-api`.

2. В файле конфигурации `findface-extraction-api` (на CPU или GPU) включите опцию `quality_estimator` для оценки качества лица при загрузке фотографии в досье.

Примечание: *Минимальное качество лица* на фотографии в досье задается параметром `MINIMUM_DOSSIER_QUALITY` в файле `/etc/ffsecurity/config.py`.

```
sudo vi /etc/findface-extraction-api.ini

quality_estimator: true
```

3. В файле конфигурации `findface-extraction-api` включите модели распознавания атрибутов лица, таких как пол, возраст, эмоции, очки и/или борода, в зависимости от ваших нужд. Удостоверьтесь, что для каждой модели вы указали правильный тип ускорения CPU или GPU: он должен совпадать с типом ускорения `findface-extraction-api`. Обратите внимание, что `findface-extraction-api` на CPU может работать только с CPU-моделями, в то время как `findface-extraction-api` на GPU поддерживает как CPU-, так и GPU-модели. Подробнее см. *Распознавание атрибутов лица*.

```
models:
  age: faceattr/age.v1.cpu.fnk
  emotions: faceattr/emotions.v1.cpu.fnk
  face: face/grapefruit_480.cpu.fnk
  gender: faceattr/gender.v2.cpu.fnk
  beard: faceattr/beard.v0.cpu.fnk
  glasses3: faceattr/glasses3.v0.cpu.fnk
```

Доступны следующие модели:

Атрибут лица	Ускорение	Параметр в файле конфигурации
биометрия лица	CPU	`face: face/grapefruit_480.cpu.fnk` `face: face/grapefruit_160.cpu.fnk`
	GPU	`face: face/grapefruit_480.gpu.fnk` `face: face/grapefruit_160.gpu.fnk`
возраст	CPU	`age: faceattr/age.v1.cpu.fnk`
	GPU	`age: faceattr/age.v1.gpu.fnk`
пол	CPU	`gender: faceattr/gender.v2.cpu.fnk`
	GPU	`gender: faceattr/gender.v2.gpu.fnk`
эмоции	CPU	`emotions: faceattr/emotions.v1.cpu.fnk`
	GPU	`emotions: faceattr/emotions.v1.gpu.fnk`
очки	CPU	`glasses3: faceattr/glasses3.v0.cpu.fnk`
	GPU	`glasses3: faceattr/glasses3.v0.gpu.fnk`
борода	CPU	`beard: faceattr/beard.v0.cpu.fnk`
	GPU	`beard: faceattr/beard.v0.gpu.fnk`

Совет: Для того чтобы отключить модель распознавания, передайте в соответствующий параметр пустое значение. Не удаляйте сам параметр, поскольку в этом случае будет выполняться поиск модели по умолчанию.

```
models:
  gender: ""
  age: ""
  emotions: ""
```

4. Откройте файл конфигурации `/etc/findface-video-worker-cpu.ini` (`/etc/findface-video-worker-gpu.ini`). В параметре `mgr-static` укажите IP-адрес сервера `findface-video-manager`, который будет обеспечивать `findface-video-worker` настройками и списком видеопотоков для обработки. В параметре `capacity` укажите максимальное количество видеопотоков, которое может быть обработано `findface-video-worker`.

```
sudo vi /etc/findface-video-worker-cpu.ini
sudo vi /etc/findface-video-worker-gpu.ini
```

(continues on next page)

(продолжение с предыдущей страницы)

```
mgr-static=127.0.0.1:18811

capacity=10
```

5. Добавьте сервисы ядра FindFace в автозагрузку Ubuntu и запустите их.

```
sudo systemctl enable findface-extraction-api findface-sf-api findface-video-manager findface-
↪video-worker-cpu
sudo systemctl start findface-extraction-api findface-sf-api findface-video-manager findface-
↪video-worker-cpu
```

Развертывание прикладного модуля FindFace Security и биометрической базы данных

Для развертывания прикладного модуля FindFace Security, выполните следующие действия:

1. Установите компоненты `findface-security` и `ffsecurity-ui`.

```
sudo apt update
sudo apt install -y ffsecurity ffsecurity-ui
```

2. Перенесите схему базы данных из FindFace Security в `PostgreSQL`, создайте группы пользователей с *предустановленными* правами и первого пользователя с правами администратора (т. н. Супер Администратора).

Важно: Супер Администратор не может лишиться прав администратора даже при смене роли.

```
sudo findface-security migrate
sudo findface-security create_groups
sudo findface-security create_default_user
```

3. Создайте структуру биометрической базы данных на основе Tarantool.

```
sudo findface-security make_tnt_schema | sudo tee /etc/ffsecurity/tnt_schema.lua
```

4. Импортируйте переменную `meta_scheme` из файла `tnt_schema.lua`. Откройте файл конфигурации `/etc/tarantool/instances.enabled/FindFace.lua`. Перед секцией `FindFace.start` добавьте строку `dofile("/etc/ffsecurity/tnt_schema.lua")`. В параметрах `FindFace.start` определите `meta_scheme=meta_scheme`.

```
sudo vi /etc/tarantool/instances.enabled/FindFace.lua

dofile("/etc/ffsecurity/tnt_schema.lua")

FindFace.start("127.0.0.1", 8101, {
    license_ntls_server="127.0.0.1:3133",
    facen_size=480,
    meta_scheme = meta_scheme
 })
```

5. Добавьте сервис `findface-tarantool-server` в автозагрузку Ubuntu и запустите его.

```
sudo systemctl enable tarantool@FindFace.service && sudo systemctl start tarantool@FindFace.
↪service
```

6. Откройте файл конфигурации `/etc/ffsecurity/config.py`. Задайте следующие параметры:

   - `SERVICE_EXTERNAL_ADDRESS`: IP-адрес или URL FindFace Security, являющимся приоритетным для Genetec и вебхуков. Если параметр не задан, система использует для работы с данным функционалом `EXTERNAL_ADDRESS`. Для использования Genetec и вебхуков обязательно укажите по крайней мере один из параметров `SERVICE_EXTERNAL_ADDRESS`/ `EXTERNAL_ADDRESS`.
   - `EXTERNAL_ADDRESS`: (Опционально) IP-адрес или URL, который используется для доступа к веб-интерфейсу FindFace Security. Если параметр не задан, система автоматически определяет его как внешний IP-адрес. Для доступа в FindFace Security вы можете использовать оба IP-адреса: как автоопределенный, так и указанный в `EXTERNAL_ADDRESS`.
   - `VIDEO_DETECTOR_TOKEN`: придумайте токен и укажите его в данном параметре, чтобы авторизовать модуль видеодетекции лиц.
   - `VIDEO_MANAGER_ADDRESS`: IP-адрес сервера `findface-video-manager`.
   - `NTLS_HTTP_URL`: IP-адрес сервера `findface-ntls`.
   - `ROUTER_URL`: IP-адрес сервера `findface-security`, который будет получать обнаруженные на видео лица от экземпляров `findface-video-worker`. Адрес указывается внутренний или внешний, в зависимости от сети, в которой `findface-video-worker` взаимодействует с `findface-security`. Измените порт по умолчанию с учетом *настроек переадресации* с HTTP на HTTPS или вообще не указывайте его, оставив только IP-адрес.
   - `SF_API_ADDRESS`: IP-адрес сервера `findface-sf-api`.

Совет: Если необходимо обеспечить безопасность данных, включите *SSL-шифрование*.

Совет: При необходимости установите `'IGNORE_UNMATCHED': True`, чтобы отключить запись события в базу данных, если обнаруженное лицо отсутствует в списках наблюдения (верификация дала отрицательный результат). Данную настройку рекомендуется использовать при большом количестве посетителей. Пороговая степень схожести при верификации лиц определяется параметром `CONFIDENCE_THRESHOLD`.

Совет: Рекомендуется отредактировать значение параметра `MINIMUM_DOSSIER_QUALITY`. Данный параметр определяет минимальное качество лица на фотографии в досье. Если качество лица хуже минимального, пользователь не сможет загрузить такую фотографию в досье. Прямые изображения лиц анфас считаются наиболее качественными. Им соответствуют значения вблизи 0, как правило, отрицательные (такие как -0.00067401276, например). Перевернутые лица и лица, повернутые под большими углами, характеризуются отрицательным значениям от -5 и меньше. По умолчанию `'MINIMUM_DOSSIER_QUALITY': -2`, что соответствует среднему качеству.

Важно: Если вы включили модели распознавания в файле конфигурации `findface-extraction-api`, добавьте следующую строку в секцию `FFSECURITY`: `'EVENTS_FEATURES': ['gender', 'age', 'emotions', 'beard', 'glasses']`, в зависимости от того, какие модели были включены. Данная строка должна быть расположена между

строками SF_API_ADDRESS и LIVENESS_THRESHOLD, как показано в примере ниже. Подробнее см. *Распознавание атрибутов лица*.

```
sudo vi /etc/ffsecurity/config.py

MEDIA_ROOT = "/var/lib/ffsecurity/uploads"
STATIC_ROOT = "/var/lib/ffsecurity/static"

# SERVICE_EXTERNAL_ADDRESS prioritized for webhooks and genetec
SERVICE_EXTERNAL_ADDRESS = 'http://localhost'
EXTERNAL_ADDRESS = "http://172.20.77.58"

DEBUG = False

LANGUAGE_CODE = 'en-us'

TIME_ZONE = 'UTC'

DATABASES = {
   'default': {
       'ENGINE': 'django.db.backends.postgresql',
       'NAME': 'ffsecurity',
   }
}

# use pwgen -sncy 50 1|tr "'" "." to generate your own unique key
SECRET_KEY = 'c8b533847bbf7142102de1349d33a1f6'

FFSECURITY = {
   'VIDEO_DETECTOR_TOKEN': '381b0f4a20495227d04185ab02f5085f',
   'CONFIDENCE_THRESHOLD': 0.75,
   'MINIMUM_DOSSIER_QUALITY': -2,
   'IGNORE_UNMATCHED': False,
   'EXTRACTION_API': 'http://127.0.0.1:18666/',
   'VIDEO_MANAGER_ADDRESS': 'http://127.0.0.1:18810',
   'EVENTS_MAX_AGE': 30,
   'NTLS_HTTP_URL': 'http://127.0.0.1:3185',
   'ROUTER_URL': 'http://172.20.77.58',
   'MONITORING_UPDATE_INTERVAL': 60,
   'SF_API_ADDRESS': 'http://127.0.0.1:18411',
   'EVENTS_FEATURES': ['gender', 'age', 'emotions', 'beard', 'glasses'],
   'LIVENESS_THRESHOLD': 0.75,
   'BEARD_THRESHOLD': 0.7,
}

ASGI_THREADS = 16

UVICORN_SETTINGS = {
    'workers': 'auto',
    'host': 'localhost',
    'port': 8002,
    'ws-workers': 'auto',
    'ws-host': 'localhost',
    'ws-port': 8003,
}

FFSECURITY_UI_CONFIG = {
```

(continues on next page)

(продолжение с предыдущей страницы)

```
    "event": {
        "features": {
            "f_gender_class": ["male", "female"],
            "age": {
                "f_age_gte": "",
                "f_age_lte": ""
            },
            "f_emotions_class": ["angry", "disgust", "fear", "happy", "sad", "surprise"],
            "f_glasses_class": ["none", "eye", "sun"],
            "f_beard_class": ["none", "beard"],
            "f_liveness_class": ["real", "fake"],
        }
    }
}

# integration plugins
INSTALLED_APPS.append('ffsecurity_genetec')  # remove or comment out this line to disable␣
↪genetec integration
```

7. Используя команду `pwgen -sncy 50 1|tr “’” “.”`, сгенерируйте ключ подписи для шифрования сессии (используется в Django) и задайте его в параметре `SECRET_KEY`.

8. Запустите сервисы.

```
sudo systemctl enable findface-security
sudo systemctl start findface-security
```

9. Отключите сервер nginx, активный по умолчанию, и добавьте сервер `findface-security` в список включенных серверов. Перезапустите nginx.

```
sudo rm /etc/nginx/sites-enabled/default

sudo ln -s /etc/nginx/sites-available/ffsecurity-nginx.conf /etc/nginx/sites-enabled/

sudo nginx -s reload
```

1.4.3 Дополнительное развертывание `findface-video-worker` на удаленных серверах

Для отдельной установки сервиса `findface-video-worker` выполните следующие действия:

Совет: Перед тем как приступить к развертыванию, обязательно ознакомьтесь с *системными требованиями*.

Совет: Если на сервере несколько видеокарт, перед развертыванием `findface-video-worker-gpu` изучите раздел *Использование нескольких видеокарт*.

---

1. Загрузите файл инсталлятора `findface-security-and-server-4.1.1.run`.
2. Поместите файл `.run` в любой каталог на сервере установки (например, `/home/username`).
3. Из данного каталога сделайте файл `.run` исполняемым.

```
chmod +x findface-security-and-server-4.1.1.run
```

4. Запустите файл `.run`.

```
sudo ./findface-security-and-server-4.1.1.run
```

Инсталлятор задаст вам несколько вопросов, после чего проверит, соответствует ли сервер системным требованиям. Вопросы следующие:

1. Устанавливаемый продукт: FindFace Video Worker.
2. Тип пакета `findface-video-worker`: CPU или GPU.
3. IP-адрес центрального сервера FindFace Security.

После этого процесс установки будет автоматически запущен.

Примечание: Ответы на вопросы будут сохранены в файл `/tmp/<findface-installer-*>.json`. Отредактируйте его и используйте для установки `findface-video-worker` на других серверах, не отвечая повторно на вопросы инсталлятора.

Примечание: Если `findface-ntls` и/или `findface-video-manager` будут установлены на серверах, отличных от сервера `findface-security`, укажите их IP-адреса в файле конфигурации `findface-video-worker` после установки компонента.

```
sudo vi /etc/findface-video-worker-cpu.ini
sudo vi /etc/findface-video-worker-gpu.ini
```

В параметре `ntls-addr` укажите IP-адрес сервера `findface-ntls`.

```
ntls-addr=127.0.0.1:3133
```

В параметре `mgr-static` укажите IP-адрес сервера `findface-video-manager`, который будет обеспечивать `findface-video-worker` настройками и списком видеопотоков для обработки.

```
mgr-static=127.0.0.1:18811
```

Совет: Для того чтобы автоматически установить `findface-video-worker` на других серверах, не отвечая на вопросы инсталлятора, используйте файл `/tmp/<findface-installer-*>.json`. Запустите инсталлятор следующей командой:

```
sudo ./findface-security-and-server-4.1.1.run -f /tmp/<findface-installer-*>.json
```

Пример файла `/tmp/<findface-installer-*>.json` можно посмотреть в разделе *Файл с параметрами установки*.

1.4.4 Установка моделей нейронных сетей

Для обнаружения и идентификации лиц и их атрибутов (пол, возраст, эмоции, борода, очки и т. д.) `findface-extraction-api` использует нейронные сети.

Если необходим ручной запуск установки моделей, выполните следующие действия:

1. Запустите подготовленный файл `findface-security-and-server-4.1.1.run`.

```
sudo ./findface-security-and-server-4.1.1.run
```

2. Тип установки: `Fully customized installation`.
3. Выберите устанавливаемый компонент FindFace Security: `findface-data`. Для этого сначала снимите выделение со всех компонентов, введя в командной строке `-*`, затем введите порядковый номер компонента: `1`. Введите `done` для сохранения выбора и перехода к следующему шагу.
4. Выберите модели для установки. После этого процесс установки будет автоматически запущен.

Примечание: Вы можете найти установленные модели для распознавания лиц и атрибутов лиц в каталогах `/usr/share/findface-data/models/face/` и `/usr/share/findface-data/models/faceattr/` соответственно.

```
ls /usr/share/findface-data/models/face/
grapefruit_480.cpu.fnk  grapefruit_480.gpu.fnk grapefruit_160.cpu.fnk  grapefruit_160.gpu.fnk

ls /usr/share/findface-data/models/faceattr/
age.v1.cpu.fnk  age.v1.gpu.fnk  beard.v0.cpu.fnk  beard.v0.gpu.fnk  emotions.v1.cpu.fnk  emotions.
↪v1.gpu.fnk  gender.v2.cpu.fnk  gender.v2.gpu.fnk  glasses3.v0.cpu.fnk  glasses3.v0.gpu.fnk ␣
↪liveness.v3.gpu.fnk
```

1.4.5 Полностью настраиваемая установка

Консольный инсталлятор FindFace Security предоставляет несколько вариантов установки, в том числе полностью настраиваемый вариант (установку отдельно выбранных пакетов). Данный вариант в основном используется при развертывании FindFace Security в сильно распределенной среде.

Для запуска полностью настраиваемой установки нужно ответить на вопросы инсталлятора следующим образом:

- Устанавливаемый продукт: `FindFace Security`.
- Тип установки: `Fully customized installation`.
- Устанавливаемые компоненты FindFace Security: для того чтобы выбрать нужные, сначала снимите выделение со всех компонентов, введя в командной строке `-*`, затем введите порядковые номера нужных компонентов через пробел, например: `1 7` (для выбора `findface-data` и `findface-extraction-api`), `13` (`findface-tarantool-server`) или `9` (`findface-upload`). Введите `done` для сохранения выбора и перехода к следующему шагу.
- Связанные вопросы, такие как как тип ускорения: CPU или GPU.

1.4.6 Типичная установка в кластере

Данный раздел посвящен развертыванию FindFace Security в кластерной среде.

Совет: Если после прочтения данного раздела у вас остались вопросы, не стесняйтесь задать их нашим экспертам по адресу support@ntechlab.com.

Развертывание FindFace Security в кластере может быть необходимо по следующим причинам:

- Нужно распределить высокую нагрузку при обработке видео.
- Требуется обработка видеопотоков от группы камер в месте их физического расположения.

Примечание: Актуально для сетей гостиниц, магазинов, при наличии нескольких проходных в одном здании и др.

См.также:

Привязка группы камер к экземпляру findface-video-worker

- Нужно распределить высокую нагрузку при извлечении биометрических образцов.
- В поиске задействовано большое количество лиц, что требует реализации распределенной базы данных.

Перед тем как приступить к развертыванию, постройте архитектурную схему с учетом будущей нагрузки системы и выделенных под нее аппаратных ресурсов (см. *Системные требования*). Наиболее распространенной схемой является следующая:

- Центральный сервер с установленными компонентами `findface-ntls`, `findface-security`, `findface-sf-api`, `findface-video-manager`, `findface-upload`, `findface-video-worker`, `findface-extraction-api`, `findface-tarantool-server`, а также сторонним программным обеспечением.
- Несколько дополнительных серверов для обработки видео с установленным компонентом `findface-video-worker`.
- (При необходимости) Несколько дополнительных серверов для извлечения биометрических образцов с установленным компонентом `findface-extraction-api`.
- (При необходимости) Дополнительные серверы базы данных с несколькими шардами Tarantool на каждом.

Инструкции в настоящем разделе приведены для описанной выше наиболее часто встречающейся схемы развертывания в кластере. В высоконагруженных системах также может потребоваться распределить обработку API-запросов, т. е. организовать несколько серверов `findface-sf-api` и `findface-video-manager`. В этом случае руководствуйтесь инструкциями в разделе *Полностью настраиваемая установка*.

Развертывание FindFace в кластерной среде состоит из следующих этапов:

- *Развертывание центрального сервера*
- *Развертывание серверов обработки видео*
- *Развертывание биометрических серверов*
- *Распределение нагрузки между биометрическими серверами*
- *Организация распределенной базы данных*
- *Настройка сетевого взаимодействия*

Развертывание центрального сервера

Для развертывания центрального сервера FindFace Security выполните следующие действия:

1. На выделенном физическом сервере *установите* FindFace Security из инсталлятора следующим образом:

   - Устанавливаемый продукт: `FindFace Security`.
   - Тип установки: `Single server, multiple video workers`. В этом случае FindFace Security будет установлен в качестве центрального сервера и настроен на взаимодействие с дополнительными удаленными экземплярами `findface-video-worker`.
   - Тип ускорения `findface-video-worker` (на центральном сервере): CPU или GPU, в зависимости от конфигурации оборудования.
   - Тип ускорения `findface-extraction-api` (на центральном сервере): CPU или GPU, в зависимости от конфигурации оборудования.

   По завершении установки в консоль будет выведена информация, необходимая для использования FindFace Security:

```
################################################################################
#                        Installation is complete                              #
################################################################################
- upload your license to http://172.20.77.17/#/license/
- user interface: http://172.20.77.17/
  superuser:      admin
  password:       admin
  documentation:  http://172.20.77.17/doc/
```

2. Загрузите файл лицензии через веб-интерфейс центрального сервера `http://<IP_адрес_сервера>/#/license`. Для доступа в веб-интерфейс используйте логин и пароль администратора, выведенные в консоли.

Примечание: IP-адрес сервера в ссылках на веб-интерфейсы FindFace имеет вид `127.0.0.1` или <IP_адрес_в_сети>, в зависимости от того, принадлежит ли сервер к сети.

Важно: Не передавайте данные `superuser` (Супер Администратора) третьим лицам. Для администрирования системы создайте назначаемого администратора. Отличие назначаемого администратора от Супер Администратора в том, что последний не может лишиться прав администратора даже при смене роли.

---

3. Разрешите лицензируемым сервисам обращаться к серверу лицензирования `findface-ntls` с любого IP-адреса. Для этого, откройте файл конфигурации `/etc/findface-ntls.cfg` и установите `listen = 0.0.0.0:3133`.

```
sudo vi /etc/findface-ntls.cfg

# Listen address of NTLS server where services will connect to.
# The format is IP:PORT
# Use 0.0.0.0:PORT to listen on all interfaces
# This parameter is mandatory and may occur multiple times
# if you need to listen on several specific interfaces or ports.
listen = 0.0.0.1:3133
```

Развертывание серверов обработки видео

На дополнительном сервере для обработки видео установите экземпляр `findface-video-worker`, руководствуясь *пошаговыми инструкциями*. Ответьте на вопросы инсталлятора следующим образом:

- Устанавливаемый продукт: FindFace Video Worker.
- Тип ускорения `findface-video-worker`: CPU или GPU, в зависимости от конфигурации оборудования.
- FindFace Security IP address: IP-адрес центрального сервера.

После этого процесс установки будет автоматически запущен. Ответы на вопросы инсталлятора будут сохранены в файл `/tmp/<findface-installer-*>.json`. Используйте данный файл, чтобы установить `FindFace Video Worker` на других серверах, не отвечая на вопросы инсталлятора повторно. Для этого запустите инсталлятор командой:

```
sudo ./findface-security-and-server-4.1.0.run -f /tmp/<findface-installer-*>.json
```

Примечание: Если `findface-ntls` и/или `findface-video-manager` установлены на других серверах, чем `findface-security`, укажите их IP-адреса после установки в файле конфигурации `findface-video-worker`.

```
sudo vi /etc/findface-video-worker-cpu.ini
sudo vi /etc/findface-video-worker-gpu.ini
```

В параметре `ntls-addr` укажите IP-адрес сервера `findface-ntls`.

```
ntls-addr=127.0.0.1:3133
```

В параметре `mgr-static` укажите IP-адрес сервера `findface-video-manager`, который будет обеспечивать `findface-video-worker` настройками и списком видеопотоков для обработки.

```
mgr-static=127.0.0.1:18811
```

Развертывание биометрических серверов

На дополнительном сервере для извлечения биометрических образцов установите экземпляр `findface-extraction-api` из консольного инсталлятора. Ответьте на вопросы инсталлятора следующим образом:

- Устанавливаемый продукт: `FindFace Security`.
- Тип установки: `Fully customized installation`.
- Устанавливаемые компоненты FindFace Security: `findface-extraction-api` и `findface-data`. Для того чтобы их выбрать, сначала снимите выделение со всех компонентов, введя в командной строке `-*`, затем введите порядковые номера `findface-extraction-api` и `findface-data` через пробел: `1 7`. Введите `done` для сохранения выбора и перехода к следующему шагу.
- Тип ускорения `findface-extraction-api`: CPU или GPU.
- Необходимость в изменении файла конфигурации `findface-extraction-api`: укажите IP-адрес сервера `findface-ntls`.

- Устанавливаемые модели нейронных сетей: CPU/GPU-модель для извлечения биометрических данных лица (обязательна для установки) и (опционально) CPU/GPU модели для распознавания пола, возраста, эмоций, очков и/или бороды. Для того чтобы выбрать нужные модели, сначала снимите установленное по умолчанию выделение всех моделей, введя в командной строке `-*`, затем введите порядковые номера нужных моделей через пробел, например: `8 2` для выбора соответственно GPU-модели для извлечения биометрических данных и GPU-модели для распознавания возраста. Введите `done` для сохранения выбора и перехода к следующему шагу. Удостоверьтесь, что для каждой модели выбран правильный тип ускорения CPU или GPU: он должен совпадать с типом ускорения `findface-extraction-api`. Обратите внимание, что `findface-extraction-api` на CPU может работать только с CPU-моделями, в то время как `findface-extraction-api` на GPU поддерживает как CPU-, так и GPU-модели. Подробнее см. *Распознавание атрибутов лица*.

  Доступны следующие модели:

Атрибут лица	Ускорение	Пакет
биометрия лица	CPU	findface-data-grapefruit-480-cpu_3.0.0_amd64.deb findface-data-grapefruit-160-cpu_3.0.0_amd64.deb
	GPU	findface-data-grapefruit-480-gpu_3.0.0_amd64.deb findface-data-grapefruit-160-gpu_3.0.0_amd64.deb
возраст	CPU	findface-data-age.v1-cpu_3.0.0_amd64.deb
	GPU	findface-data-age.v1-gpu_3.0.0_amd64.deb
пол	CPU	findface-data-gender.v2-cpu_3.0.0_amd64.deb
	GPU	findface-data-gender.v2-gpu_3.0.0_amd64.deb
эмоции	CPU	findface-data-emotions.v1-cpu_3.0.0_amd64.deb
	GPU	findface-data-emotions.v1-gpu_3.0.0_amd64.deb
очки	CPU	findface-data-glasses3.v0-cpu_3.0.0_amd64.deb
	GPU	findface-data-glasses3.v0-gpu_3.0.0_amd64.deb
борода	CPU	findface-data-beard.v0-cpu_3.0.0_amd64.deb
	GPU	findface-data-beard.v0-gpu_3.0.0_amd64.deb

После этого процесс установки будет автоматически запущен. Ответы на вопросы инсталлятора будут сохранены в файл `/tmp/<findface-installer-*>.json`. Используйте данный файл, чтобы установить `findface-extraction-api` на других серверах, не отвечая на вопросы инсталлятора повторно.

```
sudo ./findface-security-and-server-4.1.0.run -f /tmp/<findface-installer-*>.json
```

После развертывания биометрических серверов *распределите* между ними нагрузку.

Распределение нагрузки между биометрическими серверами

Распределение нагрузки между несколькими биометрическими серверами выполняется через балансировщик нагрузки. Приведенная ниже пошаговая инструкция демонстрирует балансировку нагрузки с помощью `nginx` в режиме round-robin для 3-х экземпляров `findface-extraction-api`, расположенных на различных физических серверах. Один экземпляр установлен на центральном сервере FindFace Security (`172.168.1.9`), 2 других на дополнительных удаленных серверах (`172.168.1.10`, `172.168.1.11`). Если в системе присутствует большее количество биометрических серверов, балансировка нагрузки выполняется по аналогии.

Совет: Вы можете использовать любой удобный вам балансировщик нагрузки. Руководство по его использованию ищите в соответствующей справочной документации.

Для балансировки нагрузки между экземплярами `findface-extraction-api` выполните следующие действия:

1. Назначьте т. н. сервер шлюза для балансируемой группы биометрических серверов. Им может стать центральный сервер FindFace Security (рекомендуется) или любой другой сервер с установленным nginx.

   Важно: Вам нужно будет указать IP-адрес шлюза при настройке *распределенной сети* FindFace Security.

   Совет: Вы можете установить nginx следующим образом:

   ```
   sudo apt update
   sudo apt install nginx
   ```

2. На сервере шлюза создайте новый файл конфигурации Nginx.

   ```
   sudo vi /etc/nginx/sites-available/extapi
   ```

3. Вставьте следующий текст в созданный файл конфигурации. В директиве `upstream` (`upstream extapibackends`) замените примерные IP-адреса на актуальные IP-адреса биометрических серверов. В директиве `server` в параметре `listen` укажите номер слушающего порта сервера шлюза. Вам потребуется указать данный порт при настройке *распределенной сети* FindFace Security.

   ```
   upstream extapibackends {
           server 172.168.1.9:18666; ## ``findface-extraction-api`` on principal server
           server 172.168.1.10:18666; ## 1st additional extraction server
           server 127.168.1.11:18666; ## 2nd additional extraction server
   }
   server {
           listen 18667;
           server_name extapi;
           client_max_body_size 64m;
           location / {
                   proxy_pass http://extapibackends;
                   proxy_next_upstream error;
           }
           access_log /var/log/nginx/extapi.access_log;
           error_log /var/log/nginx/extapi.error_log;
   }
   ```

4. Включите балансировщик нагрузки в nginx.

   ```
   sudo ln -s /etc/nginx/sites-available/extapi /etc/nginx/sites-enabled/
   ```

5. Перезапустите nginx.

   ```
   sudo service nginx restart
   ```

6. На центральном сервере и каждом из дополнительных биометрических серверов откройте файл конфигурации `/etc/findface-extraction-api.ini`. Замените адрес localhost в параметре `listen` на адрес, который вы указали до этого в директиве `upstream extapibackends` файла конфигурации Nginx `/etc/nginx/sites-available/extapi`. В нашем примере адрес 1-го дополнительного биометрического сервера должен быть заменен на следующий:

```
sudo vi /etc/findface-extraction-api.ini

listen: 172.168.1.10:18666
```

7. Перезапустите `findface-extraction-api` на центральном сервере и каждом дополнительном биометрическом сервере.

```
sudo systemctl restart findface-extraction-api.service
```

Балансировка нагрузки успешно настроена. Не забудьте указать актуальный IP-адрес и слушающий порт сервера шлюза при настройке *распределенной сети* FindFace Security.

Организация распределенной базы данных

Компонент `findface-tarantool-server` соединяет базу данных Tarantool и компонент `findface-sf-api`, передавая результаты поиска от базы данных в `findface-sf-api` для дальнейшей обработки. Для увеличения скорости поиска на каждом сервере с базой данных Tarantool могут быть созданы многочисленные сегменты (шарды) `findface-tarantool-server`. Их параллельное функционирование приводит к значительному увеличению производительности. Каждый шард может обрабатывать приблизительно до 10 000 000 лиц. При развертывании `findface-tarantool-server` из инсталлятора шарды создаются автоматически с учетом аппаратной конфигурации физического сервера.

Для того чтобы распределить биометрическую базу данных, установите `findface-tarantool-server` на каждом сервере базы данных. Ответьте на вопросы инсталлятора следующим образом:

- Устанавливаемый продукт: `FindFace Security`.
- Тип установки: `Fully customized installation`.
- Устанавливаемые компоненты FindFace Security: `findface-tarantool-server`. Для того чтобы его выбрать, сначала снимите выделение со всех компонентов, введя в командной строке `-*`, затем введите порядковый номер `findface-tarantool-server: 13`. Введите `done` для сохранения выбора и перехода к следующему шагу.

После этого процесс установки будет автоматически запущен. Ответы на вопросы инсталлятора будут сохранены в файл `/tmp/<findface-installer-*>.json`. Используйте данный файл, чтобы установить `findface-tarantool-server` на других серверах, не отвечая на вопросы инсталлятора повторно.

```
sudo ./findface-security-and-server-4.1.0.run -f /tmp/<findface-installer-*>.json
```

В результате установки шарды `findface-tarantool-server` будут автоматически установлены в количестве `N = max (min (mem_mb // 2000, cpu_cores), 1)`, т. е. равном размеру оперативной памяти в Мб, разделенному на 2000, или количеству физических ядер процессора (но не менее 1 шарда).

Обязательно укажите IP-адреса и порты шардов при настройке *распределенной сети* FindFace Security. Для того чтобы узнать номера портов, на каждом сервере базы данных выполните следующую команду:

```
sudo cat /etc/tarantool/instances.enabled/*shard* | grep -E ".start|(listen =)"`
```

Будет возвращен следующий результат:

```
    listen = '127.0.0.1:33001',
FindFace.start("127.0.0.1", 8101, {
    listen = '127.0.0.1:33002',
FindFace.start("127.0.0.1", 8102, {
```

Номера портов шардов указаны в секции `FindFace.start`: `8101`, `8102` и т. д.

Настройка сетевого взаимодействия

После развертывания компонентов FindFace Security настройте их взаимодействие по сети. Выполните следующие действия:

1. Откройте файл конфигурации `/etc/findface-sf-api.ini`:

```
sudo vi /etc/findface-sf-api.ini
```

Задайте следующие параметры:

Параметр	Описание
`extraction-api -> extraction-api`	IP-адрес и слушающий порт сервера, являющегося *шлюзом* для биометрических серверов с настроенной балансировкой нагрузки.
`storage-api -> shards -> master`	IP-адрес и порт мастера шарда `findface-tarantool-server`. Остальные шарды прописываются по аналогии.
`upload_url`	Путь в WebDAV nginx, по которому в компонент `findface-upload` будут отправляться исходные изображения, миниатюры и нормализованные изображения лиц.

```
...
extraction-api:
  extraction-api: http://172.168.1.9:18667

...
webdav:
  upload-url: http://127.0.0.1:3333/uploads/

...
storage-api:
  ...
  shards:
  - master: http://172.168.1.9:8101/v2/
    slave: ''
  - master: http://172.168.1.9:8102/v2/
    slave: ''
  - master: http://172.168.1.12:8101/v2/
    slave: ''
  - master: http://172.168.1.12:8102/v2/
    slave: ''
  - master: http://172.168.1.13:8102/v2/
    slave: ''
  - master: http://172.168.1.13:8102/v2/
    slave: ''
```

2. Откройте файл конфигурации `/etc/ffsecurity/config.py`.

```
sudo vi /etc/ffsecurity/config.py
```

Задайте следующие параметры:

Параметр	Описание
SERVICE_EXTERNAL_ADDRESS	IP-адрес или URL FindFace Security, являющимся приоритетным для Genetec и вебхуков. Если параметр не задан, система использует для работы с данным функционалом EXTERNAL_ADDRESS. Для использования Genetec и вебхуков обязательно укажите по крайней мере один из параметров SERVICE_EXTERNAL_ADDRESS/ EXTERNAL_ADDRESS.
EXTERNAL_ADDRESS	(Опционально) IP-адрес или URL, который используется для доступа к вебинтерфейсу FindFace Security. Если параметр не задан, система автоматически определяет его как внешний IP-адрес. Для доступа в FindFace Security вы можете использовать оба IP-адреса: как автоопределенный, так и указанный в EXTERNAL_ADDRESS.
VIDEO_DETECTOR_TOKEN	Придумайте токен и укажите его в данном параметре, чтобы авторизовать модуль видеодетекции лиц.
VIDEO_MANAGER_ADDRESS	IP-адрес сервера findface-video-manager.
NTLS_HTTP_URL	IP-адрес сервера findface-ntls.
ROUTER_URL	Внешний IP-адрес сервера findface-security, который будет получать обнаруженные лица от экземпляра(ов) findface-video-worker.
SF_API_ADDRESS	IP-адрес сервера findface-sf-api.
EXTRACTION_API	IP-адрес и слушающий порт сервера, являющегося *шлюзом* для биометрических серверов с настроенной балансировкой нагрузки.

```
sudo vi /etc/ffsecurity/config.py

...
# SERVICE_EXTERNAL_ADDRESS prioritized for webhooks and genetec
SERVICE_EXTERNAL_ADDRESS = 'http://localhost'
EXTERNAL_ADDRESS = 'http://127.0.0.1'



...
FFSECURITY = {
    'VIDEO_DETECTOR_TOKEN': '7ce2679adfc4d74edcf508bea4d67208',
    ...
    'EXTRACTION_API': 'http://172.168.1.9:18667/',
    'VIDEO_MANAGER_ADDRESS': 'http://127.0.0.1:18810',
    ...
    'NTLS_HTTP_URL': 'http://127.0.0.1:3185',
    'ROUTER_URL': 'http://172.168.1.9',
    ...
    'SF_API_ADDRESS': 'http://127.0.0.1:18411',
    ...
}
```

На этом установка FindFace Security в кластерной среде будет завершена.

1.5 Первые шаги после установки

После успешного развертывания FindFace Security пришло время открыть *веб-интерфейс* и начать работу. В этой главе вы найдете рекомендуемую последовательность шагов, следуя которой вы сможете задействовать весь спектр функциональных возможностей FindFace Security.

В этой главе:

1.5.1 Добавление камер

1. *Создайте новую группу камер* или используйте созданную по умолчанию. Группа камер — это системный объект, который позволяет группировать камеры с учетом их физического расположения. К примеру, камеры, расположенные на входе в здание, можно объединить в одну группу и работать с ними как с единым целым.
2. *Добавьте камеры* в созданную группу и *проверьте их работу*.

Дополнительно:

1. Настройте систему на обработку видеопотоков с группы камер в месте их физического расположения. Это может быть актуально в распределенной архитектуре. *Подробнее*.
2. Рассмотрите возможность включения дедупликации событий, если сцены наблюдения камер в группе перекрываются. Данная функция позволяет исключить регистрацию одинаковых событий в пределах одной группы камер. *Подробнее*.

1.5.2 Составление списков наблюдения и загрузка досье

1. *Создайте новый список наблюдения* или используйте созданный по умолчанию. Список наблюдения — это системный объект, с помощью которого выполняется классификация персон по произвольным критериям: черный список, розыск, VIP, персонал и т. д.
2. Загрузите досье и добавьте их в созданный список наблюдения *вручную*, *пакетно через веб-интерфейс* или используя функцию *пакетной загрузки через консоль*.

Дополнительно:

1. *Распределите базу данных досье* между несколькими серверами. База данных досье будет доступна для редактирования на головном сервере, а для чтения и мониторинга — на ведомых.
2. *Настройте содержимое досье*. Создайте дополнительные поля, вкладки и поисковые фильтры.

1.5.3 Создание пользователей и предоставление им прав

1. Ознакомьтесь со списком *предустановленных пользовательских ролей* и при необходимости *создайте новые*.
2. *Добавьте пользователей* в систему и предоставьте им полномочия.

1.5.4 Мониторинг лиц

По умолчанию FindFace Security отслеживает только незарегистрированные в системе лица, т. е. лица, для которых *отсутствуют совпадения с досье*. Для того чтобы включить отслеживание лиц из пользовательского списка наблюдения, сделайте это список *активным*. Вы также можете включить звуковое оповещение и сделать обязательным принятие вручную связанных со списком событий.

Дополнительно:

1. Добавьте информации в события, включив распознавание пола, возраста, эмоций, бороды и очков. *Подробнее*.
2. Защитите систему от спуфинга, включив функцию распознавания живых лиц (Liveness). *Подробнее*.

1.5.5 Организация видеонаблюдения

Настройте раскладку камер для базового видеонаблюдения.

1.5.6 Работа FindFace Security

1. *Идентифицируйте обнаруженные на видеоизображении лица*, проверяя их в реальном времени на совпадение с лицами из списков наблюдения. Работайте с историей событий, используя различные фильтры.
2. Работайте с *эпизодами*. Эпизод — это набор событий идентификации, в которых фигурируют лица одного и того же человека, обнаруженные в течение определенного периода времени. Поскольку информация о событиях отображается на вкладке *События* в произвольном порядке, обработка большого количества разнородных событий может быть делом затруднительным и неэффективным. С функцией Эпизоды, система использует искусственный интеллект для группировки входящих событий на основе времени обнаружения и схожести лиц. Это позволяет с легкостью обрабатывать разнородные события даже в больших количествах.
3. Выполняйте идентификацию (поиск) лиц по следующим базам данных:
   - База данных обнаруженных лиц. *Подробнее*.
   - База данных досье. *Подробнее*.
4. Выполняйте *поиск в архивных видео* лиц из списков наблюдения.
5. *Сравнивайте лица* вручную, чтобы проверить их на принадлежность одному человеку.
6. Используйте *мобильное приложение*.

1.5.7 Базовое обслуживание системы

1. *Настройте* автоматическое удаление старых событий из базы данных.
2. При необходимости *удалите* старые события из базы даных вручную.
3. Регулярно создавайте *резервную копию* базы данных.

1.5.8 Больше возможностей

1. Настройте *вебхуки* для автоматической отправки уведомлений об определенных событиях на заданный URL-адрес. При наступлении нужного события FindFace Security отправит HTTP-запрос на URL-адрес, указанный в настройках вебхука. Вебхуки можно использовать для решения разнообразных задач, например, для уведомления пользователя об определенном событии, вызова определенных действий на целевом веб-сайте, при решении задач безопасности, таких как удаленное автоматическое управление доступом и др. *Подробнее*.
2. Задействуйте функции FindFace Security через *HTTP API*.
3. Ознакомьтесь со списком наших *партнерских интеграций*.
4. Задействуйте *плагины*, что задать собственные правила обработки обнаруженных на видео лиц.

См.также:

- *Управление видеокамерами*
- *Мониторинг лиц. Управление базой данных досье*
- *Управление пользователями*
- *Расширенный функционал*
- *Обслуживание и устранение неисправностей*

1.6 Работа с FindFace Security

Работа с FindFace Security выполняется через веб-интерфейс. Для того чтобы отобразить веб-интерфейс, в адресной строке браузера введите базовый адрес веб-интерфейса и пройдите авторизацию.

Примечание: Базовый адрес задается при *установке* FindFace Security.

Важно: Для первого входа в систему после развертывания FindFace Security используйте учетную запись администратора, созданную при *установке*. Для создания других пользователей см. раздел *Управление пользователями*.

Веб-интерфейс имеет удобный и интуитивный дизайн и обеспечивает доступ к следующим функциям:

- Объединение камер в группы в зависимости от их физического расположения. Добавление и настройка камеры. См. *Управление камерами*.
- Управление списками наблюдения. Создание досье вручную и пакетно. См. *Мониторинг лиц. Управление базой данных досье*.
- *Управление пользователями и ролями*.

- *Идентификация лиц в видеофайле*.
- *Основные настройки*. Изменение порога срабатывания системы (порога положительной верификации лиц) и настройка автоматической очистки базы данных от старых событий.
- *Сравнение 2-х лиц*. Проверка 2-х лиц на идентичность.
- Идентификация лиц по базе событий в *режиме реального времени* как на живом (видеопоток), так и на архивном (видеофайл) видео. *Эпизоды событий*. Идентификация лиц *по базе событий и досье*. *Видеонаблюдение*.

1.6.1 Управление видеокамерами

Для настройки видео-идентификации лиц добавьте камеры в FindFace Security, сгруппировав их с учетом расположения.

Примечание: Права на создание групп камер и камер настраиваются в разрешениях пользователя (см. *Управление пользователями*).

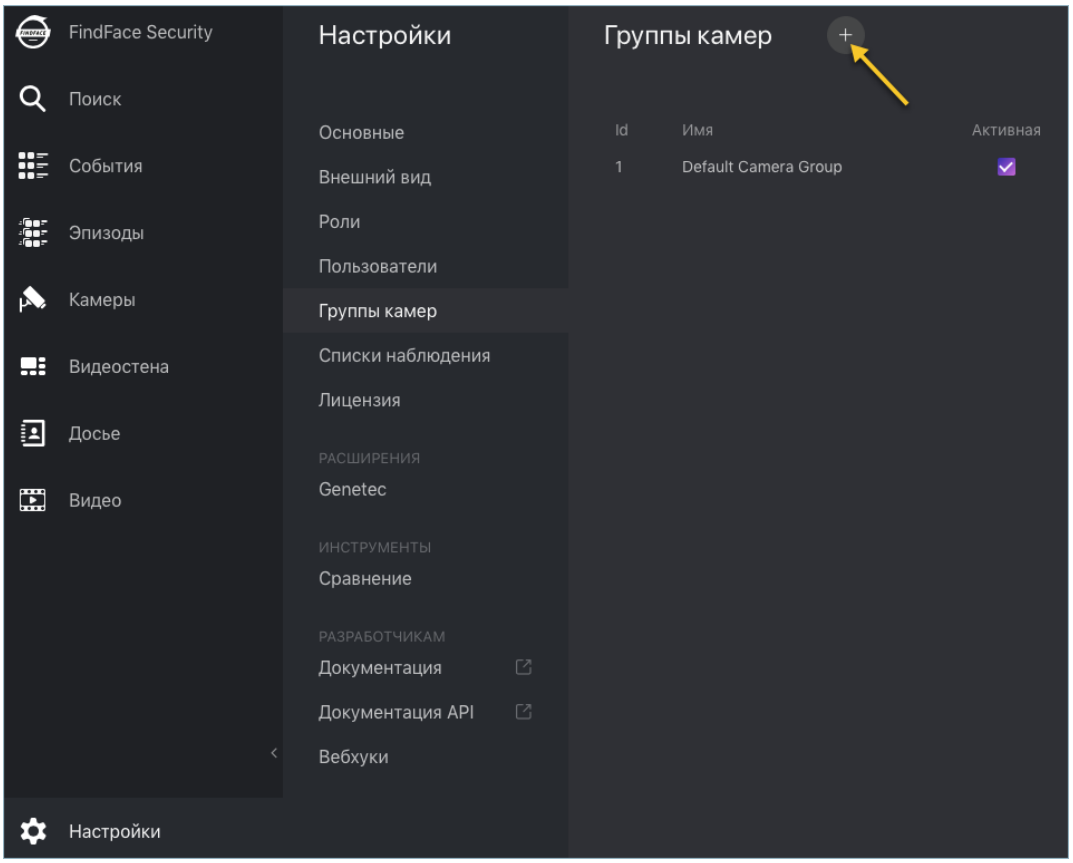
В этой главе:

Создание группы камер

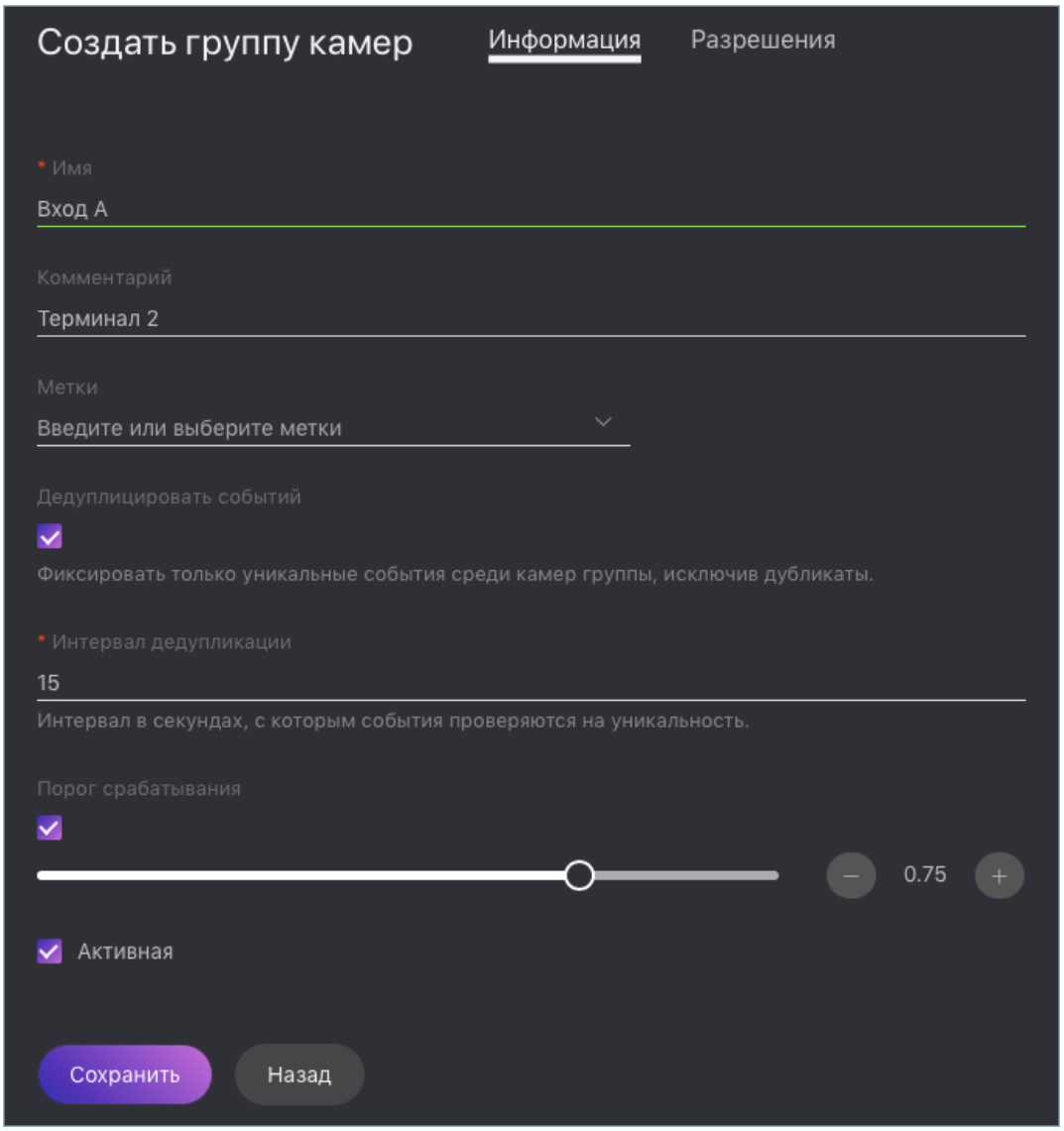
Совет: В системе доступна группа камер по умолчанию.

Для создания группы камер выполните следующие действия:

1. Перейдите на вкладку *Настройки*. Выберите *Группы камер*.
2. Нажмите *+*.



3. На вкладке *Информация*, введите имя группы и при необходимости комментарий к ней.



4. Если вам нужно выделить определенный экземпляр `findface-video-worker` для обработки видеопотоков с данной группы камер, создайте или выберите из уже созданных одну или несколько меток.

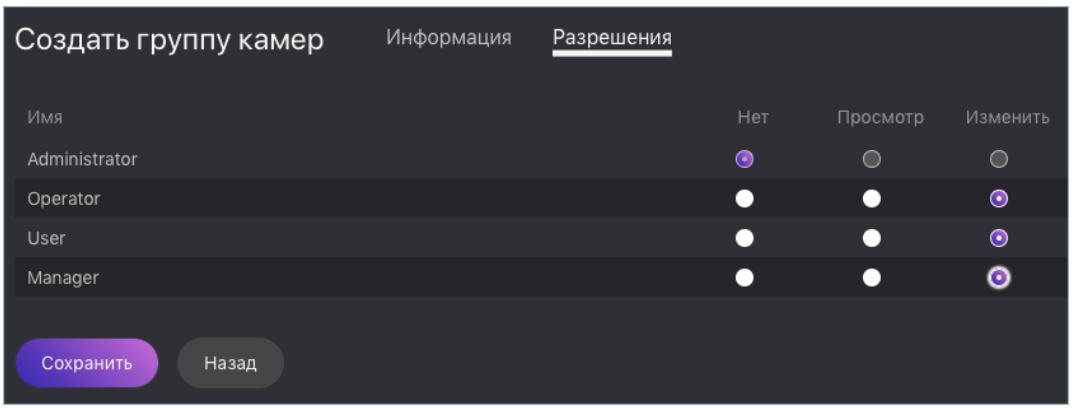
Примечание: Для того чтобы завершить выделение, перечислите метки в файле конфигурации `findface-video-worker`. Подробнее см. *Привязка группы камер к экземпляру findface-video-worker*.

---

5. Если события от камер, принадлежащих одной группе, требуется дедуплицировать, т. е. исключить одинаковые события, поставьте флажок *Дедуплицировать события* и задайте в секундах интервал дедупликации (интервал, с которым события проверяются на уникальность).

Предупреждение: Используйте дедупликацию очень осторожно. Если камеры из одной группы наблюдают разные сцены, некоторые лица могут быть пропущены. Подробнее см. *Дедупликация событий*.

6. По умолчанию на всех группах камер в системе используется *универсальный порог* срабатывания, оптимальный для большинства случаев распознавания лиц. Для того чтобы установить индивидуальный порог для группы камер, поставьте флажок *Порог срабатывания* и укажите нужное пороговое значение.
7. Поставьте флажок *Активная*.
8. Нажмите *Сохранить*.
9. На вкладке *Разрешения* назначьте права на работу с группой камер, указав, пользователям с какими ролями разрешено изменять/просматривать ее настройки.

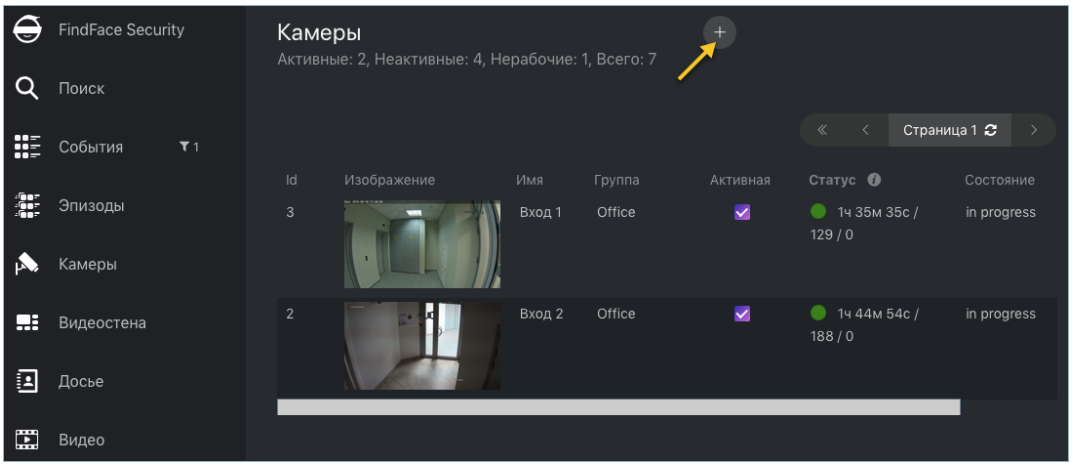


10. Нажмите *Сохранить*.

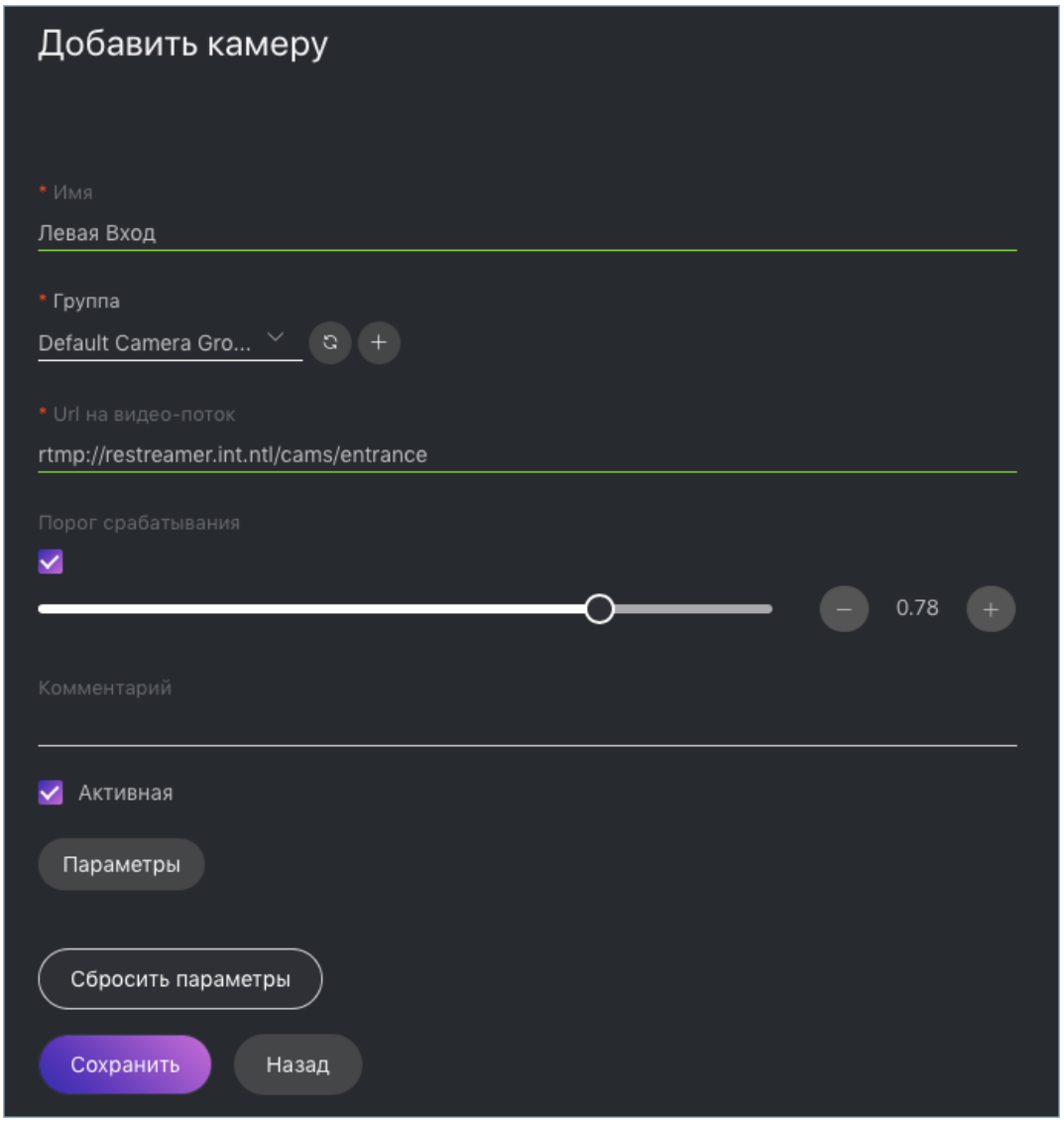
Добавление камеры

Для добавления камеры в группу выполните следующие действия:

1. Перейдите на вкладку *Камеры*.
2. Нажмите *+*.



3. Введите название камеры и добавьте ее в одну из групп. При необходимости введите комментарий к камере.



4. Задайте URL камеры или адрес видеофайла для обработки, например, `file:///data/some.mp4`.
5. По умолчанию на всех камерах в системе используется *универсальный порог* срабатывания, оптимальный для большинства случаев распознавания лиц. Для того чтобы установить индивидуальный порог для камеры, поставьте флажок *Порог срабатывания* и укажите нужное пороговое значение.
6. Поставьте флажок *Активная*.
7. Для того чтобы настроить обработку видео, нажмите на кнопку *Параметры* и внесите изменения:
   - *Минимальное качество изображения лица* (`filter_min_quality`): Минимальное качество изображения лица для отправки на сервер. Определяется эмпирически: отрицательные значения вблизи 0 = наиболее качественные прямые изображения лиц анфас, -1 = хорошее качество, -2 = удовлетворительное качество, отрицательные значения -5 и меньше = перевернутые лица и лица, повернутые под большими углами, распознавание может быть неэффективным.
   - *Минимальный размер лица* (`filter_min_face_size`): Минимальный размер лица в пикселях для отправки на сервер. Если 0, фильтр выключен.

- *Максимальный размер лица* (`filter_max_face_size`): Максимальный размер лица в пикселях для отправки на сервер.
- *Качество сжатия* (`jpeg_quality`): Качество сжатия полного кадра для отправки.
- *Опции FFMPEG* (`ffmpeg_params`): Опции ffmpeg для видеопотока. Задаются массивом строк ключ=значение, например, "rtsp_transport=tcp ss=00:20:00".
- *Буферный режим* (`overall`): Буферный режим. Отправлять для лица один кадр наилучшего качества.
- *Временной интервал* (`realtime_post_interval`): Временной интервал в миллисекундах, в течение которого в режиме реального времени выбирается лучший кадр с лицом.
- *Отправлять лучший кадр* (`realtime_post_every_interval`): Если true, отправлять лучший кадр в каждом интервале времени realtime_post_interval в режиме реального времени. Если false, отправлять лучший кадр, только если его качество улучшилось по сравнению с предыдущим отправленным кадром.
- *Время ожидания ответа на запрос* (`router_timeout_ms`): Время ожидания в миллисекундах ответа на отправленный запрос с лицом.
- *Получать временные метки из потока* (`use_stream_timestamp`): Если true, отправлять на сервер временные метки полученные из потока. Если false, отправлять текущие дату и время.
- *Прибавлять к временным меткам* (`start_stream_timestamp`): Прибавлять указанное количество секунд к временным меткам из потока.
- *Ограничение скорости проигрывания* (`play_speed`): Если меньше нуля, то скорость не ограничивается. В остальных случаях поток читается со скоростью `play_speed`. Не применимо для потоков с камер видеонаблюдения.
- *Регион слежения* (`ROT`): Включает детектирование и отслеживание лиц только внутри заданной прямоугольной области. Используйте данную опцию, чтобы уменьшить нагрузку на сервер.
- *Регион захвата лица* (`ROI`): Включает отправку на сервер лиц, обнаруженных только внутри интересующей области.

Совет: Для задания ROT/ROI удобно использовать визуальный мастер. Сначала создайте камеру без ROT/ROI, затем откройте ее для редактирования и нажмите на кнопку *Параметры*. Вы увидите визуальный мастер.

При необходимости задайте опциональные параметры обработки видео. Для это нажмите на кнопку *Дополнительные параметры*.

- *Формат FFMPEG* (`ffmpeg_format`): Передать формат FFMPEG (mxg, flv, и т. д.), если он не может быть автоматически определен.
- *Проверять SSL-сертификат* (`router_verify_ssl`): Если true, проверять SSL-сертификат сервера при отправке на него лиц через https. Если false, может быть принят самоподписанный сертификат.
- *Минимальная интенсивность движения* (`imotion_threshold`): Минимальная интенсивность движения, которая будет регистрироваться детектором движения.

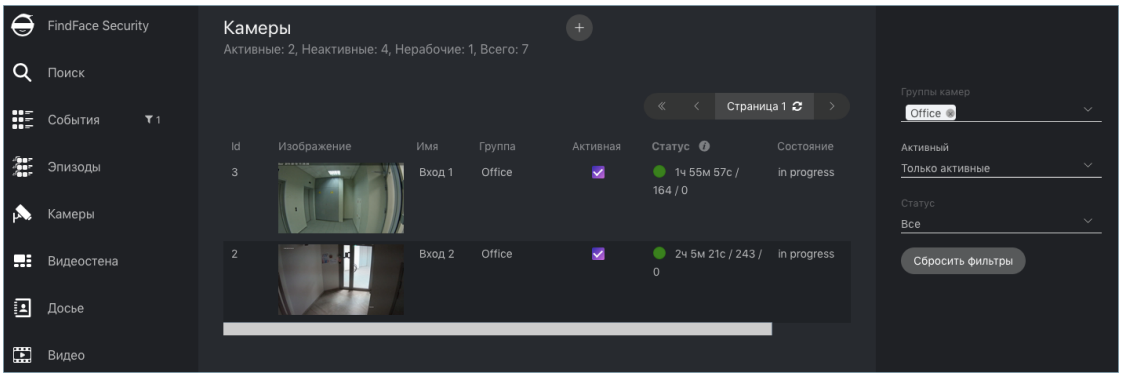
8. Нажмите *Сохранить*.

Примечание: Каждая камера связана с так называемым job-заданием, задачей на обработку видео, содержащей параметры конфигурации и данные видеопотока, которая назначается определенному эк-

экземпляру `findface-video-worker`. Данная задача может быть перезапущена (см . *Мониторинг работы камер*).

Мониторинг работы камер

Мониторинг работы камер выполняется на вкладке *Камеры*.



Статусы камер:

- Зеленый: идет обработка видеопотока с камеры, проблем не обнаружено.
- Желтый: камера работает менее 30 секунд или имеют место одна или несколько ошибок при отправке лиц.
- Красный: камера не работает.
- Серый: камера отключена.

Для каждой камеры приводятся следующие статистические данные по обработке видеопотока: длительность обработки/количество успешно отправленных лиц/количество лиц, обработанных с ошибками после последнего перезапуска job-задания.

Примечание: Каждая камера связана с так называемым job-заданием, задачей на обработку видео, содержащей параметры конфигурации и данные видеопотока, которая назначается определенному экземпляру `findface-video-worker`. Данная задача может быть перезапущена.

Для перезапуска job-задания откройте настройки камеры и нажмите на кнопку *Перезапустить*. При этом количество ошибок будет обнулено.

При большом количестве камер в системе используйте следующие фильтры:

- *Группы камер*,
- *Активный*,
- *Статус*.

См.также:

- *Привязка группы камер к экземпляру findface-video-worker*
- *Дедупликация событий*

1.6.2 Мониторинг лиц. Управление базой данных досье

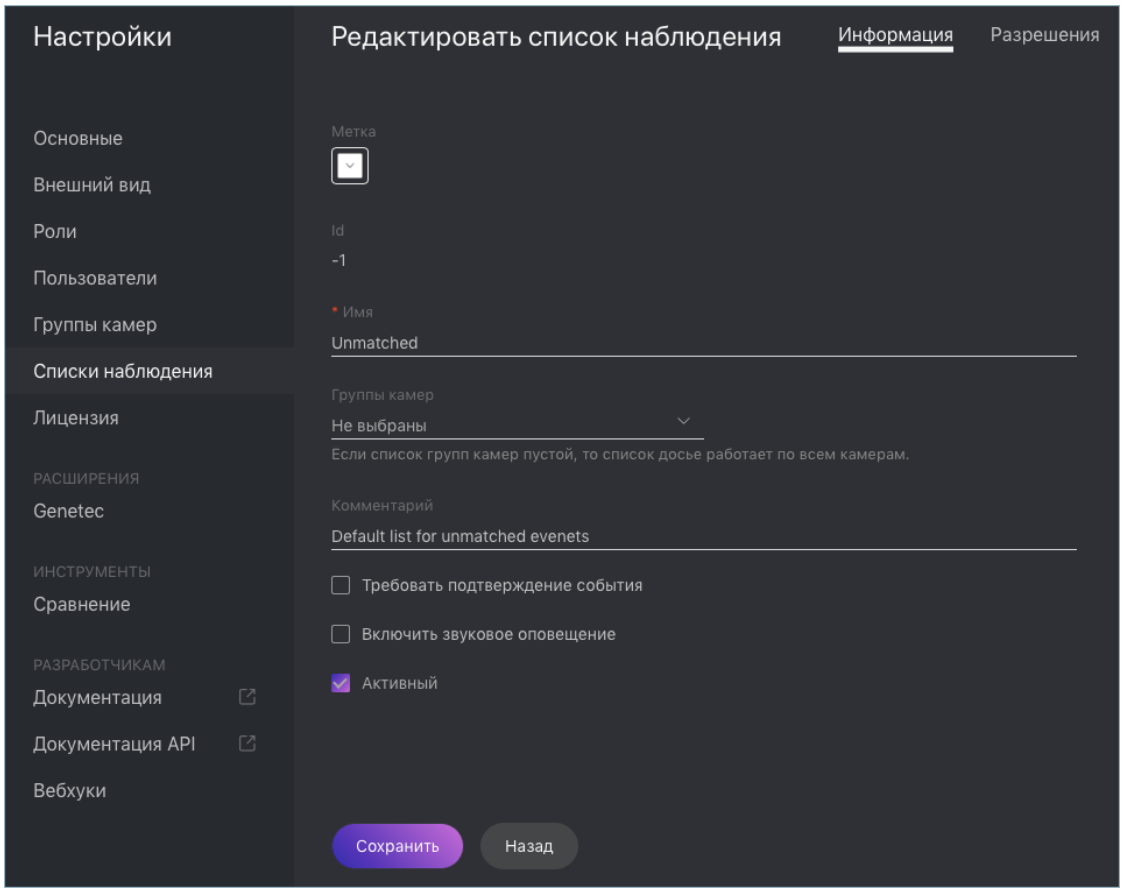
Данная глава посвящена мониторингу обнаруженных лиц и работе с базой данных досье. Каждое досье содержит одну или несколько фотографий персоны и классифицируется по принадлежности к тому или иному списку наблюдения, например, к черному или белому в самом простом случае. Вы можете создать несколько списков наблюдения, например, в зависимости от уровня опасности или, наоборот, статуса персоны.

Совет: Для автоматического создания большого количества досье используйте *функционал пакетной загрузки*.

В этом разделе:

Мониторинг незарегистрированных лиц

Базовая конфигурация FindFace Security уже содержит предустановленный список наблюдения, предназначенный для мониторинга незарегистрированных в системе лиц, т. е. лиц, для которых отсутствуют совпадения с досье. Данный список наблюдения не может быть удален из системы. Для редактирования настроек или дезактивации списка, перейдите на вкладку *Настройки*. Выберите *Списки наблюдения* и откройте настройки списка, щелкнув *Unmatched* в таблице.



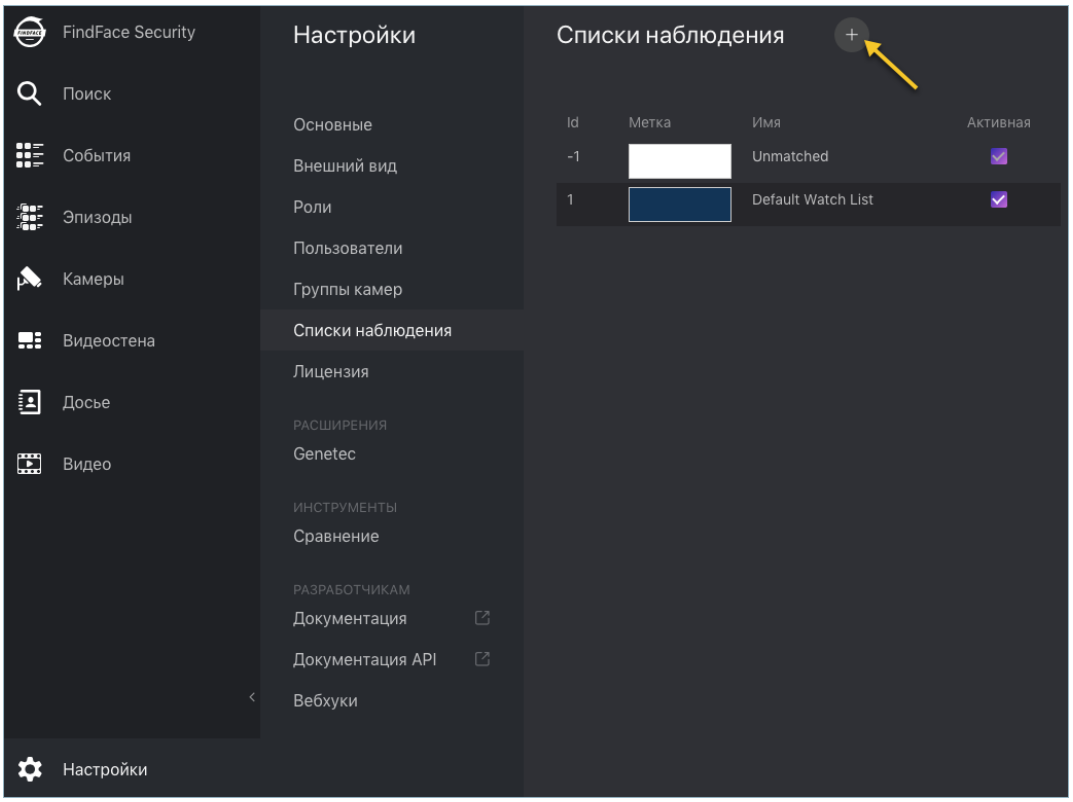
Примечание: Для того чтобы отобразить в списке событий только незарегистрированные лица, выберите значение *Unmatched* в фильтре *Списки наблюдения* на вкладке *События* (подробнее см. *Идентификация лиц в режиме реального времени*).

Создание списка наблюдения

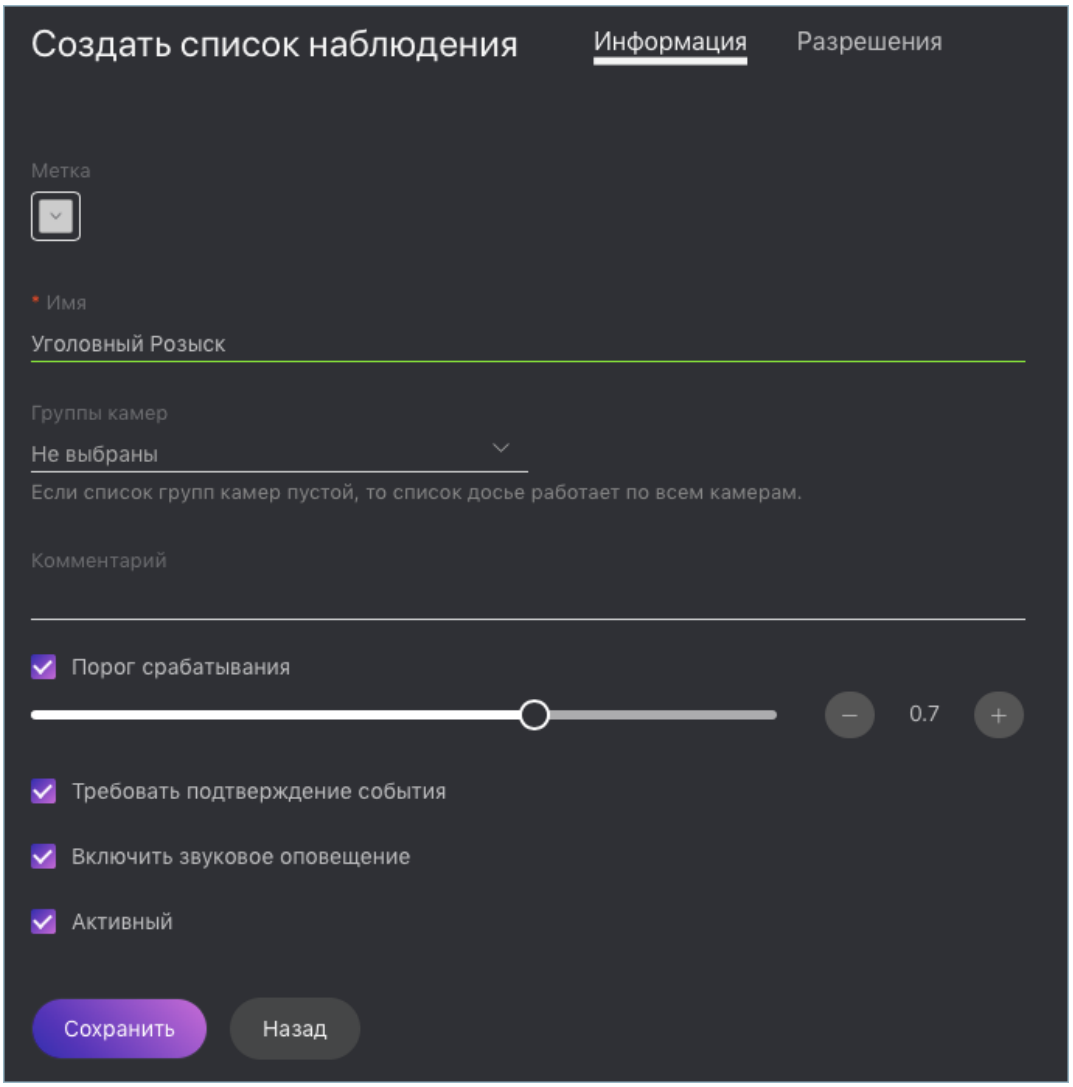
Для создания пользовательского списка наблюдения выполните следующие действия:

Совет: Помимо списка наблюдения *Unmatched* для мониторинга незарегистрированных лиц, в системе по умолчанию уже создан список для мониторинга лиц по базе досье. Данный список наблюдения не может быть удален из системы.

1. Перейдите на вкладку *Настройки*. Выберите *Списки наблюдения*.
2. Нажмите *+*.

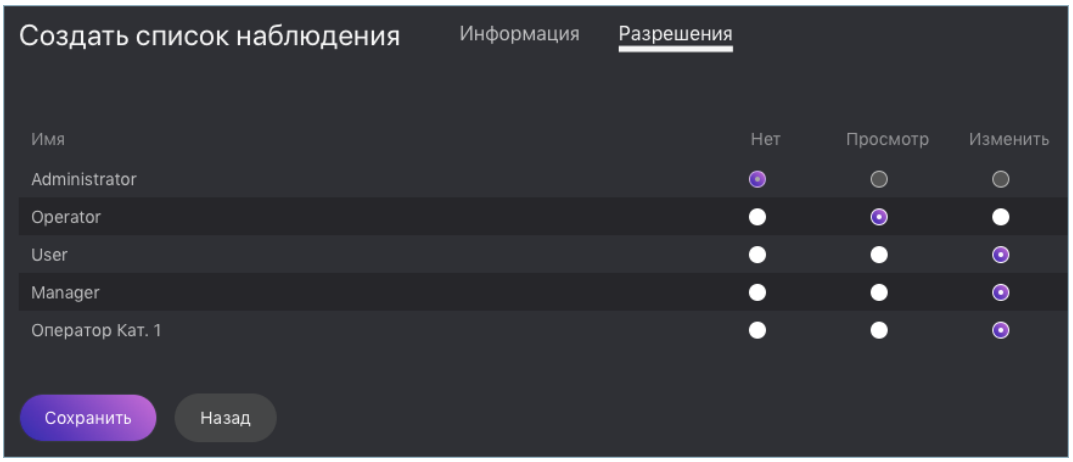


3. В палитре *Метка* выберите цвет, который будет использоваться в событиях распознавания персон из данного списка. Правильно выбранный цвет повышает быстроту реагирования оператора на событие.



4. Введите название списка наблюдения. При необходимости добавьте комментарий.
5. Выберите группу камер, которая будет использоваться для мониторинга списка наблюдения. Если группа камер не выбрана, список наблюдения будет отслеживаться всеми активными камерами в системе.
6. Поставьте флажок *Требовать подтверждение*, если для данного списка оператор должен в обязательном порядке подтвердить принятие события.
7. При необходимости включите звук при появлении события для данного списка.
8. По умолчанию ко всем спискам наблюдения в системе применяется *универсальный* порог схожести лиц, оптимальный для большинства случаев распознавания. Для того чтобы задать индивидуальный порог для списка наблюдения, поставьте флажок *Порог срабатывания* и укажите нужное значение.
9. Поставьте флажок *Активный*.
10. Нажмите *Сохранить*.
11. На вкладке *Разрешения* назначьте права на список наблюдения, указав роли пользователей, ко-

торые смогут изменять/просматривать его настройки.

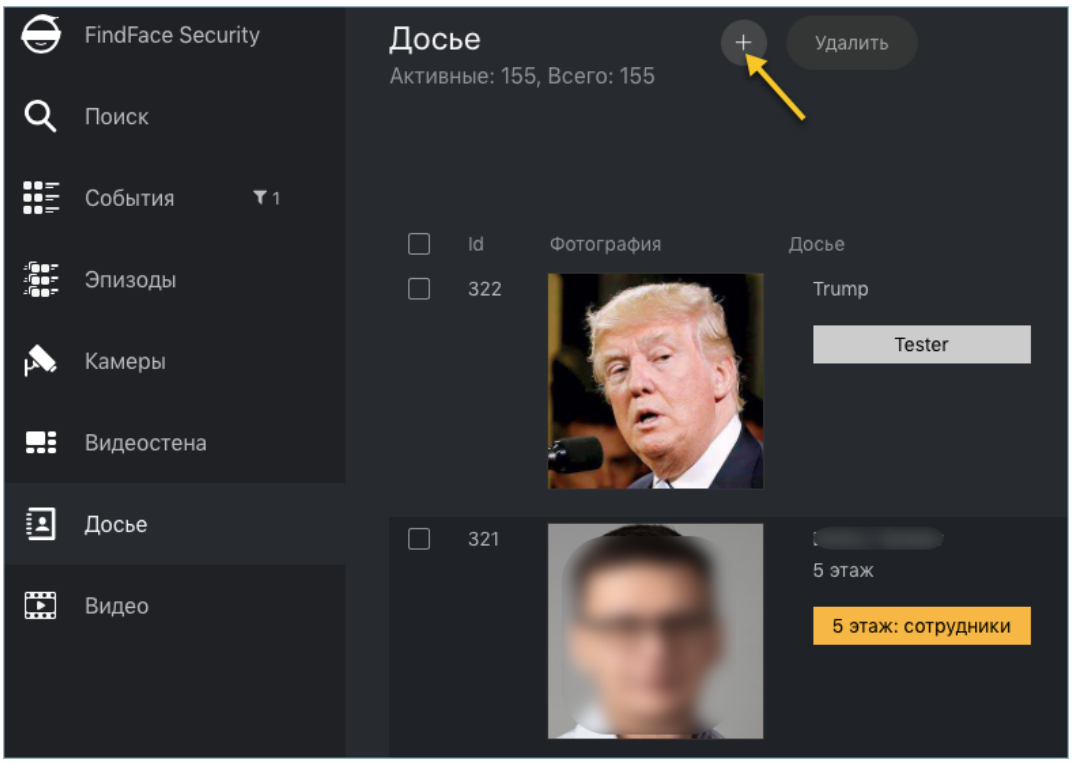


12. Нажмите *Сохранить*.

Создание досье вручную

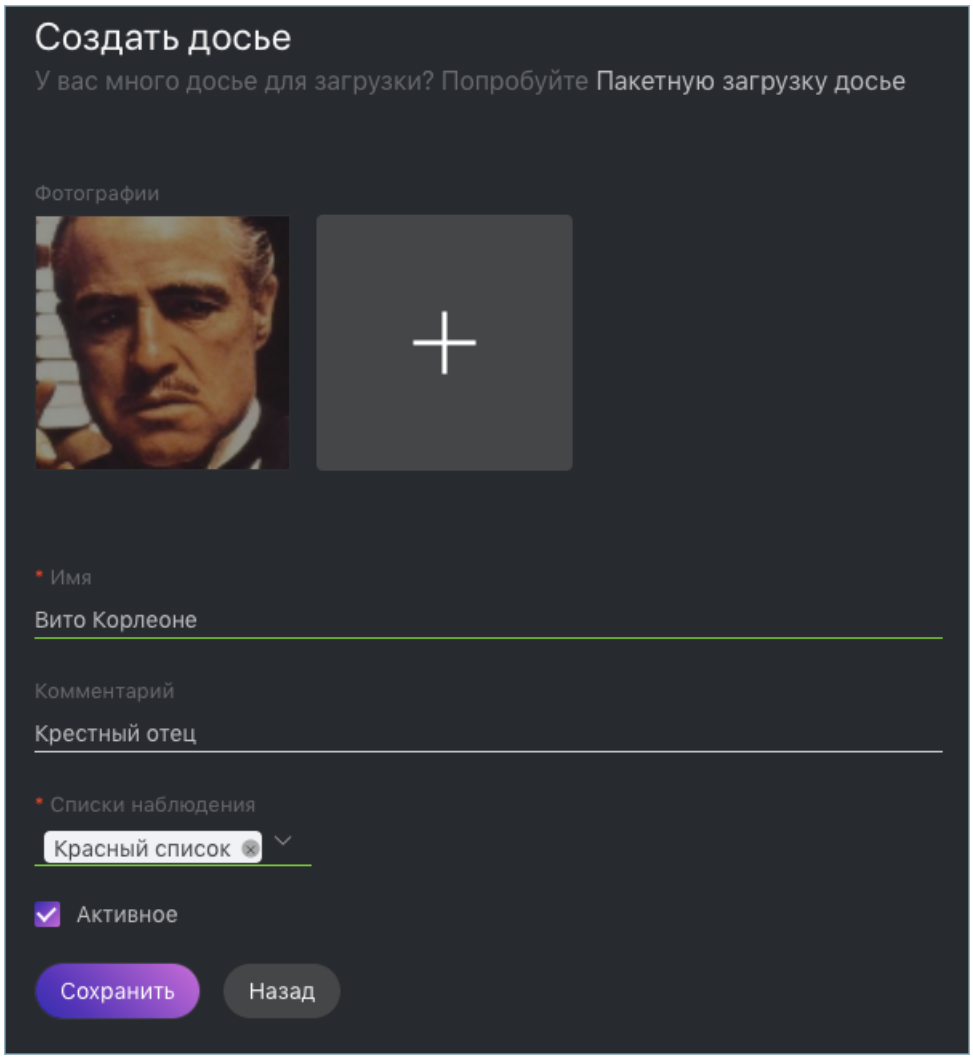
Для создания досье вручную выполните следующие действия:

1. В веб-интерфейсе перейдите на вкладку *Досье*.
2. Нажмите *+*.



3. Добавьте одну или несколько фотографий и введите имя человека. При необходимости добавьте комментарий.

Важно: Лицо на фотографии должно быть надлежащего качества, т. е. в близком к анфас положении. Расстояние между зрачками: 60 px. Поддерживаемые форматы: WEBP, JPG, BMP, PNG. При несоответствии фотографии требованиям будет выведено сообщение с описанием ошибки.



4. Из раскрывающегося списка *Списки наблюдения* выберите список (или несколько списков, по очереди), в который следует добавить досье.
5. Поставьте флажок *Активное*. Если досье неактивно, оно не будет использоваться для идентификации лица в режиме реального времени.
6. Нажмите *Сохранить*.

Пакетная загрузка фотографий

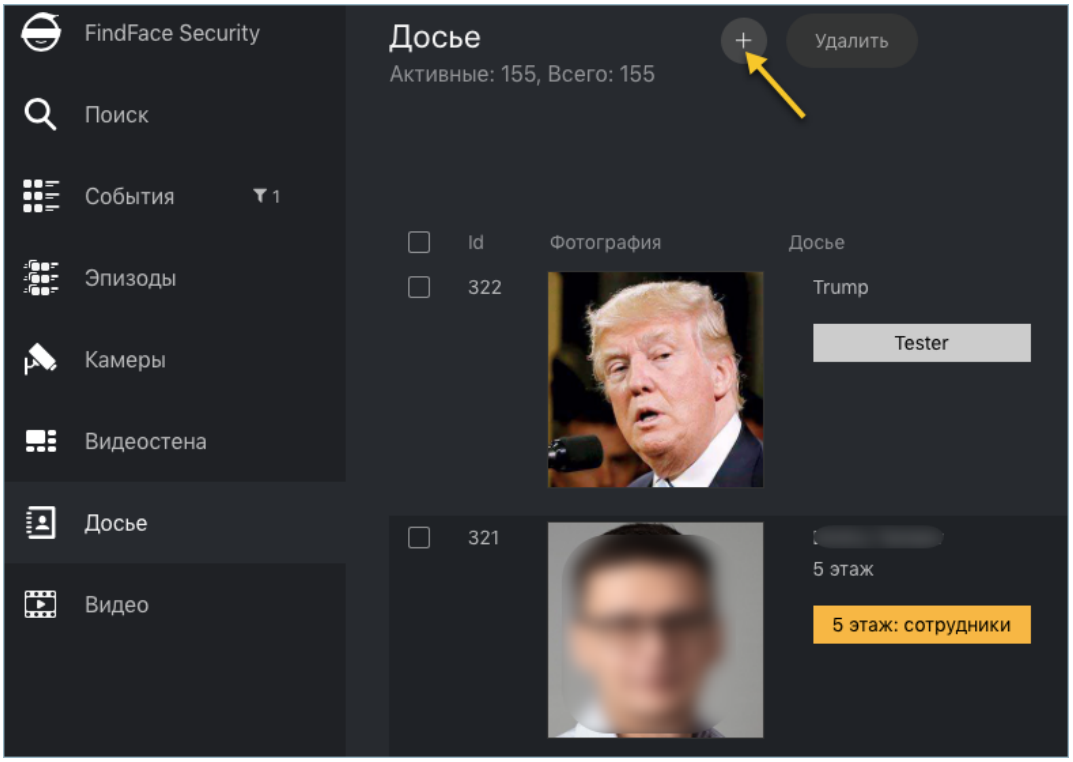
Для создания большого количества досье используйте функционал пакетной загрузки. Выполните следующие действия:

Совет: Если количество досье превышает 10000, используйте *Пакетная загрузка фотографий через консоль*.

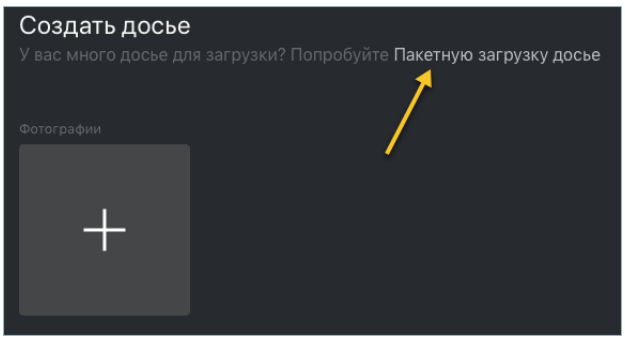
Важно: Лица на фотографиях должно быть надлежащего качества, т. е. в близком к анфас положении. Расстояние между зрачками: 60 px. Поддерживаемые форматы: WEBP, JPG, BMP, PNG. При несоответствии фотографии требованиям будет выведено сообщение с описанием ошибки.

---

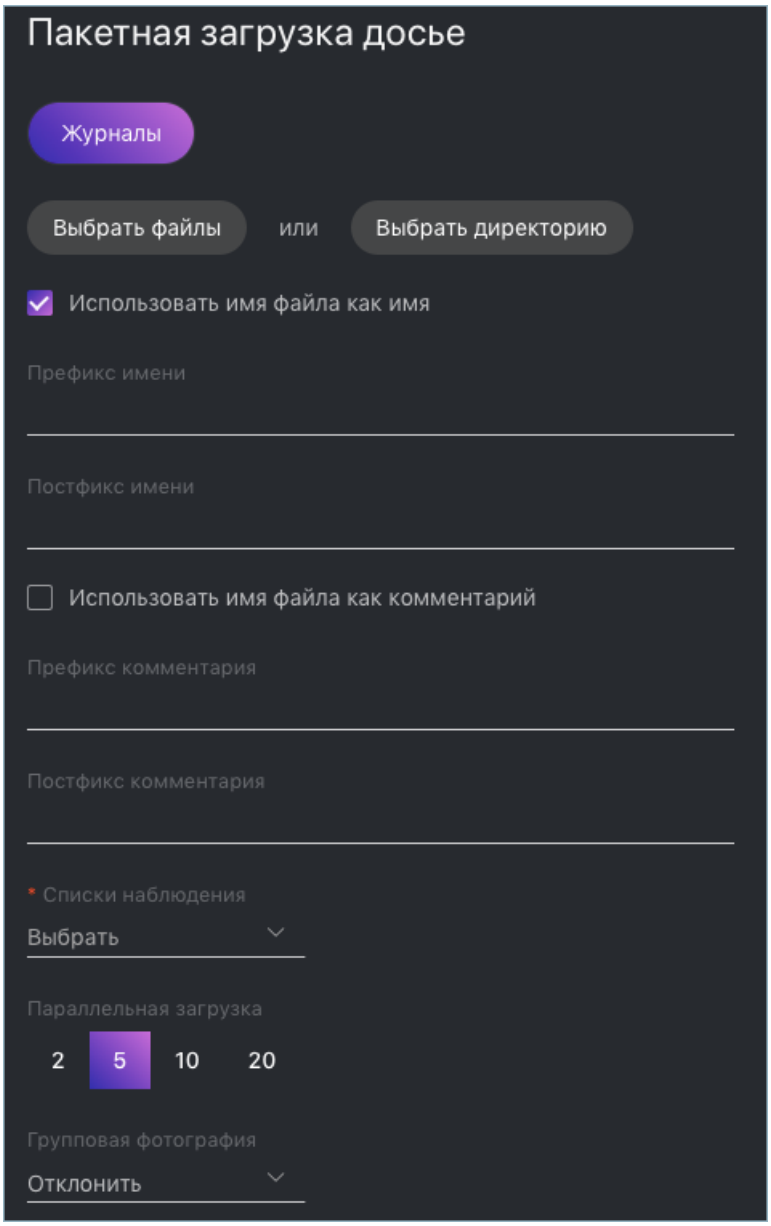
1. В веб-интерфейсе перейдите на вкладку *Досье*.
2. Нажмите *+*.



3. Нажмите *Пакетная загрузка досье*.



4. Выберите фотографии для загрузки пофайлово или укажите папку с фотографиями.



5. Имена файлов с фотографиями можно использовать как основу для имен и/или комментариев в создаваемых досье. Выберите нужный вариант(ы). Затем настройте правило формирования имени и/или комментария, добавив пользовательский префикс и/или постфикс к имени файла.

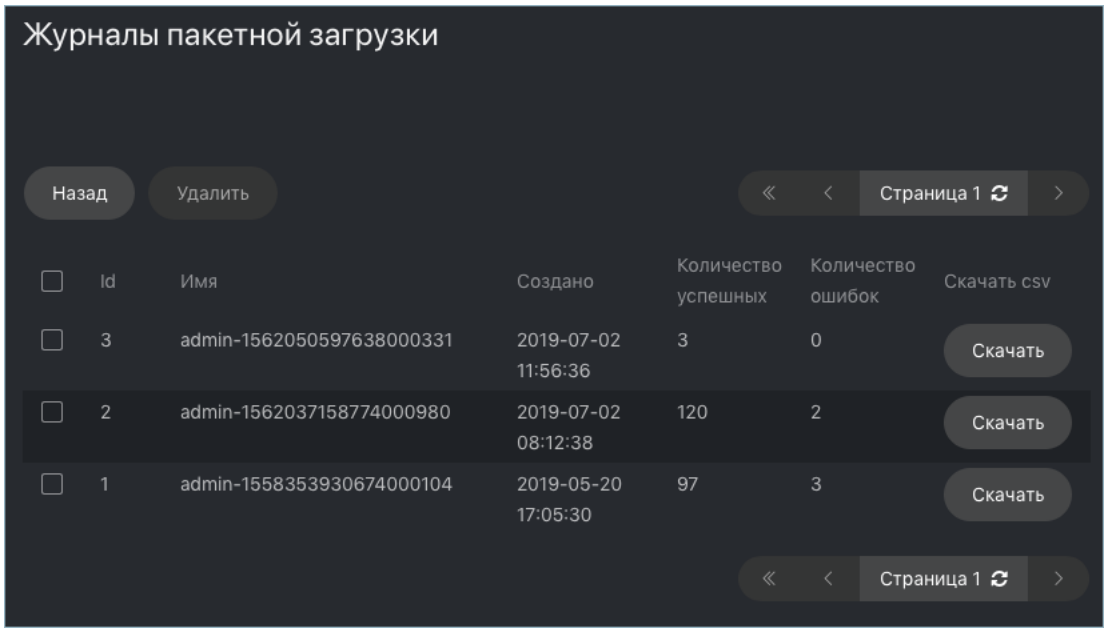
Совет: Во избежание слияние 3-х слов в одно, используйте символ подчеркивания или пробел в префиксе и постфиксе.

---

6. Из раскрывающегося списка *Списки наблюдения* выберите список (или несколько списков, по очереди), в который следует добавить создаваемые досье.

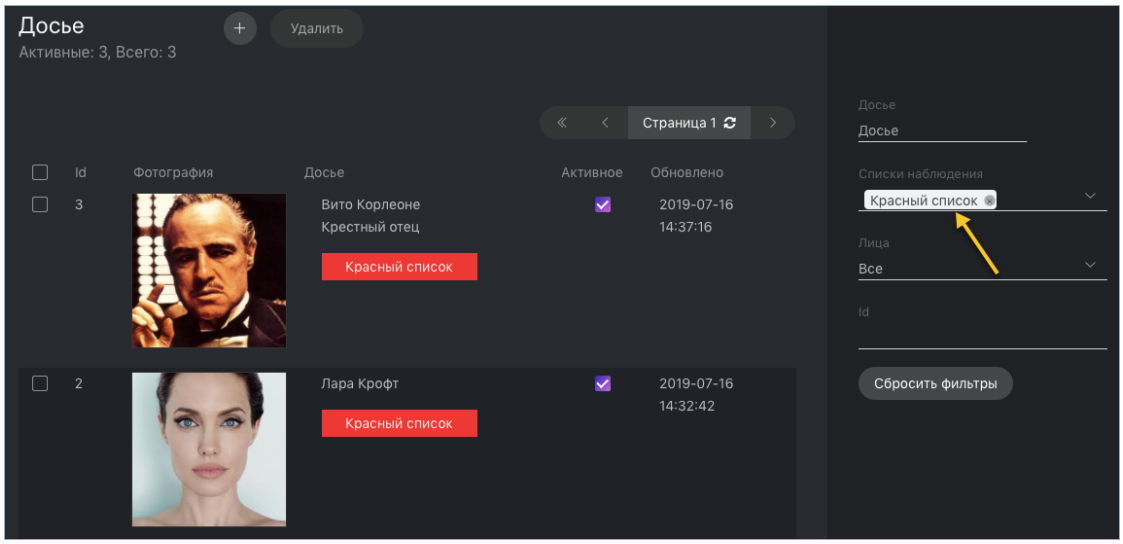
7. В параметре *Параллельная загрузка* задайте количество потоков загрузки фотографий. Чем больше потоков, тем быстрее будет завершена загрузка, однако также потребуется и большее количество ресурсов.
8. Из раскрывающегося списка *Групповая фотография* выберите, как должна поступить система при наличии нескольких лиц на фотографии: отклонить фотографию или загрузить самое большое лицо.
9. Для запуска пакетного создания досье нажмите на кнопку *Старт*.

Важно: Для просмотра лога пакетной загрузки нажмите на кнопку *Журналы*. Затем при необходимости можно скачать лог в формате `.csv`.



Фильтрация досье по спискам наблюдения

Все созданные в FindFace Security досье отображаются на вкладке *Досье*. Используйте фильтр *Списки наблюдения*, чтобы отфильтровать досье по спискам.



1.6.3 Управление пользователями

В этой главе:

Предустановленные роли

Для работы с FindFace Security предусмотрены следующие предустановленные роли:

- Администратор. Обладает полными правами на *управление видеокамерами*, *базой данных досье*, событий, пользователями FindFace Security, а также полным доступом ко всем остальным функциям.

  Важно: Супер Администратор не может лишиться прав администратора даже при смене роли.

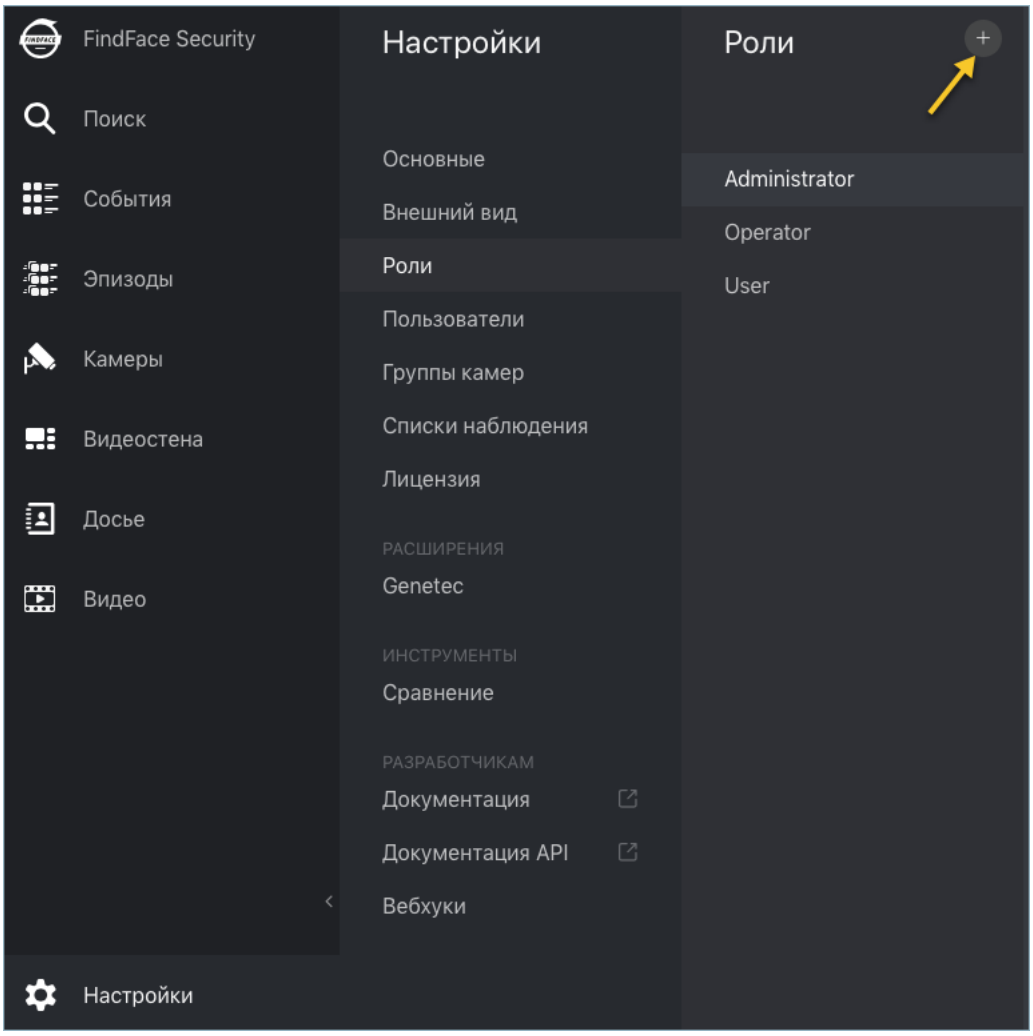
- Оператор. Обладает правами на *создание досье вручную*, подтверждение событий и поиск лиц в базах событий и досье. Остальная информация доступна в режиме чтения. *Пакетное* создание досье невозможно.
- Пользователь. Обладает правами только на подтверждение событий и поиск лиц в базе событий и досье. Остальная информация доступна в режиме чтения.

Вы можете изменить привилегии предустановленных ролей, а также создать новые роли.

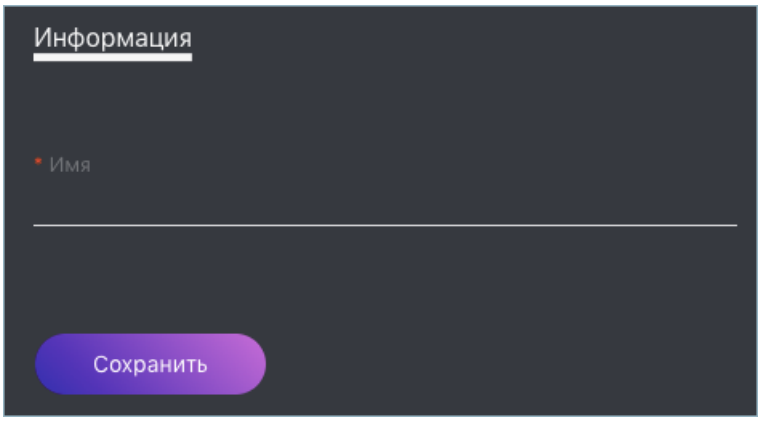
Создание новой роли

Для создания новой роли выполните следующие действия:

1. Перейдите на вкладку *Настройки*. Выберите *Роли*.
2. Нажмите *+*.



3. На вкладке *Информация* задайте имя роли.



4. Нажмите *Сохранить*. Рядом с вкладкой *Информация* появятся дополнительные вкладки. На данных вкладках можно задать права на определенные списки наблюдения (вкладка *Списки наблюдения*) и группы камер (*Группы камер*), а также привилегии на работу с системными функциями и объектами (*Разрешения*).

Примечание: Например, если вы устанавливаете `Нет` для определенной группы камер на вкладке *Группы камер*, пользователи с данной ролью не смогут работать с **этой** группой камер. Установка `Нет` для `cameragroup` на вкладке *Разрешения* не позволит пользователям просматривать и работать со **всеми** группами камер.

Примечание: Право на событие складывается из прав на соответствующую камеру и список наблюдения. Для просмотра событий, для которых не найдены совпадения с досье, требуются только права на камеру.

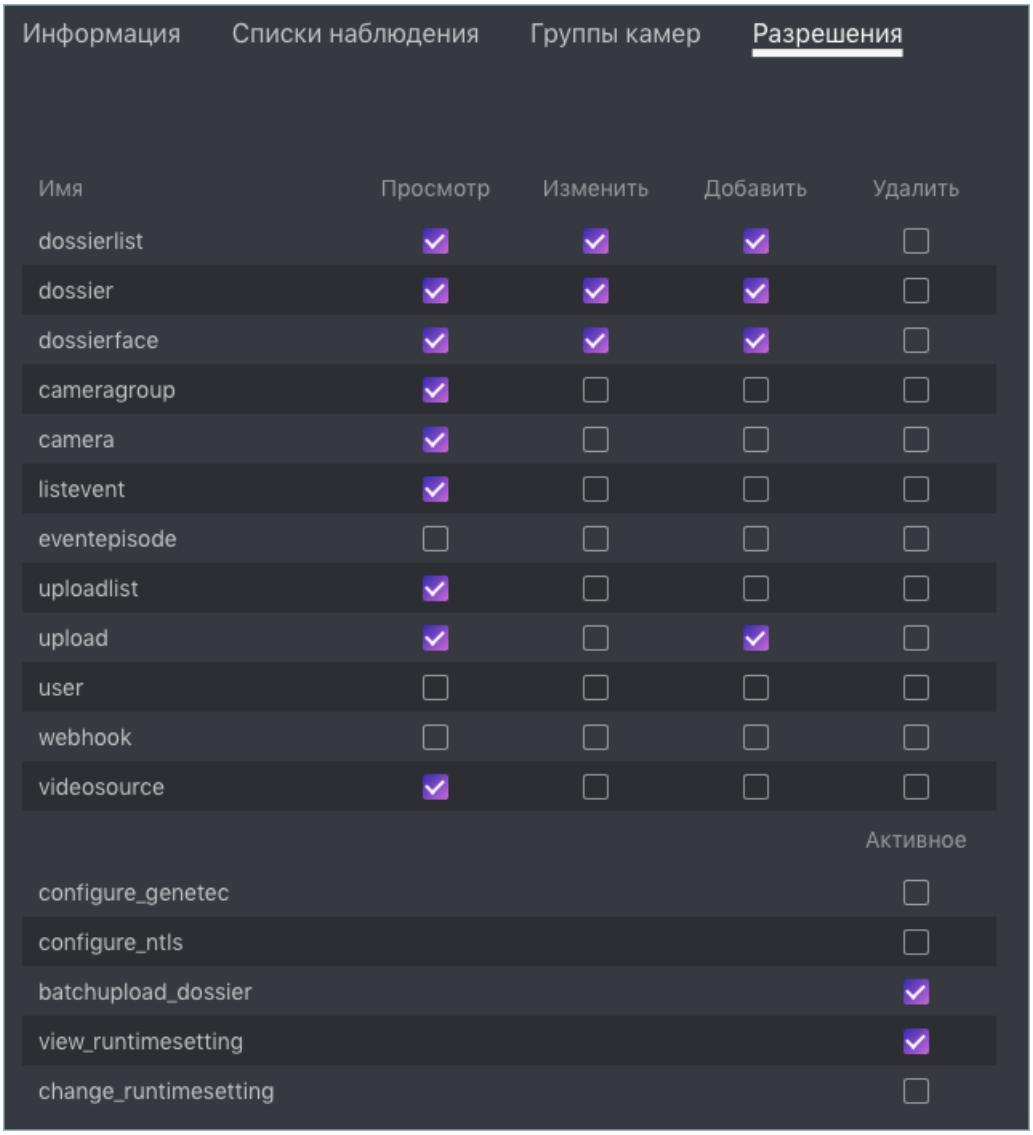
Полный список объектов FindFace Security:

- `dossierlist`: *списки наблюдения*;
- `dossier`: *досье*;
- `dossierface`: *фото в досье*;
- `cameragroup`: *группа камер*;
- `camera`: *камера*;
- `listevent`: *список событий*;
- `eventepisode`: *эпизоды*
- `uploadlist`: список фотографий в *пакетной загрузке*;
- `upload`: элемент (фото) в пакетной загрузке;
- `user`: *пользователь*;
- `group`: *пользовательская роль*;
- `hook`: *вебхук*;
- `videosource`: *идентификация лиц в видеофайлах*.

Вы также можете включать и отключать права на следующий функционал:

- `configure_genetec`: конфигурация *интеграции с Genetec*;

- `configure_ntls`: конфигурация *сервера лицензий* `findface-ntls`.
- `batchupload_dossier`: *пакетное создание досье*;
- `view_runtimesetting`: просмотр *основных настроек* FindFace Security;
- `change_runtimesetting`: изменение основных настроек FindFace Security



Главная и дополнительная роль пользователя

Вы можете назначить пользователю привилегии, используя следующие роли:

- *Главная роль*: основная роль пользователя, обязательная для назначения. Пользователю можно назначить только одну главную роль.
- *Роль*: дополнительная роль пользователя, необязательная для назначения. Одному пользователю можно назначить несколько ролей. Связанные с ними права будут добавлены к правам, предоставляемым главной ролью.

Различие между главной и дополнительными ролями заключается в следующем. Если пользователю назначена определенная главная роль, данной роли будут **автоматически** предоставлены права на изменение всех объектов, впоследствии созданных данным пользователем (т. е. на изменение созданных им камер, списков наблюдения, досье и т. д.). Этого не произойдет, если вы назначите ему дополнительную роль. Например, если пользователю назначена главная роль `Менеджер`, все пользователи с ролью `Менеджер` смогут изменять объекты, впоследствии созданные этим пользователем. Напротив, если вы назначите роль `Менеджер` как дополнительную, другим пользователям с ролью `Менеджер` понадобятся отдельные разрешения на изменение объектов, созданных этим пользователем.

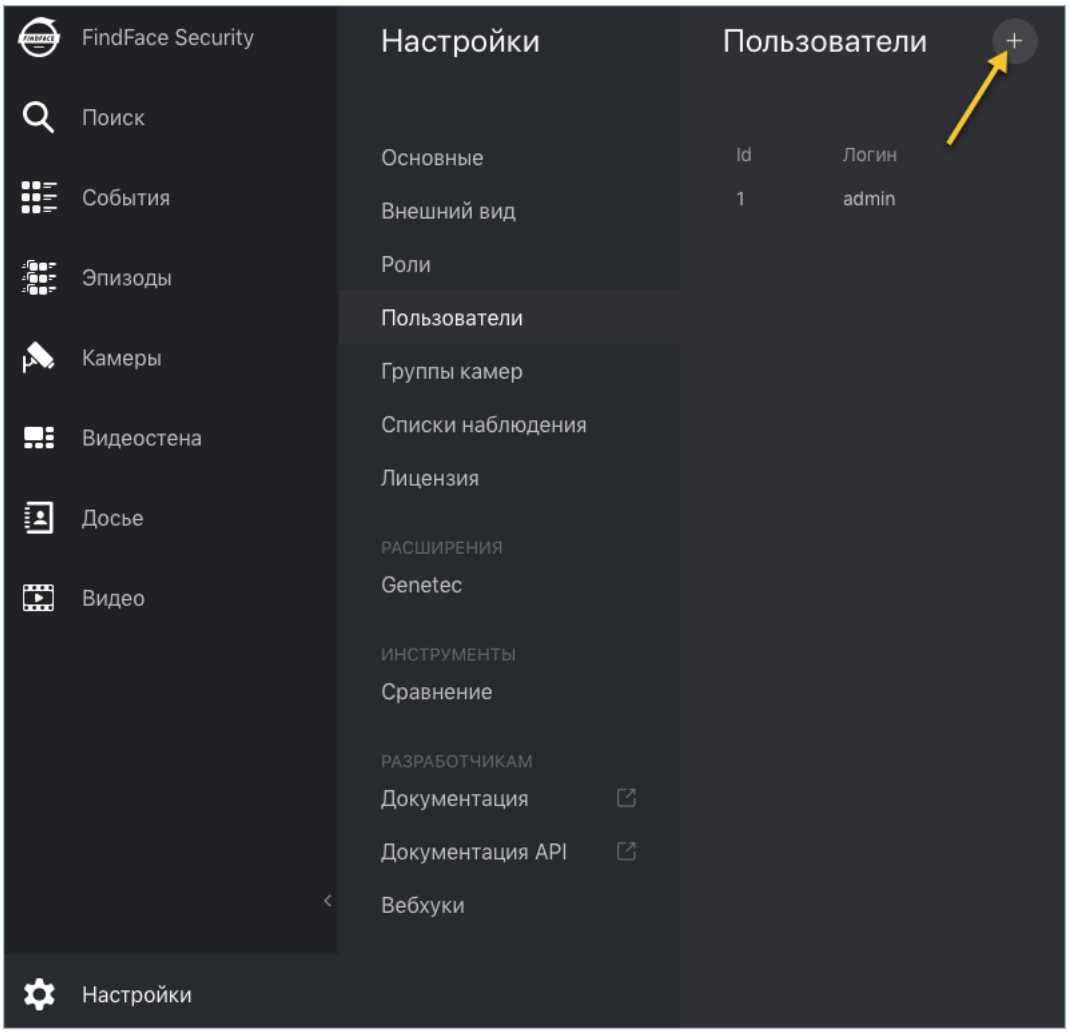
См.также:

Создание пользователя

Создание пользователя

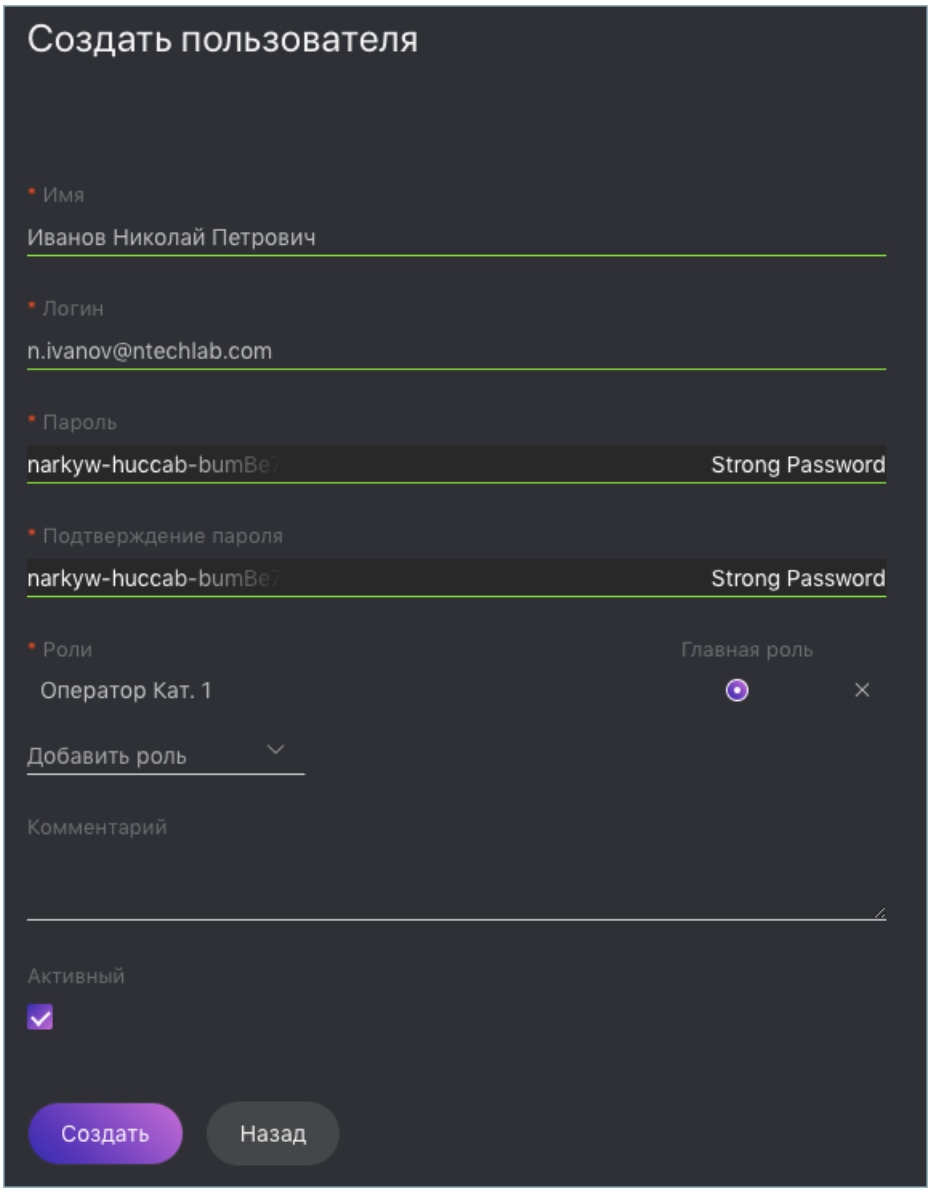
Для создания нового пользователя выполните следующие действия:

1. Перейдите на вкладку *Настройки*. Выберите *Пользователи*.
2. Нажмите *+*.



3. Задайте имя пользователя, логин и пароль. При необходимости добавьте комментарий.

4. Из раскрывающегося меню *Roles* выберите одну или несколько пользовательских ролей. Назначьте они из них основной.



5. Поставьте флажок *Активный*.
6. Нажмите *Создать*.

Деактивация или удаление пользователя

Для того чтобы деактивировать пользователя, снимите флажок *Активный* в списке пользователей (*Настройки -> Пользователи*).

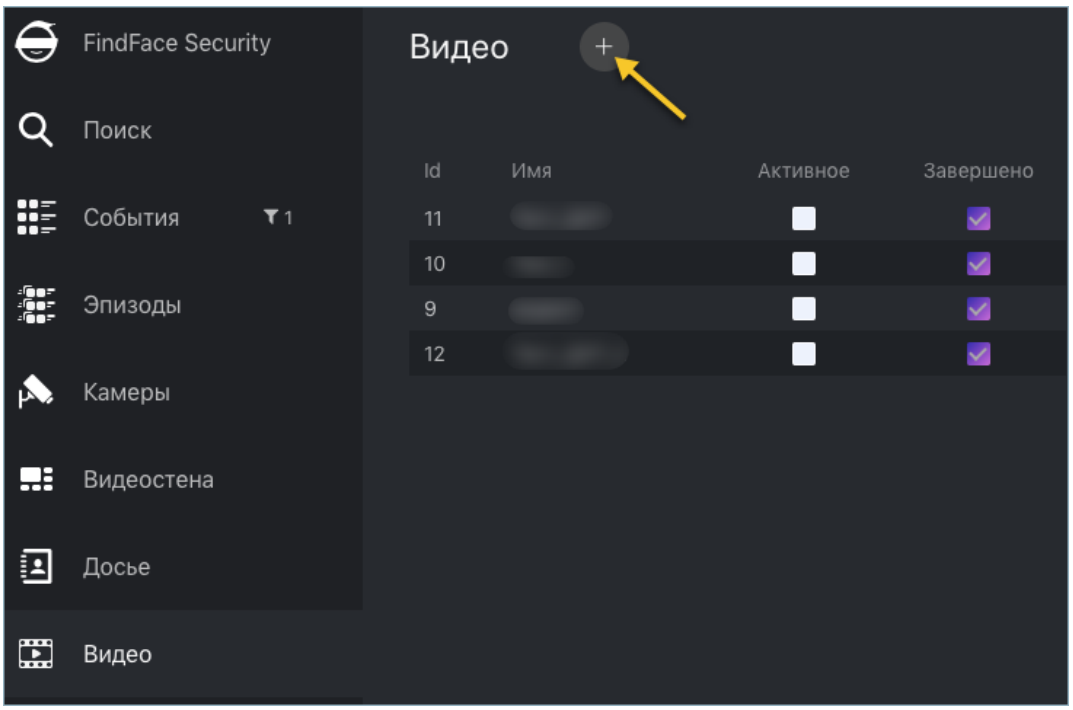
Для удаления пользователя из FindFace Security щелкните по его логину в списке. Нажмите *Удалить*.

1.6.4 Идентификация лиц в оффлайн видео

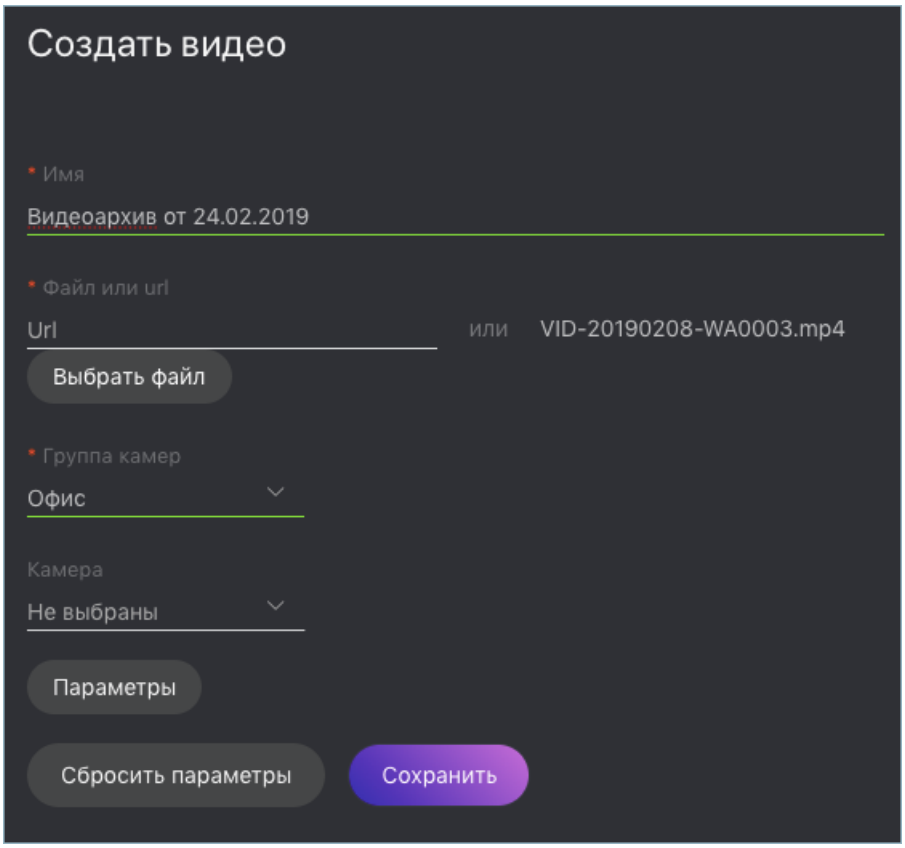
Помимо идентификации лица в режиме реального времени, FindFace Security также позволяет выполнять идентификацию лиц в оффлайн видео. Данная функциональность имеет широкий спектр возможных применений, среди которых наиболее распространенным случаем является обнаружение и распознавание лиц в архивных видео.

Для идентификации лица в оффлайн видео выполните следующие действия:

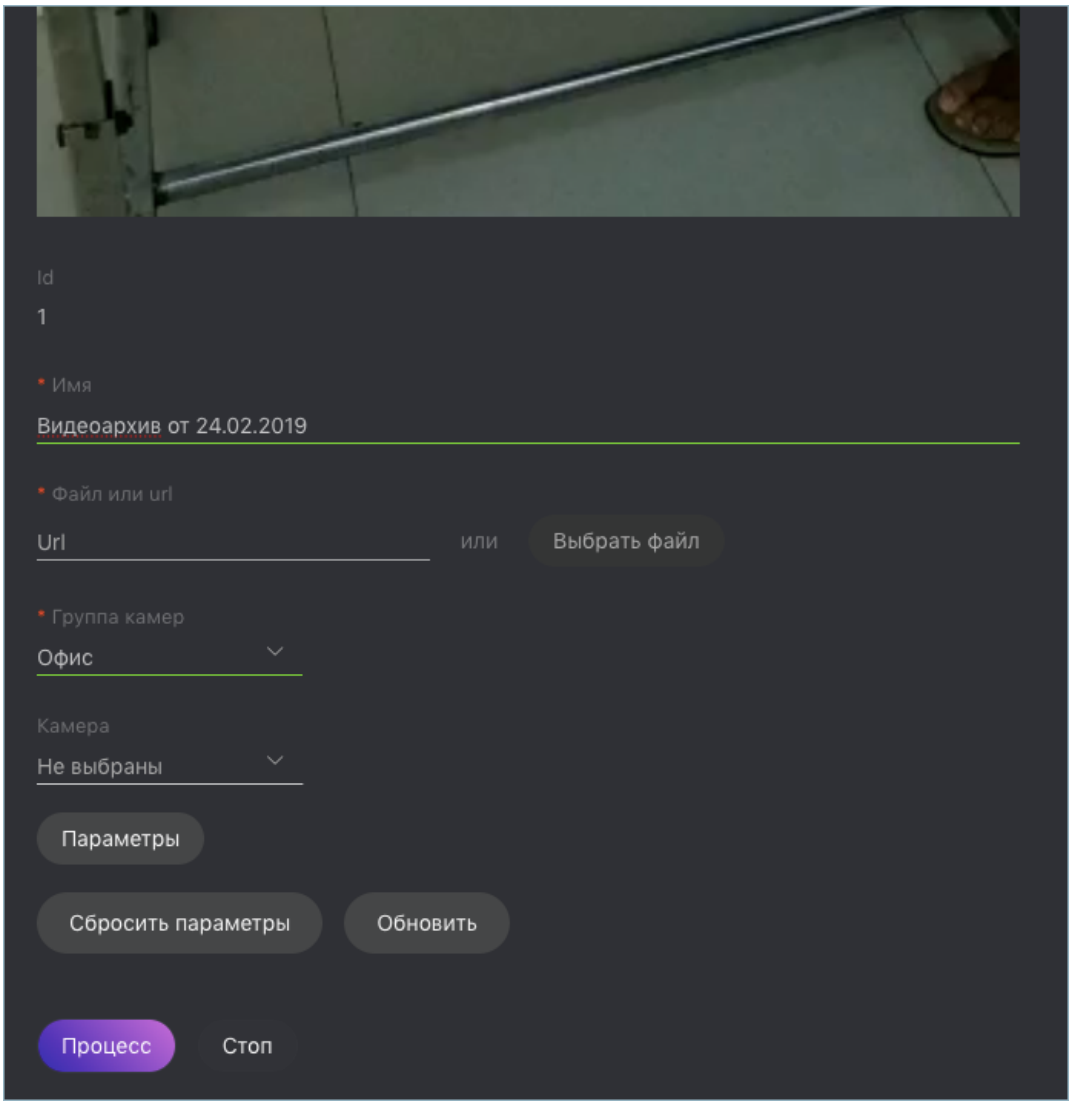
1. Создайте *группу камер* с базовыми настройками.
2. Укажите данную группу камер в настройках тех *списков наблюдения*, лица в которых нужно найти на видео.
3. Создайте видео в FindFace Security, загрузив его из файла или онлайн-хранилища/облака. Для этого, перейдите на вкладку *Видео*.
4. Нажмите *+*.



5. Укажите название видео.



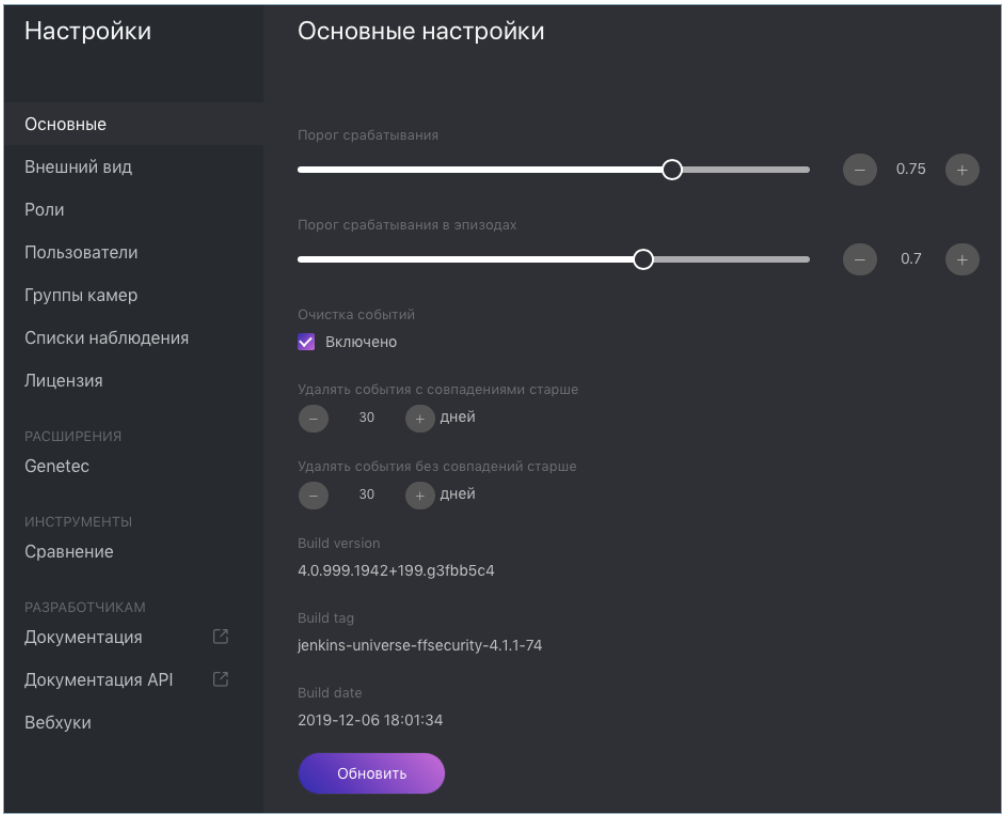
6. Укажите URL видеоизображения в онлайн-хранилище или выберите видеофайл.
7. Выберите только что созданную группу камер.
8. (Опционально) Выберите камеру, с которой будут ассоциироваться лица, найденные на видео.
9. (Опционально) Укажите параметры обработки видео по аналогии с параметрами обработки видеопотока с *камеры*.
10. Нажмите *Сохранить* для загрузки видео на сервер.



11. После того как видео загрузится, нажмите *Обработать* для запуска процедуры идентификации лиц. Для просмотра найденных лиц перейдите на вкладку *События* и отфильтруйте список событий по связанной группе камер.

1.6.5 Основные настройки

Для изменения порога срабатывания при верификации лиц, порога срабатывания для эпизодов и настройки автоматического удаления старых лиц (событий и эпизодов) из базы данных, перейдите на вкладку *Настройки*. Выберите *Основные*. После внесения изменений нажмите *Обновить*.



В этом разделе:

Универсальный порог срабатывания

FindFace Security принимает решение о совпадении обнаруженного лица с лицом из досье (т. е. о положительной верификации) на основании предустановленной пороговой степени схожести. По умолчанию установлено оптимальное пороговое значение, равное 0.75. При необходимости вы можете изменить данное значение.

Примечание: Чем выше пороговая степень схожести, тем меньше шансов на положительную ложную верификацию человека, однако некоторые подходящие фотографии могут также не пройти верификацию.

Совет: Вы можете настроить порог срабатывания индивидуально для каждой *камеры*, *группы камер* и *списка наблюдения*.

Порог срабатывания для эпизодов

При формировании эпизода система ищет в биометрической базе данных *недавние* события с похожими лицами выше определенной пороговой степени схожести. Порог по умолчанию установлен на 0.7. При необходимости вы можете изменить это значение.

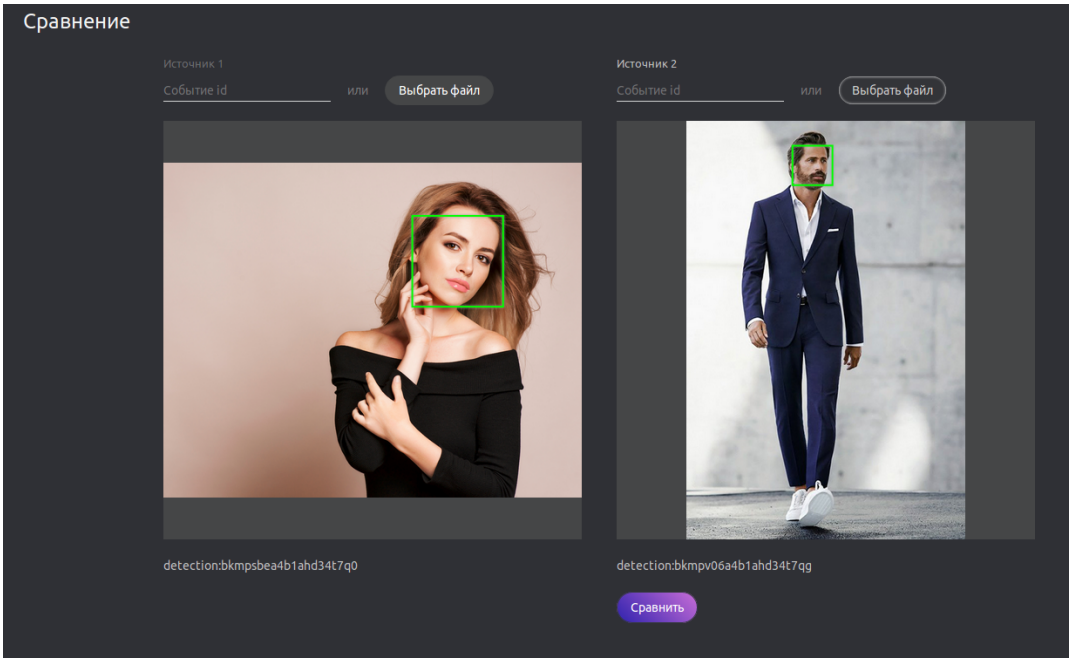
Автоматическое удаление старых событий и эпизодов

Используйте эту же вкладку, чтобы задать расписание автоматического удаления старых событий и эпизодов из базы данных. Можно настроить удаление событий и эпизодов по разным расписаниям в зависимости от наличия совпадений с базой данных досье.

1.6.6 Сравнение лиц

FindFace Security позволяет выполнять сравнение 2-х лиц. Выполните следующие действия:

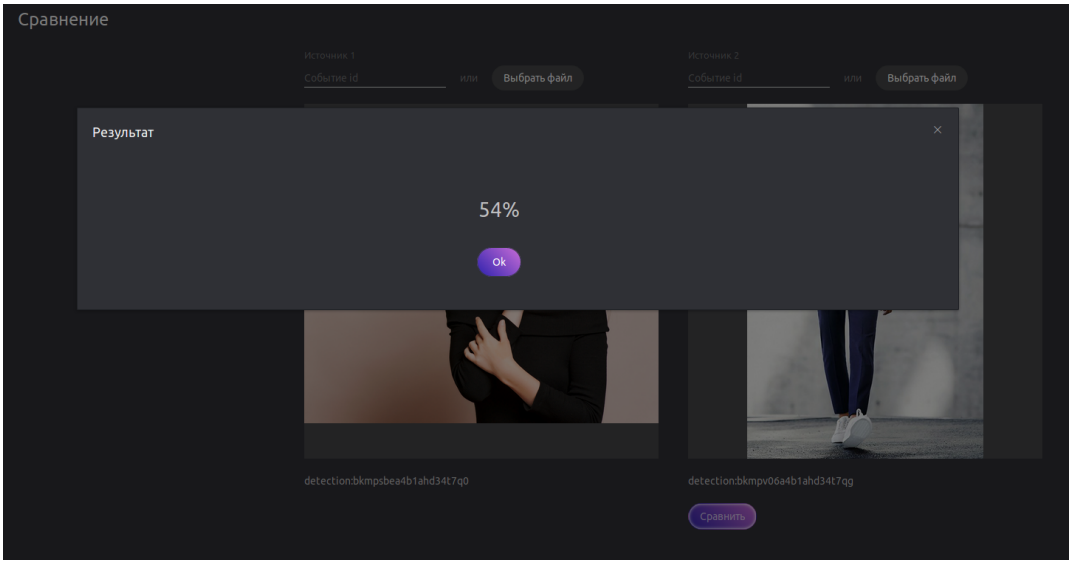
1. Перейдите на вкладку *Настройки*. Выберите *Сравнение*.



2. Укажите id событий, лица из которых нужно сравнить, и/или загрузите фотографии с лицами.

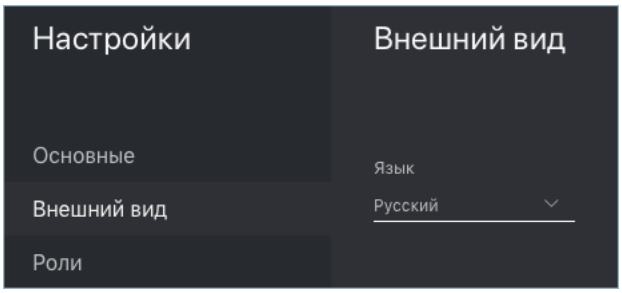
   Совет: Узнать ID события можно на вкладке *События*.

3. Нажмите *Сравнить*. В результате будет отображена вероятность принадлежности лиц одному человеку.



1.6.7 Переключение языка

Для того чтобы переключить язык интерфейса, перейдите в меню *Настройки -> Внешний вид*.



1.7 Расширенный функционал

1.7.1 Настройка эпизодов

В этом разделе:

Об эпизодах

Эпизод — это набор событий идентификации, в которых фигурируют лица одного и того же человека, обнаруженные в течение определенного периода времени.

Эпизоды бывают двух типов:

- LIVE: открытый на данный момент эпизод, в который могут добавлены новые события.
- Закрытый: закрытый эпизод, добавление событий невозможно.

Параметры эпизода

Для настройки эпизодов вам понадобится файл конфигурации `findface-security`. Добавьте следующие параметры в секцию `FFSECURITY`:

- `EPISODE_SEARCH_INTERVAL`: период времени, предшествующий событию, в течение которого система ищет в биометрической базе данных события с похожими лицами. Если такого события не найдено, система создает новый эпизод. В противном случае она выбирает наиболее подходящее событие из открытого (LIVE) эпизода, отсортировав 100 последних похожих лиц.

  Примечание: Порог срабатывания в эпизодах отличается от порога при верификации лиц в событиях. См. *Основные настройки*.

- `EPISODE_MAX_DURATION`: максимальная продолжительность эпизода в секундах. По истечении этого времени эпизод автоматически закрывается.
- `EPISODE_EVENT_TIMEOUT`: максимальное время в секундах с момента добавления последнего события в эпизод. По истечении этого времени эпизод автоматически закрывается.

```
sudo vi /etc/ffsecurity/config.py

...

FFSECURITY = {
      ...
      'EPISODE_SEARCH_INTERVAL': 60,
      'EPISODE_MAX_DURATION': 300,
      'EPISODE_EVENT_TIMEOUT': 30,
      ...
}




...
```

См.также:

Для того чтобы посмотреть, как работают эпизоды, перейдите на вкладку *Эпизоды*. Подробнее см. *Эпизоды событий*.

Назначение прав на эпизоды

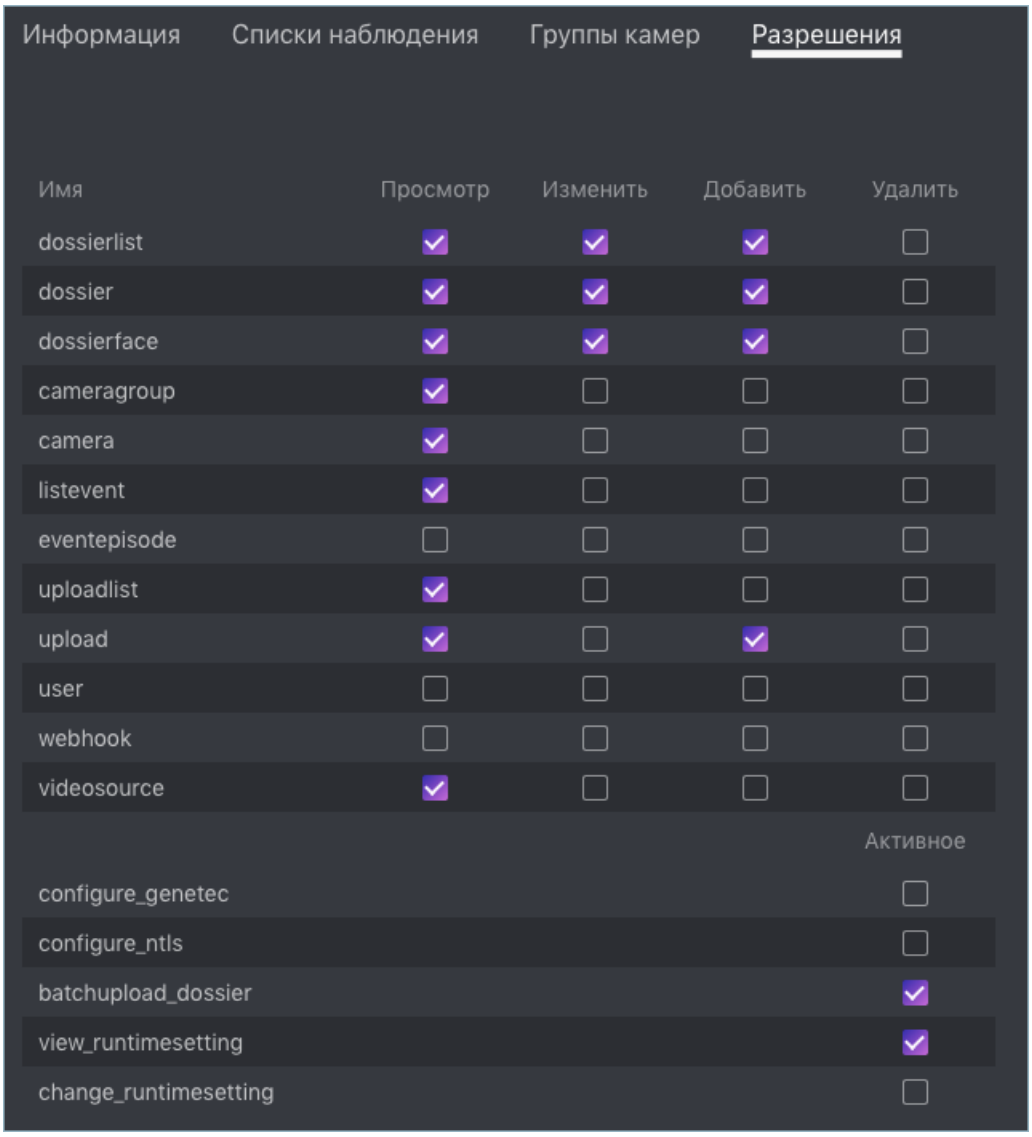
Пользователь получает уведомление о новом эпизоде, если у него есть права на открывающее этот эпизод событие. Просмотр новых событий в эпизоде также требует соответствующих прав.

Право на событие состоит из прав на соответствующие камеру и список наблюдения.

Примечание: Чтобы увидеть несопоставленные с досье события, вам понадобятся только права на камеру.

Для управления правами на объект `Эпизод` перейдите в разрешения для соответствующей роли и настройте разрешение `eventepisode`.

Совет: См. *Управление пользователями*.



1.7.2 Привязка группы камер к экземпляру `findface-video-worker`

Часто в распределенной архитектуре обработку видеоизображения с группы камер требуется выполнять локально, не обращаясь к центральному серверу и не перераспределяя видеопотоки между удаленными экземплярами `findface-video-worker`.

Примечание: Например, это может быть актуальным для сетей гостиниц, магазинов, при наличии нескольких проходных в одном здании и т. д.

В этом случае группу камер привязывают к локально установленному экземпляру `findface-video-worker`.

Выполните следующие действия:

1. Перейдите на вкладку *Настройки*. Выберите *Группы камер*.
2. Откройте настройки группы камер.
3. В поле *Метки* создайте или выберите из уже созданных одну или несколько меток для привязки группы камер к экземпляру `findface-video-worker`. Сохраните изменения.
4. Откройте файл конфигурации экземпляра `findface-video-worker` и укажите в нем заданные метки в формате `имя_метки=true` (`terminal_1` в примере ниже).

```
sudo vi /etc/findface-video-worker-cpu.ini
sudo vi /etc/findface-video-worker-gpu.ini

labels = terminal_1=true
```

5. Перезапустите `findface-video-worker`.

```
sudo systemctl restart findface-video-worker-cpu.service
sudo systemctl restart findface-video-worker-gpu.service
```

Примечание: Если камере присвоена метка, то видеопоток с нее может обрабатываться как экземпляром `findface-video-worker(-gpu)` с аналогичной меткой, так и экземпляром `findface-video-worker(-gpu)` без меток.

Предупреждение: Если камера с меткой обрабатывается экземпляром `findface-video-worker(-gpu)` без меток и появляется свободный экземпляр с меткой, камера автоматически на него не переключится. Чтобы переключить камеру, перезапустите экземпляр `findface-video-worker(-gpu)` с меткой.

1.7.3 Распределенная база данных досье

В распределенной архитектуре часто необходимо, чтобы база данных досье распределялась по нескольким серверам.

В текущей реализации база данных досье доступна для редактирования только на главном сервере, т. н. master-сервере. Master-сервер синхронизирован с несколькими дополнительными slave-экземплярами FindFace Security. На slave-серверах база данных досье доступна только для чтения и мониторинга.

В этом разделе:

Настройка синхронизации Master/Slave

Для настройки синхронизации master/slave выполните следующие действия:

1. На master-сервере откройте файл конфигурации `findface-security`. Придумайте токен синхронизации и укажите его в параметре `SYNC_TOKEN`.

```
sudo vi /etc/ffsecurity/config.py

...

SYNC_TOKEN = 'ABC_123456789'
```

2. Скопируйте токен синхронизации в файл конфигурации `findface-security` на slave-серверах.
3. Убедитесь, что параметр `EXTERNAL_ADDRESS` в файлах конфигурации `findface-security` на slave-серверах содержит тот же адрес, что и на master-сервере.

Синхронизация master/slave теперь настроена и будет активирована, как только вы настроите репликацию списка наблюдения с master-сервера на slave.

Репликация списка наблюдения master -> slave

Для того чтобы реплицировать список наблюдения от master-сервера на slave-экземпляры, отправьте на slave POST-запрос со следующими параметрами в теле:

- `remote_dossier_list`: id исходного списка наблюдения на master-сервере
- `remote_url`: URL master-сервера
- `slave_dossier_list`: id списка наблюдения на slave-сервере, который будет репликой исходного списка наблюдения.

```
POST /sync/dossier-lists/
{remote_dossier_list: 1,
remote_url: "http://172.17.46.14",
slave_dossier_list: 3}
```

Задание времени синхронизации

По умолчанию реплицированные списки наблюдения на slave-серверах синхронизируются с master-сервером каждую ночь в 3:00. Время синхронизации можно изменить следующим образом:

1. Откройте файл конфигурации `findface-security` на master-сервере.

```
sudo vi /etc/ffsecurity/config.py
```

2. Вставьте следующие строки и задайте собственный график репликации.

```
...

SYNC_TIME = {
    'hour': 3,  # 24 hour format
    'minute': 0,
}
```

Отмена репликации и синхронизации списка наблюдения

Для того чтобы отменить репликацию и синхронизацию списка наблюдения, отправьте на slave-сервер API-запрос с `{id}` соответствующего списка на slave-сервере:

```
DELETE /sync/dossier-lists/{id}/
```

Включение функционала в веб-интерейсе

По умолчанию вы можете включать и отключать репликацию списка наблюдения только через API. Для того чтобы данный функционал был также доступен через веб-интерфейс, выполните следующие действия:

1. Откройте файл конфигурации `findface-security` на master-сервере.

```
sudo vi /etc/ffsecurity/config.py
```

2. Включите плагин `ffsecurity_sync`, добавив строку `INSTALLED_APPS.append ('ffsecurity_sync')` в раздел плагинов:

```
...

# integration plugins
...
INSTALLED_APPS.append('ffsecurity_sync')
```

3. Выполните аналогичные действия на каждом slave-сервере.
4. На каждом сервере выполните перенос основной архитектуры базы данных из FindFace Security в `PostgreSQL` и перезапустите службу `findface-security`.

```
sudo findface-security migrate
sudo systemctl restart findface-security.service
```

1.7.4 Пользовательские вкладки, поля и фильтры в досье

Часто необходимо, чтобы досье содержало дополнительные вкладки и поля, по которым можно было осуществлять поисковые запросы.

Для добавления пользовательских вкладок и полей в досье выполните следующие действия:

1. Подготовьте список пользовательских вкладок и полей для добавления в досье.

2. Откройте файл конфигурации `findface-security`.

```
sudo vi /etc/ffsecurity/config.py
```

3. В секцию `FFSECURITY` вставьте новую секцию `CUSTOM_FIELDS` со следующим содержимым:

- `'items'`: список полей в досье. Опишите каждое поле следующими параметрами:
  - `'name'`: внутреннее имя поля, string.
  - `'label'`: название поля в веб-интерфейсе, string.
  - `'display'`: формат отображения (`form` или `list`), string или array.
  - `'tab'`: вкладка, на которой располагается поле. Если не задана, поле появится на главной странице досье (той, что с фотографией).
  - `'editable'`: редактируемость поля, boolean.
  - `'type'`: тип данных поля, string. Возможные значения:
    - `list`: требует задания `items`, дополнительного параметра для списков (см. ниже), ожидает объекты {id, name} в словарях;
    - `valuelist`: ожидает элементы примитивных типов.
    - `objectlist`: позволяет создавать массивы объектов нужного типа.
    - `datetime`: примитивный тип данных, отображаемый как список datetime.
    - `date`: примитивный тип данных, отображаемый как выбор даты.
    - `boolean`: примитивный тип данных, отображаемый как флажок.
    - `string`: примитивный тип данных `string`.
  - дополнительные параметры для списков (`type=list`, `type=valuelist`):
    - `multiple`: возможность выбора нескольких элементов в списке, boolean.
    - `items`: словарь, используемый как источник данных для списка.
    - `allow_create`: возможность добавления новых элементов в список, boolean.
    - `custom_id`: пользовательское поле для id (`type=list`).
  - дополнительные параметры для списков объектов (`type=objectlist`).
    - `object`: объекты, используемые как источник данных для списка объектов.
    - `simple`: указывает, что поле ожидает данные примитивного типа вместо объектов, например, ожидает строки с телефонными номерами.
- `'filters'`: список фильтров для поиска по пользовательским полям. Параметры:
  - `'name'`: внутреннее имя фильтра,
  - `'label'`: название фильтра в веб-интерфейсе,
  - `'field'`: связанное поле в формате `meta__[имя поля]` (с двойным подчеркиванием).
- `'tabs'`: список вкладок в досье. Первая вкладка в списке соответствует главной странице досье.

```
FFSECURITY = {

...

    'CUSTOM_FIELDS': {
        'dossier_meta': {
            'items': [
                {'name': 'recid','label':"",'display':['form','list']},
                {'name': 'name','label':"",'display':'form'},
                {'name': 'address','label':"",'display':'form'},
                {'name': 'notation','label':"",'display':'form'},
                {'name': 'nullcolumn','label':"",'display':'form'},
                {'name': 'photo','label':"",'display':'form'},
                {'name': 'age', 'label':"",'display':'form', 'tab': 'look'},
                {'name': 'growth', 'label':"",'display':'form', 'tab': 'look'},
            ],
            'filters': [
                {
                    'name': 'recid',
                    'label': 'meta Field',
                    'field': 'meta__recid',
                },
            ],
            'tabs': [
                {'name': 'main'},
                {'name': 'look'},
                {'name': 'crime'},
            ],

        }
    }
}
```

4. Вы увидите, что новая форма досье появилась в веб-интерфейсе:

1.7.5 Пакетная загрузка фотографий через консоль

Помимо веб-интерфейса, для пакетной загрузки фотографий в базу данных досье можно использовать поставляемую вместе с FindFace Security утилиту `findface-security-uploader`. Используйте утилиту, когда требуется загрузить большое количество фотографий (более 10000).

Совет: Для вызова справки `findface-security-uploader` выполните команду:

```
findface-security-uploader --help
```

Выполните следующие действия:

1. Подготовьте CSV- или TSV-файл со списком фотографий и метаданными.

   Важно: В качестве источника метаданных файл должен иметь следующий формат: `путь к фотографии | метаданные`.

   Для подготовки TSV-файла можно использовать скрипт, аналогичный `данному` или команду `find`.

Примечание: Как скрипт, так и команда в примерах создают файл `images.tsv` с данными в формате `полный путь к файлу с фотографией | метаданные`. В качестве метаданных будет создана строка с именем файла.

Для запуска скрипта на создание TSV-файла со списком фотографий из указанного каталога (`/home/user/25_celeb/` в примере) выполните следующую команду:

```
python3 tsv_builder.py /home/user/25_celeb/
```

Пример использования команды `find`:

```
find photos/ -type f -iname '*g' | while read x; do y="${x%.*}"; printf "%s\t%s\n" "$x" "${y#
↪#*/}"; done
```

2. Создайте файл задания (job-файл) из CSV- или TSV-файла, используя метод `add` утилиты. В результате в текущем каталоге будет создан файл `enroll-job.db`.

```
findface-security-uploader add images.tsv
```

Опции `add`:

- `-format`: формат файла, по умолчанию `tsv`,
- `-delimiter`: используемый разделитель, по умолчанию ''\t'' для TSV-файла, '','' для формата CSV.

Примечание: Файл job представляет собой sqlite-базу, которая может быть открыта в консоли `sqlite3`.

3. Выполните задание job, используя метод `run` утилиты.

```
findface-security-uploader run --dossier-lists 2 --api http://127.0.0.1:80 --user admin --
↪password password
```

Опции `run`:

- `-parallel`: количество потоков загрузки фотографий, по умолчанию 10. Чем больше потоков, тем быстрее будет завершена загрузка, однако также потребуется и большее количество ресурсов.
- `-api`: URL API компонента `findface-security`, по умолчанию `http://127.0.0.1:80/`.
- `-user`: имя пользователя.
- `-password`: пароль.
- `-dossier-lists`: перечень разделенных запятой id списков наблюдения, в которые нужно добавить фотографии.
- `-failed`: в случае неудачи при обработке job-файла исправьте ошибку и повторите попытку с данной опцией.

1.7.6 Дедупликация событий

В этом разделе:

Рассмотрите возможность включения дедупликации, чтобы исключить дублирование событий распознавания лиц в пределах одной группы камер.

Включение дедупликации

Для того чтобы активировать функцию дедупликации, выполните следующие действия:

1. Для каждой камеры в группе включите буферный режим детектирования лиц. См. подробнее *Добавление камеры*.
2. Перейдите на вкладку *Настройки*. Выберите *Группы камер*.
3. Откройте настройки группы камер.
4. Поставьте флажок *Дедуплицировать события* и задайте в секундах интервал дедупликации.

Алгоритм работы дедупликации

Алгоритм дедупликации работает следующим образом:

1. В буферном режиме за один сеанс отслеживания на камере сервер получает одно, наилучшее, изображение лица (сеанс отслеживания продолжается до тех пор, пока лицо не исчезнет из поля зрения камеры).
2. Если в пределах одной группы камер произошло несколько сеансов отслеживания на одной или нескольких камерах в течение указанного интервала дедупликации, FindFace Security обработает полученные изображения лиц следующим образом:
   - Если в течение предшествующего периода, равного интервалу дедупликации, есть совпадение с досье, FindFace Security удаляет вновь полученное изображение. В противном случае изображение сохраняется в базе данных.
   - При выполнении дедупликации не совпавших с досье лиц FindFace Security учитывает как сходство между лицами на различных изображениях, так и качество лиц. FindFace Security удаляет все изображения похожих лиц в пределах интервала дедупликации, если они более низкого качества, чем первое в данном интервале. Если новое лицо более высокого качества, оно сохраняется. Это гарантирует, что система дедуплицирует события, не пропуская высококачественные лица, необходимые для видеоаналитики.

1.7.7 Распознавание живых лиц в реальном времени (Liveness)

Примечание: *Liveness-детектор* на CPU работает гораздо медленнее, чем на GPU.

Для обнаружения фальшивых лиц и предотвращения фото-атак используйте интегрированную антиспуфинговую систему, отличающую живые лица от их изображений. Алгоритм анализирует несколько последовательных кадров, регистрируя изменения в мимике и текстуре кожи, и благодаря этому

определяет, является ли лицо перед камерой живым или фальшивым. Это исключает возможность мошенничества с использованием изображения лица на бумаге или экране мобильного устройства.

Liveness-детектор оценивает живость лица с определенным уровнем достоверности и возвращает оценку достоверности вместе с бинарным результатом `Живой человек/изображение`, в зависимости от предустановленного порога достоверности.

В этом разделе:

Включение Liveness-детектора

Для включения Liveness-детектора выполните следующие действия:

1. Откройте файл конфигурации `/etc/findface-video-worker-gpu.ini` (`/etc/findface-video-worker-cpu.ini`). В параметре `liveness` –> `fnk` укажите путь к модели liveness-детектора, как показано ниже.

```
sudo vi /etc/findface-video-worker-gpu.ini

[liveness]
#------------------------------
## path to liveness fnk
fnk = /usr/share/findface-data/models/faceattr/liveness.v3.gpu.fnk
```

```
sudo vi /etc/findface-video-worker-cpu.ini

[liveness]
#------------------------------
## path to liveness fnk
fnk = /usr/share/findface-data/models/faceattr/liveness.v3.cpu.fnk
```

2. Перезапустите `findface-video-worker`.

```
sudo systemctl restart findface-video-worker-gpu
sudo systemctl restart findface-video-worker-cpu
```

Настройка порога Liveness

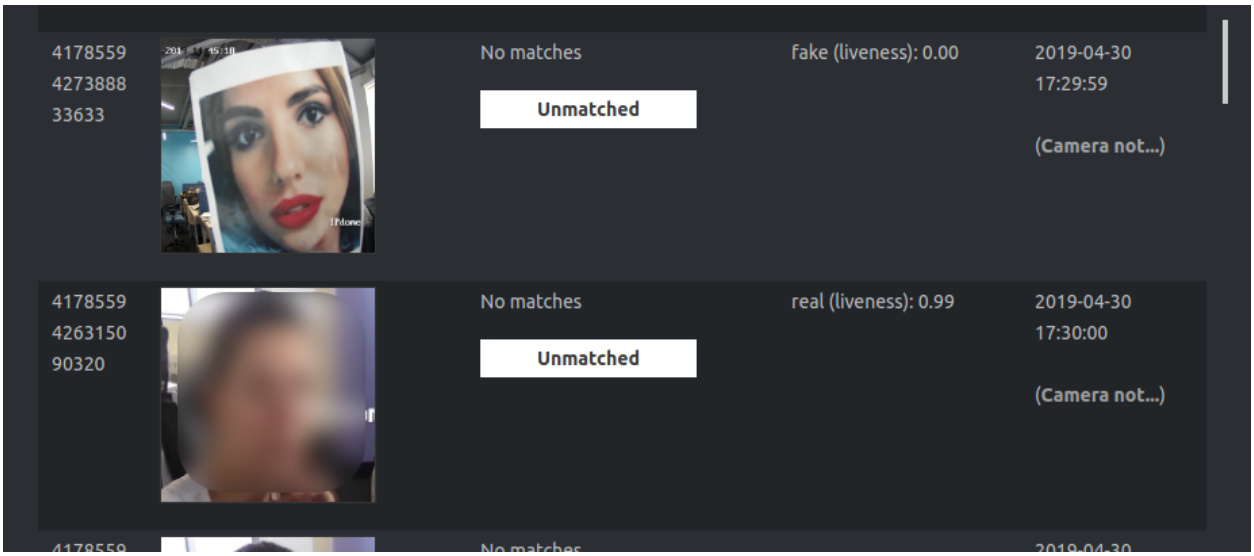
При необходимости вы можете настроить пороговое значение Liveness в файле конфигурации `/etc/ffsecurity/config.py`. Liveness-детектор оценивает "живость" лица с определенной достоверностью. В зависимости от порогового значения достоверности, он возвращает бинарный результат `Живой человек` или `Изображение`.

Примечание: Значение по умолчанию является оптимальным. Перед изменением порога проконсультируйтесь у наших специалистов по адресу support@ntechlab.com.

```
sudo vi /etc/ffsecurity/config.py

'LIVENESS_THRESHOLD': 0.75,
```

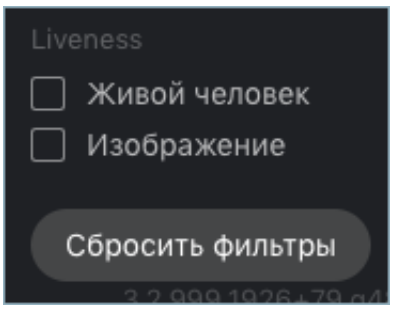
Информация о живом лице в веб-интерфейсе

После настройки Liveness-детектора для каждого события будет отображаться оценка liveness обнаруженного лица.



Примечание: Liveness может принимать значение `null`, если детектор живых лиц отключен или на предоставленном изображении невозможно достоверно оценить Liveness.

Используйте фильтр *Liveness* для просмотра событий только с живыми людьми или только с изображениями, если имели место спуфинговые атаки.



1.7.8 Распознавание атрибутов лица

FindFace Security позволяет распознавать в реальном времени такие атрибуты лица, как пол, возраст, эмоции, очки и/или борода. Данный функционал доступен как на GPU-, так и на CPU-видеодетекторе лиц.

В этом разделе:

Включение распознавания атрибутов лица

Важно: На данном шаге включается распознавание атрибутов лица на уровне HTTP API.

Для того чтобы включить автоматическое распознавание атрибутов лица, откройте файл конфигурации `/etc/findface-extraction-api.ini` и включите соответствующие модели: пол, возраст, эмоции, очки и/или борода. Удостоверьтесь, что для каждой модели вы указали правильный тип ускорения CPU или GPU: он должен совпадать с типом ускорения `findface-extraction-api`. Обратите внимание, что `findface-extraction-api` на CPU может работать только с CPU-моделями, в то время как `findface-extraction-api` на GPU поддерживает как GPU-, так и CPU-модели.

```
sudo vi /etc/findface-extraction-api.ini

models:
  age: faceattr/age.v1.cpu.fnk
  emotions: faceattr/emotions.v1.cpu.fnk
  face: face/grapefruit_480.cpu.fnk
  gender: faceattr/gender.v2.cpu.fnk
  beard: faceattr/beard.v0.cpu.fnk
  glasses3: faceattr/glasses3.v0.cpu.fnk
```

Доступны следующие модели:

Примечание: Вы можете найти модели для распознавания атрибутов лица в каталоге `/usr/share/findface-data/models/faceattr/`.

```
ls /usr/share/findface-data/models/faceattr/
age.v1.cpu.fnk  age.v1.gpu.fnk  beard.v0.cpu.fnk  beard.v0.gpu.fnk  emotions.v1.cpu.fnk  emotions.
↪v1.gpu.fnk  gender.v2.cpu.fnk  gender.v2.gpu.fnk  glasses3.v0.cpu.fnk  glasses3.v0.gpu.fnk ␣
↪liveness.v3.gpu.fnk
```

Атрибут лица	Ускорение	Параметр в файле конфигурации
биометрия лица	CPU	`face: face/grapefruit_480.cpu.fnk  face: face/grapefruit_160.cpu.fnk`
	GPU	`face: face/grapefruit_480.gpu.fnk  face: face/grapefruit_160.gpu.fnk`
возраст	CPU	`age: faceattr/age.v1.cpu.fnk`
	GPU	`age: faceattr/age.v1.gpu.fnk`
пол	CPU	`gender: faceattr/gender.v2.cpu.fnk`
	GPU	`gender: faceattr/gender.v2.gpu.fnk`
эмоции	CPU	`emotions: faceattr/emotions.v1.cpu.fnk`
	GPU	`emotions: faceattr/emotions.v1.gpu.fnk`
очки	CPU	`glasses3: faceattr/glasses3.v0.cpu.fnk`
	GPU	`glasses3: faceattr/glasses3.v0.gpu.fnk`
борода	CPU	`beard: faceattr/beard.v0.cpu.fnk`
	GPU	`beard: faceattr/beard.v0.gpu.fnk`

Совет: Для того чтобы отключить модель распознавания, передайте в соответствующий параметр пустое значение. Не удаляйте сам параметр, поскольку в этом случае будет выполняться поиск модели по умолчанию.

```
models:
  gender: ""
  age: ""
  emotions: ""
```

Перезапустите `findface-extraction-api`.

```
sudo systemctl restart findface-extraction-api
```

После включения моделей обязательно *настройте* отображение атрибутов в веб-интерфейсе.

Отображение результатов распознавания атрибутов в событиях

Для отображения результатов распознавания атрибутов лица в списке событий, добавьте следующую строку в секцию `FFSECURITY`:“ ‘EVENTS_FEATURES’: [‘gender’, ‘age’, ‘emotions’, ‘beard’, ‘glasses’]“, в зависимости от того, какие модели были включены.

Предупреждение: Данная строка должна быть расположена между строками `SF_API_ADDRESS` и `LIVENESS_THRESHOLD`, как показано в примере ниже.

```
sudo vi /etc/ffsecurity/config.py

...
FFSECURITY = {
...
'SF_API_ADDRESS': 'http://127.0.0.1:18411',
'EVENTS_FEATURES': ['gender', 'age', 'emotions', 'beard', 'glasses'],
```

(continues on next page)

(продолжение с предыдущей страницы)

```
'LIVENESS_THRESHOLD': 0.75,
'BEARD_THRESHOLD': 0.7,
 }
```

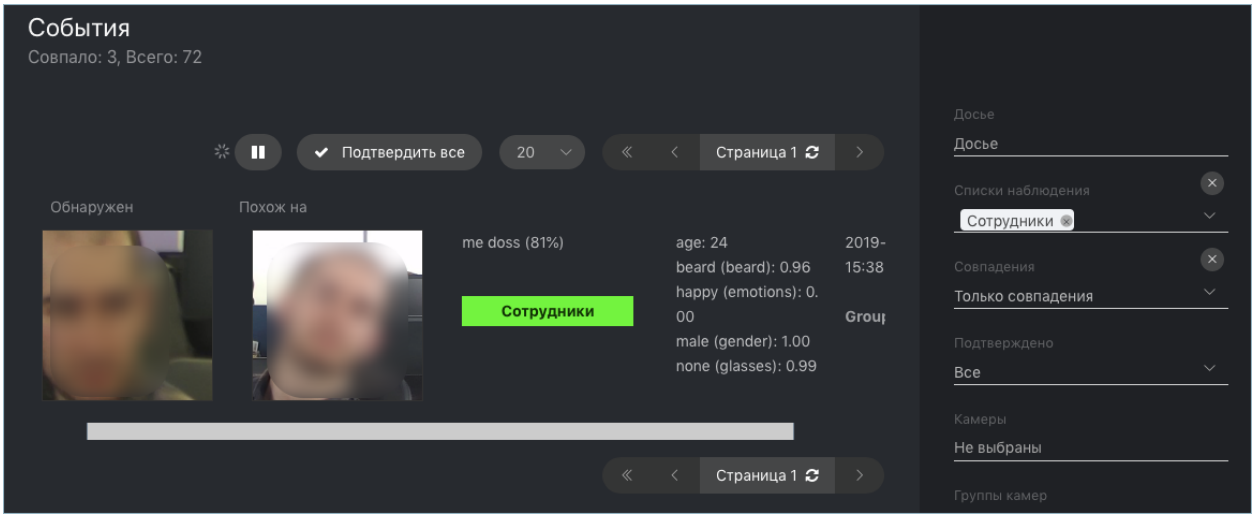
Перезапустите `findface-security`.

```
sudo systemctl restart findface-security
```

Атрибуты лиц в событиях

После настройки распознавания атрибутов лиц результат распознавания будет выводится для каждого обнаруженного лица в следующем формате:

Атрибут лица	Формат результата	Пример
Возраст	Атрибут: `возраст`: количество лет	возраст: 33
Пол	Результат: `мужской/женский` (атрибут: `пол`): уверенность алгоритма в результате	женский (пол): 0.95
Эмоции	Результат: `злость/отвращение/страх/счастье/грусть/удивление` (атрибут: `эмоции`): уверенность алгоритма в результате	счастье (эмоции): 0.99
Очки	Результат: `медицинские/солнечные/нет` (атрибут: `очки`): уверенность алгоритма в результате	нет (очки): 0.87
Борода	Результат: `борода/нет` (атрибут: `борода`): уверенность алгоритма в результате	нет (борода): 0.91



При необходимости выполните фильтрацию событий по атрибутам лица.

1.7.9 Использование нескольких видеокарт

Если на физическом сервере установлено несколько видеокарт, вы можете создать дополнительные экземпляры `findface-extract-api-gpu` или `findface-video-worker-gpu` и распределить их по одному экземпляру на карту.

В этом разделе:

Распределение экземпляров `findface-extraction-api-gpu` по видеокартам

Для распределения экземпляров `findface-extraction-api-gpu` по нескольким видеокартам выполните следующие действия:

1. Остановите исходный сервис `findface-extraction-api-gpu`.

```
sudo service findface-extraction-api stop
```

2. Создайте несколько копий файла конфигурации `findface-extract-api-gpu`, в зависимости от того, какое количество видеокарт будет использоваться для извлечения биометрических образцов. Добавьте соответствующие номера устройств GPU к именам новых файлов конфигурации, как показано в примере ниже (устройства GPU №0 и №6).

```
/etc/findface-extraction-api@0.ini
/etc/findface-extraction-api@6.ini
```

3. Откройте новые файлы конфигурации. Укажите номера устройств GPU, а также номера слушающих портов.

```
sudo vi /etc/findface-extraction-api@0.ini

listen: 127.0.0.1:18666
...

gpu_device: 0
...
```

```
sudo vi /etc/findface-extraction-api@6.ini

listen: 127.0.0.1:18667
...

gpu_device: 6
...
```

4. Запустите новые сервисы.

```
sudo service findface-extraction-api@0 start
sudo service findface-extraction-api@6 start
```

Привязка `findface-video-worker-gpu` к дополнительной видеокарте

Для создания дополнительного экземпляра `findface-video-worker-gpu` и его привязки к свободной видеокарте выполните следующие действия:

1. Отобразите статус исходного сервиса `findface-video-worker-gpu`, выполнив команду:

```
sudo systemctl status findface-video-worker-gpu.service
```

2. Найдите полный путь к сервису в строке `Loaded: loaded (/lib/systemd/system/findface-video-worker-gpu.service; enabled; vendor preset: enabled`. В приведенном примере это `findface-video-worker-gpu.service` (имя может быть другим). Создайте копию сервиса под новым именем.

```
sudo cp /lib/systemd/system/findface-video-worker-gpu.service /lib/systemd/system/findface-
↪video-worker-gpu2.service`
```

3. Таким же образом создайте под новым именем копию исходного файла конфигурации.

```
sudo cp /etc/findface-video-worker-gpu.ini /etc/findface-video-worker-gpu2.ini
```

4. Откройте только что созданный файл конфигурации и актуализируйте номер используемого GPU-устройства.

```
sudo vim /etc/findface-video-worker-gpu2.ini

## cuda device number
device_number = 1
```

5. Откройте новый сервис и укажите только что созданный файл конфигурации.

```
sudo vim /lib/systemd/system/findface-video-worker-gpu2.service

ExecStart=/usr/bin/findface-video-worker-gpu --config /etc/findface-video-worker-gpu2.ini
```

6. Для применения изменений перезагрузите демон `systemd`.

```
sudo systemctl daemon-reload
```

7. Добавьте новый сервис в автозагрузку.

```
sudo systemctl enable findface-video-worker-gpu2.service

Created symlink from /etc/systemd/system/multi-user.target.wants/findface-video-worker-gpu2.
↪service to /lib/systemd/system/findface-video-worker-gpu2.service
```

8. Запустите новый сервис.

```
sudo systemctl start findface-video-worker-gpu2.service
```

9. Проверьте статус обоих сервисов `findface-video-worker-gpu`.

```
sudo systemctl status findface-video-worker-* | grep -i 'Active:' -B 3

 findface-video-worker-gpu2.service - findface-video-worker-gpu daemon
  Loaded: loaded (/lib/systemd/system/findface-video-worker-gpu2.service; enabled; vendor␣
↪preset: enabled)
  Active: active (running) since Thu 2019-07-18 10:32:02 MSK; 1min 11s ago

...

 findface-video-worker-gpu.service - findface-video-worker-gpu daemon
```

(continues on next page)

(продолжение с предыдущей страницы)

```
  Loaded: loaded (/lib/systemd/system/findface-video-worker-gpu.service; enabled; vendor␣
↪preset: enabled)
  Active: active (running) since Mon 2019-07-15 15:18:33 MSK; 2 days ago
```

1.7.10 Прямые API-запросы к базе данных Tarantool

Вы можете использовать HTTP API для извлечения напрямую данных из базы данных Tarantool.

В этом разделе:

Общие сведения

API-запросы к базе данных Tarantool нужно отправлять по адресу `http://<tarantool_host_ip:port>`.

Совет: Порт для API-запросов можно узнать в разделе `FindFace.start` файла конфигурации Tarantool:

```
cat /etc/tarantool/instances.enabled/FindFace.lua

##8001:
FindFace.start("127.0.0.1", 8001)
```

Примечание: В случае если FindFace Security развернут на одиночном физическом сервере, база данных Tarantool по умолчанию будет доступна только локально (`127.0.0.1`). Если необходимо открыть доступ к базе данных Tarantool с удаленного сервера, *внесите изменения* в файл конфигурации `findface-tarantool-server`.

API-запросы к Tarantool могут содержать следующие параметры в сегментах пути:

- `:ver`: версия API (v2 на данный момент).

- `:name`: имя галереи.

Совет: Для получения списка имен галерей на шарде введите следующую команду в адресном поле браузера:

```
http://<tarantool_host_ip:shard_port>/stat/list/1/99
```

Та же самая команда в консоли:

```
curl <tarantool_host_ip:shard_port>/stat/list/1/99 \| jq
```

Вы также можете получить список имен галерей, отправив в Tarantool прямой запрос:

```
echo 'box.space.galleries:select()' | tarantoolctl connect <tarantool_host_ip:shard_port>
```

Имейте в виду, что при значительном количестве шардов в системе произвольно выбранный шард может не включать в себя все существующие галереи. В этом случае отобразите список галерей на нескольких шардах.

Добавление лица

```
POST /:ver/faces/add/:name
```

Параметры в теле:

Массив лиц в представлении JSON со следующими полями:

- `"id"`: id лица в галерее, uint64_t,
- `''facen''`: необработанный вектор признаков, base64,
- `"meta"`: метаданные лица, словарь.

Возвращает:

- HTTP 200 и пустое тело в случае успеха.
- HTTP 404 с описанием ошибки, если галерея с заданным именем не существует.
- HTTP с отличным от 200 статусом и описание ошибки в теле в случае неудачи.

Пример

Запрос

```
curl -D - -s 'http://localhost:8001/v2/faces/add/testgal' --data '
[
  {
    "id": 9223372036854776000,
    "facen": "qgI3vZRv/z...NpO9MdHavW1WuTO=",
    "meta": {
"cam_id": "223900",
```

(continues on next page)

(продолжение с предыдущей страницы)

```
"person_name": "Mary Ostin",

    }
  }
]
```

Ответ

```
HTTP/1.1 200 Ok
Content-length: 1234
Server: Tarantool http (tarantool v1.7.3-673-g23cc4dc)
Connection: keep-alive
```

Удаление лица

```
POST /v2/faces/delete/:name
```

Параметры в теле:

Массив в представлении JSON из списка id лиц, подлежащих удалению.

Возвращает:

- HTTP 200 и пустое тело в случае успеха.
- HTTP 404, если лицо с заданным id не найдено в галерее.
- HTTP с отличным от 200 статусом и описание ошибки в теле в случае неудачи.

Пример

Запрос

```
curl -D -  -s 'http://localhost:8001/v2/faces/delete/testgal' --data '[1, 4, 922, 3]'
```

Ответ

```
HTTP/1.1 200 Ok
Content-length: 111
Server: Tarantool http (tarantool v1.7.3-673-g23cc4dc)
Connection: keep-alive
```

Поиск лица

```
POST /v2/faces/search/:name
```

Параметры в теле:

Поисковый запрос в представлении JSON со следующими полями:

- `limit`: максимальное количество лиц в ответе.
- `sort`: включает сортировку по следующим параметрам: `id`: по возрастанию id, `-id` по убыванию id, `-score`: по убыванию степени схожести (если поиск выполняется по схожим векторам признаков).
- `filter` (фильтры):
  - `facen`: опциональный фильтр по схожести вектора признаков. Передайте словарь со следующими полями: `data`: вектор признаков в формате base64; `score`: диапазон качества лица, поддерживаются только запросы с правой границей 1 (максимальное качество).
  - `id` и `meta/<meta_key>`: фильтры по пользовательскому id лиц и содержимому поля `meta`. Для задания фильтра используются следующие операторы:
    - `range`: диапазон значений, только для числовых полей.
    - `set`: набор значений, одно из которых должно присутствовать в id или метаданных, для числовых и строковых полей.
    - `subset`: набор значений, каждое из которых должно присутствовать в id или метаданных, для числовых и строковых полей.
    - `like`: аналогично `like` в SQL-запросах: поддерживаются только 'aa%' или 'aa%' или '%aa%'. Только для полей `string` и `set[string]`. При использовании `set[string]` фильтр вернет результат, если хотя бы одно из значений прошло проверку.
    - `ilike` (только для полей `string` и `set[string]`): аналогично `like`, но без учета регистра.

Возвращает:

- В случае успеха массив JSON с лицами. Значение в заголовке `X-search-stat` показывает, был ли использован быстрый индекс для поиска: `with_index` или `without_index`.

  Примечание: В API v2 быстрый индекс не используется.

- HTTP с отличным от 200 статусом и описание ошибки в теле в случае неудачи.

Пример

Запрос

```
curl -D - -s 'http://localhost:8001/v2/testgal/search' --data '
{
        "limit": 2,
    "sort": {
                "score": -1
        },
        "filter": {
                "facen": {
                        "data": "qgI3vZRv/zOBQTk9rcirOyZrNpO9MdHavW1WuTO=",
                        "score": [0.75, 1]
```

(continues on next page)

(продолжение с предыдущей страницы)

```
                },
                "id": {
                        "range": [9223372036854000000, 9223372036854999000]
                },
                "meta": {
                        "person_id": {
                                "range": [444, 999]
                        },
                        "cam_id": {
                                "set": ["12767", "8632", "23989"]
                        }
                }
        }
}'
```

Ответ

```
HTTP/1.1 200 Ok
Content-length: 1234
X-search-stat: without_index
Server: Tarantool http (tarantool v1.7.3-673-g23cc4dc)
Connection: keep-alive

{
  "results": [
    {
      "facen": " qgI3vZRv/zOBQTk9rcirOyZrNpO9MdHavW1WuTO=",
      "meta": {
        "timestamp": 0,
        "photo_hash": "",
        "person_id": 777,
        "cam_id": "8632"
      },
      "score": 0.9964,
      "id": 9223372036854776000
    }
  ]
}
```

Редактирование метаданных и/или вектора признаков

```
POST /v2/faces/update/:name
```

Параметры в теле:

Массив лиц в представлении JSON со следующими полями:

- `"id"`: id лица, uint64_t,
- `''facen''`: (опционально) новый вектор признаков, base64. Если параметр отсутствует или `null`, поле в базе данных не обновляется.

- ''meta'': словарь, в котором передаются новые метаданные. Если поле `meta` отсутствует или `null`, оно не обновляется в базе данных.

Возвращает:

- HTTP 200 и словарь со всеми параметрами лица, в том числе неизменными, в случае успеха.
- HTTP 404 с описанием ошибки, если лица с таким id не существует.
- HTTP с отличным от 200 статусом и описание ошибки в теле в случае неудачи.

Пример

Запрос

```
curl -D - -s 'http://localhost:8001/v2/faces/update/sandbox' --data '[{"id":1,"facen":null,"meta":{
↪"m:timestamp":1848}}]'
```

Ответ

```
HTTP/1.1 200 Ok
Content-length: 151
Server: Tarantool http (tarantool v1.7.3-673-g23cc4dc)
Connection: keep-alive

{"meta":{"m:timestamp":1848,"normalized_id":"1_b9pkrf00mjt6h1vmq1kg.png","m:cam_id":"a9f7a973-f07e-
↪469d-a3bd-41ddd510b26f","feat":"{\"score\":0.123}"}, "id":1, ... }
```

Получение списка галерей

```
POST /v2/galleries/list
```

Возвращает:

Массив с галереями, для каждой из которой возвращается имя (`name`) и число лиц (`faces`).

Пример

Запрос

```
curl -D - -s -X POST http://localhost:8001/v2/galleries/list
```

Ответ

```
HTTP/1.1 200 Ok
Content-length: 42
Server: Tarantool http (tarantool v1.7.3-673-g23cc4dc)
Connection: keep-alive

{
  "results": [
    {
      "name": "testgal",
      "faces": 2
    }
  ]
}
```

Получение информации о галерее

```
POST /v2/galleries/get/:name
```

Возвращает:

- HTTP 200 и словарь с параметрами галереи в случае успеха.
- HTTP 404 с описанием ошибки, если галереи с таким именем не существует.
- HTTP с отличным от 200 статусом и описание ошибки в теле в случае неудачи.

Пример

Запрос

```
curl -D - -s -X POST http://localhost:8001/v2/galleries/get/testgal
```

```
HTTP/1.1 200 Ok
Content-length: 11
Server: Tarantool http (tarantool v1.7.3-673-g23cc4dc)
Connection: keep-alive

{"faces":2}
```

Создание галереи

```
POST /v2/galleries/add/:name
```

Возвращает:

- HTTP 200 и пустое тело в случае успеха.
- HTTP с отличным от 200 статусом и описание ошибки в теле в случае неудачи.

Пример

Запрос

```
curl -D - -X POST -s 'http://localhost:8001/v2/galleries/add/123'
```

Ответ

```
HTTP/1.1 409 Conflict
Content-length: 57
Server: Tarantool http (tarantool v1.7.3-673-g23cc4dc)
Connection: keep-alive

{"error":{"message":"gallery already exists","code":409}}
```

Удаление галереи

```
POST /v2/galleries/delete/:name
```

Возвращает:

- HTTP 200 и пустое тело в случае успеха.
- HTTP с отличным от 200 статусом и описание ошибки в теле в случае неудачи.

Пример

Запрос

```
curl -D - -X POST -s 'http://localhost:8001/v2/galleries/delete/123'
```

Ответ

```
HTTP/1.1 204 No content
Content-length: 0
Server: Tarantool http (tarantool v1.7.3-673-g23cc4dc)
Connection: keep-alive
```

1.7.11 Аутентификация по сертификатам КриптоПро

По умолчанию вход пользователей в систему осуществляется по логину и паролю. При необходимости можно включить аутентификацию по сертификатам КриптоПро.

В этом разделе:

Настройка аутентификации по сертификатам КриптоПро

Для настройки аутентификации по сертификатам КриптоПро выполните следующие действия:

1. Загрузите на сервер FindFace Security архивы КриптоПро CSP 4.0 для Linux (x64, deb) и КриптоПро ЭЦП SDK 2.0 Linux x64.
2. Распакуйте загруженные архивы.
3. Из директории `/opt/ffsecurity/lib/python3.5/site-packages/ffsecurity_cproauth` запустите скрипт `build.sh`.

```
chmod +x build.sh
sudo ./build.sh
```

   По требованию скрипта установите дополнительные deb-пакеты из распакованных архивов.

```
Please install lsb-cprocsp-devel from CryptoPro CSP ( https://www.cryptopro.ru/
↪products/csp/downloads )

apt install lsb-cprocsp-devel
...
```

4. Откройте файл конфигурации `findface-security`.

```
sudo vi /etc/ffsecurity/config.py
```

5. Добавьте в него следующие настройки:

```
...
UVICORN_SETTINGS = {
...
'proxy_headers': True
}
...
INSTALLED_APPS.append('ffsecurity_cproauth')
#REST_FRAMEWORK['DEFAULT_AUTHENTICATION_CLASSES'] = ['ffsecurity.auth.TokenAuthentication',
↪'ffsecurity_cproauth.auth.CryptoProOrTokenAuthentication']
```

6. Перезапустите `findface-security`.

```
sudo systemctl restart findface-security.service
```

На этом настройка аутентификации по сертификатам КриптоПро будет завершена. Можно приступить к загрузке сертификатов в систему.

Загрузка сертификата пользователя

Для того чтобы загрузить сертификат пользователя в систему, выполните следующие действия:

1. Перейдите на вкладку *Настройки*. Выберите *Пользователи*.
2. Откройте карточку пользователя.
3. Нажмите *Добавить сертификат*.

1.8 Обслуживание и устранение неисправностей

1.8.1 Обновление FindFace Security до 4.1.x

Предупреждение: FindFace Security 4.1.x использует новую версию Tarantool. Биометрическая база данных из предыдущих версий FindFace Security (4.0 и более ранних) НЕСОВМЕСТИМА с FindFace Security 4.1.x. При обновлении продукта до 4.1.x обязательно используйте функцию *резервного копирования / восстановления* (см. полный алгоритм ниже).

Для обновления FindFace Security с любой предыдущей версии до 4.1.x выполните следующие действия:

1. Откройте файл конфигурации `findface-security`. Сохраните для последующего использования значения следующих параметров: `EXTERNAL_ADDRESS`, `SECRET_KEY`, `VIDEO_DETECTOR_TOKEN`, `ROUTER_URL`.

```
sudo vi /etc/ffsecurity/config.py

EXTERNAL_ADDRESS = "http://172.20.77.58"

...
# use pwgen -sncy 50 1|tr "'" "." to generate your own unique key
SECRET_KEY = 'c8b533847bbf7142102de1349d33a1f6'

FFSECURITY = {
    'VIDEO_DETECTOR_TOKEN': '381b0f4a20495227d04185ab02f5085f',
    ...
    'ROUTER_URL': 'http://172.20.77.58',
    ...
}
```

2. Остановите сервис `findface-security`.

```
sudo systemctl stop findface-security*
```

3. Создайте резервную копию биометрической базы данных на основе Tarantool в любой выбранной директории, например, `/tmp/dump`.

Совет: Подробнее см. *Резервное копирование и восстановление хранилищ данных*.

```
mkdir -p /tmp/dump
cd /tmp/dump
sudo findface-storage-api-dump -config /etc/findface-sf-api.ini
```

4. Установите apt-репозиторий с новой версией FindFace Security, используя консольный инсталлятор согласно инструкции в *этом разделе*.

5. Установите сервисы из репозитория в соответствии с текущей архитектурной схемой.

   CPU-версия:

```
sudo apt update
sudo apt install ffsecurity ffsecurity-ui findface-extraction-api findface-ntls findface-sf-
↪api findface-tarantool-server findface-upload findface-video-manager findface-video-worker-
↪cpu
```

   GPU-версия:

```
sudo apt update
sudo apt install ffsecurity ffsecurity-ui findface-extraction-api-gpu findface-ntls findface-
↪sf-api findface-tarantool-server findface-upload findface-video-manager findface-video-
↪worker-gpu findface-gpudetector-data
```

6. Откройте файл конфигурации `findface-security` и вставьте в него сохраненные значения параметров `EXTERNAL_ADDRESS`, `SECRET_KEY`, `VIDEO_DETECTOR_TOKEN` и `ROUTER_URL`.

```
sudo vi /etc/ffsecurity/config.py
```

7. Измените структуру базы данных Tarantool с помощью файла `tnt_schema.lua` из новой версии.

```
sudo findface-security make_tnt_schema | sudo tee /etc/ffsecurity/tnt_schema.lua
```

8. Остановите шарды `findface-tarantool-server`. Удалите базу данных Tarantool (базу по умолчанию или шарды).

```
sudo systemctl stop 'tarantool@*'

sudo rm -R /opt/ntech/var/lib/tarantool/shard-00*/index/*
sudo rm -R /opt/ntech/var/lib/tarantool/shard-00*/snapshots/*
sudo rm -R /opt/ntech/var/lib/tarantool/shard-00*/xlogs/*
```

9. Перезапустите шарды `findface-tarantool-server`.

```
TNT=$(ls /etc/tarantool/instances.enabled/ | wc -l)
for i in $(seq 1 $TNT); do sudo systemctl start tarantool@shard-00$i.service ; done
```

10. Восстановите базу данных Tarantool из резервной копии.

```
cd /tmp/dump

for x in *.json; do sudo findface-storage-api-restore -config /etc/findface-sf-api.ini < "$x";
↪ done
```

11. Перенесите схему базы данных из FindFace Security в `PostgreSQL`, заново создайте группы пользователей с *предустановленными* правами и первого пользователя с правами администратора.

```
sudo findface-security migrate
sudo findface-security create_groups
sudo findface-security create_default_user
```

12. Перезагрузите сервисы.

```
sudo systemctl restart findface-security.service
sudo systemctl restart findface-extraction-api findface-video-worker* findface-video-manager␣
↪findface-sf-api
```

1.8.2 Резервное копирование и восстановление хранилищ данных

Этот раздел посвящен резервному копированию и восстановлению хранилищ данных FindFace Security. Система использует следующие хранилища:

- Биометрическая база данных на основе Tarantool, в которой хранятся биометрические образцы (векторы признаков) и события идентификации лиц.
- Основная база данных системы на основе PostgreSQL, в которой хранятся внутренние данные системы, досье, учетные записи пользователей и настройки камер.
- Каталог `/var/lib/ffsecurity/uploads`, в котором хранятся загруженные в досье фотографии, видеофайлы, а также такие артефакты событий, как полные кадры, миниатюры лиц и нормализованные изображения лиц.
- Каталог `/var/lib/ffupload/`, в котором хранятся только такие артефакты событий, как миниатюры лиц.

В этом разделе:

Резервное копирование и восстановление биометрической базы данных

В биометрической базе данных на основе Tarantool есть 3 галереи:

- `ffsec_dossier_face`: биометрические образцы, извлеченные из фотографий в досье.
- `ffsec_events`: биометрические образцы, извлеченные из лиц, обнаруженных на видео.
- `ffsec_monitoring`: биометрические образцы из всех досье в мониторинге (активных).

Функционал резервного копирования и восстановления базы данных позволяет при необходимости восстанавливать содержимое данных галерей.

Для предотвращения потери данных создание резервной копии биометрической базы данных рекомендуется выполнять по крайней мере 1 раз в неделю.

Резервную копию базы данных также нужно создать перед *миграцией* системы на другую биометрическую модель.

Утилиты

Для резервного копирования и восстановления биометрической базы данных FindFace Security необходимы следующие утилиты:

1. резервное копирование: `findface-storage-api-dump`,

2. восстановление: `findface-storage-api-restore`.

Данные утилиты автоматически устанавливаются вместе с компонентом `findface-sf-api`.

Резервное копирование базы данных

Для резервного копирования биометрической базы данных используйте утилиту `findface-storage-api-dump` следующим образом:

Важно: Сервисы `findface-tarantool-server` и `findface-sf-api` должны быть активны.

Примечание: Резервное копирование можно также применить к распределенной базе данных. В этом случае утилита `findface-storage-api-dump` создаст резервные копии всех шардов, указанных в `/etc/findface-sf-api.ini`.

---

1. На сервере с установленным `findface-sf-api` создайте каталог для хранения резервных копий (`/tmp/backup` в примере ниже).
2. Запустите утилиту `findface-storage-api-dump` следующей командой:

```
sudo findface-storage-api-dump -output-dir=/tmp/backup -config /etc/findface-sf-api.ini
```

   Утилита создаст резервные копии всех галерей и запишет их в указанный каталог в виде файлов с соответствующими именами `ffsec_dossier_face.json`, `ffsec_events.json` и `ffsec_monitoring.json`. Эти файлы содержат все данные, необходимые для полного восстановления галерей.

Восстановление базы данных

Для восстановления биометрической базы данных выполните следующие действия:

1. Используя HTTP API, создайте галереи в базе данных: `ffsec_dossier_face`, `ffsec_events`, `ffsec_monitoring`.

   Совет: См. *HTTP API*.

```
curl -D - -X POST -s 'http://localhost:8101/v2/galleries/add/ffsec_dossier_face'
curl -D - -X POST -s 'http://localhost:8101/v2/galleries/add/ffsec_events'
curl -D - -X POST -s 'http://localhost:8101/v2/galleries/add/ffsec_monitoring'
```

2. Запустите утилиту `findface-storage-api-restore` для всех файлов в папке с резервными копиями:

```
sudo findface-storage-api-restore -config /etc/findface-sf-api.ini /tmp/backup/*.json
```

   Процесс восстановления можно при необходимости прервать с сохранением выполненной работы. Для того чтобы продолжить процесс после прерывания, снова запустите утилиту `findface-storage-api-restore`.

См.также:

- *Опции резервного копирования базы данных*
- *Опции восстановления базы данных*

Резервное копирование основной базы данных

Для резервного копирования базы данных PostgreSQL выполните команду:

```
sudo -u postgres pg_dump ffsecurity > ffsecurity_postgres_backup.sql
```

Резервное копирование артефактов

Артефакты FindFace Security, такие как загруженные фотографии, видеофайлы и артефакты событий (полные кадры, миниатюры лиц и нормализованные изображения лиц) хранятся в следующих каталогах:

- `/var/lib/ffsecurity/uploads`
- `/var/lib/ffupload/`

Для резервного копирования артефактов выполните команду:

```
tar -cvzf var_lib_ffsecurity_uploads.tar.gz /var/lib/ffsecurity/uploads
tar -cvzf var_lib_ffupload.tar.gz /var/lib/ffupload/
```

1.8.3 Миграция на другую модель биометрического образца

Совет: Не стесняйтесь обращаться к нашим специалистам по вопросам миграции по адресу support@ntechlab.com.

Важно: Перед миграцией обязательно создайте *резервную копию* базы данных.

Иногда вам может потребоваться миграция экземпляра FindFace Security на другую модель нейронной сети. Как правило, данная процедура необходима при обновлении до последней версии продукта.

Для того чтобы мигрировать систему на другую модель биометрического образца, используйте утилиту `findface-sf-api-migrate`. Для того чтобы передать настройки миграции, запустите ее с опцией `-config` и укажите файл конфигурации, показанный в примере ниже.

```
findface-sf-api-migrate -config <migration.ini>
```

Пример файла конфигурации:

```
extraction-api:
  timeouts:
    connect: 5s
    response_header: 30s
    overall: 35s
    idle_connection: 0s
  extraction-api: http://127.0.0.1:18666
storage-api-from: # current location of the gallery
```

(continues on next page)

(продолжение с предыдущей страницы)

```
  timeouts:
    connect: 5s
    response_header: 30s
    overall: 35s
    idle_connection: 10s
  max-idle-conns-per-host: 20
  shards:
    - master: http://127.0.0.1:8001/v2/
      slave: ""
storage-api-to:
  timeouts:
    connect: 5s
    response_header: 30s
    overall: 35s
    idle_connection: 10s
  max-idle-conns-per-host: 20
  shards:
    - master: http://127.0.0.1:8002/v2/
      slave: ""
workers_num: 3
faces_limit: 100
extraction_batch_size: 8
normalized_storage:
   type: webdav
  enabled: True
  webdav:
    upload-url: http://127.0.0.1:3333/uploads/
  s3:
    endpoint: 172.20.77.75:9000
    bucket-name: sf-api-normalized
    access-key: W0G6EQT6MC3BZC8136DW
    secret-access-key: XnottrdxRFp70wfEGdkvKgkzKZ3mEa2Y9bYmob4I
    secure: False
    region: ""
    operation-timeout: 10
    public-url: 123
```

Параметр	Описание
`extraction-api -> extraction-api`	Адрес `findface-extraction-api` с новой моделью биометрического образца в файле конфигурации.
`storage-api-from`	Прежнее хранилище биометрических образцов (до миграции)
`storage-api-to`	Новое хранилище биометрических образцов (после миграции)
`normalized_storage -> upload-url`	Хранилище нормализованных изображений лиц

1.8.4 Изменение структуры биометрической базы данных

В некоторых случаях вам может потребоваться применить новую структурную схему к биометрической базе данных Tarantool, например, при обновлении до последней версии продукта или если необходимо усовершенствовать структуру базы данных, добавив в нее дополнительные параметры, расширенные метаданные лиц и т. д.

В этом разделе:

О структуре биометрической базы данных

В FindFace Security структура базы данных задается через файл `tnt_schema.lua`.

Структура представляет собой набор полей, каждое из которых описывается следующими параметрами:

- id: id поля;
- name: название поля, должно совпадать с названием соответствующего параметра лица;
- field_type: тип данных;
- default: значение по умолчанию. Если значение по умолчанию больше '1e14 – 1', то его следует записывать в виде строки, т. е. `"123123.."` вместо `123123..`

Используемый по умолчанию файл `tnt_schema.lua` приведен ниже:

```
scheme = {
    -- internal.normalized_id:
    {
        default = '',
        field_type = 'string',
        id = 1,
        name = 'normalized_id',
    },
    -- internal.feat:
    {
        default = '',
        field_type = 'string',
        id = 2,
        name = 'feat',
    },
    -- <class 'ffsecurity.entities_tnt.event.models.ListEvent'>.m:acknowledged:
    {
        default = 0,
        field_type = 'unsigned',
        id = 3,
        name = 'm:acknowledged',
    },
    -- <class 'ffsecurity.entities_tnt.event.models.ListEvent'>.m:acknowledged_by:
    {
        default = 0,
        field_type = 'unsigned',
        id = 4,
        name = 'm:acknowledged_by',
    },
    -- <class 'ffsecurity.entities_tnt.event.models.ListEvent'>.m:acknowledged_date:
    {
        default = 0,
```

(continues on next page)

(продолжение с предыдущей страницы)

```
    field_type = 'unsigned',
    id = 5,
 name = 'm:acknowledged_date',
},
-- <class 'ffsecurity.entities_tnt.event.models.ListEvent'>.m:acknowledged_reaction:
{
    default = '',
    field_type = 'string',
    id = 6,
    name = 'm:acknowledged_reaction',
},
-- <class 'ffsecurity.entities_tnt.event.models.ListEvent'>.m:camera:
{
    default = 0,
    field_type = 'unsigned',
    id = 7,
    name = 'm:camera',
},
-- <class 'ffsecurity.entities_tnt.event.models.ListEvent'>.m:camera_group:
{
    default = 0,
    field_type = 'unsigned',
    id = 8,
    name = 'm:camera_group',
},
-- <class 'ffsecurity.entities_tnt.event.models.ListEvent'>.m:confidence:
{
    default = "10000000000000000000",
    field_type = 'unsigned',
    id = 9,
    name = 'm:confidence',
},
-- <class 'ffsecurity.entities_tnt.event.models.ListEvent'>.m:created_date:
{
    default = 0,
    field_type = 'unsigned',
    id = 10,
    name = 'm:created_date',
},
-- <class 'ffsecurity.entities_tnt.event.models.ListEvent'>.m:episode:
{
    default = 0,
    field_type = 'unsigned',
    id = 11,
    name = 'm:episode',
},
-- <class 'ffsecurity.entities_tnt.event.models.ListEvent'>.m:f_age:
{
    default = "10000000000000000000",
    field_type = 'unsigned',
    id = 12,
    name = 'm:f_age',
},
-- <class 'ffsecurity.entities_tnt.event.models.ListEvent'>.m:f_beard_class:
{
    default = '',
```

(continues on next page)

(продолжение с предыдущей страницы)

```
        field_type = 'string',
        id = 13,
        name = 'm:f_beard_class',
    },
    -- <class 'ffsecurity.entities_tnt.event.models.ListEvent'>.m:f_beard_score:
    {
        default = "10000000000000000000",
        field_type = 'unsigned',
        id = 14,
        name = 'm:f_beard_score',
    },
    -- <class 'ffsecurity.entities_tnt.event.models.ListEvent'>.m:f_emotions_class:
    {
        default = '',
        field_type = 'string',
        id = 15,
        name = 'm:f_emotions_class',
    },
    -- <class 'ffsecurity.entities_tnt.event.models.ListEvent'>.m:f_emotions_score:
    {
        default = "10000000000000000000",
        field_type = 'unsigned',
        id = 16,
        name = 'm:f_emotions_score',
    },
    -- <class 'ffsecurity.entities_tnt.event.models.ListEvent'>.m:f_gender_class:
    {
        default = '',
        field_type = 'string',
        id = 17,
        name = 'm:f_gender_class',
    },
    -- <class 'ffsecurity.entities_tnt.event.models.ListEvent'>.m:f_gender_score:
    {
        default = "10000000000000000000",
        field_type = 'unsigned',
        id = 18,
        name = 'm:f_gender_score',
    },
    -- <class 'ffsecurity.entities_tnt.event.models.ListEvent'>.m:f_glasses_class:
    {
        default = '',
        field_type = 'string',
        id = 19,
        name = 'm:f_glasses_class',
    },
    -- <class 'ffsecurity.entities_tnt.event.models.ListEvent'>.m:f_glasses_score:
    {
        default = "10000000000000000000",
        field_type = 'unsigned',
        id = 20,
        name = 'm:f_glasses_score',
    },
    -- <class 'ffsecurity.entities_tnt.event.models.ListEvent'>.m:f_liveness_class:
    {
        default = '',
```

(continues on next page)

(продолжение с предыдущей страницы)

```
        field_type = 'string',
        id = 21,
        name = 'm:f_liveness_class',
    },
    -- <class 'ffsecurity.entities_tnt.event.models.ListEvent'>.m:f_liveness_score:
    {
        default = "10000000000000000000",
        field_type = 'unsigned',
        id = 22,
        name = 'm:f_liveness_score',
    },
    -- <class 'ffsecurity.entities_tnt.event.models.ListEvent'>.m:f_race_class:
    {
        default = '',
        field_type = 'string',
        id = 23,
        name = 'm:f_race_class',
    },
    -- <class 'ffsecurity.entities_tnt.event.models.ListEvent'>.m:f_race_score:
    {
        default = "10000000000000000000",
        field_type = 'unsigned',
        id = 24,
        name = 'm:f_race_score',
    },
    -- <class 'ffsecurity.entities_tnt.event.models.ListEvent'>.m:thumbnail:
    {
        default = '',
        field_type = 'string',
        id = 25,
        name = 'm:thumbnail',
    },
    -- <class 'ffsecurity.entities_tnt.event.models.ListEvent'>.m:frame:
    {
        default = '',
        field_type = 'string',
        id = 26,
        name = 'm:frame',
    },
    -- <class 'ffsecurity.entities_tnt.event.models.ListEvent'>.m:frame_coords_bottom:
    {
        default = 0,
        field_type = 'unsigned',
        id = 27,
        name = 'm:frame_coords_bottom',
    },
    -- <class 'ffsecurity.entities_tnt.event.models.ListEvent'>.m:frame_coords_left:
    {
        default = 0,
        field_type = 'unsigned',
        id = 28,
        name = 'm:frame_coords_left',
    },
    -- <class 'ffsecurity.entities_tnt.event.models.ListEvent'>.m:frame_coords_right:
    {
        default = 0,
```

(continues on next page)

(продолжение с предыдущей страницы)

```
        field_type = 'unsigned',
        id = 29,
        name = 'm:frame_coords_right',
    },
    -- <class 'ffsecurity.entities_tnt.event.models.ListEvent'>.m:frame_coords_top:
    {
        default = 0,
        field_type = 'unsigned',
        id = 30,
        name = 'm:frame_coords_top',
    },
    -- <class 'ffsecurity.entities_tnt.event.models.ListEvent'>.m:pk:
    {
        default = 0,
        field_type = 'unsigned',
        id = 31,
        name = 'm:pk',
    },
    -- <class 'ffsecurity.entities_tnt.event.models.ListEvent'>.m:matched:
    {
        default = 0,
        field_type = 'unsigned',
        id = 32,
        name = 'm:matched',
    },
    -- <class 'ffsecurity.entities_tnt.event.models.ListEvent'>.m:matched_dossier:
    {
        default = 0,
        field_type = 'unsigned',
        id = 33,
        name = 'm:matched_dossier',
    },
    -- <class 'ffsecurity.entities_tnt.event.models.ListEvent'>.m:matched_face:
    {
        default = 0,
        field_type = 'unsigned',
        id = 34,
        name = 'm:matched_face',
    },
    -- <class 'ffsecurity.entities_tnt.event.models.ListEvent'>.m:matched_lists:
    {
        default = {},
        field_type = 'set[unsigned]',
        id = 35,
        name = 'm:matched_lists',
    },
    -- <class 'ffsecurity.entities_tnt.event.models.ListEvent'>.m:normalized_photo:
    {
        default = '',
        field_type = 'string',
        id = 36,
        name = 'm:normalized_photo',
    },
    -- <class 'ffsecurity.entities_tnt.event.models.ListEvent'>.m:quality:
    {
        default = "10000000000000000000",
```

(continues on next page)

(продолжение с предыдущей страницы)

```
        field_type = 'unsigned',
        id = 37,
        name = 'm:quality',
    },
    -- <class 'ffsecurity.entities_tnt.event.models.ListEvent'>.m:scores:
    {
        default = '',
        field_type = 'string',
        id = 38,
        name = 'm:scores',
    },
    -- <class 'ffsecurity.entities_tnt.event.models.ListEvent'>.m:video_source:
    {
        default = 0,
        field_type = 'unsigned',
        id = 39,
        name = 'm:video_source',
    },
    -- <class 'ffsecurity.entities_tnt.dossier_face.models.DossierFace'>.m:dossier:
    {
        default = 0,
        field_type = 'unsigned',
        id = 40,
        name = 'm:dossier',
    },
    -- <class 'ffsecurity.entities_tnt.dossier_face.models.DossierFace'>.m:modified_date:
    {
        default = 0,
        field_type = 'unsigned',
        id = 41,
        name = 'm:modified_date',
    },
    -- <class 'ffsecurity.entities_tnt.dossier_face.models.DossierFace'>.m:source_photo:
    {
        default = '',
        field_type = 'string',
        id = 42,
        name = 'm:source_photo',
    },
    -- <class 'ffsecurity.entities_tnt.dossier_face.models.DossierFace'>.m:source_photo_name:
    {
        default = '',
        field_type = 'string',
        id = 43,
        name = 'm:source_photo_name',
    },
}
-- Fields referenced by multiple models: m:frame_coords_left, m:pk, m:frame_coords_top, m:created_
↪date, m:frame_coords_right, m:thumbnail, m:frame_coords_bottom
```

Изменение структуры

Для изменения структуры базы данных выполните следующие действия:

1. Остановите сервис `findface-security`.

```
sudo systemctl stop findface-security.service
```

2. Создайте резервную копию биометрической базы данных в любой выбранной директории, например, `/tmp/dump`.

Совет: Подробнее см. *Резервное копирование и восстановление хранилищ данных*.

```
mkdir -p /tmp/dump
cd /tmp/dump
sudo findface-storage-api-dump -config /etc/findface-sf-api.ini
```

3. Подготовьте файл `tnt_schema.lua` с новой структурой базы данных.
4. Измените структуру базы данных с помощью подготовленного файла `tnt_schema.lua`.

```
sudo findface-security make_tnt_schema | sudo tee /etc/ffsecurity/tnt_schema.lua
```

5. Откройте файл конфигурации Tarantool. Убедитесь, что перед секцией `FindFace.start` добавлена строка `dofile("/etc/ffsecurity/tnt_schema.lua")`, а переменная `meta_scheme=meta_scheme` определена в параметрах `FindFace.start`.

```
sudo vi /etc/tarantool/instances.enabled/<shard_00N>.lua

dofile("/etc/ffsecurity/tnt_schema.lua")

FindFace.start("127.0.0.1", 8101, {
    license_ntls_server="127.0.0.1:3133",
    facen_size=480,
    meta_scheme = meta_scheme
 })
```

6. Удалите базу данных Tarantool (базу по умолчанию или шарды).

```
sudo rm -f /opt/ntech/var/lib/tarantool/default/{index,snapshots,xlogs}/*

sudo rm -f /opt/ntech/var/lib/tarantool/shard-001/{index,snapshots,xlogs}/*
...
sudo rm -f /opt/ntech/var/lib/tarantool/shard-00N/{index,snapshots,xlogs}/*
```

7. Восстановите базу данных Tarantool из резервной копии.

Важно: Если некоторые прежние поля отсутствуют в новой структуре базы данных, сначала потребуется вручную удалить соответствующие данные из резервной копии.

```
cd /tmp/dump
for x in *.json; do curl -X POST "http://127.0.0.1:18411/v2/galleries/${x%%.json}"; done
for x in *.json; do sudo findface-storage-api-restore -config /etc/findface-sf-api.ini < "$x";
↪ done
```

8. Перезапустите сервис `findface-security`.

```
sudo systemctl stop findface-security.service
```

1.8.5 Удаление экземпляра продукта

Вы можете автоматически удалить FindFace Security вместе с базой данных с помощью скрипта `ffsec_uninstall.sh`. Перед удалением будут созданы резервные копии файлов конфигурации и базы данных.

Выполните следующие действия:

1. Загрузите скрипт `ffsec_uninstall.sh` в любой каталог на сервере установки (например, в `/home/username/`).
2. Из данного каталога сделайте скрипт исполняемым.

```
chmod +x ffsec_uninstall.sh
```

3. Запустите скрипт.

```
sudo ./ffsec_uninstall.sh
```

4. Ответьте **all** на вопрос интерактивного мастера удаления, чтобы полностью удалить FindFace Security вместе с базой данных.

1.8.6 Проверка статуса компонентов

Проверьте статус компонентов, если вы столкнулись с проблемой в системе.

Компонент	Команда для просмотра статуса сервиса
`findface-extraction-api`	sudo systemctl status findface-extraction-api.service
`findface-sf-api`	sudo systemctl status findface-sf-api.service
`findface-tarantool-server`	sudo systemctl status tarantool@FindFace.service
`findface-video-manager`	sudo systemctl status findface-video-manager.service
`findface-video-worker`	sudo systemctl status findface-video-worker*.service
`findface-ntls`	sudo systemctl status findface-ntls
`findface-security`	sudo systemctl status findface-security*
`etcd`	sudo systemctl status etcd.service
`NginX`	sudo systemctl status nginx.service
`memcached`	sudo systemctl status memcached.service
`postgresql`	sudo systemctl status postgresql*
`redis`	sudo systemctl status redis.service

1.8.7 Лог-файлы

При разборе нештатных ситуаций используйте логи, содержащие подробную деталировку всех событий, произошедших в системе.

Компонент	Команда для просмотра лога
findface-extraction-api	sudo tail -f /var/log/syslog | grep extraction-api
findface-sf-api	sudo tail -f /var/log/syslog | grep sf-api
findface-tarantool-server	sudo tail -f /var/log/tarantool/FindFace.log
findface-video-manager	sudo tail -f /var/log/syslog | grep video-manager
findface-video-worker	sudo tail -f /var/log/syslog | grep video-worker
findface-security	sudo tail -f /var/log/syslog | grep findface-security
findface-ntls	sudo tail -f /var/log/syslog | grep ntls
findface-security	sudo tail -f /var/log/syslog | grep security
etcd	sudo tail -f /var/log/syslog | grep etcd

Вы также можете посмотреть аудит-логи для каждого компонента. Для этого используйте команду `journalctl -u <component>`, например:

```
journalctl -u findface-extraction-api
```

Важно: Для того чтобы включить хранение аудит-логов на жестком диске в ОС Ubuntu, в файле `/etc/systemd/journald.conf` раскомментируйте и измените параметр `Storage` следующим образом:

```
sudo vi /etc/systemd/journald.conf
...
[Journal]
Storage=persistent
```

При необходимости также раскомментируйте и измените значение параметра `SystemMaxUse`. Данный параметр определяет в процентах максимальный объем логов на жестком диске (по умолчанию 10%).

```
SystemMaxUse=15
```

Для того чтобы просмотреть аудит-логи в ОС Ubuntu, выполните следующую команду:

```
journalctl -o verbose SYSLOG_IDENTIFIER=ffsecurity
```

При расшифровке аудит-логов в первую очередь обращайте внимание на следующие параметры:

- `REQUEST_USER`: пользователь, который выполнил изменения;
- `REQUEST_PATH`: URL запроса;
- `REQUEST_DATA`: данные запроса.

Ниже приведен пример лога создания досье с `id=1879` пользователем `admin`:

```
Fr 2017-12-22 17:53:32.436258 MSK [s=0b5566699751426983e13241301205e9;i=e26015;
↪b=907c34cc1fde4398af63bb575587d9ba;m=246f620c449;t=560eefaf59bc5;x=ed60a136c8fc6362]
   PRIORITY=6
   _UID=123
   _GID=130
   _CAP_EFFECTIVE=0
   _BOOT_ID=907c34cc1fde4398af63bb575587d9ba
   _MACHINE_ID=a3eea61c03e041ef8e64d5c72f5fce40
   _HOSTNAME=ntechadmin
   SYSLOG_IDENTIFIER=ffsecurity
   THREAD_NAME=MainThread
```

(continues on next page)

(продолжение с предыдущей страницы)

```
   _TRANSPORT=journal
   _PID=6579
   _COMM=findface-securi
   _EXE=/opt/ffsecurity/bin/python3
   _CMDLINE=/opt/ffsecurity/bin/python /opt/ffsecurity/bin/findface-security runworker
   _SYSTEMD_CGROUP=/system.slice/system-findface\x2dsecurity\x2dworker.slice/findface-security-
↪worker@4.service
   _SYSTEMD_UNIT=findface-security-worker@4.service
   _SYSTEMD_SLICE=system-findface\x2dsecurity\x2dworker.slice
   CODE_FILE=/opt/ffsecurity/lib/python3.5/site-packages/ffsecurity/mixins.py
   CODE_LINE=94
   CODE_FUNC=finalize_response
   REQUEST_USER=admin
   LOGGER=ffsecurity.audit
   MESSAGE=N8Be05il POST /dossier-faces/ 201 by admin
   REQUEST_DATA={"dossier": "'1879'", "source_photo": "<InMemoryUploadedFile: 14927016033292449.
↪jpeg (image/jpeg)>"}
   REQUEST_PATH=/dossier-faces/
   REQUEST_ID=N8Be05il
   _SOURCE_REALTIME_TIMESTAMP=1513954412436258
```

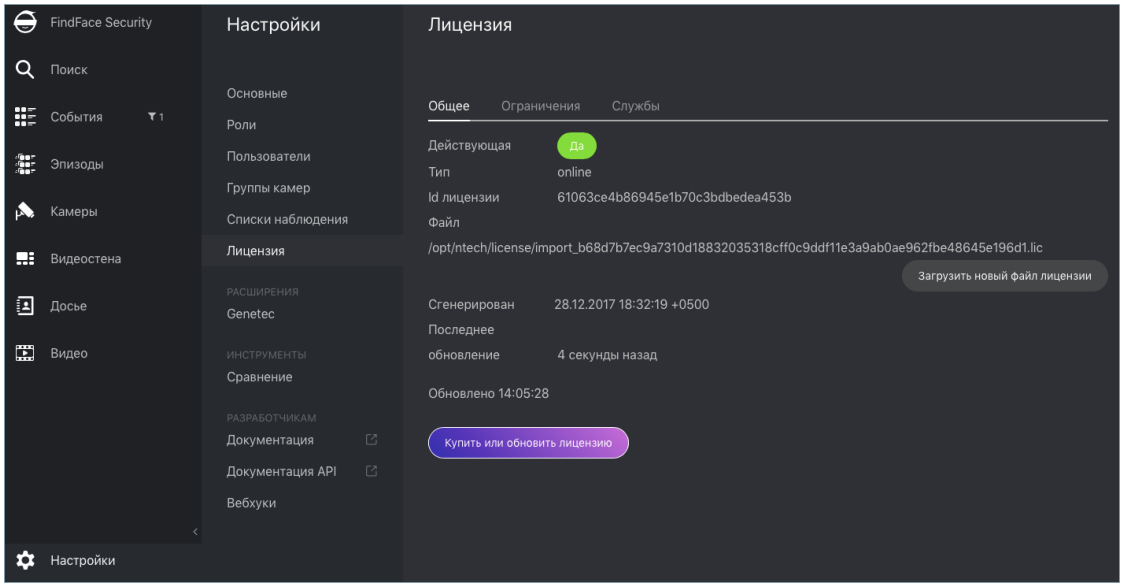
В следующем примере для досье с `id=1879` запрашивается список лиц:

```
Fr 2017-12-22 17:53:32.475467 MSK [s=0b5566699751426983e13241301205e9;i=e26016;
↪b=907c34cc1fde4398af63bb575587d9ba;m=246f6215d82;t=560eefaf634fe;x=b1374a144a46b5cd]
   PRIORITY=6
   _UID=123
   _GID=130
   _CAP_EFFECTIVE=0
   _BOOT_ID=907c34cc1fde4398af63bb575587d9ba
   _MACHINE_ID=a3eea61c03e041ef8e64d5c72f5fce40
   _HOSTNAME=ntechadmin
   SYSLOG_IDENTIFIER=ffsecurity
   THREAD_NAME=MainThread
   _TRANSPORT=journal
   _COMM=findface-securi
   _EXE=/opt/ffsecurity/bin/python3
   _CMDLINE=/opt/ffsecurity/bin/python /opt/ffsecurity/bin/findface-security runworker
   _SYSTEMD_SLICE=system-findface\x2dsecurity\x2dworker.slice
   _PID=6588
   _SYSTEMD_CGROUP=/system.slice/system-findface\x2dsecurity\x2dworker.slice/findface-security-
↪worker@2.service
   _SYSTEMD_UNIT=findface-security-worker@2.service
   CODE_FILE=/opt/ffsecurity/lib/python3.5/site-packages/ffsecurity/mixins.py
   CODE_LINE=94
   CODE_FUNC=finalize_response
   REQUEST_USER=admin
   REQUEST_DATA={}
   LOGGER=ffsecurity.audit
   MESSAGE=Dee7Qvy4 GET /dossier-faces/?dossier=1879&limit=1000 200 by admin
   REQUEST_ID=Dee7Qvy4
   REQUEST_PATH=/dossier-faces/?dossier=1879&limit=1000
   _SOURCE_REALTIME_TIMESTAMP=1513954412475467
```

1.8.8 Лицензирование

Просмотр и обновление лицензии

Для просмотра текущей информации по лицензии или ее обновления, перейдите в *Настройки -> Лицензия*.



Устранение неполадок с лицензированием и `findface-ntls`

При устранении неполадок с лицензией и сервером **`findface-ntls`** (см. *Принцип лицензирования*) первым шагом является получение информации о лицензии и статусе сервера. Это можно сделать, отправив API-запрос в **`findface-ntls`**. Действия по устранению неполадок предпринимаются в учетом содержания API-ответа.

Совет: По вопросам устранения неполадок обращайтесь к нашим специалистам по адресу support@ntechlab.com.

Для получения информации о *лицензии* FindFace Security и статусе **`findface-ntls`**, выполните в консоли сервера **`findface-ntls`** следующую команду:

```
curl http://localhost:3185/license.json -s | jq
```

Ответ будет возвращен в формате JSON. Одним из наиболее значимых параметров в ответе является **`last_updated`**. Он показывает в секундах, как давно в последний раз проверялась локальная лицензия.

Интерпретируйте значение параметра **`last_updated`** следующим образом:

- [0, 5] — все работает отлично.
- (5, 30] — возможно имеют место быть какие-то проблемы со связью, либо с локальным накопителем, где хранятся файлы лицензий.
- (30; 120] — почти наверняка случилось что-то нехорошее.
- (120; ∞) — не удается получить ответ от источника лицензирования в течение длительного времени. Необходимо вмешательство.

- `"valid": false`: связь с источником лицензирования так и не была установлена.

```
curl http://localhost:3185/license.json -s | jq
{
  "name": "NTLS",
  "time": 1565186356,
  "type": "online",
  "license_id": "61063ce4b86945e1b70c3bdbedea453b",
  "generated": 1514467939,
  "last_updated": 5,
  "valid": {
    "value": true,
    "description": ""
  },
  "source": "/opt/ntech/license/import_
↪b68d7b7ec9a7310d18832035318cff0c9ddf11e3a9ab0ae962fbe48645e196d1.lic",
  "limits": [
    {
      "type": "time",
      "name": "end",
      "value": 1609161621
    },
    {
      "type": "number",
      "name": "faces",
      "value": 9007199254740991,
      "current": 0
    },
    {
      "type": "number",
      "name": "cameras",
      "value": 4294967295,
      "current": 0
    },
    {
      "type": "number",
      "name": "extraction_api",
      "value": 256,
      "current": 0
    },
    {
      "type": "boolean",
      "name": "gender",
      "value": true
    },
    {
      "type": "boolean",
      "name": "age",
      "value": true
    },
    {
      "type": "boolean",
      "name": "emotions",
      "value": true
    },
    {
      "type": "boolean",
      "name": "fast-index",
```

(continues on next page)

(продолжение с предыдущей страницы)


```
      "value": true
    },
    {
      "type": "boolean",
      "name": "sec-genetec",
      "value": false
    },
    {
      "type": "boolean",
      "name": "countries",
      "value": false
    },
    {
      "type": "boolean",
      "name": "beard",
      "value": false
    },
    {
      "type": "boolean",
      "name": "race",
      "value": false
    },
    {
      "type": "boolean",
      "name": "glasses",
      "value": false
    },
    {
      "type": "boolean",
      "name": "liveness",
      "value": false
    }
  ],
  "services": [
    {
      "name": "video-worker",
      "ip": "127.0.0.1:53276"
    },
    {
      "name": "FindFace-tarantool",
      "ip": "127.0.0.1:53284"
    },
    {
      "name": "FindFace-tarantool",
      "ip": "127.0.0.1:53288"
    }
  ]
}
```


1.8.9 Автоматическое восстановление Tarantool

Если архитектура вашей системы не обеспечивает бесперебойную доступность серверов Tarantool, рекомендуется включить автоматическое восстановление базы данных. В этом случае каждый раз при возникновении ошибки во время чтения файла `.snap` или `.xlog`, Tarantool попытается прочитать как можно больше информации и восстановить файл, игнорируя битые записи.

Для включения автоматического восстановления базы данных выполните следующие действия:

1. Откройте файл конфигурации Tarantool.

```
sudo vi /etc/tarantool/instances.enabled/FindFace.lua
```

2. Раскомментируйте `force_recovery = true`.

```
box.cfg{

    force_recovery = true,
}
```

1.8.10 Ручная очистка базы данных от старых событий и эпизодов

Совет: Для настройки автоматического удаления событий и эпизодов см. *Автоматическое удаление старых событий и эпизодов*.

Для ручного удаления старых событий и эпизодов из базы данных FindFace Security используйте утилиту `cleanup_events`. Вы можете выбрать удаление только событий, для которых отсутствуют совпадения с досье, или только событий с совпадениями.

Справка по утилите `cleanup_events` вызывается следующей командой:

```
findface-security cleanup_events --help
usage: findface-security cleanup_events [-h] [--matched-age MATCHED_AGE]
                                        [--unmatched-age UNMATCHED_AGE]
                                        [--version] [-v {0,1,2,3}]
                                        [--settings SETTINGS]
                                        [--pythonpath PYTHONPATH]
                                        [--traceback] [--no-color]

Delete old events

optional arguments:
  -h, --help               show this help message and exit
  --matched-age MATCHED_AGE
                        Minimum age in days of matched events to clean up
  --unmatched-age UNMATCHED_AGE
                        Minimum age in days of unmatched events to clean up
  --version             show program's version number and exit
  -v {0,1,2,3}, --verbosity {0,1,2,3}
                        Verbosity level; 0=minimal output, 1=normal output,
                        2=verbose output, 3=very verbose output
  --settings SETTINGS   The Python path to a settings module, e.g.
                        "myproject.settings.main". If this isn't provided, the
                        DJANGO_SETTINGS_MODULE environment variable will be
                        used.
  --pythonpath PYTHONPATH
                        A directory to add to the Python path, e.g.
                        "/home/djangoprojects/myproject".
  --traceback           Raise on CommandError exceptions
  --no-color            Don't colorize the command output.
```

Для удаления событий и эпизодов старше определенного количества дней используйте опцию `-age`. Например, для удаления событий без совпадений старше 5 дней выполните команду:

```
sudo findface-security cleanup_events --unmatched-age 5
```

Для удаления событий с совпадениями старше 5 дней выполните команду:

```
sudo findface-security cleanup_events --matched-age 5
```

Важно: Должен быть задан хотя бы один из аргументов `--matched-age`/`--unmatched-age`.

1.9 Приложения

1.9.1 Настройка шифрования данных

Для обеспечения безопасности данных включите SSL-шифрование. Выполните следующие действия:

Важно: Использование самоподписанного сертификата не рекомендуется.

1. В директории с конфигурацией nginx создайте каталог для хранения информации о SSL-шифровании:

   ```
   sudo mkdir /etc/nginx/ssl
   ```

2. Создайте ключ и сертификат SSL:

   ```
   sudo openssl req -x509 -nodes -days 365 -newkey rsa:2048 -keyout /etc/nginx/ssl/my-example-
   ↪domain.com.key -out /etc/nginx/ssl/my-example-domain.com.crt
   ```

   Для заполнения полей сертификата вам будет предложено несколько вопросов. Ответьте на них, уделив особое внимание строке `Common Name`. В ней нужно ввести имя или публичный IP-адрес домена, связанного с сервером. Созданные файлы ключа `my-example-domain.com.key` и сертификата `my-example-domain.com.crt` будут сохранены в каталоге `/etc/nginx/ssl`.

3. Configure nginx to use SSL. Open the nginx configuration file `/etc/nginx/sites-available/ffsecurity-nginx.conf`. Copy the code from the example below into the file.

   ```
   sudo vi /etc/nginx/sites-available/ffsecurity-nginx.conf

   upstream ffsecurity {
           server 127.0.0.1:8002;
   }
   upstream ffsecurity-ws {
           server 127.0.0.1:8003;
   }
   map $http_upgrade $ffsec_upstream {
           default "http://ffsecurity-ws";
           "" "http://ffsecurity";
   }
   server {
           listen 80;
           server_name 172.20.77.10;
           rewrite ^(.*) https://172.20.77.10$1 permanent;
           access_log off;
   ```

(continues on next page)

(продолжение с предыдущей страницы)

```
}
server {
        #listen 80 default_server;
        #listen [::]:80 default_server;
        listen 443 ssl;
        ssl_certificate     /etc/nginx/ssl/my-example-domain.com.crt;
        ssl_certificate_key /etc/nginx/ssl/my-example-domain.com.key;
        root /var/lib/ffsecurity;
        autoindex off;
        server_name _;
        location = / {
                alias /usr/share/ffsecurity-ui/;
                try_files /index.html =404;
        }
        location /static/ {
        }
        location /uploads/ {
                add_header 'Access-Control-Allow-Origin' '*';
                add_header 'Access-Control-Allow-Methods' 'GET';
                add_header 'Access-Control-Allow-Headers' 'DNT,User-Agent,X-Requested-With,If-
↪Modified-Since,Cache-Control,Content-Type,Range,Authorization';
                add_header 'Access-Control-Expose-Headers' 'Content-Length,Content-Range';
                add_header 'Access-Control-Max-Age' 2592000;
        }
        location /ui-static/ {
                alias /usr/share/ffsecurity-ui/ui-static/;
        }
        location /doc/ {
                alias /opt/ffsecurity/doc/;
        }
        location ~ /videos/(?<video_id>[0-9]+)/upload/(.*)$ {
                if ($request_method = 'OPTIONS') {
                        add_header 'Content-Type' 'text/plain; charset=utf-8';
                        add_header 'Content-Length' 0;
                        return 204;
                }
                set $auth_request_uri "http://ffsecurity/videos/$video_id/auth-upload/";
                auth_request /video-upload-auth/;
                alias "/var/lib/ffsecurity/uploads/videos/$video_id.bin";
                client_max_body_size 15g;
                dav_access user:rw group:rw all:rw;
                dav_methods PUT;
                create_full_put_path on;
                autoindex off;
                autoindex_exact_size off;
                autoindex_localtime on;
                charset utf-8;
                add_header 'Access-Control-Allow-Origin' '*';
                add_header 'Access-Control-Allow-Methods' 'PUT, OPTIONS';
                add_header 'Access-Control-Allow-Headers' 'authorization';
        }
        location = /video-upload-auth/ {
                internal;
                client_max_body_size 15g;
                proxy_set_header Content-Length "";
                proxy_set_header Host $http_host;
```

(continues on next page)

(продолжение с предыдущей страницы)

```
                proxy_set_header X-Forwarded-For $remote_addr;
                proxy_set_header X-Forwarded-Proto $scheme;
                proxy_pass_request_body off;
                proxy_pass $auth_request_uri;
        }
        location / {
                client_max_body_size 300m;
                proxy_set_header Host $http_host;
                proxy_set_header X-Forwarded-For $remote_addr;
                proxy_set_header X-Forwarded-Proto $scheme;
                proxy_http_version 1.1;
                proxy_set_header Upgrade $http_upgrade;
                proxy_set_header Connection "upgrade";
                proxy_pass $ffsec_upstream;
                location ~ ^/(cameras|videos)/([0-9]+)/stream/?$ {
                        proxy_set_header Host $http_host;
                        proxy_set_header X-Forwarded-For $remote_addr;
                        proxy_set_header X-Forwarded-Proto $scheme;
                        proxy_pass http://ffsecurity;
                }
                location ~ ^/streams/(.*)$ {
                        internal;
                        proxy_pass $1;
                }
        }
}
```

4. Перезапустите nginx.

```
sudo service nginx restart
```

5. Внесите изменения в файл конфигурации `findface-security`. В параметрах `EXTERNAL_ADDRESS` и `ROUTER_URL` измените приставку `http://` на `https://`.

```
sudo vi /etc/ffsecurity/config.py

...
EXTERNAL_ADDRESS="https://my-example-domain.com"
...
ROUTER_URL="https://IP_address"
```

6. Если есть запущенные процессы `findface-video-worker`, нужно либо пересоздать камеры в веб-интерфейсе, либо изменить значение параметра `router_url` в job-заданиях, заменив приставку `http://` на `https://`. Это можно сделать с помощью команды, аналогичной следующей:

```
curl -s localhost:18810/jobs | jq -r '.[]["id"]' | xargs -I {} curl -X PATCH -d '{"router_url
↪": "https://domain.ru/video-detector/frame"}' http://localhost:18810/job/{}
```

1.9.2 Подробно о компонентах

findface-extraction-api

Компонент `findface-extraction-api` с помощью нейронных сетей обнаруживает лицо на изображении, извлекает из лица биометрический образец, а также распознает пол, возраст, эмоции и другие атрибуты лица.

Компонент взаимодействует с сервисом `findface-sf-api` следующим образом:

- Получает от него фотографию с лицом или нормализованное изображение лица.
- Возвращает координаты рамки с лицом, а также вектор признаков, данные о поле, возрасте, эмоциях и других атрибутах лица (если они были запрошены `findface-sf-api`).

Полный список функций:

- детекция (обнаружение) лица на исходном изображении с возвращением координат рамки с лицом,
- получение из исходного изображения нормализованного изображения лица,
- извлечение из нормализованного изображения лица вектора признаков (биометрического образца),
- распознавание атрибутов лица (пол, возраст, эмоции, борода, очки и др.).

Сервис `findface-extraction-api` может работать с ускорением на CPU (устанавливается из пакета `findface-extraction-api`) или GPU (устанавливается из пакета `findface-extraction-api-gpu`). Как для CPU-, так и для GPU-сервиса, настройка выполняется через файл конфигурации `/etc/findface-extraction-api.ini`, однако содержимое данного файла на CPU и GPU отличается.

Файл конфигурации сервиса на CPU:

```
detectors:
  max_batch_size: 1
  instances: 1
  models:
    cheetah:
      model: facedet/cheetah.cpu.fnk
      options:
        min_object_size: 32
        resolutions: [256x256, 384x384, 512x512, 768x768, 1024x1024, 1536x1536, 2048x2048]
  quality_estimator: true
normalizers:
  max_batch_size: 8
  instances: 1
  models:
    bee:
      model: facenorm/bee.v2.cpu.fnk
    crop1x:
      model: ""
    crop2x:
      model: ""
extractors:
  max_batch_size: 8
  instances: 1
  models:
    age: faceattr/age.v1.cpu.fnk
    beard: ""
    emotions: faceattr/emotions.v1.cpu.fnk
```

(continues on next page)

(продолжение с предыдущей страницы)

```
    face: face/grapefruit_480.cpu.fnk
    gender: faceattr/gender.v2.cpu.fnk
    glasses3: ""
    liveness: ""
    quality: faceattr/quality.v0.cpu.fnk
gpu_device: 0
models_root: /usr/share/findface-data/models
cache_dir: /var/cache/findface/models_cache
listen: :18666
license_ntls_server: 127.0.0.1:3133
fetch:
  enabled: true
  size_limit: 10485760
max_dimension: 6000
allow_cors: false
ticker_interval: 5000
prometheus:
  timing_buckets: [0.001, 0.005, 0.01, 0.02, 0.03, 0.05, 0.1, 0.2, 0.3, 0.5, 0.75,
    0.9, 1, 1.1, 1.3, 1.5, 1.7, 2, 3, 5, 10, 20, 30, 50]
  resolution_buckets: [10000, 20000, 40000, 80000, 100000, 200000, 400000, 800000,
    1e+06, 2e+06, 3e+06, 4e+06, 5e+06, 6e+06, 8e+06, 1e+07, 1.2e+07, 1.5e+07, 1.8e+07,
    2e+07, 3e+07, 5e+07, 1e+08]
  faces_buckets: [0, 1, 2, 5, 10, 20, 50, 75, 100, 200, 300, 400, 500, 600, 700, 800,
    900, 1000]
```

Файл конфигурации сервиса на GPU:

```
detectors:
  max_batch_size: 16
  gpu_device: 0
  instances: 1
  models:
    cheetah:
      model: facedet/cheetah.gpu.fnk
      options:
        min_object_size: 32
        resolutions:
        - 256x256
        - 384x384
        - 512x512
        - 768x768
        - 1024x1024
        - 1536x1536
        - 2048x2048
  quality_estimator: true
normalizers:
  max_batch_size: 16
  gpu_device: 0
  instances: 1
  models:
    bee:
      model: facenorm/bee.v2.gpu.fnk
    crop1x:
      model: ''
    crop2x:
```

(continues on next page)

(продолжение с предыдущей страницы)

```
      model: ''
extractors:
  max_batch_size: 16
  gpu_device: 0
  instances: 1
  models:
    age: ''
    beard: ''
    emotions: ''
    face: face/grapefruit_480.gpu.fnk
    gender: ''
    glasses3: ''
    liveness: ''
    quality: faceattr/quality.v0.gpu.fnk
models_root: /usr/share/findface-data/models
cache_dir: /var/cache/findface/models_cache
listen: 127.0.0.1:18666
license_ntls_server: 127.0.0.1:3133
fetch:
  enabled: true
  size_limit: 10485760
max_dimension: 6000
allow_cors: false
ticker_interval: 5000
prometheus:
  timing_buckets:
  - 0.001
  - 0.005
  - 0.01
  - 0.02
  - 0.03
  - 0.05
  - 0.1
  - 0.2
  - 0.3
  - 0.5
  - 0.75
  - 0.9
  - 1
  - 1.1
  - 1.3
  - 1.5
  - 1.7
  - 2
  - 3
  - 5
  - 10
  - 20
  - 30
  - 50
  resolution_buckets:
  - 10000
  - 20000
  - 40000
  - 80000
  - 100000
```

(continues on next page)

(продолжение с предыдущей страницы)

```
 - 200000
 - 400000
 - 800000
 - 1e+06
 - 2e+06
 - 3e+06
 - 4e+06
 - 5e+06
 - 6e+06
 - 8e+06
 - 1e+07
 - 12000000.0
 - 15000000.0
 - 18000000.0
 - 2e+07
 - 3e+07
 - 5e+07
 - 1e+08
faces_buckets:
 - 0
 - 1
 - 2
 - 5
 - 10
 - 20
 - 50
 - 75
 - 100
 - 200
 - 300
 - 400
 - 500
 - 600
 - 700
 - 800
 - 900
 - 1000
```

Пользовательская настройка `findface-extraction-api` (как CPU, так и GPU) выполняется с использованием следующих параметров:

Параметр	Описание
`detectors -> quality_estimator`	Включает/отключает оценку качества лица. В этом случае компонент `findface-extraction-api` будет возвращать показатель качества лица в параметре `quality_estimator_score`. Показатель качества можно анализировать. Прямые изображения лиц анфас считаются наиболее качественными. Для них возвращаются значения вблизи 0, как правило, отрицательные (такие как -0.00067401276, например). Перевернутые лица и лица, повернутые под большими углами, оцениваются отрицательными значениями от -5 и меньше.
`cheetah -> min_object_size`	Минимальный размер лица, которое будет гарантированно найдено. Определяется размером рамки с лицом (bbox). Чем больше значение, тем менее ресурсоемок процесс обнаружения лица.
`license_manager`	Порт сервера лицензирования `findface-ntls`.
`gpu_device`	(Только для GPU) Номер GPU-устройства, используемого `findface-extraction-api-gpu`.

В зависимости от нужд вашего бизнеса, вам также может потребоваться включить модели распознавания атрибутов лица, таких как пол, возраст, эмоции, очки и/или борода. Удостоверьтесь, что для каждой модели вы указали правильный тип ускорения CPU или GPU: он должен совпадать с типом ускорения `findface-extraction-api`. Обратите внимание, что `findface-extraction-api` на CPU может работать только с CPU-моделями, в то время как `findface-extraction-api` на GPU поддерживает как GPU-, так и CPU-модели.

```
models:
  age: faceattr/age.v1.cpu.fnk
  emotions: faceattr/emotions.v1.cpu.fnk
  face: face/grapefruit_480.cpu.fnk
  gender: faceattr/gender.v2.cpu.fnk
  beard: faceattr/beard.v0.cpu.fnk
  glasses3: faceattr/glasses3.v0.cpu.fnk
```

Доступны следующие модели:

Атрибут лица	Ускорение	Параметр в файле конфигурации
биометрия лица	CPU	`face: face/grapefruit_480.cpu.fnk` `face: face/grapefruit_160.cpu.fnk`
	GPU	`face: face/grapefruit_480.gpu.fnk` `face: face/grapefruit_160.gpu.fnk`
возраст	CPU	`age: faceattr/age.v1.cpu.fnk`
	GPU	`age: faceattr/age.v1.gpu.fnk`
пол	CPU	`gender: faceattr/gender.v2.cpu.fnk`
	GPU	`gender: faceattr/gender.v2.gpu.fnk`
эмоции	CPU	`emotions: faceattr/emotions.v1.cpu.fnk`
	GPU	`emotions: faceattr/emotions.v1.gpu.fnk`
очки	CPU	`glasses3: faceattr/glasses3.v0.cpu.fnk`
	GPU	`glasses3: faceattr/glasses3.v0.gpu.fnk`
борода	CPU	`beard: faceattr/beard.v0.cpu.fnk`
	GPU	`beard: faceattr/beard.v0.gpu.fnk`

Совет: Для того чтобы отключить модель распознавания, передайте в соответствующий параметр пустое значение. Не удаляйте сам параметр, поскольку в этом случае будет выполняться поиск модели по умолчанию.

```
models:
  gender: ""
  age: ""
  emotions: ""
```

findface-sf-api

Компонент `findface-sf-api` представляет собой сервис, реализующий HTTP API основного функционала ядра FindFace (обнаружение и распознавание лиц, при этом сами операции обнаружения и распознавания лиц выполняются компонентом `findface-extracton-api`). Взаимодействует с базой биометрических данных Tarantool через компонент `findface-tarantool-server`, а также с компонентами `findface-extraction-api` (обнаружение и распознавание лиц) и `findface-upload` (хранилище исходных изображений и артефактов работы ядра FindFace).

Для обнаружения лица на фотографии в компонент `findface-sf-api` должен быть отправлен API-запрос, передающий данную фотографию в виде файла или URL. Данный запрос затем перенаправляется в компонент `findface-extraction-api`.

При наличии в системе видеодетектора лиц (присутствует в FindFace Security) компонент `findface-sf-api` получает данные об обнаруженных лицах вместе с правилами их обработки от компонента `findface-facerouter` и затем выполняет полученные директивы (например, сохраняет лица в определенную галерею базы данных).

Примечание: В FindFace Security, функции `findface-facerouter` выполняет сервис `findface-security`.

Полный список функций:

- реализация HTTP API по части обнаружения и распознавания лиц (операции выполняются `findface-extraction-api`).
- сохранение лиц в базу биометрических данных (через сервис `findface-tarantool-server`),
- сохранение исходных изображений, миниатюр и нормализованных изображений лиц на веб-сервере nginx (через сервис `findface-upload`).
- обеспечение взаимодействия всех компонентов системы.

Настройка компонента `findface-sf-api` выполняется через файл конфигурации `/etc/findface-sf-api.ini`.

```
listen: 127.0.0.1:18411
extraction-api:
  timeouts:
    connect: 5s
    response_header: 30s
    overall: 35s
    idle_connection: 10s
  extraction-api: http://127.0.0.1:18666
storage-api:
  timeouts:
    connect: 5s
    response_header: 30s
    overall: 35s
    idle_connection: 10s
  max-idle-conns-per-host: 20
  shards:
  - master: http://127.0.0.1:8101/v2/
    slave: ''
  - master: http://127.0.0.1:8102/v2/
    slave: ''
  - master: http://127.0.0.1:8103/v2/
    slave: ''
  - master: http://127.0.0.1:8104/v2/
    slave: ''
limits:
  url-length: 4096
  deny-networks: 127.0.0.0/8,192.168.0.0/16,10.0.0.0/8,::1/128,fe00::/8
  body-image-length: 33554432
  allow-return-facen: false
cache:
  type: memcache
```

(continues on next page)

(продолжение с предыдущей страницы)

```
  inmemory:
    size: 16384
  memcache:
    nodes:
    - 127.0.0.1:11211
    timeout: 100ms
  redis:
    network: tcp
    addr: localhost:6379
    password: ''
    db: 0
    timeout: 5s
normalized-storage:
  type: webdav
  enabled: true
  webdav:
    upload-url: http://127.0.0.1:3333/uploads/
    timeouts:
      connect: 5s
      response_header: 30s
      overall: 35s
      idle_connection: 10s
  s3:
    endpoint: ''
    bucket-name: ''
    access-key: ''
    secret-access-key: ''
    secure: true
    region: ''
    public-url: ''
    operation-timeout: 30
```

Пользовательская настройка `findface-sf-api` выполняется с использованием следующих параметров:

Параметр	Описание
`extraction-api -> extraction-api`	IP-адрес и порт сервера `findface-extraction-api`.
`storage-api -> shards -> master`	IP-адрес физического сервера с мастером шарда `findface-tarantool-server`.
`storage-api -> shards -> slave`	IP-адрес физического сервера с репликой шарда `findface-tarantool-server`.
`limits -> body-image-length`	Максимальный размер в байтах изображения, передаваемого через API-запрос.
`normalized-storage -> webdav -> upload_url`	Путь в WebDAV nginx, по которому в компонент `findface-upload` будут отправляться исходные изображения, миниатюры и нормализованные изображения лиц.

findface-tarantool-server

Сервис `findface-tarantool-server` обеспечивает взаимодействие между сервисом `findface-sf-api` и биометрической базой данных Tarantool следующим образом:

Совет: Подробнее см. официальную документацию Tarantool.

- `findface-tarantool-server` получает от `findface-sf-api` данные для записи в базу данных (например, об обнаруженных лицах).
- По запросу от `findface-sf-api` `findface-tarantool-server` выполняет поиск по базе данных и возвращает его результат.

Для увеличения скорости поиска на каждом сервере с базой данных Tarantool могут быть созданы многочисленные сегменты («шарды») `findface-tarantool-server`. Их параллельное функционирование приводит к значительному увеличению производительности (в 70-100 раз).

Полный список функций:

- сохранение лиц в базу биометрических данных,
- выполнение поиска по базе биометрических данных,
- реализация прямых запросов в базу данных Tarantool (см. *Прямые API-запросы к базе данных Tarantool*).

Настройка компонента `findface-tarantool-server` выполняется через файл конфигурации `/etc/tarantool/instances.enabled/<shard-*>.lua`. В кластерной среде файл конфигурации настраивается отдельно для каждого шарда.

```
-
-- Please, read the tarantool cfg doc:
-- https://tarantool.org/doc/reference/configuration/index.html#box-cfg-params
--
box.cfg{
    --port to listen, direct tarantool access
    --Only need for admin operations
    --THIS IS NOT PORT YOU NEED FOR facenapi/sf-api
    listen = '127.0.0.1:32001',
    --Directory to store data
    vinyl_dir = '/opt/ntech/var/lib/tarantool/shard-001',
    work_dir = '/opt/ntech/var/lib/tarantool/shard-001',
    memtx_dir = '/opt/ntech/var/lib/tarantool/shard-001/snapshots',
    wal_dir = '/opt/ntech/var/lib/tarantool/shard-001/xlogs',
    --Maximum mem usage in bytes
    memtx_memory = 200 * 1024 * 1024,
    checkpoint_interval = 3600*4,
    checkpoint_count = 3,
    --uncomment only if you know what you are doing!!! and don't forget box.snapshot()
    -- wal_mode = 'none',
    --if true, tarantool tries to continue if there is an error while reading a snapshot/xlog␣
↪files:  skips invalid records, reads as much data as possible and re-builds the file
    -- force_recovery = true,
}
pcall(function() box.schema.user.grant('guest', 'execute,read,write', 'universe') end)
dofile("/etc/ffsecurity/tnt_schema.lua")
-- host,port to bind for http server
-- this is what you need for facenapi
FindFace = require("FindFace")
FindFace.start("127.0.0.1", 8101, {
    license_ntls_server="127.0.0.1:3133",
    facen_size=480,
    meta_indexes=meta_indexes,
```

(continues on next page)

(продолжение с предыдущей страницы)

```
    meta_scheme = meta_scheme
})
```

Пользовательская настройка `findface-tarantool-server` выполняется с использованием следующих параметров:

Параметр	Описание
`memtx_memory`	Максимальный размер оперативной памяти в байтах, который может быть использован шардом Tarantool. Перед изменением данного параметра обратитесь к нашим экспертам за консультацией по адресу support@ntechlab.com.
`force_recovery`	Включает/отключает автоматическое восстановление базы данных Tarantool. Если автоматическое восстановление данных включено (true), каждый раз при возникновении ошибки во время чтения файла .snap или .xlog, Tarantool попытается прочитать как можно больше информации и восстановить файл, игнорируя битые записи.
`license_address`	IP-адрес сервера лицензирования `findface-ntls`.
`facen_size`	Размер вектора признаков на выходе из нейронной сети для распознавания лиц. Не редактируйте данное значение, не посоветовавшись с нашими специалистами.
`meta_scheme`	Структура базы данных для хранения биометрических параметров. Представляет собой набор полей, для каждого из которых указываются следующие параметры: `id`: id поля, `name`: название поля, должно совпадать с названием соответствующего параметра лица, `field_type`: тип данных, `default`: значение по умолчанию. Если значение по умолчанию для поля больше '1e14 – 1', то его следует записывать в виде строки, т. е. "123123" вместо 123123.

Структура базы данных передается из файла `/etc/ffsecurity/tnt_schema.lua` в параметр `meta_scheme`.

findface-upload

Компонент `findface-upload` представляет собой веб-сервер на базе WebDAV nginx, который используется для хранения исходных изображений, миниатюр и нормализованных изображений лиц (получает их от компонента `findface-sf-api`).

По умолчанию исходные изображения, миниатюры и нормализованные изображения лиц хранятся в каталоге `/var/lib/ffupload/uploads/`.

Компонент findface-upload автоматически настраивается при установке. Настройка не предусмотрена.

Видеодетекция лиц: `findface-video-manager` и `findface-video-worker`

Примечание: Компонент `findface-video-worker` поставляется в пакетах с ускорением на CPU (`findface-video-worker-cpu`) и GPU (`findface-video-worker-gpu`).

В этом разделе:

Функции findface-video-manager

Сервис `findface-video-manager` является частью модуля видеодетекции лиц и используется для непосредственного управления детекцией лиц на видео.

Сервис `findface-video-manager` взаимодействует с `findface-video-worker` следующим образом:

- Обеспечивает `findface-video-worker` настройками и списком видеопотоков для обработки. Для этого он выдает `findface-video-worker` так называемое *job-задание*, задачу на обработку видео, которая содержит параметры конфигурации и сведения о видеопотоке.
- В распределенной системе распределяет видеопотоки (job-задания) по свободным экземплярам `findface-video-worker`.

Примечание: Параметры конфигурации, передаваемые через job-задания, имеют больший приоритет, чем аналогичные параметры в файле конфигурации `findface-video-manager`.

Для работы `findface-video-manager` требуется установленный сервис ETCD. ETCD представляет собой стороннее программное обеспечение, реализующее распределенное хранилище ключей `findface-video-manager`. Используется в качестве координационной службы в распределенной системе, обеспечивая отказоустойчивость работы видеодетектора лиц.

Полный список функций `findface-video-manager`:

- конфигурирование параметров видеодетектора лиц,
- управление списком видеопотоков для обработки,
- управление видеодетекцией лиц.

Функции findface-video-worker

Компонент `findface-video-worker` (или `findface-video-worker-gpu`) является частью модуля видеодетекции лиц и служит для обнаружения лиц «на лету» в видеопотоке или видеофайле. Он поддерживает большинство видеоформатов и кодеков, которые могут быть декодированы FFmpeg.

Сервис `findface-video-worker` взаимодействует с сервисами `findface-video-manager` и `findface-facerouter` следующим образом:

- По запросу `findface-video-worker` получает от `findface-video-manager` job-задание с настройками и списком видеопотоков для обработки.
- Сервис `findface-video-worker` отправляет полученные нормализованные изображения лиц вместе с полными кадрами и метаданными, такими как рамка вокруг лица, ID камеры и время детекции, в сервис `findface-facerouter` для дальнейшей обработки.

Примечание: В FindFace Security функции `findface-facerouter` выполняются сервисом `findface-security`.

Полный список функций `findface-video-manager`:

- обнаружение лиц на видео,
- извлечение нормализованных изображений лиц,
- поиск наилучшего изображения лица,
- дедупликация кадров с лицом (только один кадр на каждое событие распознавания лица).

При обработки видео `findface-video-worker` последовательно использует следующие алгоритмы:

- `Детектор движения`. Данный алгоритм позволяет снизить потребление ресурсов, поскольку детектор лиц включается только по движению в кадре.
- `Детектор лиц`. Алгоритм детектирует, отслеживает и захватывает лица на видео. Может работать одновременно с несколькими лицами в кадре. С помощью встроенной нейронной сети выполняет поиск кадра с лучшим изображением лица. Как только лучшее изображение найдено, отправляет его в компонент `findface-facerouter`.

Подбор лучшего изображения лица может быть выполнен в одном из следующих режимов:

- Режим реального времени
- Буферный режим

Режим реального времени

В режиме реального времени `findface-video-worker` начинает отправлять в компонент `findface-facerouter` изображения лица сразу после появления лица в поле зрения видеокамеры.

- Если параметр `rt-perm=True`, детектор лиц выбирает лучший кадр в течение каждого из последовательных промежутков времени, равных `rt-delay`, и отправляет его в `findface-facerouter`.
- Если `rt-perm=False`, детектор лиц выбирает лучшее изображение лица динамически:
  1. Сначала оценивается качество изображения лица. Если оно превышает некое предустановленное пороговое значение, то лицо отправляется в `findface-facerouter`.
  2. Порог повышается после каждой отправки изображения лица в `findface-facerouter`. Каждый раз, когда детектор лиц получает изображение того же лица лучшего качества, оно отправляется.
  3. При исчезновении лица из поля зрения видеокамеры снова устанавливается пороговое значение по умолчанию.

По умолчанию режим реального времени отключен (параметр `realtime=false` в файле конфигурации `/etc/findface-video-manager.conf`).

Буферный режим

Буферный режим требует меньший объем дискового пространства по сравнению с режимом реального времени, поскольку для каждого лица компонент `findface-video-worker` отправляет только одно изображение из трека, но наивысшего качества.

Буферный режим включен по умолчанию (параметр `overall=true` в файле конфигурации `/etc/findface-video-manager.conf`).

Настройка видеодетекции лиц

Настройка видеодетектора лиц выполняется через следующие файлы конфигурации:

1. Файл конфигурации компонента `findface-video-manager` `/etc/findface-video-manager.conf`:

```
listen: :18810
etcd:
  endpoints: 127.0.0.1:2379
  dial_timeout: 3s
kafka:
  enabled: false
  endpoints: 127.0.0.1:9092
master:
  lease_ttl: 10
  self_url: 127.0.0.1:18811
  self_url_http: 127.0.0.1:18810
rpc:
  listen: 127.0.0.1:18811
  heart_beat_timeout: 4s
router_url: http://127.0.0.1:18820/v0/frame
exp_backoff:
  enabled: false
  min_delay: 1s
  max_delay: 1m0s
  factor: 2
  flush_interval: 2m0s
ntls:
  enabled: false
  url: http://127.0.0.1:3185/
  update_interval: 1m0s
prometheus:
  jobs_processed_duration_buckets:
  - 1
  - 30
  - 60
  - 500
  - 1800
  - 3600
  - 21600
  - .inf
job_scheduler_script: ""
stream_settings:
  ffmpeg_params: []
  md_threshold: 0.002
  md_scale: 0.3
  fd_frame_height: -1
  uc_max_time_diff: 30
  uc_max_dup: 3
  uc_max_avg_shift: 10
  det_period: 8
  realtime: false
  npersons: 4
  disable_drops: false
```

(continues on next page)

(продолжение с предыдущей страницы)

```
  tracker_threads: 4
  parse_sei: false
  image_arg: photo
  additional_headers: []
  additional_body: []
  api_timeout: 15000
  api_ssl_verify: true
  post_uniq: true
  min_score: -2
  min_d_score: -1000
  realtime_dly: 500
  realtime_post_perm: false
  rot: ""
  roi: ""
  draw_track: false
  send_track: 0
  min_face_size: 0
  max_face_size: 0
  overall: true
  only_norm: false
  max_candidates: 0
  jpeg_quality: 95
  ffmpeg_format: ""
stream_settings_gpu:
  play_speed: -1
  filter_min_quality: -2
  filter_min_face_size: 1
  filter_max_face_size: 8192
  normalized_only: false
  jpeg_quality: 95
  overall_only: false
  use_stream_timestamp: false
  ffmpeg_params: []
  router_timeout_ms: 15000
  router_verify_ssl: true
  router_headers: []
  router_body: []
  start_stream_timestamp: 0
  imotion_threshold: 0
  rot: ""
  roi: ""
  realtime_post_interval: 1
  realtime_post_every_interval: false
  ffmpeg_format: ""
  disable_drops: true
  router_full_frame_png: false
  router_disable_normalized: false
```

Пользовательская настройка `findface-video-manager` выполняется с использованием следующих параметров:

Опция	Описание
`router_url`	IP-адрес и порт сервера `findface-facerouter`, который получает обнаруженные лица из `findface-video-worker`. В FindFace Security функции `findface-facerouter` выполняет компонент `findface-security`. Значение по умолчанию: `http://127.0.0.1:18820/v0/frame`.
`etcd -> endpoints`	IP-адрес и порт сервиса `etcd`. Значение по умолчанию: `127.0.0.1:2379`.
`ntls -> enabled`	Если `true`, компонент `findface-video-manager` отправляет в компонент `findface-video-worker` задания только на обработку того количества видеокамер, которое указано в лицензии. Значение по умолчанию: `false`.
`ntls -> url`	IP-адрес и порт сервера `findface-ntls`. Значение по умолчанию: `http://127.0.0.1:3185/`.

Вы также можете использовать следующие параметры:

Примечание: В разделе файла `stream_settings-gpu` вы найдете настройки, общие для всех видеопотоков. Настройки в данном разделе работают как в GPU, так и CPU-конфигурации. Настройки определенного потока, переданные в job-задании, имеют приоритет над настройками в файле конфигурации (см. *Job-задания*).

Примечание: Раздел `stream_settings` файла устарел и необходим только для обратной совместимости.

Опция	Описание
play_speed	Если меньше нуля, то скорость не ограничивается. В остальных случаях поток читается со скоростью play_speed. Не применимо для потоков с камер видеонаблюдения.
filter_min_quality	Минимальное значение качества изображения лица, отправляемого компонентом findface-video-worker в компонент findface_facerouter (findface-security в стандартной конфигурации FindFace Security). Значение определяется эмпирически: отрицательные рациональные числа до 0. Реперные точки: 0 = наиболее качественные лица, -1 = хорошее качество, -2 = удовлетворительное качество, -5 = последующее распознавание лиц может быть неэффективным. Значение по умолчанию: -2.
filter_min_face_size	Определяет минимальный размер лица в пикселях. Лица меньшего размера не отправляются. Значение по умолчанию: 0 (фильтр выключен).
filter_max_face_size	Определяет максимальный размер лица в пикселях. Лица большего размера не отправляются. Значение по умолчанию: 0 (фильтр выключен).
normalized_only	Включает/отключает отправку только нормализованных лиц без исходных кадров. Значение по умолчанию: false.
jpeg_quality	Качество сжатия исходного кадра в JPEG. Значение по умолчанию: 95 % от исходного размера.
overall_only	Буферный режим. Отправлять для лица один кадр наилучшего качества. Значение по умолчанию: true.
use_stream_timestamp	Если true, отправлять на сервер временные метки полученные из потока. Если false, отправлять текущие дату и время.
ffmpeg_params	Список ffmpeg-параметров видеопотока со значениями в виде массива ключ=значение: [“rtsp_transpotr=tcp”, .., “ss=00:20:00”]. Полный список параметров на сайте FFMPEG. Значение по умолчанию: параметры не указаны.
router_timeout_ms	Время ожидания в миллисекундах ответа от компонента findface-facerouter (findface-security в стандартной конфигурации FindFace Security) на API-запрос компонента findface-video-worker. Если время ожидания истекло, регистрируется ошибка. Значение по умолчанию: 15000.
router_verify_ssl	Включает/отключает проверку подписи SSL-сертификата при взаимодействии по https findface-video-worker с компонентом findface-facerouter (findface-security в стандартной конфигурации FindFace Security). Значение по умолчанию: true. Если false, может быть принят самоподписанный сертификат.
router_headers	Массив дополнительных заголовков в POST-запросе с изображением лица в формате [“ключ=значение”]. По умолчанию дополнительные заголовки не передаются.
router_body	Массив дополнительных полей в POST-запросе с изображением лица в формате [“ключ=значение”]. По умолчанию дополнительные поля не передаются.
start_stream_timestamp	Прибавить указанное количество секунд к временным меткам из потока.
imotion_threshold	Минимальная интенсивность движения, которая будет регистрироваться детектором движения. Пороговое значение определяется эмпирически. Реперные точки: 0 = детектор выключен, 0.002 = значение по умолчанию, 0.05 = минимальная интенсивность слишком высока, чтобы зарегистрировать движение.
rot	Включает детектирование и отслеживание лиц только внутри заданной прямоугольной области WxH+X+Y. Используйте данную опцию, чтобы уменьшить нагрузку на findface-video-worker. По умолчанию область не задана.
roi	Включает отправку на Сервер лиц, обнаруженных только внутри интересующей области WxH+X+Y. По умолчанию область не задана.
realtime_post_interval	Тонкая настройка режима реального времени. Если realtime_post_perm=True, период времени в миллисекундах, в течение которого детектор лиц выбирает лучший кадр и отправляет его в компонент findface-facerouter. Если realtime_post_perm=False, максимальный период времени в милли-

1. Если вы выбрали пакет `findface-video-worker-cpu` с ускорением на CPU, используйте файл конфигурации `/etc/findface-video-worker-cpu.ini`:

```
## read streams from file, do not use VideoManager
input =
## exit on first finished job, only when --input specified
exit_on_first_finished = false
## batch size
batch_size = 4
## http server port for metrics, 0=do not start server
metrics_port = 0
## resize scale, 1=do not resize
resize_scale = 1.000000
## maximum number of streams
capacity = 10
## command to obtain videomanager's grpc ip:port
mgr_cmd =
## videomanager grpc ip:port
mgr_static = 127.0.0.1:18811
## ntls server ip:port
ntls_addr = 127.0.0.1:3133
## debug: save faces to dir
save_dir =
## minimum face size
min_face_size = 60
## preinit detector for specified resolutions: "640x480;1920x1080"
resolutions =
## worker labels: "k=v;group=enter"
labels =
## use timestamps from SEI packet
use_time_from_sei = false
#------------------------------
[streamer]
#------------------------------
## streamer/shots webserver port, 0=disabled
port = 18999
## streamer url - how to access this worker on streamer_port
url = 127.0.0.1:18999
#------------------------------
[liveness]
#------------------------------
## path to liveness fnk
fnk =
## liveness threshold
threshold = 0.945000
## liveness internal algo param
interval = 1.000000
## liveness internal algo param
stdev_cnt = 1
#------------------------------
[send]
#------------------------------
## posting faces threads
threads = 8
## posting faces maximum queue size
queue_limit = 256
#------------------------------
[tracker]
```

(continues on next page)

(продолжение с предыдущей страницы)

```
#------------------------------
## max face miss duration, sec
miss_interval = 1.000000
## overlap threshold
overlap_threshold = 0.250000
#------------------------------
[models]
#------------------------------
## path to detector fnk
detector = /usr/share/findface-data/models/facedet/mtcnn.cpu.fnk
## path to quality fnk
quality = /usr/share/findface-data/models/faceattr/quality.v0.cpu.fnk
## path to norm for quality fnk
norm_quality = /usr/share/findface-data/models/facenorm/ant.v2.cpu.fnk
## path to norm200 fnk, for face send
norm_200 = /usr/share/findface-data/models/facenorm/ant.v2.cpu.fnk
## path to norm_crop2x fnk, for face send
norm_crop2x = /usr/share/findface-data/models/facenorm/crop2x.v2_maxsize400.cpu.fnk
```

Если вы выбрали пакет `findface-video-worker-gpu` с ускорением на GPU, используйте файл конфигурации `/etc/findface-video-worker-gpu.ini`.

```
## cuda device number
device_number = 0
## old gpu detector models directory
models_dir = /usr/share/findface-gpudetector/models
## read streams from file, do not use VideoManager
input =
## exit on first finished job, only when --input specified
exit_on_first_finished = false
## batch size
batch_size = 8
## http server port for metrics, 0=do not start server
metrics_port = 0
## resize scale, 1=do not resize
resize_scale = 1.000000
## maximum number of streams
capacity = 30
## command to obtain videomanager's grpc ip:port
mgr_cmd =
## videomanager grpc ip:port
mgr_static = 127.0.0.1:18811
## ntls server ip:port
ntls_addr = 127.0.0.1:3133
## debug: save faces to dir
save_dir =
## minimum face size
min_face_size = 60
## preinit detector for specified resolutions: "640x480;1920x1080"
resolutions =
## worker labels: "k=v;group=enter"
labels =
## use timestamps from SEI packet
use_time_from_sei = false
#------------------------------
[streamer]
```

(continues on next page)

(продолжение с предыдущей страницы)

```
#------------------------------
## streamer/shots webserver port, 0=disabled
port = 18999
## streamer url - how to access this worker on streamer_port
url = 127.0.0.1:18999
#------------------------------
[liveness]
#------------------------------
## path to liveness fnk
fnk =
## liveness threshold
threshold = 0.945000
## liveness internal algo param
interval = 1.000000
## liveness internal algo param
stdev_cnt = 1
#------------------------------
[send]
#------------------------------
## posting faces threads
threads = 8
## posting faces maximum queue size
queue_limit = 256
#------------------------------
[tracker]
#------------------------------
## max face miss duration, sec
miss_interval = 1.000000
## overlap threshold
overlap_threshold = 0.250000
#------------------------------
[models]
#------------------------------
## path to detector fnk
detector =
## path to quality fnk
quality =
## path to norm for quality fnk
norm_quality =
## path to norm200 fnk, for face send
norm_200 = /usr/share/findface-data/models/facenorm/ant.v2.gpu.fnk
## path to norm_crop2x fnk, for face send
norm_crop2x = /usr/share/findface-data/models/facenorm/crop2x.v2_maxsize400.gpu.fnk
## path to cache directory
cache_dir =
#------------------------------
[video_decoder]
#------------------------------
## decode video on cpu
cpu = false
```

Пользовательская настройка `findface-video-worker` на CPU/GPU выполняется с использованием следующих параметров:

CPU	GPU	Описание
`ntls-addr`		IP-адрес и порт сервера `findface-ntls`.
`mgr-static`		IP-адрес сервера `findface-video-manager`, который обеспечивает `findface-video-worker` настройками и списком видеопотоков для обработки.
`capacity`		Максимальное количество видеопотоков, обрабатываемых `findface-video-worker`.
`mgr-exec`		Возможность подключить скрипт, описывающий динамическое изменение адреса компонента `findface-videomanager-api` (вместо `mgr-static`).
`labels`		Метки, используемые для привязки экземпляра `findface-video-worker` к определенной группе камер. См. *Привязка группы камер к экземпляру findface-video-worker*.
Н/п	`fnk`	Путь к детектору *живых* лиц (Liveness).
`input`		Обрабатывать видеопотоки из файла, игнорируя данные потоков, поступающие от `findface-video-manager`.
`exit_on_first_finished`		(Только если указан `input`) Выйти после завершения первого job-задания.
`resize_scale`		Масштабировать видеокадры с заданным коэффициентом.
`save_dir`		(Для отладки) Сохранять обнаруженные лица в заданный каталог.
`min_face_size`		Минимальный обнаруживаемый размер лица.
`resolutions`		Предварительно инициализируйте `findface-video-worker` для конкретных разрешений, чтобы ускорить его работу.
Н/п	`device_number`	Номер используемого GPU-устройства.
Н/п	`models_dir`	Старый каталог с моделями GPU-детектора. В противном случае используйте данные из секции `[models]`.
Н/п	`cpu`	При необходимости декодировать видео на CPU.

Job-задания

Сервис `findface-video-manager` выдает `findface-video-worker` так называемое job-задание, задачу на обработку видео, которая содержит параметры конфигурации и сведения о видеопотоке.

Содержимое типичного job-задания показано в примере ниже.

```
curl http://127.0.0.1:18810/job/1 | jq
 % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                Dload  Upload   Total   Spent    Left  Speed
 100  1771  100  1771    0     0   447k      0 --:--:-- --:--:-- --:--:--  576k
{
  "id": "1",
  "enabled": true,
  "stream_url": "rtmp://restreamer.int.ntl/cams/openspace",
  "labels": {},
  "router_url": "http://172.17.46.13/video-detector/frame",
  "single_pass": false,
  "stream_settings": {
    "ffmpeg_params": [],
    "md_threshold": 0.002,
    "md_scale": 0.3,
    "fd_frame_height": -1,
    "uc_max_time_diff": 30,
    "uc_max_dup": 3,
    "uc_max_avg_shift": 10,
    "det_period": 8,
```

(continues on next page)

(продолжение с предыдущей страницы)

```
    "realtime": false,
    "npersons": 4,
    "disable_drops": false,
    "tracker_threads": 4,
    "parse_sei": false,
    "image_arg": "photo",
    "additional_headers": [
      "Authorization=Token b612396adc3a6dd71b82b5fe333a0a30"
    ],
    "additional_body": [],
    "api_timeout": 15000,
    "api_ssl_verify": true,
    "post_uniq": true,
    "min_score": -2,
    "min_d_score": -1000,
    "realtime_dly": 500,
    "realtime_post_perm": false,
    "rot": "",
    "roi": "",
    "draw_track": false,
    "send_track": 0,
    "min_face_size": 0,
    "max_face_size": 0,
    "overall": true,
    "only_norm": false,
    "max_candidates": 0,
    "jpeg_quality": 95,
    "ffmpeg_format": ""
  },
  "stream_settings_gpu": {
    "play_speed": -1,
    "filter_min_quality": -2,
    "filter_min_face_size": 1,
    "filter_max_face_size": 8192,
    "normalized_only": false,
    "jpeg_quality": 95,
    "overall_only": false,
    "use_stream_timestamp": false,
    "ffmpeg_params": [],
    "router_timeout_ms": 15000,
    "router_verify_ssl": true,
    "router_headers": [
      "Authorization=Token b612396adc3a6dd71b82b5fe333a0a30"
    ],
    "router_body": [],
    "start_stream_timestamp": 0,
    "imotion_threshold": 0,
    "rot": "",
    "roi": "",
    "realtime_post_interval": 1,
    "realtime_post_every_interval": false,
    "ffmpeg_format": "",
    "disable_drops": true
    "router_full_frame_png": false,
    "router_disable_normalized": false
  },
```

(continues on next page)

(продолжение с предыдущей страницы)


```
  "status": "INPROGRESS",
  "status_msg": "",
  "statistic": {
    "processed_duration": 14879,
    "faces_posted": 777,
    "faces_failed": 3,
    "faces_not_posted": 1206,
    "processing_fps": 18.816668,
    "frames_dropped": 0,
    "frames_processed": 0,
    "frames_imotion_skipped": 0,
    "decoding_soft_errors": 0,
    "job_starts": 56
  },
  "restream_url": "",
  "worker_id": "ffsec40_213ab8c0ed5d954e",
  "version": "bl068taaa7tcafrfsmq0"
}
```


Каждое job-задание имеет следующие параметры:

- `id`: id job-задания.
- `enabled`: статус активности.
- `stream_url`: URL/адрес видеопотока или файла для обработки.
- `labels`: метки, по которым будет осуществляться обработка обнаруженных лиц в компоненте `findface-facerouter` (`findface-security` в стандартной конфигурации FindFace Security).
- `single_pass`: если true (по умолчанию false), то не перезапускать обработку потока в случае ошибки.
- `router_url`: IP-адрес и порт компонента `findface-facerouter` (`findface-security` в стандартной конфигурации FindFace Security), в который компонент `findface-video-worker` отправляет обнаруженные лица для последующей обработки.
- `stream_settings`: используется только для обратной совместимости.
- `stream_settings_gpu`: параметры видеопотока, дублирующие *параметры* в файле конфигурации `findface-video-manager` (обладая при этом приоритетом).
- `status`: статус job-задания.
- `status_msg`: дополнительная информация о статусе job-задания.
- `statistic`: статистика выполнения job-задания (продолжительность процесса обработки, количество отправленных и неотправленных лиц, кадровая частота обработки, количество обработанных и пропущенных кадров, время начала обработки и т. д.).
- `worker_id`: id экземпляра `findface-video-worker`, выполняющего job-задание.

findface-ntls

Локальный сервер лицензирования `findface-ntls` – это сервис, который устанавливается на выбранном физическом сервере и служит для верификации лицензии FindFace. Для верификации используются следующие источники:

- Глобальный сервер лицензий NtechLab. Служит для лицензирования в сетях с доступом в интернет, в том числе с доступом через прокси-сервер.

- Ключ аппаратной защиты. Служит для лицензирования в закрытых сетях.

Для управления `findface-ntls` используйте основной веб-интерфейс FindFace Security. Доступны следующие возможности:

- просмотр списка приобретенных функций,
- просмотр списка текущих ограничений,
- загрузка файла лицензии,
- просмотр списка подключенных в данный момент компонентов.

Лицензируются следующие компоненты:

- `findface-tarantool-server`,
- `findface-extraction-api`,
- `findface-video-manager`,
- `findface-video-worker`.

Важно: После разрыва соединения между сервером лицензирования `findface-ntls` и лицензируемым компонентом или между сервером лицензирования `findface-ntls` и глобальным сервером лицензирования, время работы компонента составляет 6 часов.

Настройка компонента `findface-ntls` выполняется через файл конфигурации `/etc/findface-ntls.cfg`.

```
# Listen address of NTLS server where services will connect to.
# The format is IP:PORT
# Use 0.0.0.0:PORT to listen on all interfaces
# This parameter is mandatory and may occur multiple times
# if you need to listen on several specific interfaces or ports.
listen = 127.0.0.1:3133


# Directory with license files.
# NTLS use most recently generated one.
# Note: "recentness" of a license file is detected not by
#       mtime/ctime but from its internal structure.
#
# This parameter is mandatory and must occur exactly once.
license-dir = /opt/ntech/license

# You can specify proxy which NTLS will use to access
# global license server. The syntax is the same that is used by curl.
# Proxy is optional
#proxy = http://192.168.1.1:12345

# This is bind address for NTLS web-interface.
# Note: there're no authorization or access restriction mechanisms
#       in NTLS UI. If you need one, consider using nginx as proxy
#       with .htaccess / ip-based ACLs.
# This parameter may be specified multiple times.
ui = 127.0.0.1:3185
```

Пользовательская настройка `findface-ntls` выполняется с использованием следующих параметров:

Параметр	Описание
`listen`	IP-адрес сервера, с которого осуществляется обращение лицензируемых компонентов в `findface-ntls`. Для того чтобы разрешить обращение с любого IP-адреса, установите значение `0.0.0.0:3133`.
`license_dir`	Каталог для хранения файла лицензии.
`proxy`	IP-адрес и порт прокси-сервера (опционально).
`ui`	IP-адрес сервера, с которого будет доступен веб-интерфейс `findface-ntls`. Для того чтобы разрешить доступ со всех адресов, измените значение на "`0.0.0.0`".

findface-security

Компонент `findface-security` обеспечивает доступ конечного пользователя к функциям ядра FindFace. Отвечает за взаимодействие между ядром FindFace Core и веб-интерфейсом, а также функционирование системы как единого целого, реализует HTTP- и веб-сокеты (вместе с Django), обновление базы данных и *вебхуки*.

Компонент `findface-security` также выполняет функции компонента `findface-facerouter` (часть ядра FindFace), задавая правила обработки обнаруженных лиц. Он получает рамку с лицом и нормализованное изображение лица вместе с исходным кадром и другими данными (например, датой и временем детекции) от сервиса `findface-video-worker` и перенаправляет их для дальнейшей обработки в сервис `findface-sf-api`.

Настройка компонента `findface-security` выполняется через файл конфигурации `/etc/ffsecurity/config.py`.

```
sudo vi /etc/ffsecurity/config.py

MEDIA_ROOT = "/var/lib/ffsecurity/uploads"
STATIC_ROOT = "/var/lib/ffsecurity/static"
# SERVICE_EXTERNAL_ADDRESS prioritized for webhooks and genetec
SERVICE_EXTERNAL_ADDRESS = 'http://172.20.77.10'
EXTERNAL_ADDRESS = ''
DEBUG = False
LANGUAGE_CODE = 'en-us'
TIME_ZONE = 'UTC'
DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.postgresql',
        'NAME': 'ffsecurity',
    }
}
# use pwgen -sncy 50 1|tr "'" "." to generate your own unique key
SECRET_KEY = 'b9bd9f8e2ae9df0ab4671f87983a4738'
FFSECURITY = {
    'VIDEO_DETECTOR_TOKEN': '638f4788aa1401bf1f4d6f6b56604774',
    'CONFIDENCE_THRESHOLD': 0.75,
    'EPISODES_THRESHOLD': 0.7,
    'MINIMUM_DOSSIER_QUALITY': -2,
    'IGNORE_UNMATCHED': False,
    'EXTRACTION_API': 'http://127.0.0.1:18666/',
    'VIDEO_MANAGER_ADDRESS': 'http://127.0.0.1:18810',
    'EVENTS_MAX_MATCHED_AGE': 30,
    'EVENTS_MAX_UNMATCHED_AGE': 30,
    'NTLS_HTTP_URL': 'http://127.0.0.1:3185',
```

(continues on next page)

(продолжение с предыдущей страницы)

```
    'ROUTER_URL': 'http://127.0.0.1',
    'MONITORING_UPDATE_INTERVAL': 60,
    'SF_API_ADDRESS': 'http://127.0.0.1:18411',
    'LIVENESS_THRESHOLD': 0.75,
    'BEARD_THRESHOLD': 0.7,
}
ASGI_THREADS = 32
UVICORN_SETTINGS = {
    'workers': 'auto',
    'host': 'localhost',
    'port': 8002,
    'ws-workers': 'auto',
    'ws-host': 'localhost',
    'ws-port': 8003,
}
FFSECURITY_UI_CONFIG = {
    "event": {
        "features": {
            "f_gender_class": ["male", "female"],
            "age": {
                "f_age_gte": "",
                "f_age_lte": ""
            },
            "f_emotions_class": ["angry", "disgust", "fear", "happy", "sad", "surprise
↪"],
            "f_glasses_class": ["none", "eye", "sun"],
            "f_beard_class": ["none", "beard"],
            "f_liveness_class": ["real", "fake"],
        }
    }
}
# integration plugins
INSTALLED_APPS.append('ffsecurity_genetec')  # remove or comment out this line to␣
↪disable genetec integration
```

Пользовательская настройка `findface-security` выполняется с использованием следующих параметров:

Параметр	Описание
EXTERNAL_ADDRESS	IP-адрес или URL, который будет использован для доступа к веб-интерфейсу FindFace Security.
VIDEO_DETECTOR_TOKEN	Перейдите в раздел Токен и укажите его в данном параметре, чтобы авторизовать модуль видеодетекции лиц.
VIDEO_MANAGER_ADDRESS	IP-адрес сервера `findface-video-manager`.
EVENTS_MAX_MATCHED_AGE	Параметр, при достижении которого должно происходить автоматическое удаление из базы данных событий, для которых есть совпадения с досье.
EVENTS_MAX_UNMATCHED_AGE	Параметр, при достижении которого должно происходить автоматическое удаление из базы данных событий, для которых отсутствуют совпадения с досье.
NTLS_HTTP_URL	IP-адрес сервера `findface-ntls`.
ROUTER_URL	IP-адрес сервера `findface-security`, который будет получать обнаруженные на видео лица от экземпляров `findface-video-worker`. Адрес указывается внутренний или внешний, в зависимости от сети, в которой `findface-video-worker` взаимодействует с `findface-security`.
EXTRACTION_API	IP-адрес сервера `findface-extraction-api`.
SF_API_ADDRESS	IP-адрес сервера `findface-sf-api`.
IGNORE_UNMATCHED	Отключает запись события в базу данных, если обнаруженное лицо отсутствует в списках наблюдения (верификация дала отрицательный результат). Данную настройку рекомендуется использовать при большом количестве посетителей. Пороговая степень схожести при верификации лиц определяется параметром `CONFIDENCE_THRESHOLD`.
CONFIDENCE_THRESHOLD	Пороговая степень схожести при верификации
MINIMUM_DOSSIER_QUALITY	Минимальное качество лица на фотографии в досье. Если качество лица хуже минимального, пользователь не сможет загрузить такую фотографию в досье. Прямые изображения лиц анфас считаются наиболее качественными. Им соответствуют значения вблизи 0, как правило, отрицательные (такие как -0.00067401276, например). Перевернутые лица и лица, повернутые под большими углами, характеризуются отрицательным значениям от -5 и меньше. По умолчанию `'MINIMUM_DOSSIER_QUALITY': -2`, что соответствует среднему качеству.
EVENTS_FEATURES	Перечислите здесь модели для распознавания атрибутов лица, которые вы прописали в файле конфигурации `findface-extraction-api`.
LIVENESS_THRESHOLD	Детектор живых лиц оценивает живость лица с определенной достоверностью. В зависимости от порогового значения достоверности, он возвращает бинарный результат `Живой человек` или `Изображение`.
BEARD_THRESHOLD	Присутствие на лице бороды оценивается с определенной достоверностью. В зависимости от порогового значения достоверности, система возвращает бинарный результат `нет` или `борода`.
EPISODE_SEARCH_INTERVAL	(Доступен только для эпизодов) Период времени, предшествующий событию, в течение которого система ищет в биометрической базе данных события с похожими лицами. Если такого события не найдено, система создает новый эпизод. В противном случае она выбирает наиболее подходящее событие из открытого (LIVE) эпизода, отсортировав 100 последних похожих лиц. См. *Настройка эпизодов*.
EPISODE_MAX_DURATION	(Доступен только для эпизодов) Максимальная продолжительность эпизода в секундах. По истечении этого времени эпизод автоматически закрывается.
EPISODE_EVENT_TIMEOUT	(Доступен только для эпизодов) Максимальное время в секундах с момента добавления последнего события в эпизод. По истечении этого времени эпизод автоматически закрывается.
CUSTOM_FIELDS	Не трогайте этот раздел вручную, чтобы настроить содержание досье. Подробнее см. *Пользовательские вкладки, поля и фильтры в досье*.

Предупреждение: Секция `FFSECURITY` должна заканчиваться параметрами `EVENTS_FEATURES`/ `LIVENESS_THRESHOLD`/ `BEARD_THRESHOLD`, которые должны быть расположены друг за другом в указанном порядке.

```
...
'SF_API_ADDRESS': 'http://127.0.0.1:18411',
'EVENTS_FEATURES': ['gender', 'age', 'emotions', 'beard', 'glasses'],
'LIVENESS_THRESHOLD': 0.945,
'BEARD_THRESHOLD': 0.7,
```

`findface-facerouter`

Важно: Компонент `findface-facerouter` не входит в состав стандартной конфигурации FindFace Security. При необходимости используйте его для интеграции. См. *Плагины*.

Компонент `findface-facerouter` представляет собой сервис, через который задаются правила обработки обнаруженных на видео лиц. Правила задаются в виде пользовательских плагинов.

Компонент `findface-facerouter` принимает нормализованные изображения лиц вместе с исходным кадром и другой информацией (например, датой и временем детекции) от компонента `findface-video-worker`. В общем случае позволяет обрабатывать лица произвольным способом, в том числе отправлять их напрямую в партнерское приложение. В базовой реализации перенаправляет их в компонент `findface-sf-api` для дальнейшей обработки в соответствии с заданными правилами.

Полный список функций:

- задание правил обработки обнаруженных лиц на видео,
- перенаправление обнаруженных лиц в компонент `findface-sf-api` или другой сервис (в том числе стороннее приложение) для последующей обработки.

Настройка компонента `findface-facerouter` выполняется через файл конфигурации `/etc/findface-facerouter.py`.

```
# main.py options:

# debug                         = False
## debug - debug mode
# detector                      = ''
## detector - Detector to use if client fails to provide normalized face
## (nnd).Use "nnd" if you need to detect faces in such requests. Empty value
## rejects requests without face0.
# host                          = ''
## host - host to listen
# port                          = 18820
## port - port to listen
# sfapi_url                     = 'http://localhost:18411'
## sfapi_url - SF-API URL
# version                       = False
## version - print version

# plugin_dir.py options:

# plugin_dir                    = ''
```

(continues on next page)

(продолжение с предыдущей страницы)

```
## plugin_dir - Plugin directory for plugin_source='dir'

# abstract_define.py options:

# plugin_source                   = 'dir'
## plugin_source - Plugin source (dir)

# log.py options:

# log_file_max_size               = 100000000
## log_file_max_size - max size of log files before rollover
# log_file_num_backups            = 10
## log_file_num_backups - number of log files to keep
# log_file_prefix                 = None
## log_file_prefix - Path prefix for log files. Note that if you are running
## multiple tornado processes, log_file_prefix must be different for each of
## them (e.g. include the port number)
# log_rotate_interval             = 1
## log_rotate_interval - The interval value of timed rotating
# log_rotate_mode                 = 'size'
## log_rotate_mode - The mode of rotating files(time or size)
# log_rotate_when                 = 'midnight'
## log_rotate_when - specify the type of TimedRotatingFileHandler interval other
## options:('S', 'M', 'H', 'D', 'W0'-'W6')
# log_to_stderr                   = None
## log_to_stderr - Send log output to stderr (colorized if possible). By default
## use stderr if --log_file_prefix is not set and no other logging is
## configured.
# logging                         = 'info'
## logging - Set the Python log level. If 'none', tornado won't touch the
## logging configuration.
```

Пользовательская настройка `findface-facerouter` выполняется с использованием следующих параметров:

Параметр	Описание
`sfapi_url`	IP-адрес и порт сервера `findface-sf-api`.
`plugin_dir`	Список каталогов с плагинами, определяющими правила обработки обнаруженных на видео лиц.

1.9.3 Файл с параметрами установки

При установке FindFace Security из инсталлятора параметры установки автоматически сохраняются в файл `/tmp/<findface-installer-*>.json`. Вы можете отредактировать данный файл и использовать его при установке FindFace Security на других серверах, не отвечая на вопросы инсталлятора повторно.

Совет: Подробная информация об инсталляторе приведена в разделе *Развертывание из консольного инсталлятора*.

Важно: Обязательно удалите поля `*.config`, `exp_ip` и `int_ip` перед установкой FindFace Security на сервере с другим IP-адресом.

Пример файла с параметрами установки:

```
{
  "findface-security.config": {
    "EXTERNAL_ADDRESS": "http://172.20.77.17"
  },
  "product": "security",
  "ext_ip.bind": "0.0.0.0",
  "findface-ntls.config": {
    "NTLS_LISTEN": "127.0.0.1:3133",
    "NTLS_LISTEN_UI": "127.0.0.1:3185",
    "NTLS_LICENSE_DIR": "/opt/ntech/license"
  },
  "components": [
    "findface-data",
    "memcached",
    "etcd",
    "redis",
    "postgresql",
    "findface-ntls",
    "findface-extraction-api",
    "findface-sf-api",
    "findface-upload",
    "findface-video-manager",
    "findface-video-worker",
    "findface-security",
    "findface-tarantool-server"
  ],
  "memcached.config": {
    "max_memory": 1024,
    "listen_host": "127.0.0.1",
    "item_size": 16
  },
  "findface-video-manager.config": {
    "listen": "127.0.0.1:18810",
    "master": {
      "self_url_http": "127.0.0.1:18811",
      "self_url": "127.0.0.1:18811"
    },
    "rpc": {
      "listen": "127.0.0.1:18811"
    },
    "ntls": {
      "url": "http://127.0.0.1:3185/",
      "enabled": false
    }
  },
  "findface-video-worker.variant": "cpu",
  "findface-extraction-api.variant": "cpu",
  "ignore_lowmem": true,
  "findface-video-worker.config": {
    "FKVD_WRK_CAP": "10",
    "FKVD_MGR_ADDR": "127.0.0.1:18811",
    "FKVD_NTLS_ADDR": "127.0.0.1:3133"
  },
  "findface-extraction-api.config": {
    "listen": "127.0.0.1:18666",
    "extractors": {
```

(continues on next page)

(продолжение с предыдущей страницы)


```
      "instances": 1,
      "models": {
        "gender": "",
        "face": "face/elderberry_576.cpu.fnk",
        "age": "",
        "emotions": ""
      }
    },
    "nnd": {
      "quality_estimator": true
    },
    "license_ntls_server": "127.0.0.1:3133"
  },
  "ext_ip.advertised": "172.20.77.17",
  "findface-tarantool-server.config": {
    "shard-002": {
      "TNT_META_SCHEME": "meta_scheme",
      "TNT_LISTEN": "127.0.0.1:33002",
      "TNT_FF_LISTEN_IP": "127.0.0.1",
      "TNT_EXTRA_LUA": "\ndofile(\"/etc/ffsecurity/tnt_schema.lua\")\n",
      "TNT_FF_NTLS": "127.0.0.1:3133",
      "TNT_DATA_DIR": "/opt/ntech/var/lib/tarantool/shard-002",
      "TNT_FF_LISTEN_PORT": "8102"
    },
    "shard-001": {
      "TNT_META_SCHEME": "meta_scheme",
      "TNT_LISTEN": "127.0.0.1:33001",
      "TNT_FF_LISTEN_IP": "127.0.0.1",
      "TNT_EXTRA_LUA": "\ndofile(\"/etc/ffsecurity/tnt_schema.lua\")\n",
      "TNT_FF_NTLS": "127.0.0.1:3133",
      "TNT_DATA_DIR": "/opt/ntech/var/lib/tarantool/shard-001",
      "TNT_FF_LISTEN_PORT": "8101"
    }
  },
  "tnt_instances": 2,
  "inter_ip.bind": "127.0.0.1",
  "type": "stand-alone",
  "findface-sf-api.config": {
    "listen": "127.0.0.1:18411",
    "extraction-api": {
      "extraction-api": "http://127.0.0.1:18666"
    },
    "storage-api": {
      "shards": [
        {
          "master": "http://127.0.0.1:8101/v2/",
          "slave": ""
        },
        {
          "master": "http://127.0.0.1:8102/v2/",
          "slave": ""
        }
      ]
    }
  },
  "findface-facerouter.config": {
```


(continues on next page)

(продолжение с предыдущей страницы)

```
    "plugin_source": "dir",
    "port": "18820",
    "plugin_dir": "/etc/findface-facerouter-plugins",
    "sfapi_url": "http://127.0.0.1:18411",
    "host": "127.0.0.1"
  },
  "inter_ip.advertised": "127.0.0.1"
}
```

1.9.4 Модели нейронных сетей

В этом разделе вы найдете сводную информацию по моделям нейронных сетей, созданным в нашей лаборатории и используемым в FindFace Security:

Примечание: Конфигурация для теста производительности на CPU и GPU:

- CPU - Intel® Core™ i7-5930K CPU @ 3.50GHz × 12
- GPU - GeForce GTX 1080

Важно: При чистой установке биометрической моделью по умолчанию является `grapefruit_480`.

Предупреждение: Настоятельно не рекомендуется использовать для работы `face/elderberry_160`.

Модель	CPU, FPS	GPU, FPS	Тип нейронной сети
face/elderberry_160	14.99	204.98	Биометрия лица
face/elderberry_576.r2	2.07	71.14	
face/grapefruit_160	Требуется тестирование		
face/grapefruit_480	Требуется тестирование		
faceattr/age.v1	14.99	529.35	Распознавание возраста
faceattr/beard.v0	15.03	532.05	Распознавание бороды
faceattr/emotions.v1	10.99	235.59	Распознавание эмоций
faceattr/gender.v2	15.01	523.22	Распознавание пола
faceattr/glasses3.v0	15.01	529.64	Распознавание очков

1.9.5 Хранилища данных FindFace Security

В этом разделе:

Список хранилищ

FindFace Security использует следующие хранилища данных:

- Биометрическая база данных на основе Tarantool, в которой хранятся биометрические образцы (векторы признаков) и события идентификации лиц.
- Основная база данных системы на основе PostgreSQL, в которой хранятся внутренние данные системы, досье, учетные записи пользователей и настройки камер.
- Каталог `/var/lib/ffsecurity/uploads`, в котором хранятся загруженные в досье фотографии, видеофайлы и артефакты событий (полные кадры, миниатюры лиц и нормализованные изображения лиц).
- Каталог `/var/lib/ffupload/`, в котором хранятся только такие артефакты событий, как миниатюры лиц.

Галереи биометрической базы данных

В биометрической базе данных на основе Tarantool есть 3 галереи:

- `ffsec_dossier_face`: биометрические образцы, извлеченные из фотографий в досье.
- `ffsec_events`: биометрические образцы, извлеченные из лиц, обнаруженных на видео.
- `ffsec_monitoring`: биометрические образцы из всех досье в мониторинге (активных).

1.9.6 Опции резервного копирования базы данных

Для резервного копирования биометрической базы данных вам понадобиться утилита `findface-storage-api-dump`. Данная утилита может быть запущена со следующими опциями:

Примечание: Вы можете найти подробную информацию по использованию `findface-storage-api-dump` в разделе *Резервное копирование и восстановление хранилищ данных*.

```
findface-storage-api-dump --help

Usage of findface-storage-api-dump:
  -cache string
        Cache type: inmemory, redis or memcache (default "memcache")
  -cache-inmemory-size int
        Maximum number of items in ARC cache (default 16384)
  -cache-memcache-nodes value
        Comma-separated list of memcache shards (default 127.0.0.1:11211)
  -cache-memcache-timeout duration
        Specifies read/write timeout (default 100ms)
  -cache-redis-addr string
        Host:Port address (default "localhost:6379")
  -cache-redis-db int
        Database to be selected after connecting to the server.
  -cache-redis-network string
        Network type, either tcp or unix (default "tcp")
  -cache-redis-password string
        Optional password. Must match the password specified in the requirepass server␣
↪configuration option.
```

(continues on next page)

(продолжение с предыдущей страницы)

```
  -cache-redis-timeout duration
        Specifies dial/read/write timeout (default 5s)
  -config string
        Path to config file
  -config-template
        Output config template and exit
  -continue-on-errors
        Continue on errors instead of exiting
  -cpu-profile string
        Enable CPU profile and set output file
  -extraction-api-extraction-api string
        Extraction API address (default "http://127.0.0.1:18666")
  -extraction-api-timeouts-connect duration
        extraction-api-timeouts-connect (default 5s)
  -extraction-api-timeouts-idle-connection duration
        extraction-api-timeouts-idle-connection (default 10s)
  -extraction-api-timeouts-overall duration
        extraction-api-timeouts-overall (default 35s)
  -extraction-api-timeouts-response-header duration
        extraction-api-timeouts-response-header (default 30s)
  -limits-allow-return-facen
        Allow returning raw feature vectors to detect responses if ?return_facen=true
  -limits-body-image-length int
        Maximum length of image supplied in request body (default 33554432)
  -limits-deny-networks string
        Comma-separated list of subnets that are not allowed to fetch from (default "127.0.0.0/8,
↪192.168.0.0/16,10.0.0.0/8,::1/128,fe00::/8")
  -limits-url-length int
        Maximum supported url length in bytes (default 4096)
  -listen string
        IP:port to listen on (default ":18411")
  -normalized-storage-enabled
        Enables normalize saving (default true)
  -normalized-storage-s3-access-key string
        Access key for the object storage
  -normalized-storage-s3-bucket-name string
        S3 storage bucket name
  -normalized-storage-s3-endpoint string
        S3 compatible object storage endpoint
  -normalized-storage-s3-operation-timeout int
        Storage operations (Get,Put,Delete) timeout in seconds (default 30)
  -normalized-storage-s3-public-url string
        Storage public url
  -normalized-storage-s3-region string
        Storage region
  -normalized-storage-s3-secret-access-key string
        Secret key for the object storage
  -normalized-storage-s3-secure
        If 'true' API requests will be secure (HTTPS), and insecure (HTTP) otherwise (default true)
-normalized-storage-webdav-timeouts-connect duration
        normalized-storage-webdav-timeouts-connect (default 5s)
  -normalized-storage-webdav-timeouts-idle-connection duration
        normalized-storage-webdav-timeouts-idle-connection (default 10s)
  -normalized-storage-webdav-timeouts-overall duration
        normalized-storage-webdav-timeouts-overall (default 35s)
  -normalized-storage-webdav-timeouts-response-header duration
```

(continues on next page)

(продолжение с предыдущей страницы)

```
        normalized-storage-webdav-timeouts-response-header (default 30s)
  -normalized-storage-webdav-upload-url string
        webdav storage for normalized, disable normalized if empty string (default "http://127.0.0.
↪1:3333/uploads/")
  -normalized_storage string
        Normalized storage type: webdav, s3 (default "webdav")
  -output-dir string
        Output directory (default ".")
  -storage-api-max-idle-conns-per-host int
        storage-api-max-idle-conns-per-host (default 20)
  -storage-api-timeouts-connect duration
        storage-api-timeouts-connect (default 5s)
  -storage-api-timeouts-idle-connection duration
        storage-api-timeouts-idle-connection (default 10s)
  -storage-api-timeouts-overall duration
        storage-api-timeouts-overall (default 35s)
  -storage-api-timeouts-response-header duration
        storage-api-timeouts-response-header (default 30s)
```

1.9.7 Опции восстановления базы данных

Для восстановления биометрической базы данных из резервной копии вам понадобится утилита `findface-storage-api-restore`. Данная утилита может быть запущена со следующими опциями:

Примечание: Вы можете найти подробную информацию по использованию `findface-storage-api-restore` в разделе *Резервное копирование и восстановление хранилищ данных*.

```
findface-storage-api-restore --help

Usage of findface-storage-api-restore:
  -cache string
        Cache type: inmemory, redis or memcache (default "memcache")
  -cache-inmemory-size int
        Maximum number of items in ARC cache (default 16384)
  -cache-memcache-nodes value
        Comma-separated list of memcache shards (default 127.0.0.1:11211)
  -cache-memcache-timeout duration
        Specifies read/write timeout (default 100ms)
  -cache-redis-addr string
        Host:Port address (default "localhost:6379")
  -cache-redis-db int
        Database to be selected after connecting to the server.
  -cache-redis-network string
        Network type, either tcp or unix (default "tcp")
  -cache-redis-password string
        Optional password. Must match the password specified in the requirepass server␣
↪configuration option.
  -cache-redis-timeout duration
        Specifies dial/read/write timeout (default 5s)
  -config string
        Path to config file
  -config-template
```

(continues on next page)

(продолжение с предыдущей страницы)

```
        Output config template and exit
  -cpu-profile string
        Enable CPU profile and set output file
  -dont-create-gallery
        Don't create gallery, fail if doesn't exist
  -extraction-api-extraction-api string
        Extraction API address (default "http://127.0.0.1:18666")
  -extraction-api-timeouts-connect duration
        extraction-api-timeouts-connect (default 5s)
  -extraction-api-timeouts-idle-connection duration
        extraction-api-timeouts-idle-connection (default 10s)
  -extraction-api-timeouts-overall duration
        extraction-api-timeouts-overall (default 35s)
  -extraction-api-timeouts-response-header duration
        extraction-api-timeouts-response-header (default 30s)
  -limits-allow-return-facen
        Allow returning raw feature vectors to detect responses if ?return_facen=true
  -limits-body-image-length int
        Maximum length of image supplied in request body (default 33554432)
  -limits-deny-networks string
        Comma-separated list of subnets that are not allowed to fetch from (default "127.0.0.0/8,
↪192.168.0.0/16,10.0.0.0/8,::1/128,fe00::/8")
  -limits-url-length int
        Maximum supported url length in bytes (default 4096)
  -listen string
        IP:port to listen on (default ":18411")
  -normalized-storage-enabled
        Enables normalize saving (default true)
  -normalized-storage-s3-access-key string
        Access key for the object storage
  -normalized-storage-s3-bucket-name string
        S3 storage bucket name
  -normalized-storage-s3-endpoint string
        S3 compatible object storage endpoint
  -normalized-storage-s3-operation-timeout int
        Storage operations (Get,Put,Delete) timeout in seconds (default 30)
  -normalized-storage-s3-public-url string
        Storage public url
  -normalized-storage-s3-region string
        Storage region
  -normalized-storage-s3-secret-access-key string
        Secret key for the object storage
  -normalized-storage-s3-secure
        If 'true' API requests will be secure (HTTPS), and insecure (HTTP) otherwise (default true)
  -normalized-storage-webdav-timeouts-connect duration
        normalized-storage-webdav-timeouts-connect (default 5s)
  -normalized-storage-webdav-timeouts-idle-connection duration
        normalized-storage-webdav-timeouts-idle-connection (default 10s)
  -normalized-storage-webdav-timeouts-overall duration
        normalized-storage-webdav-timeouts-overall (default 35s)
  -normalized-storage-webdav-timeouts-response-header duration
        normalized-storage-webdav-timeouts-response-header (default 30s)
  -normalized-storage-webdav-upload-url string
        webdav storage for normalized, disable normalized if empty string (default "http://127.0.0.
↪1:3333/uploads/")
  -normalized_storage string
```

(continues on next page)

(продолжение с предыдущей страницы)

```
      Normalized storage type: webdav, s3 (default "webdav")
-rename string
      Ignore dump header and use this string as gallery name
-storage-api-max-idle-conns-per-host int
      storage-api-max-idle-conns-per-host (default 20)
-storage-api-timeouts-connect duration
      storage-api-timeouts-connect (default 5s)
-storage-api-timeouts-idle-connection duration
      storage-api-timeouts-idle-connection (default 10s)
-storage-api-timeouts-overall duration
      storage-api-timeouts-overall (default 35s)
-storage-api-timeouts-response-header duration
      storage-api-timeouts-response-header (default 30s)
```

Глава 2

Руководство оператора

2.1 Веб-интерфейс

Работа с FindFace Security выполняется через веб-интерфейс. Для того чтобы отобразить веб-интерфейс, введите его адрес в адресной строке браузера и пройдите авторизацию.

Примечание: Логин и пароль для авторизации выдаются администратором.

Веб-интерфейс имеет удобный и интуитивный дизайн и обеспечивает доступ к следующим функциям:

- *Идентификация по базам данных*.
- *Идентификация лиц в режиме реального времени*.
- *Работа с досье* (только для пользователей с правами оператора).
- *Видеостена*.

2.2 Идентификация по базам данных

FindFace Security позволяет выполнять идентификацию (поиск) лиц по следующим базам данных:

- База данных обнаруженных на видео лиц (вкладка *События*).
- База данных досье (вкладка *Досье*). Содержит эталонные изображения лиц.

Поиск лиц выполняется на вкладке *Поиск*.

В этой главе:

- *Идентификация лица по базе данных обнаруженных лиц*
- *Идентификация лица по базе данных досье*

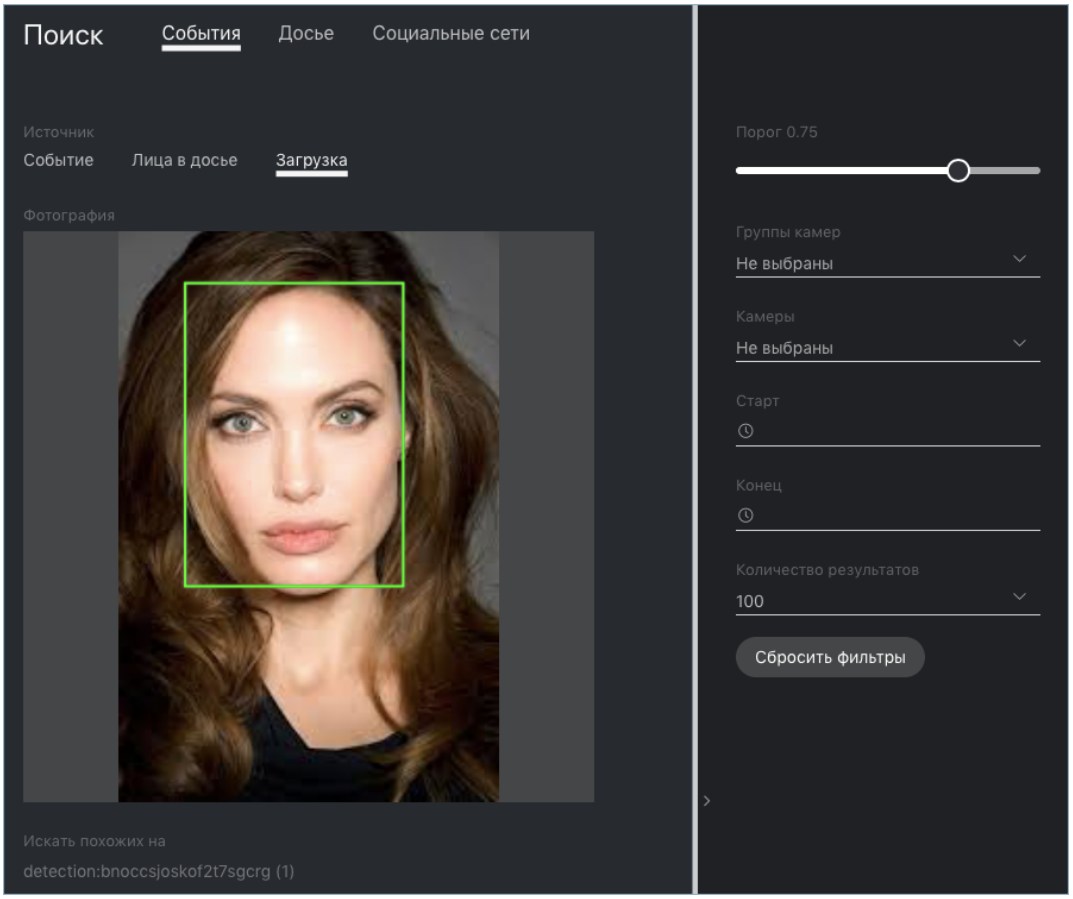
2.2.1 Идентификация лица по базе данных обнаруженных лиц

FindFace Security позволяет выполнять идентификацию лица по базе данных обнаруженных на видео лиц.

Примечание: В интерфейсе база данных представлена списком событий (вкладка *События*).

Для идентификации лица по базе данных выполните следующие действия:

1. Перейдите на вкладку *Поиск*.



2. Укажите место поиска: *События*.
3. Задайте лицо, которое требуется найти, одним из следующих способов:
   - Задав ID события с искомым лицом.
   - Задав ID досье с искомым лицом. Если досье содержит несколько фотографий, выберите те, которые будут использованы для поиска.

- Загрузив фотографию. Фотография будет отображена в одноименном поле. Если на фотографии присутствует несколько лиц, выберите нужное.

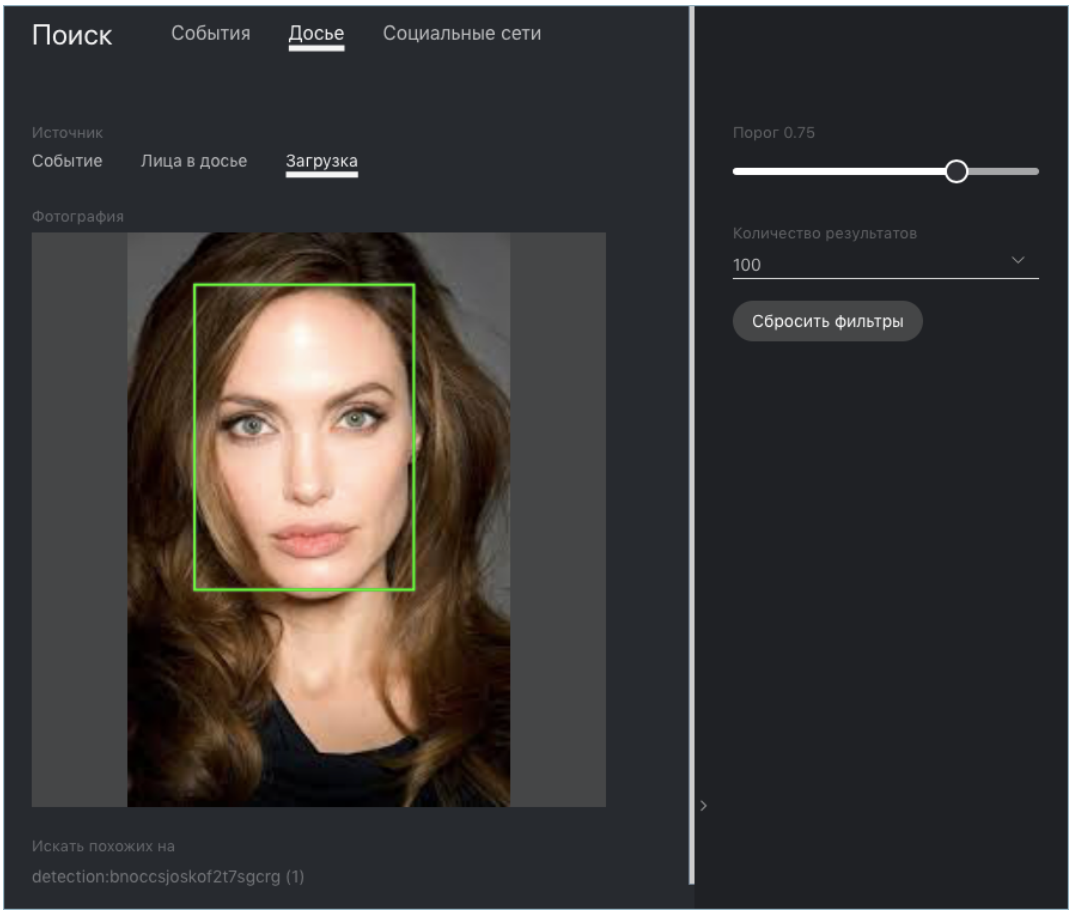
4. По умолчанию в результатах поиска отображаются лица, степень схожести которых с искомым равна или превышает 0.75. При необходимости измените данное значение.
5. При необходимости укажите группу камер и период времени, в течение которого произошло событие.
6. Выберите метод сортировки результатов поиска: по степени уверенности алгоритма в совпадении лиц (степени схожести между лицами) или дате.
7. Укажите максимальное количество событий в результатах поиска.
8. Нажмите *Поиск*. Результаты поиска будут отображены ниже. Для каждого найденного лица будет указана вероятность его совпадения с лицом на фотографии.

2.2.2 Идентификация лица по базе данных досье

FindFace Security позволяет выполнять идентификацию лица по базе данных, содержащей досье с эталонными изображениями лиц.

Для идентификации лица по базе данных выполните следующие действия:

1. Перейдите на вкладку *Поиск*.



2. Укажите место поиска: *Досье*.

3. Задайте лицо, которое требуется найти, одним из следующих способов:
   - Задав ID события с искомым лицом.
   - Задав ID досье с искомым лицом. Если досье содержит несколько фотографий, выберите те, которые будут использованы для поиска.
   - Загрузив фотографию. Фотография будет отображена в одноименном поле. Если на фотографии присутствует несколько лиц, выберите нужное.
4. По умолчанию в результатах поиска отображаются лица, степень схожести которых с искомым равна или превышает 0.75. При необходимости измените данное значение.
5. Выберите метод сортировки результатов поиска: по степени уверенности алгоритма в совпадении лиц (степени схожести между лицами) или дате.
6. Укажите максимальное количество событий в результатах поиска.
7. Нажмите *Поиск*. Результаты поиска будут отображены ниже. Для каждого найденного лица будет указана вероятность его совпадения с лицом на фотографии.

2.3 Идентификация лиц в режиме реального времени

Результат работы системы по части идентификации лиц на видеоизображении в режиме реального времени отображается на вкладках *События* и *Эпизоды*. Помимо работы с текущими событиями идентификации, данные вкладки также предоставляют доступ к истории событий. Данный раздел полностью посвящен работе с вкладкой *События*.

Совет: *Эпизоды* позволяют поднять безопасность на новый уровень.

Совет: Поиск лица в списке событий и базе данных досье с эталонными изображениями лиц выполняется на вкладке *Поиск*.

Совет: Для идентификации лиц в архивных видео см. *Идентификация лиц в оффлайн видео*.

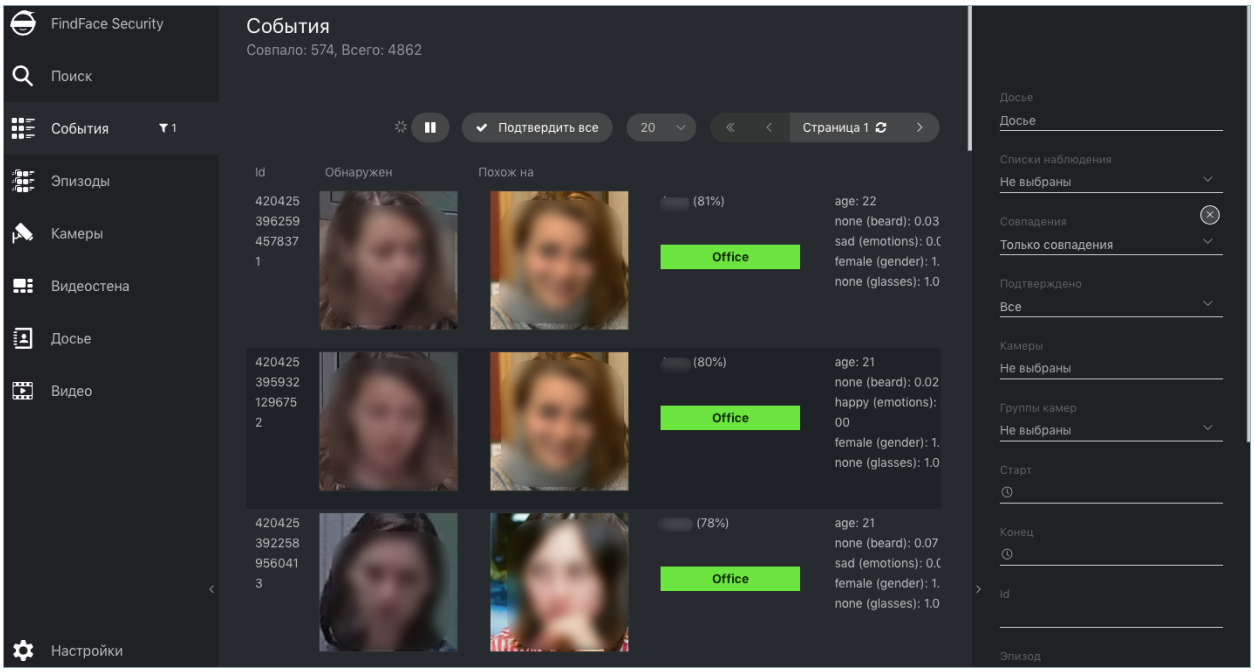
Важно: Вы можете *включить звуковые уведомления* для событий, связанных с конкретными списками наблюдения. В некоторых браузерах для того чтобы воспроизводился звук, вкладка с событиями должна оставаться в фокусе. Чтобы выделить вкладку, откройте ее и щелкните в любом месте страницы.

В этой главе:

- *Карточка события. Поиск лица*

2.3.1 Просмотр событий идентификации в режиме реального времени

При обнаружении лица в списке событий выводится уведомление.



Уведомление содержит следующую информацию:

- Если на лицо отсутствует досье: нормализованное изображение лица, дата и время обнаружения лица, группа камер.
- Если на лицо заведено досье: нормализованное изображение лица, фотография из досье, имя персоны, степень схожести лиц, комментарий из досье, список досье, дата и время обнаружения лица, группа камер.

Примечание: Система может быть настроена таким образом, что уведомления будут выводиться только для лиц с досье.

Важно: Для того чтобы остановить вывод новых уведомлений, нажмите на кнопку над списком событий.

К событиям (уведомлениям) в списке можно применить следующие фильтры:

- *Досье*: отображать только события по определенному досье.
- *Списки наблюдения*: отображать только события по определенному списку наблюдения.

Примечание: Для просмотра только лиц без совпадений в списке событий, установите в филь-

тре *Без совпадений*.

- *Совпадения*: отображать только события с совпадением лиц/без совпадения лиц или все события.
- *Подтверждено*: отображать только принятые/непринятые или все события.
- *Камеры*: отображать только события по определенной камере.
- *Группы камер*: отображать только события по определенной группе камер.
- *Старт*, *Конец*: отображать только события, случившиеся в определенный период времени.
- *id*: отобразить событие с определенным ID.
- *Episode*: отобразить события из эпизода с определенным ID.

2.3.2 Распознавание живых лиц и атрибутов лица

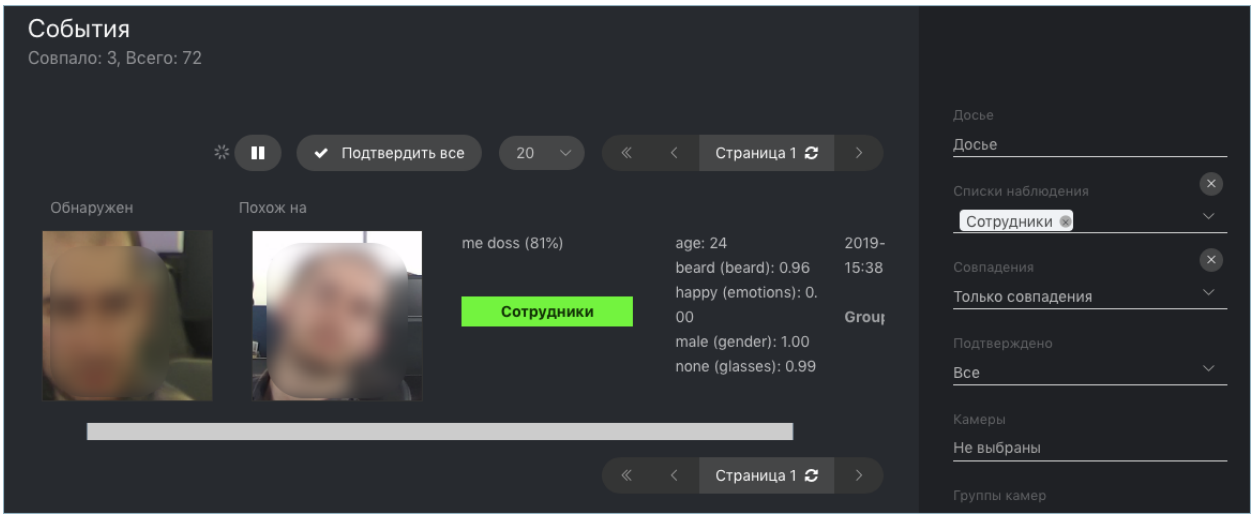
В зависимости от настроек системы, вы можете видеть оценку Liveness лица, а также результат распознавания атрибутов лица, таких как пол, возраст, эмоции, наличие очков и бороды.

Детектор живых лиц (Liveness) в автоматическом режиме отличает настоящее лицо от лица на фото- или видеоизображении, предотвращая мошенничество.

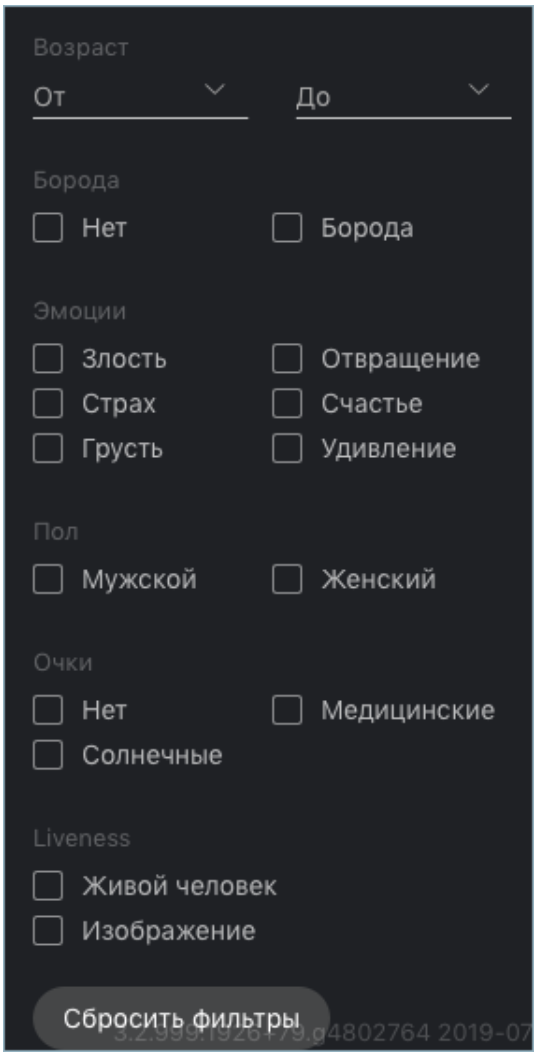
Примечание: Результат Liveness может быть `null`. Так происходит, когда детектор живых лиц отключен или не может достоверно оценить Liveness на предоставленном изображении.

Результат распознавания атрибутов лица возвращается в следующем формате:

Атрибут лица	Формат результата	Пример
Возраст	Атрибут: `возраст`: число лет	возраст: 33
Пол	Результат: `мужской/женский` (атрибут: `пол`): уверенность алгоритма в результате	женский (пол): 0.95
Эмоции	Результат: `злость/отвращение/страх/счастье/грусть/удивление` (атрибут: `эмоции`): уверенность алгоритма в результате	счастье (эмоции): 0.99
Очки	Результат: `медицинские/солнечные/нет` (атрибут: `очки`): уверенность алгоритма в результате	нет (очки): 0.87
Борода	Результат: `борода/нет` (атрибут: `борода`): уверенность алгоритма в результате	нет (борода): 0.91



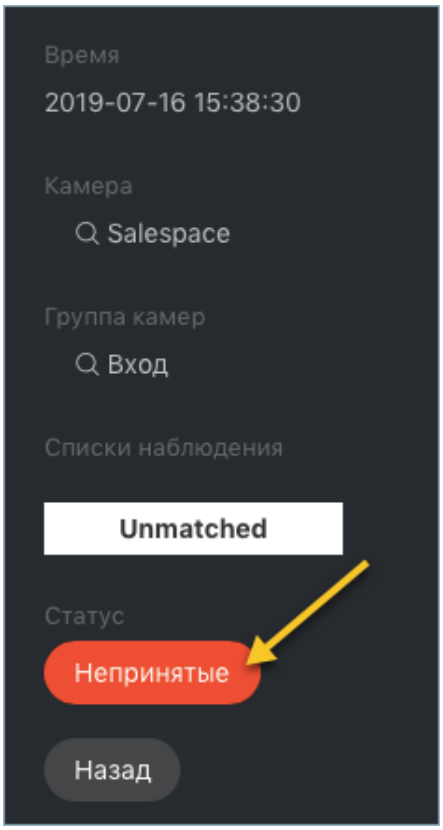
При необходимости выполните фильтрацию событий по атрибутам лиц, отобразите только живые лица/попытки спуфинга.



2.3.3 Карточка события. Принятие события

Для того чтобы перейти в карточку события из списка событий, щелкните в уведомлении по результату распознавания (*Нет совпадений* или имя из досье).

Карточка содержит ту же информацию, что и *уведомление*, а также предоставляет возможность принять событие. Для того чтобы это сделать, поставьте флажок *Подтверждение события*. Нажмите на кнопку *Сохранить*.



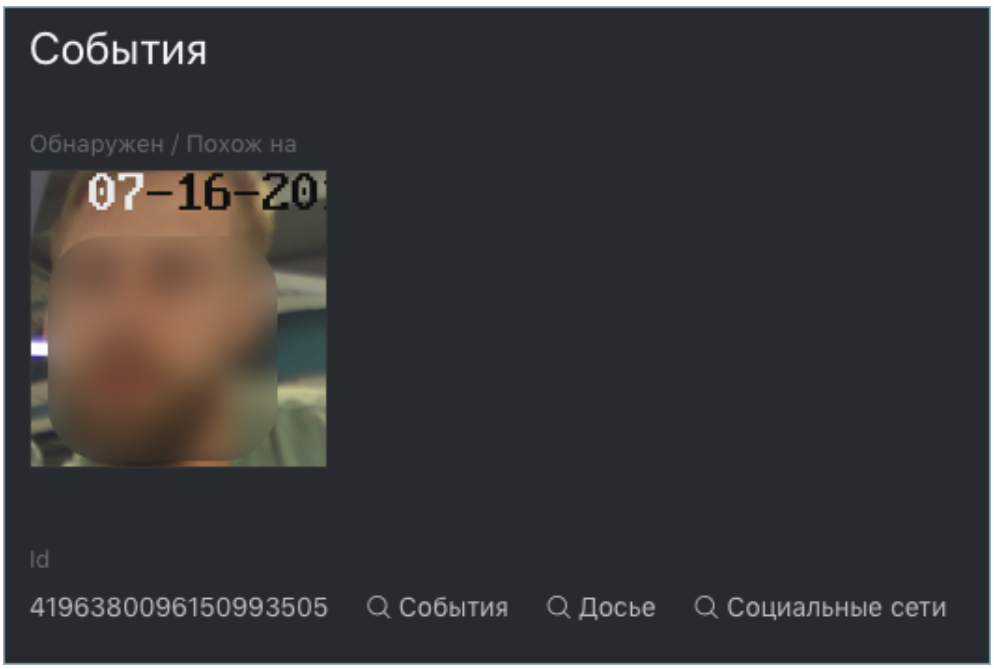
Совет: Если на обнаруженное лицо заведено досье, в него можно перейти, щелкнув по имени персоны в карточке события.

Совет: Для того чтобы принять все события, нажмите на кнопку  над списком событий.

Примечание: Принятие события может быть автоматизировано для выбранных списков наблюдения.

2.3.4 Карточка события. Поиск лица

FindFace Security позволяет искать обнаруженные лица в базе данных обнаруженных лиц и в базе данных досье с эталонными изображениями лиц. Для перехода на вкладку поиска из карточки события нажмите *События* и *Досье*.



См.также:

- *Идентификация по базам данных*.

2.4 Эпизоды событий

Результат работы системы по части идентификации лиц на видеоизображении в режиме реального времени отображается на вкладках *События* и *Эпизоды*. Помимо работы с текущими событиями идентификации, данные вкладки также предоставляют доступ к истории событий. Данный раздел полностью посвящен работе с вкладкой *Эпизоды*.

См.также:

- *Идентификация лиц в режиме реального времени*
- *Настройка эпизодов*

Эпизод — это набор событий идентификации, в которых фигурируют лица одного и того же человека, обнаруженные в течение определенного периода времени. Поскольку события на вкладке *События* отображаются в произвольном порядке, обработка большого количества разнородных событий может быть делом затруднительным и неэффективным. Функция Эпизоды позволяет автоматически объединять входящие события на основе времени обнаружения и схожести лиц. Это позволяет с легкостью обрабатывать разнородные события даже в больших количествах.

Совет: Поиск лица в списке событий и базе данных досье с эталонными изображениями лиц выполняется на вкладке *Поиск*.

Совет: Для идентификации лиц в архивных видео см. *Идентификация лиц в оффлайн видео*.

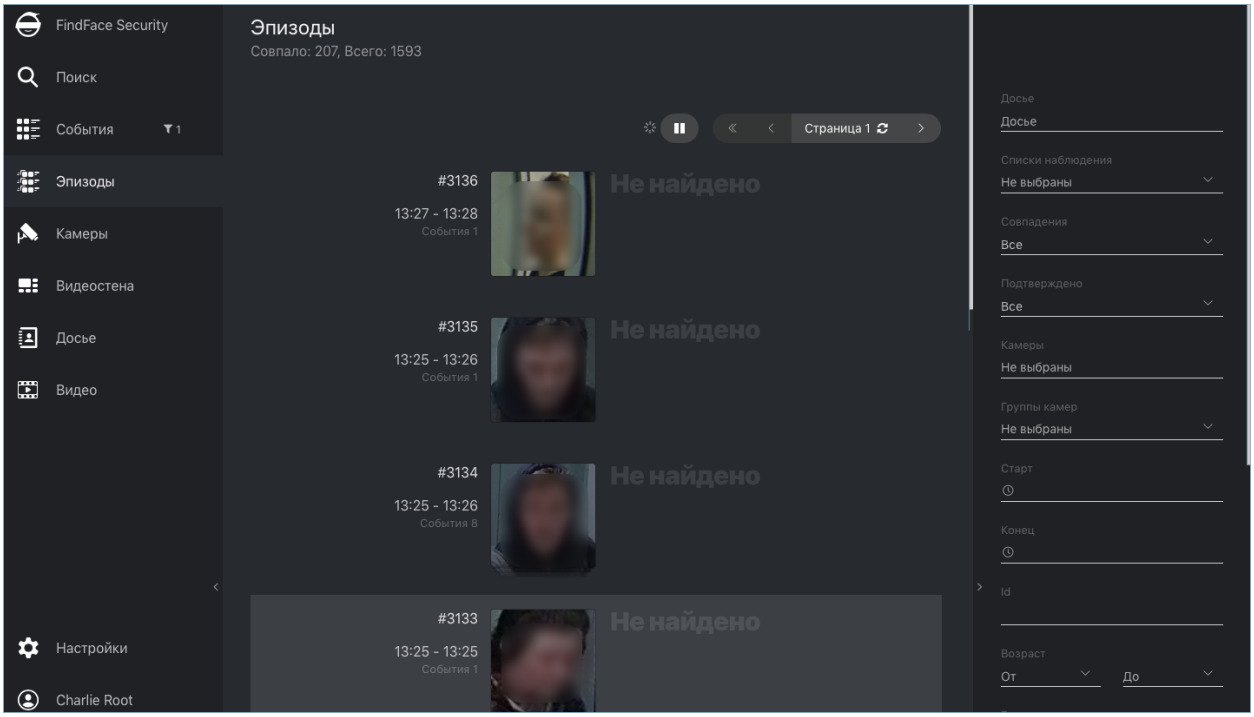
В этой главе:

2.4.1 Просмотр эпизодов

Эпизоды бывают двух типов:

- LIVE: открытый на данный момент эпизод, в который могут добавлены новые события.
- Закрытый: закрытый эпизод, добавление событий невозможно.

Список эпизодов с фильтрами и статистикой отображается на вкладке *Эпизоды*. Событие обнаружения лица либо добавляется в существующий LIVE-эпизод, либо инициирует создание нового эпизода. Каждому эпизоду присваивается id, который впоследствии можно использовать для фильтрации событий и эпизодов.



К эпизодам в списке можно применить следующие фильтры:

- *Досье*: отображать только эпизоды по определенному досье.
- *Списки наблюдения*: отображать только эпизоды по определенному списку наблюдения.

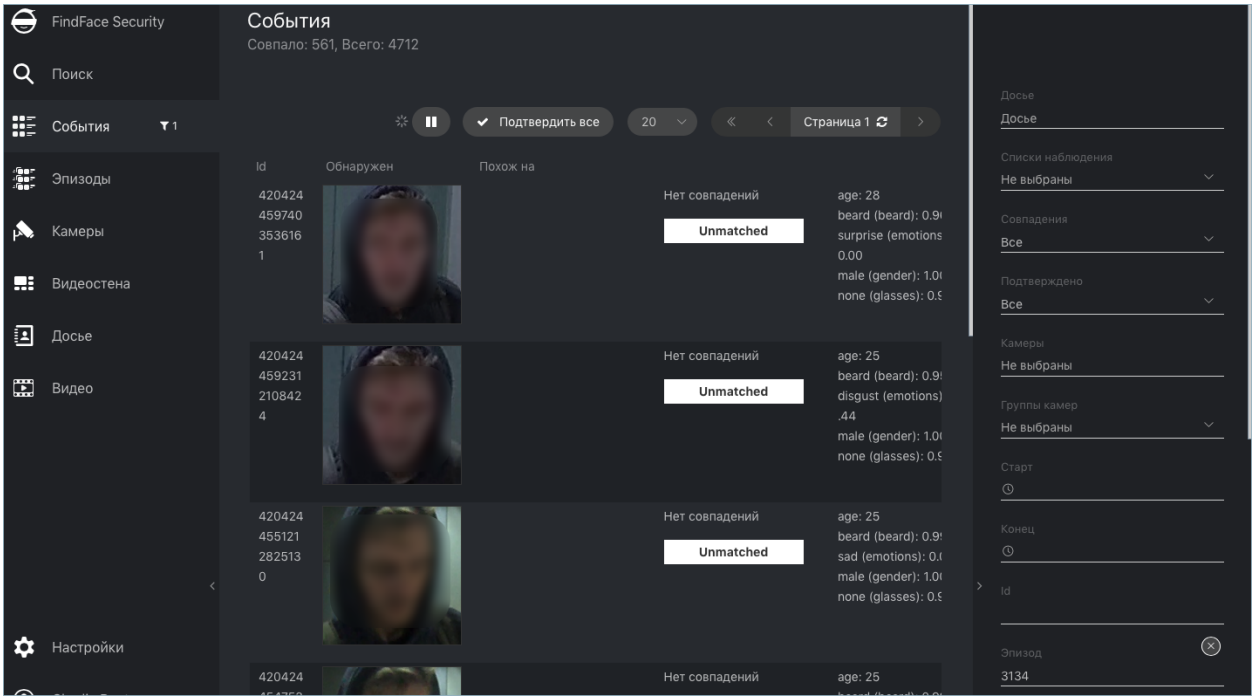
  Примечание: Для просмотра только лиц без совпадений в списке эпизодов, установите в фильтре *Без совпадений*.

- *Совпадения*: отображать только эпизоды с совпадением лиц/без совпадения лиц или все эпизоды.

- *Подтверждено*: отображать только принятые/непринятые или все эпизоды.
- *Камеры*: отображать только эпизоды по определенной камере.
- *Группы камер*: отображать только эпизоды по определенной группе камер.
- *Старт*, *Конец*: отображать только эпизоды, случившиеся в определенный период времени.
- *id*: отобразить эпизод с определенным ID.

Вы также можете отфильтровать эпизоды по Liveness и атрибутам лица (если это применимо к вашей системе).

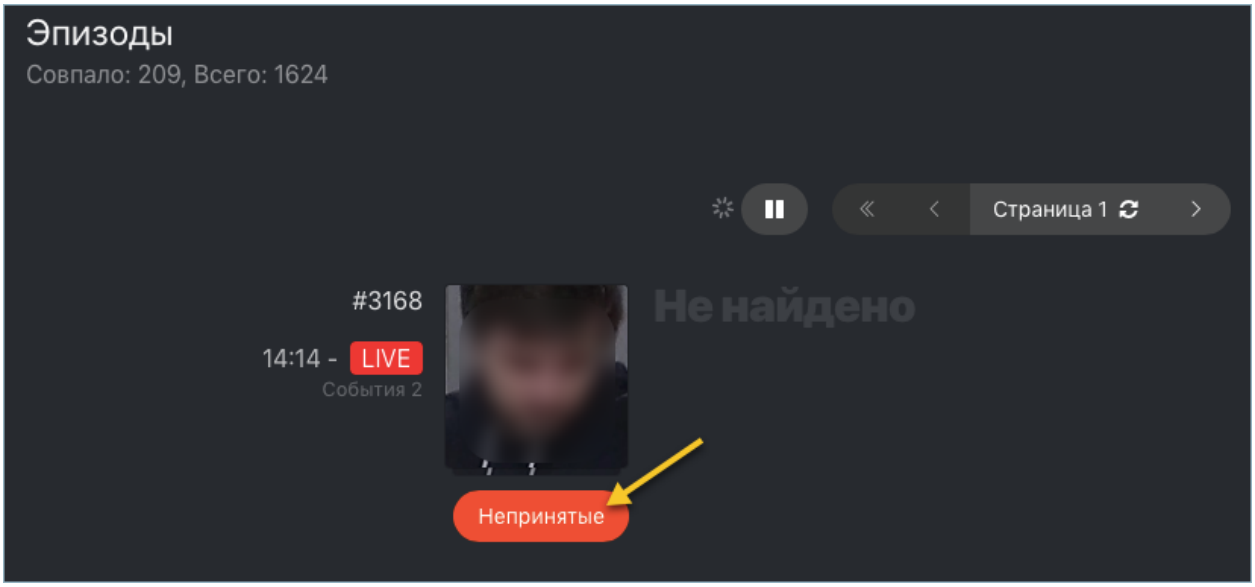
Для просмотра событий щелкните по нужному эпизоду в списке. Вы будете переправлены на вкладку *События* с соответствующим ID эпизода в фильтре *Эпизод*:



Работа с вкладкой *События* описана в разделе *Идентификация лиц в режиме реального времени*.

2.4.2 Принятие события и эпизода

Для того чтобы подтвердить эпизод целиком, нажмите *Непринятые* в списке эпизодов. В результате все события в эпизоде будут автоматически подтверждены, включая события, которые еще не добавлены (в случае эпизода LIVE).



Эпизод также автоматически подтверждается, если вы приняли все события по отдельности.

2.4.3 Фильтрация событий по ID эпизода

Для того чтобы отобразить события по ID эпизода, используйте фильтр *id* на вкладке *Эпизоды* или фильтр *ID эпизода* на вкладке *События*.

2.5 Работа с досье

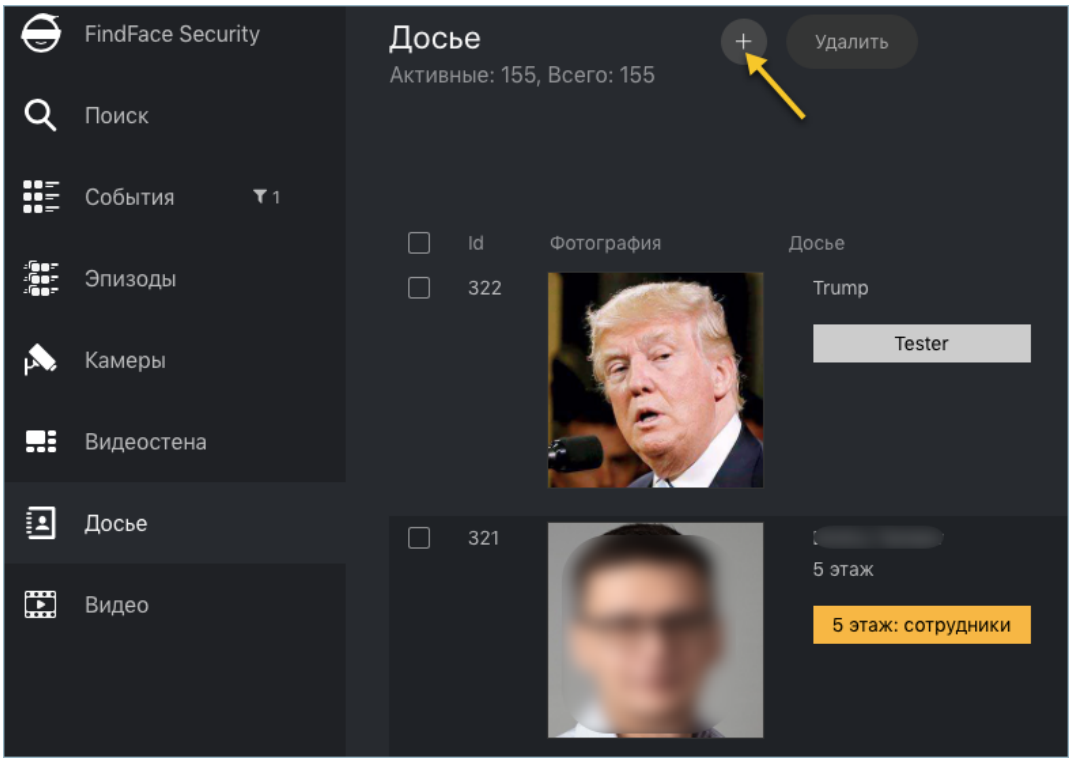
FindFace Security позволяет создавать досье на интересующих лиц, содержащее одну или несколько фотографий. Досье классифицируется по принадлежности к тому или иному списку наблюдения.

В этом разделе:

2.5.1 Создание досье

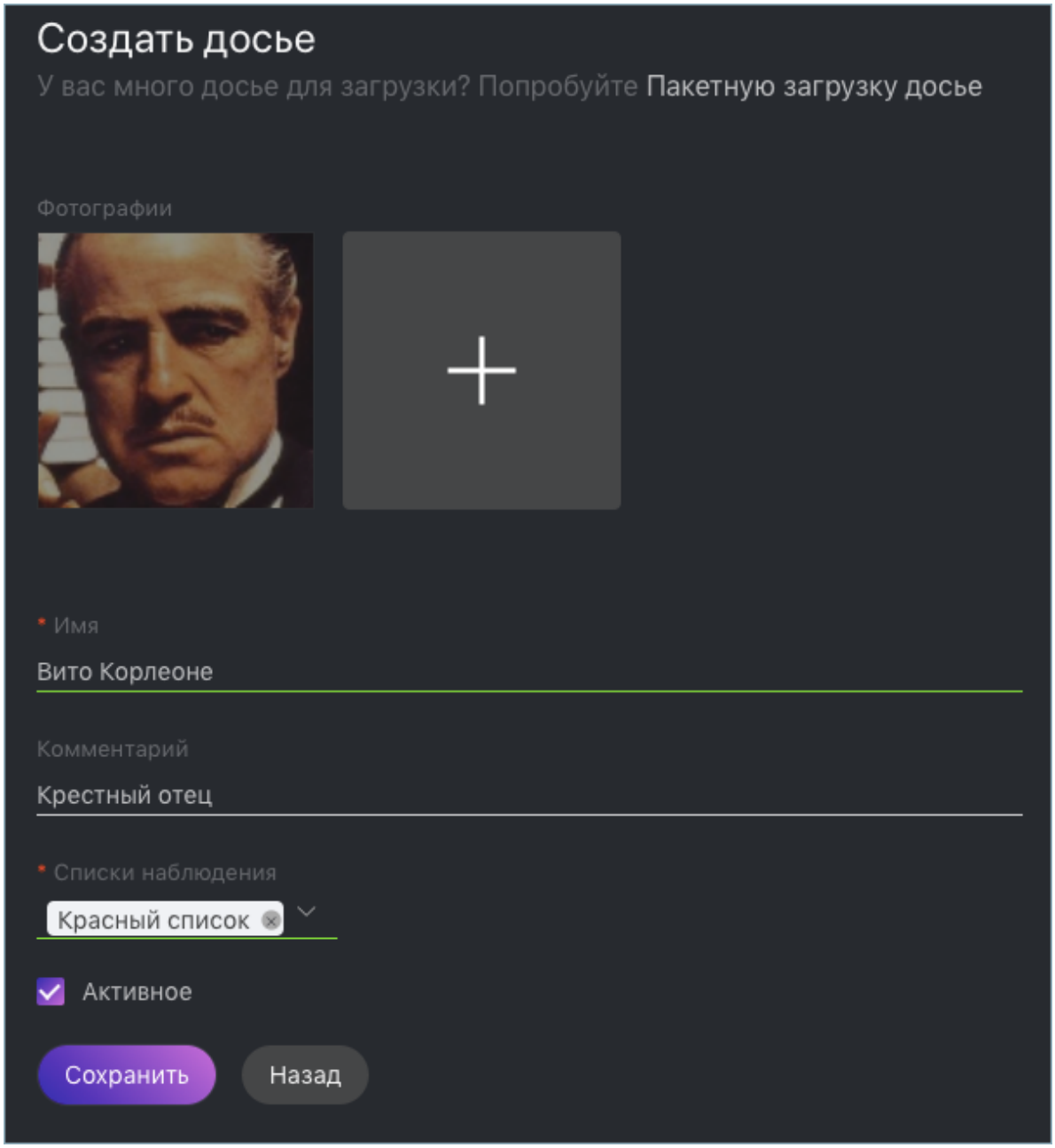
Для создания досье выполните следующие действия:

1. В веб-интерфейсе перейдите на вкладку *Досье*.
2. Нажмите *+*.



3. Добавьте одну или несколько фотографий и введите имя человека. При необходимости добавьте комментарий.

Важно: Лицо на фотографии должно быть надлежащего качества, т. е. в близком к анфас положении. При несоответствии фотографии данному требованию будет выведено сообщение с описанием ошибки.



4. Из раскрывающегося списка *Списки наблюдения* выберите список (или несколько списков, по очереди), в который следует добавить досье.

Примечание: Если ни один список наблюдения не подходит, *создайте* новый или попросите администратора сделать это.

---

5. Убедитесь, что поставлен флажок *Активное*. Если досье неактивно, оно не будет использоваться для *идентификации лица* в режиме реального времени.
6. Нажмите *Сохранить*.

2.5.2 Просмотр досье из списка наблюдения

Все созданные в FindFace Security досье отображаются на вкладке *Досье*. Используйте фильтр *Списки наблюдения*, чтобы отфильтровать досье по спискам.

2.6 Видеостена

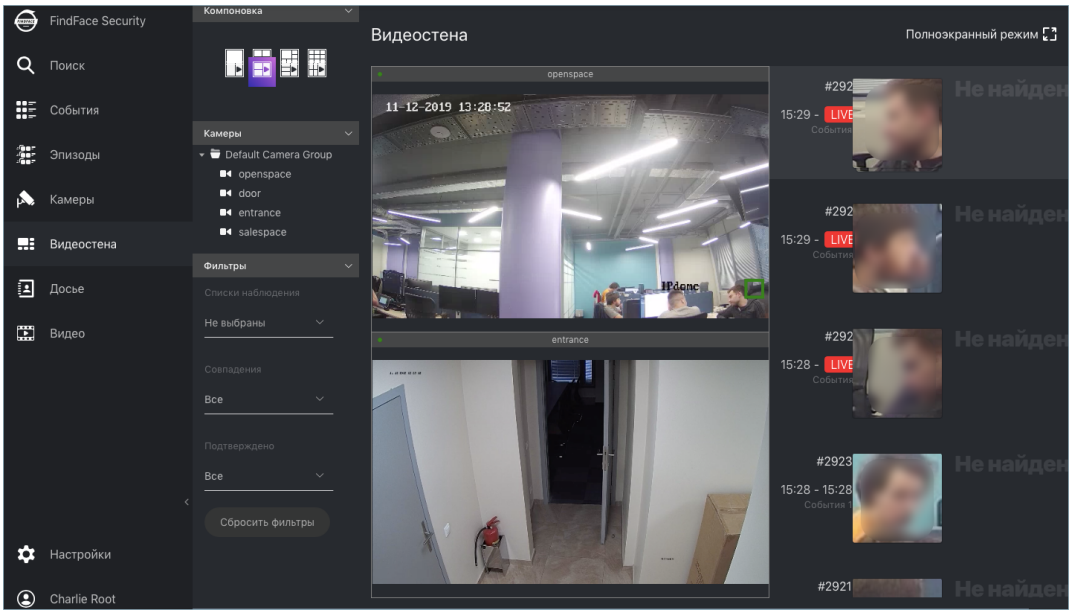
В FindFace Security встроен базовый функционал видеонаблюдения. Используйте видеостену для отображения видео с камер и видеофайлов.

Видеостена может работать в двух режимах (по 4 раскладки в каждом):

- видеотрансляция,
- видеотрансляция с детектированием лиц и лентой эпизодов.

Для отображения на видеостене видеоизображения выполните следующие действия:

1. Перейдите на вкладку *Видеостена*.
2. Выберите режим работы видеостены и раскладку камер.



3. Перетащите на видеостену выбранные камеры.

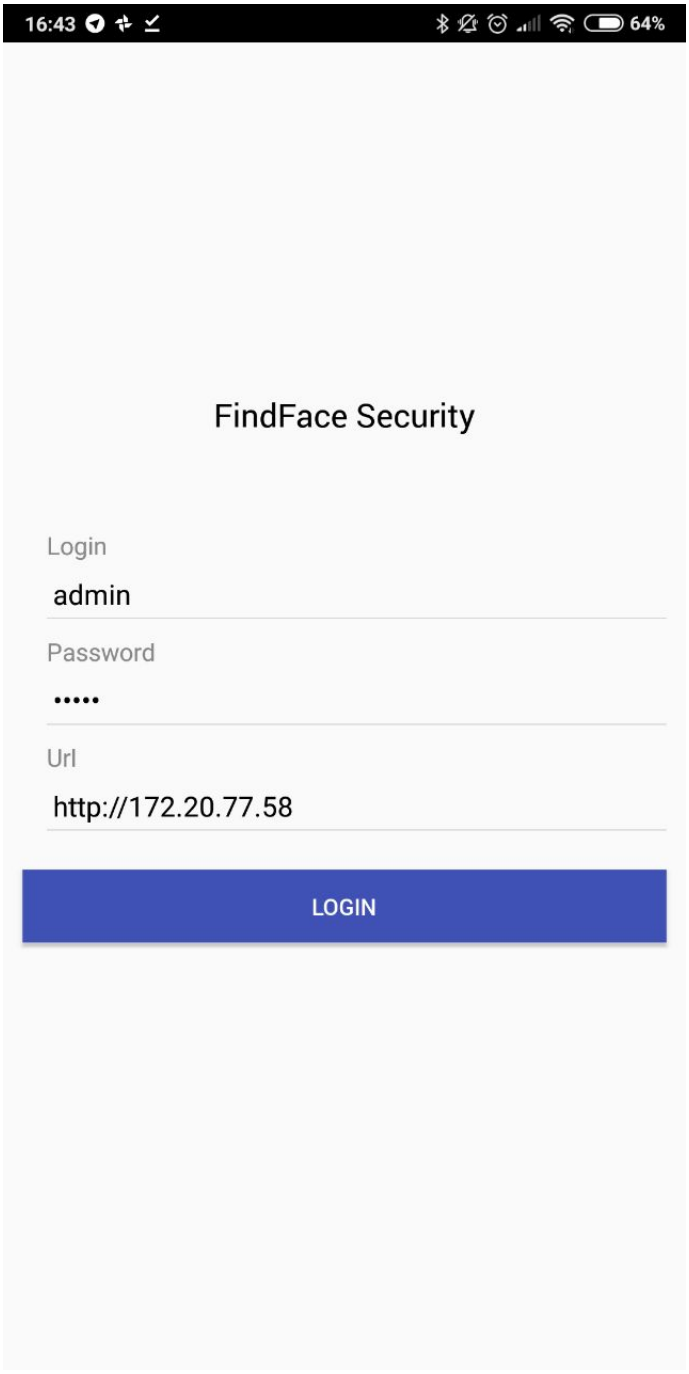
Вы можете работать с лентой эпизодов на видеостене *по аналогии* с вкладкой *Эпизоды*, включая следующие основные фильтры:

- *Списки наблюдения*
- *Совпадения*.
- *Подтверждено*.

2.7 Мобильный веб-интерфейс

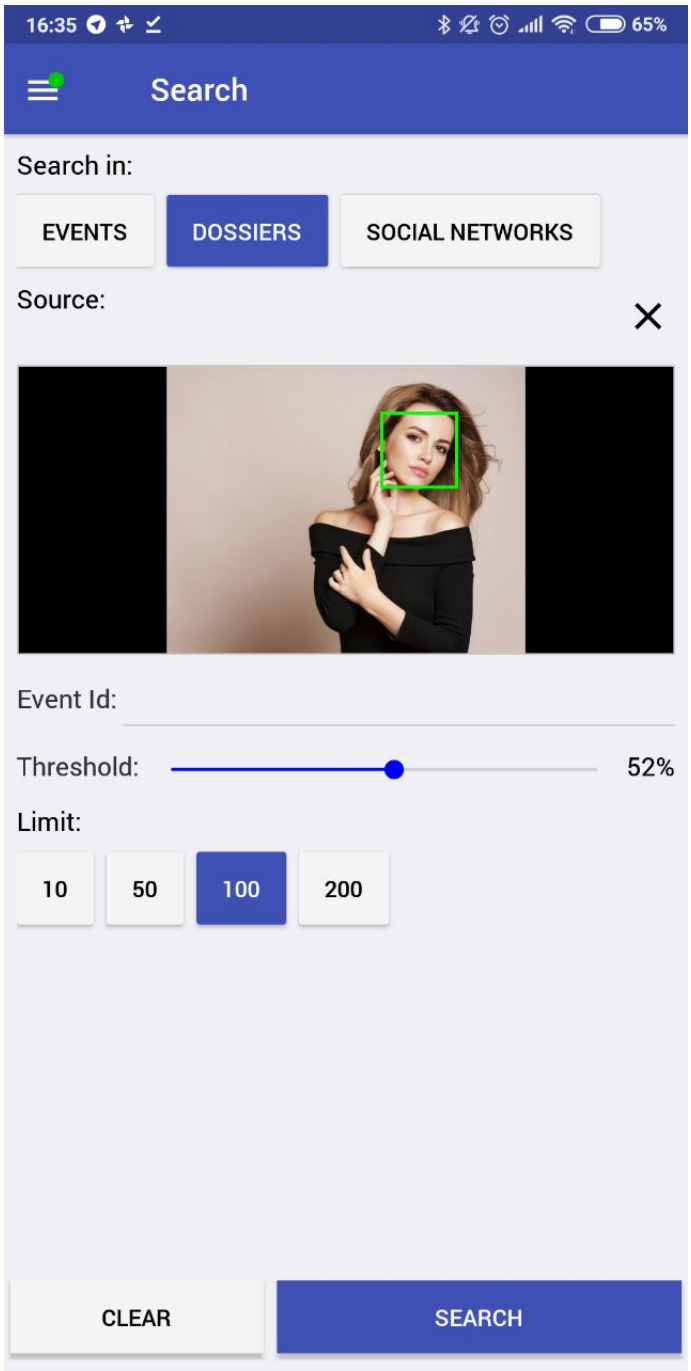
Для работы с FindFace Security также можно использовать упрощенную мобильную версию системы. Мобильное приложение FindFace Security поставляется по запросу для Android.

В приложении введите свой логин и пароль FindFace Security, а также адрес сервера FindFace Security и пройдите авторизацию.



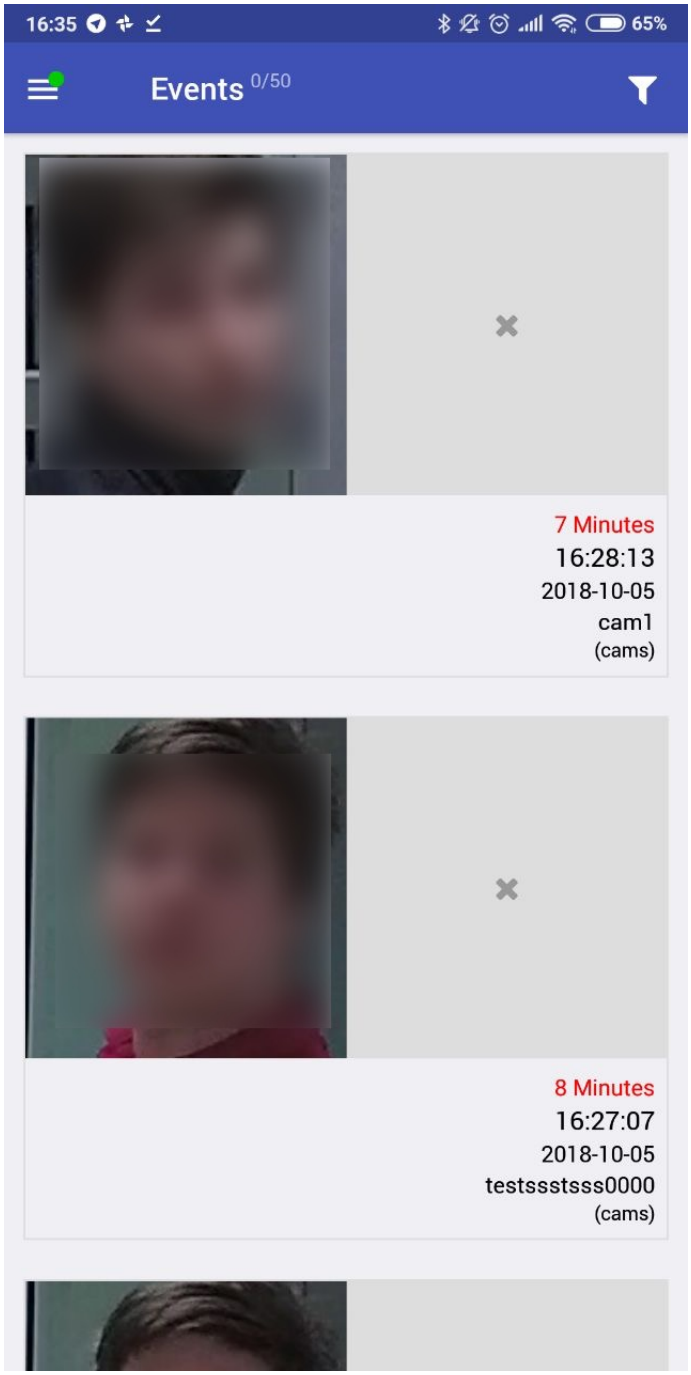
Мобильный веб-интерфейс имеет удобный и интуитивный дизайн и обеспечивает доступ к следующим функциям:

- Поиск лиц в базах данных.

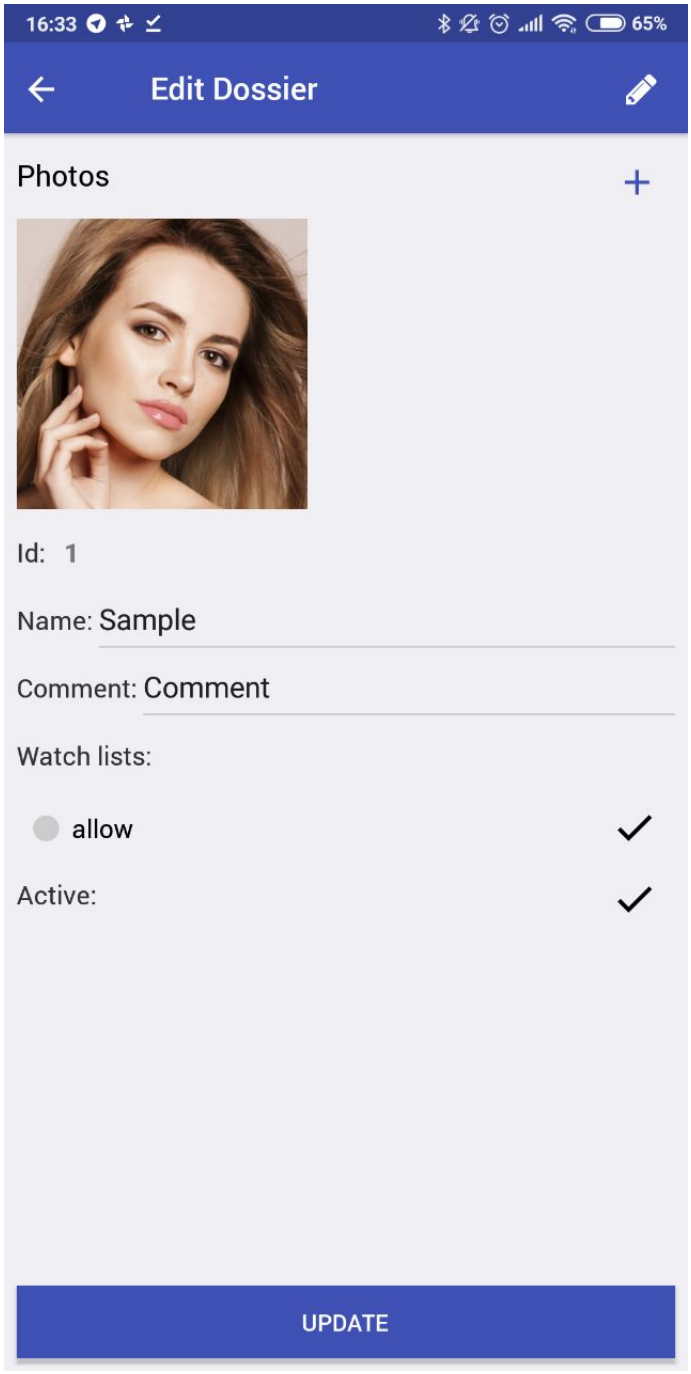


- Идентификация лиц по базам данных в режиме реального времени

Важно: Для того чтобы в мобильной версии получать события в формате push-оповещений, для соответствующего списка наблюдения нужно поставить флажки *Требовать подтверждение события* и *Включить звуковое оповещение* (через полноформатный веб-интерфейс).



- Работа с досье на персону.



Работа с данными функциями аналогична полноформатной версии.

Важно: Для доступа в *Настройки*, вам потребуется ввести PIN-код, по умолчанию 1234.

Глава 3

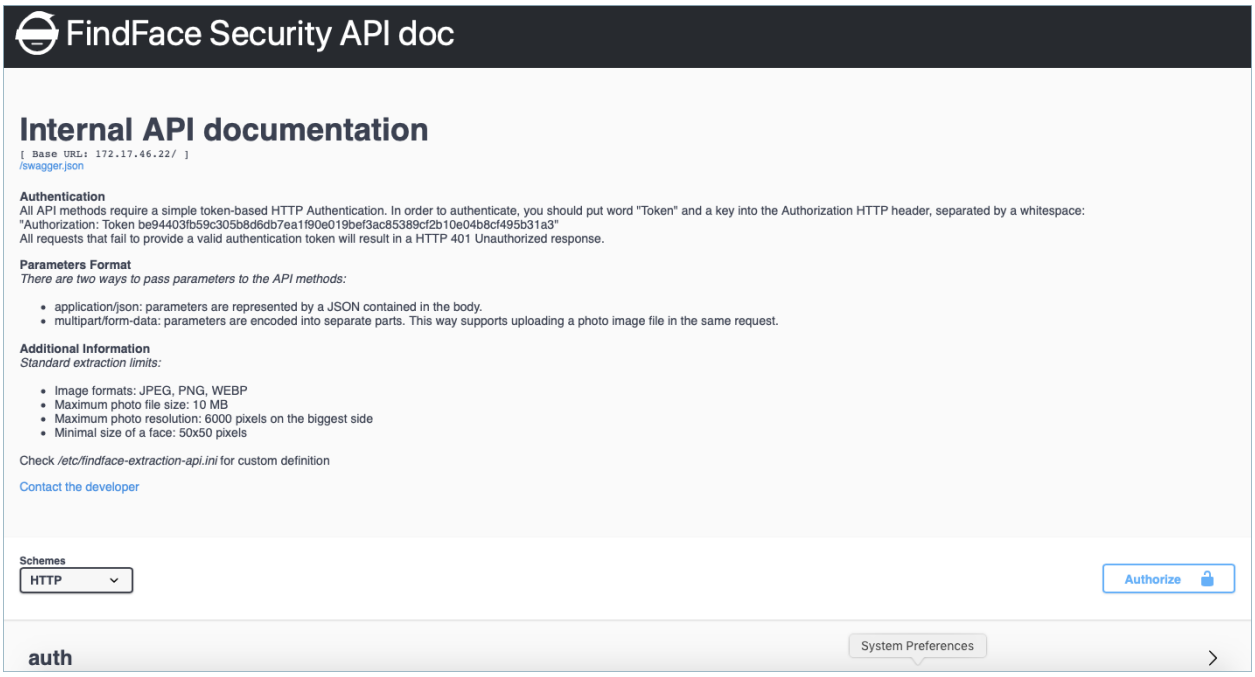
Руководство по интеграции

Данная глава посвящена возможностям интеграции с FindFace Security. Для интеграции своей системы используйте HTTP API, веб-хуки и плагины. Также обратите внимание на список наших готовых интеграций с партнерами.

3.1 HTTP API

Подробная интерактивная документация HTTP API FindFace Security доступна после установки по адресу `http://<ffsecurity_ip:port>/api-docs`. Изучайте и пробуйте.

Совет: Документацию также можно найти в веб-интерфейсе, перейдя в меню по пунктам *Настройки -> Документация API*.



3.2 Вебхуки

Вы можете настроить FindFace Security для автоматической отправки уведомлений об определенных событиях на заданный URL-адрес. Для этого создайте и настройте вебхук. При наступлении нужного события FindFace Security отправит HTTP-запрос на URL-адрес, указанный в настройках вебхука.

Вебхуки можно использовать для решения разнообразных задач, например, для уведомления пользователя об определенном событии, вызова определенных действий на целевом веб-сайте, при решении задач безопасности, таких как удаленное автоматическое управление доступом и др.

В этом разделе:

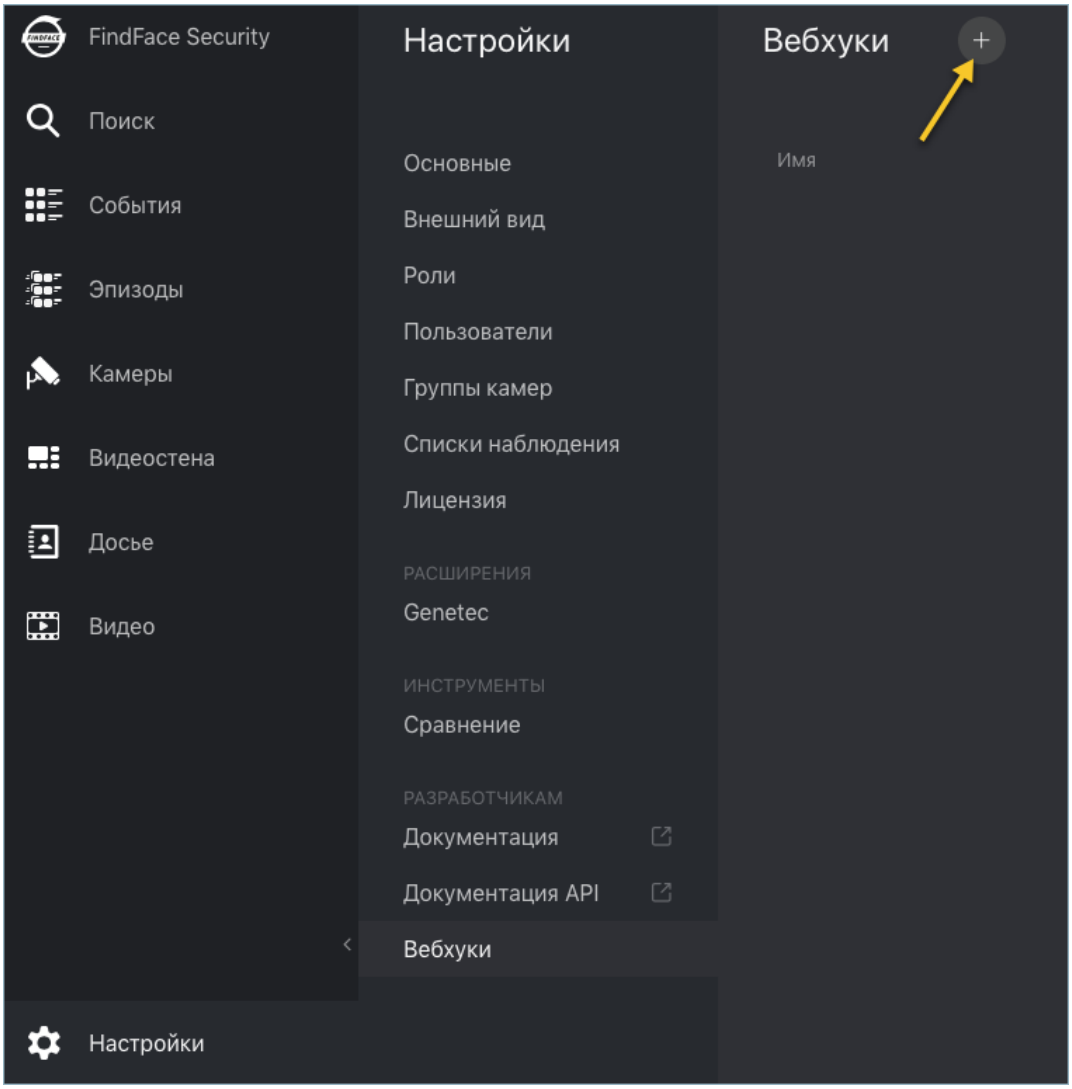
3.2.1 Настройка вебхука

Важно: Для создания вебхука необходимы права администратора.

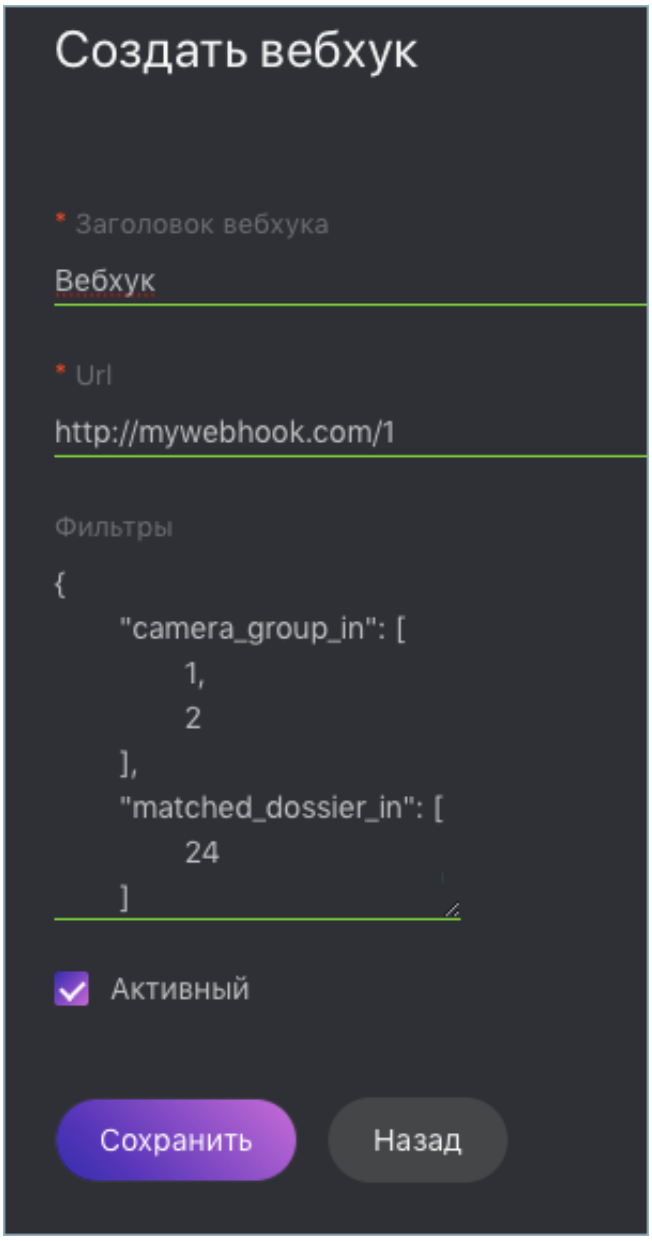
Примечание: Для того чтобы использовать вебхуки, обязательно укажите по крайней мере один из параметров `SERVICE_EXTERNAL_ADDRESS`/`EXTERNAL_ADDRESS` в файле `/etc/ffsecurity/config.py`.

Для создания вебхука выполните следующие действия:

1. Перейдите на вкладку *Настройки*. Выберите *Вебхуки*.
2. Нажмите *+*.



3. Введите имя вебхука.



4. Укажите адрес, на который будут отправляться оповещения.
5. FindFace Security будет автоматически отправлять оповещения о событиях, удовлетворяющих заданным фильтрам. Фильтровать события можно по следующим параметрам:
   - `camera_group_in`: id группы камер, число.
   - `matched_dossier_in`: id совпавшего досье, число.
   - `matched`: статус события `совпадение` (`true` или `false`), логический.
   - `camera_in`: id камеры, число.

Важно: Используйте только фильтры, соответствующие цели поиска. Для выключения фильтра удалите его из вебхука. Не оставляйте фильтр пустым (`[]`), поскольку в этом случае фильтр

вернет пустой результат.

Примечание: Для получения оповещений обо всех совпадениях с досье передайте скобки без вложенных фильтров:

```
{}
```

Примечание: За исключением `matched`, для каждого фильтра можно задать несколько значений. В этом случае вебхук будет активироваться при совпадении одного из значений фильтра. В примере ниже вы будете оповещены о событии с группы камер `1` или `3`, если совпало досье с id `12` или `25`.

```
{
  "camera_group_in": [1, 3],
  "matched_dossier_in": [12,25]
}
```

6. Поставьте флажок *Активный*.
7. Нажмите *Сохранить*.

3.2.2 Как работает вебхук

Для тестирования работы вебхука можно использовать следующий простой веб-сервер на Python, обеспечивающий захват отправленных вебхуком оповещений о событиях:

```
from pprint import pprint
from aiohttp import web


async def handle(request):
    pprint(await request.json())
    return web.Response(status=200)


app = web.Application()
# for aiohttp v 3.x
# app.add_routes([web.post('/', handle)])

# for aiohttp v 2.x
app.router.add_post('/', handle)

web.run_app(app, port=8888)
```

Если для вебхука не задано ни одного фильтра, данный веб-сервер будет получать оповещения о каждом произошедшем в системе событии. Оповещения отправляются в следующем формате:

```
======== Running on http://0.0.0.0:8888 ========
(Press CTRL+C to quit)
[{'acknowledged': True,
  'acknowledged_by': None,
```

(continues on next page)

(продолжение с предыдущей страницы)


```
 'acknowledged_date': '2019-04-09T12:29:23Z',
 'acknowledged_reaction': None,
 'camera': 2,
 'confidence': 0.9098,
 'created_date': '2019-04-09T12:29:23Z',
 'face': 'http://172.20.77.17/uploads/2019/04/09/event/122955_face_aT3ZZh.jpg',
 'features': {'age': None,
              'beard': None,
              'emotions': None,
              'gender': None,
              'glasses': None,
              'liveness': None},
 'frame': 'http://172.20.77.17/uploads/2019/04/09/event/122955_image_3msdHH.jpg',
 'frame_coords_bottom': 981,
 'frame_coords_left': 1630,
 'frame_coords_right': 1911,
 'frame_coords_top': 701,
 'id': '4173669353687265180',
 'looks_like_confidence': None,
 'matched': True,
 'matched_dossier': 1,
 'matched_face': '4173665826982243136',
 'matched_lists': [1],
 'normalized_photo': 'http://172.20.77.17/uploads/2019/04/09/event/122955_face0_E638aW.png',
 'quality': -0.000158,
 'scores': {'direction_score': -2.62964,
            'frame_no': 800,
            'score': -0.000158435,
           'tracking_duration': 34000}}]
```


Для просмотра статуса отправки вебхука в FindFace Security, выполните следующую команду:

```
sudo journalctl -u  findface-security.service | grep webhook
```

Ответ в случае успеха:

```
`Apr 09 16:02:28 ubuntu ffsecurity[1524]: INFO     [-] hook 1 was pulled on http://172.20.77.
↪70:8888`
```

Ответ, если попытка была неудачной:

```
`Apr 09 15:59:02 ubuntu ffsecurity[1524]: INFO     [-] While working on hook 1 Exception occured:␣
↪Cannot connect to host 172.20.77.70:8888 ssl:False [Connection refused]`
```

3.3 Интеграции с партнерами

3.3.1 Genetec Security Center

Интеграция FindFace Security с программным комплексом Genetec Security Center позволяет добавлять функционал распознавания лиц в системы безопасности на базе Genetec.

Настройка интеграции

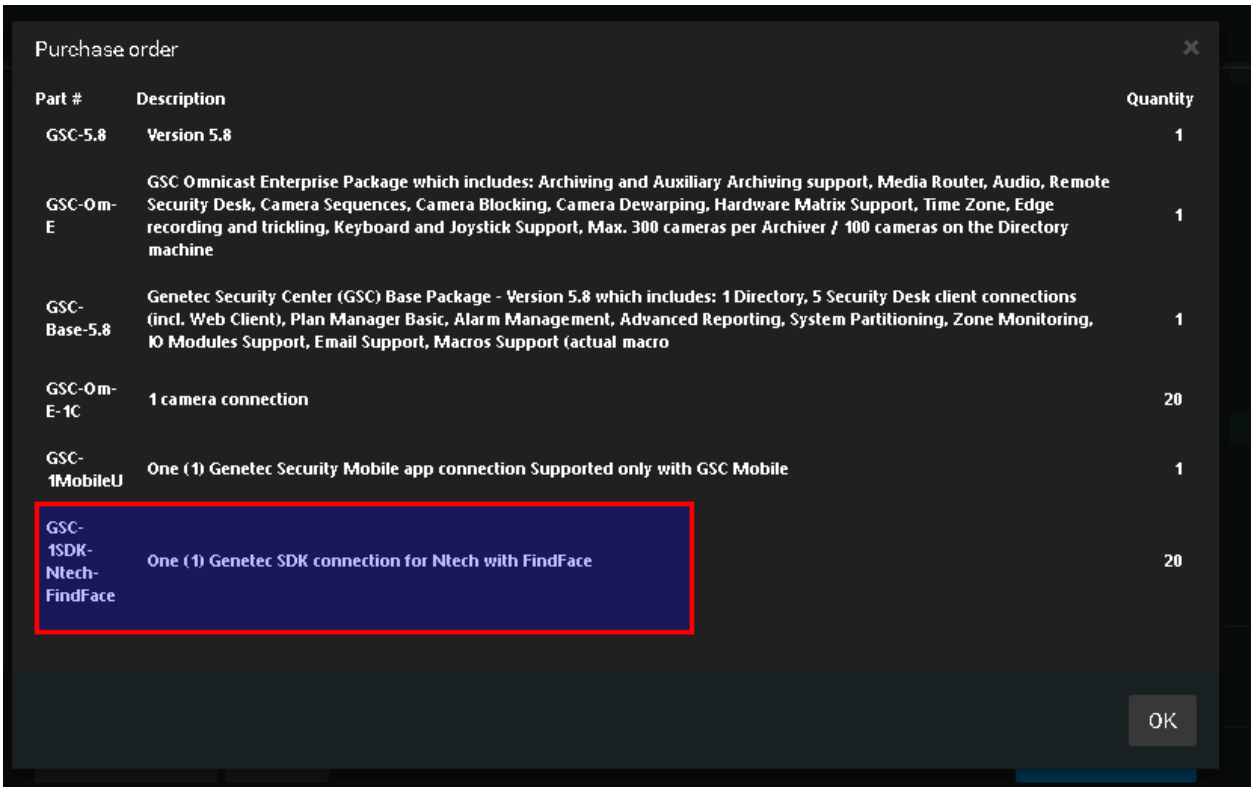
Интеграция с Genetec Security Center реализуется через плагин `findface-genetec`. По умолчанию плагин активен и на панели навигации FindFace Security есть вкладка *Genetec*.

Примечание: Если это не так в FindFace Security, откройте файл конфигурации `findface-security` и проверьте, есть ли в нем активная строка `INSTALLED_APPS.append('ffsecurity_genetec')`. Также убедитесь, что по крайней мере один из следующих параметров задан: `SERVICE_EXTERNAL_ADDRESS` или `EXTERNAL_ADDRESS`.

```
sudo vi /etc/ffsecurity/config.py

...
# SERVICE_EXTERNAL_ADDRESS prioritized for webhooks and genetec
SERVICE_EXTERNAL_ADDRESS = 'http://localhost'
EXTERNAL_ADDRESS = 'http://127.0.0.1'

...
FFSECURITY_UI_CONFIG = {
    "event": {
        "features": {
            "f_gender_class": ["male", "female"],
            "age": {
                "f_age_gte": "",
                "f_age_lte": ""
            },
            "f_emotions_class": ["angry", "disgust", "fear", "happy", "sad", "surprise"],
            "f_glasses_class": ["none", "eye", "sun"],
            "f_beard_class": ["none", "beard"],
            "f_liveness_class": ["real", "fake"],
        }
    }
}

# integration plugins
INSTALLED_APPS.append('ffsecurity_genetec') # remove or comment out this line to disable
```

Перед настройкой интеграции на стороне FindFace Security разверните программное обеспечение Genetec Web SDK и Media Gateway и создайте в Genetec Security Center оповещение `Alarm`, которое будет отображаться при наступлении в FindFace Security события распознавания лица.

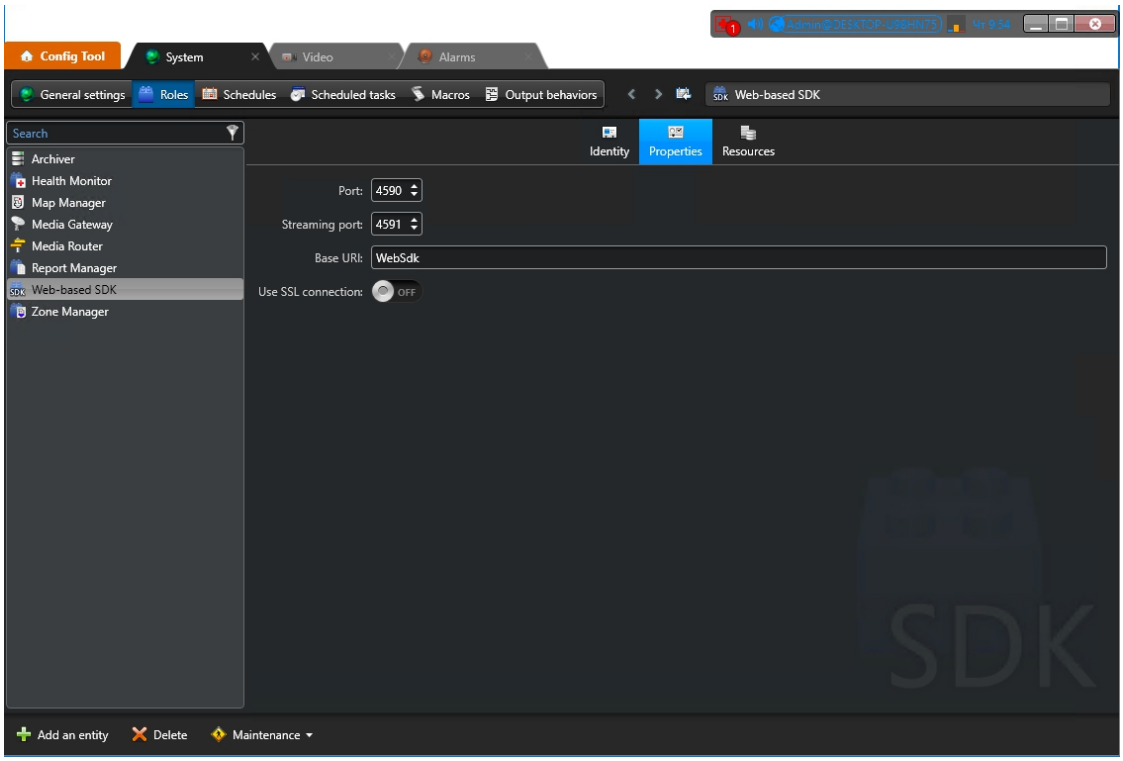
Важно: Для того чтобы интеграция Genetec-FindFace работала, вам также понадобится приобрести соответствующую лицензию у Genetec (номер по каталогу лицензий `GSC-1SDK-Ntech-FindFace`) и активировать ее в Genetec Security Center.



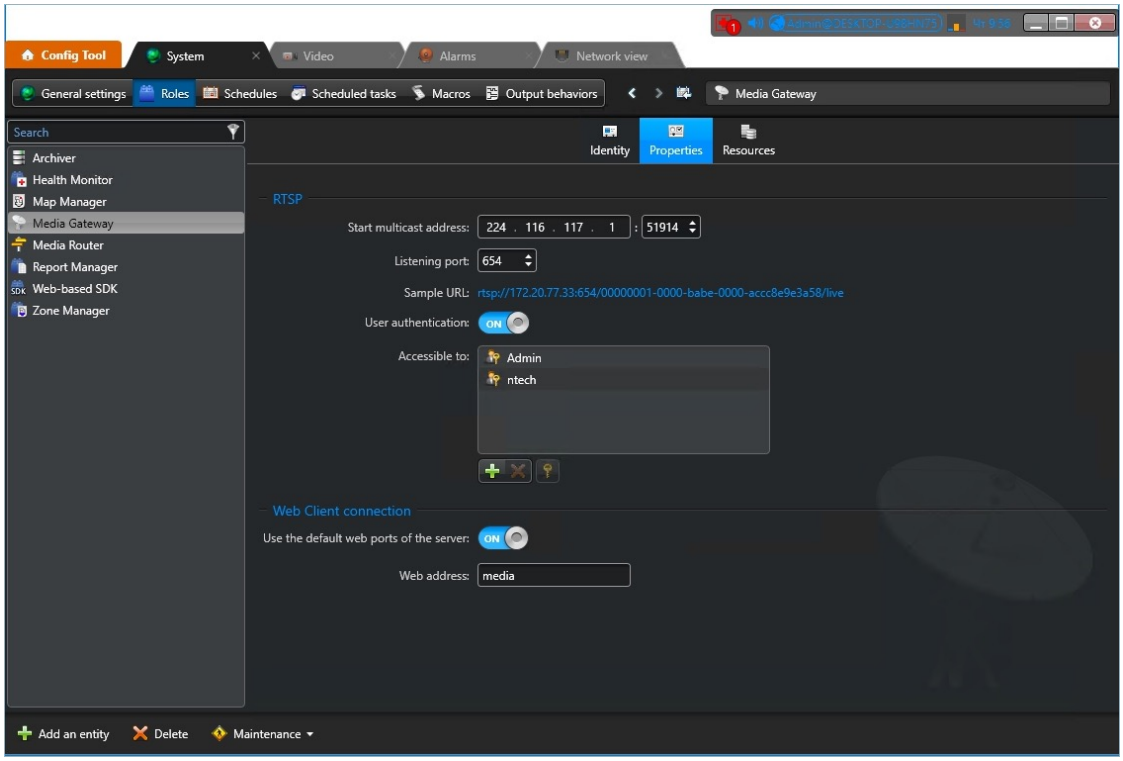
В этой главе:

Настройка Genetec Web SDK и Media Gateway

Для того чтобы развернуть Web SDK, используйте ПО Genetec Config Tool. Детали настройки приведены в официальной справочной документации *Security Center Administrator Guide -> Chapter 52: Role Types -> Web-based SDK configuration tabs.*



Для того чтобы развернуть Media Getaway в Genetec Config Tool, ознакомьтесь с содержанием главы *Security Center Administrator Guide -> Chapter 24: Video Deployment.*

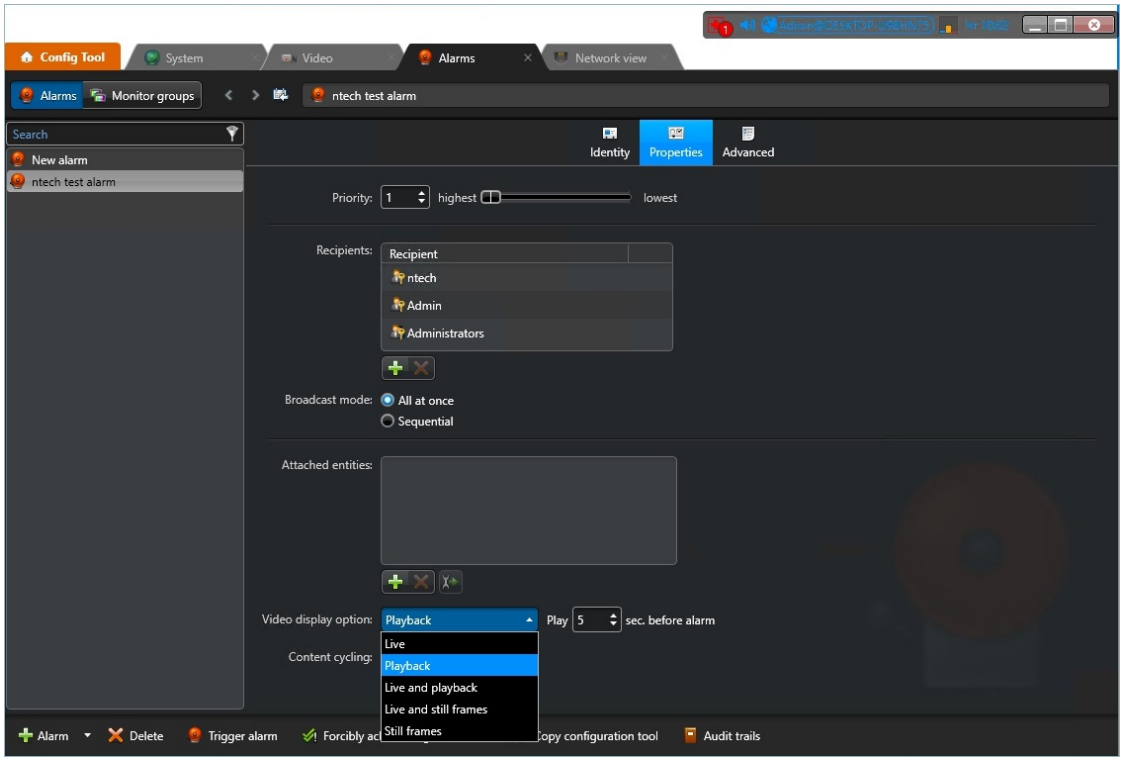


Важно: Убедитесь, что файрвол настроен таким образом, что порты WebSDK и Media Gateway

остаются открытыми.

Создание оповещения в Genetec Security Center

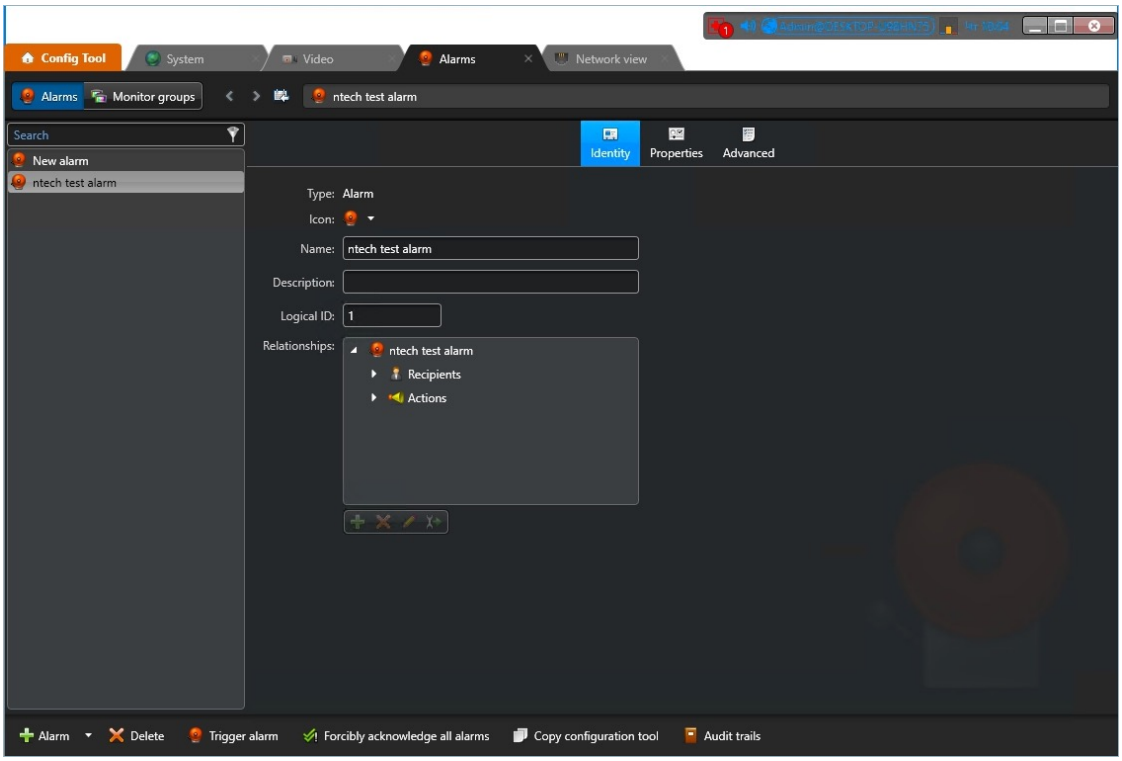
Создайте и настройте новое оповещение `Alarm` в Genetec Config Tool, руководствуясь разделом *Security Center Administrator Guide -> Chapter 48: Alarms -> Creating Alarms*.



Совет: На вкладке *Properties* выберите ту опцию отображения видео *Video display option*, которая в наибольшей степени соответствует вашим нуждам. Доступные опции *Live*, *Playback*, и т. д.

Совет: Для того чтобы активировать операции с оповещением Alarm Procedures и автоповорот видео непосредственно во всплывающем окне оповещения, включите *Content cycling*.

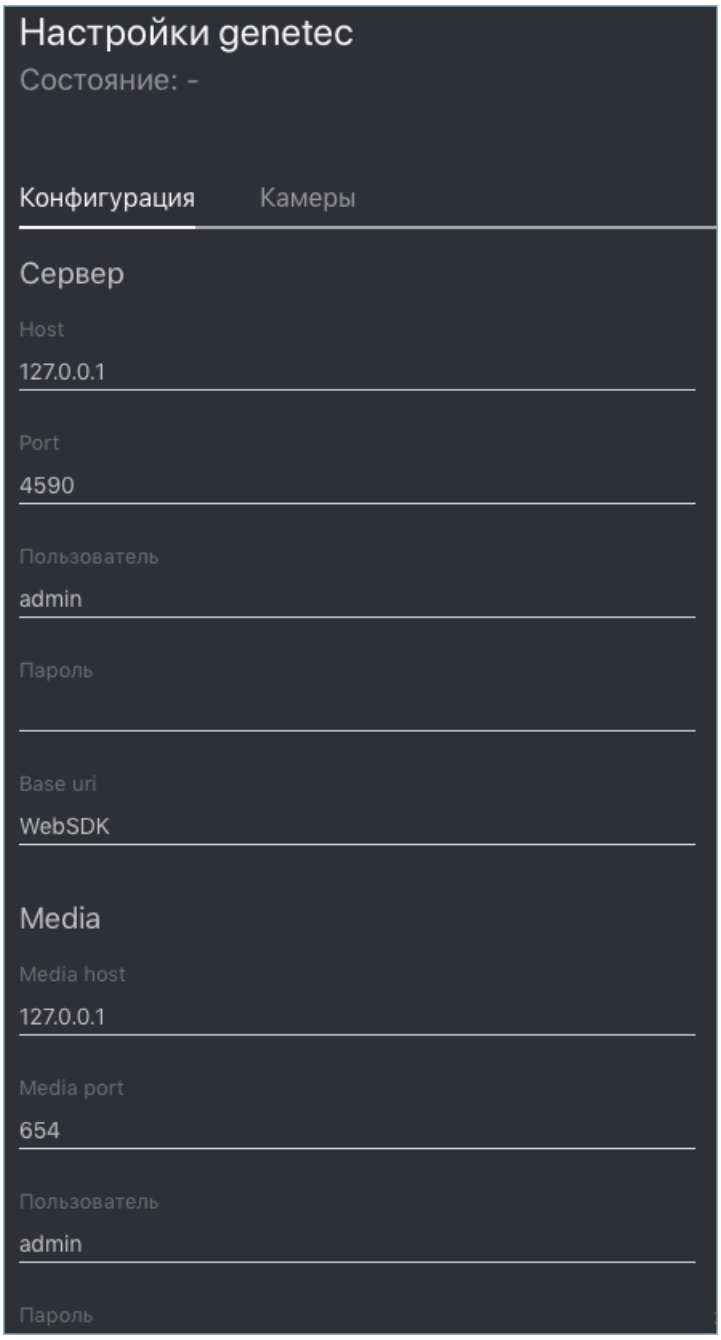
При настройке интеграции на стороне FindFace Security вам потребуется ввести логическое id оповещения, которое задается на вкладке *Identity*.



Настройка точек доступа в FindFace Security

Для того чтобы установить соединение между FindFace Security и Genetec Security Center, выполните следующие действия:

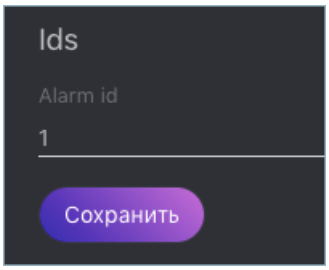
1. Перейдите на вкладку *Настройки*. Выберите *Genetec*.



2. В секциях *Сервер* и *Media*, укажите *настройки* точек доступа Web SDK и Media Gateway.

Важно: Порты WebSDK и Media Gateway должны быть открыты.

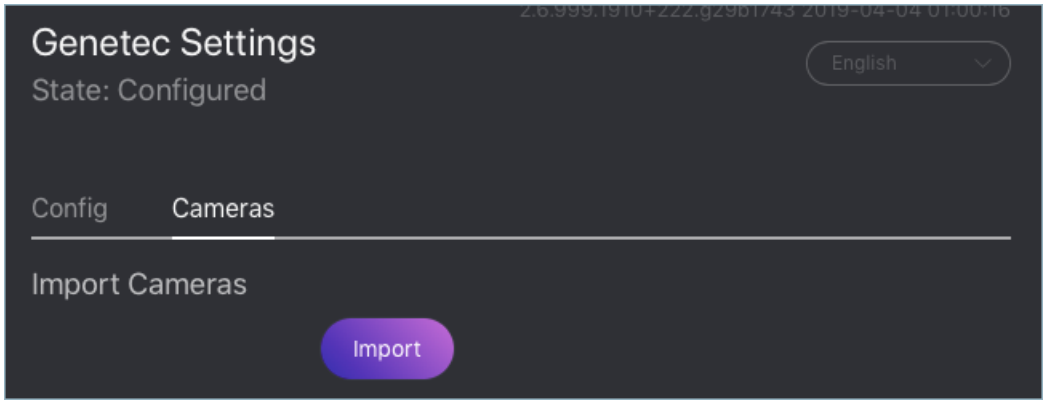
3. В секции guilabel:*Ids* укажите *логический id* оповещения `Alarm`, которое будет отображаться в Genetec Security Center при наступлении события распознавания лица в FindFace Security.



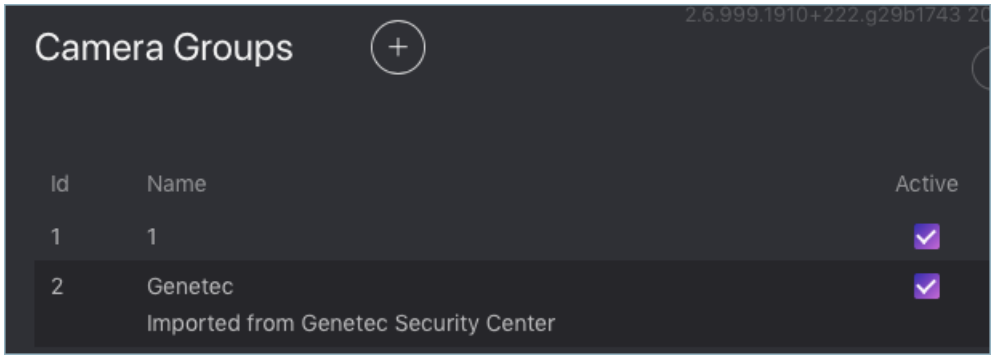
4. Нажмите *Сохранить*. Если соединение с Genetec Security Center успешно установлено, статус будет автоматически изменен на *Сконфигурирован*.

Импорт камер из Genetec Security Center

Как только соединение с Genetec Security Center установлено, можно импортировать камеры. Для этого выберите *Камеры* на вкладке *Genetec* и нажмите *Импорт*.



Данное действие создаст *группу камер* `Genetec`, включающую в себя все камеры из Genetec Security Center.



Для того чтобы посмотреть список камер, на панели навигации FindFace Security перейдите на вкладку *Камеры*. Для того чтобы исключить камеру из распознавания лиц, просто дезактивируйте ее в этом списке.

Создание списков наблюдения и досье в FindFace Security

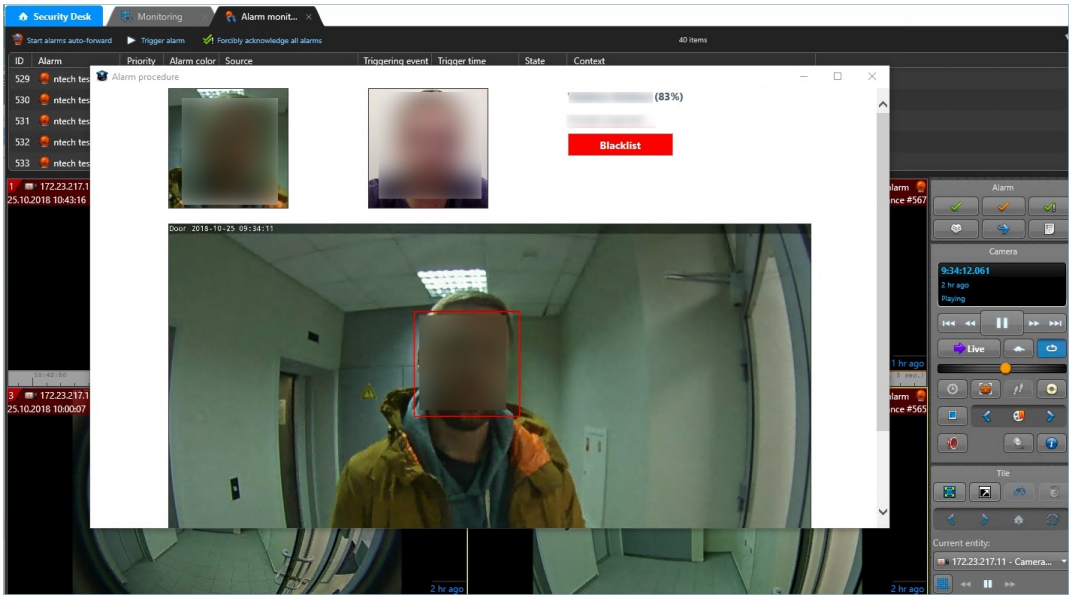
После настройки точек доступа и импорта камер завершите интеграцию, создав *базу данных досье*. После этого оповещения о событиях распознания лиц будут автоматически отправляться в Genetec

Security Center. См. *Оповещения в Genetec Security Center*.

Оповещения в Genetec Security Center

Каждое событие распознавания лица с камеры Genetec, для которого найдено досье, активирует соответствующее оповещение `alarm` в Genetec Security Center. Каждое оповещение, отправленное FindFace Security, связывается с камерой-источником события распознавания лица, поэтому вы можете сразу же просматривать живое или архивное видео в задаче Alarm Monitoring в Genetec Security Desk. FindFace Security также использует операции с оповещением Alarm Procedures для обеспечения пользователя дополнительными данными по событию, такими как:

- обнаруженное на видео лицо
- найденное похожее лицо из базы данных досье
- имя человека и комментарий из досье
- степень схожести лиц (уверенность алгоритма в совпадении)
- название списка наблюдения
- полный кадр



Обработка полученного оповещения о распознавания лица выполняется аналогично другим оповещениям в Genetec Security Center.

3.3.2 Axxon Next

Интеграция FindFace Security с программным комплексом Axxon Next позволяет обрабатывать видеопотоки из системы безопасности на базе Axxon и анализировать их на предмет наличия лиц из досье.

Важно: Один экземпляр FindFace Security может взаимодействовать только с одним сервером Axxon Next.

Интеграция с Axxon Next выполняется с использованием плагина `ffsecurity_axxon`.

Для того чтобы настроить интеграцию с Axxon Next в ОС Ubuntu, выполните следующие действия:

1. Активируйте плагин, добавив в файл конфигурации `/etc/ffsecurity/config.py` строку `INSTALLED_APPS.append('ffsecurity_axxon')`.

```
sudo vi /etc/ffsecurity/config.py

...

# integration plugins
INSTALLED_APPS.append('ffsecurity_axxon') # remove or comment out this line to disable
```

2. В файл конфигурации добавьте секцию `FFSECURITY->AXXON`. Заполните ее так, как показано в примере ниже. В параметре `api` укажите адрес сервера Axxon Next, по которому FindFace Security будет обращаться к API Axxon и за HLS-потоками архива. В параметре `rtsp` укажите общий адрес видеопотоков на сервере Axxon Next.

```
 FFSECURITY = {
'AXXON': {
   'api': 'http://user:password@example.com/',
   'rtsp': 'rtsp://user:password@example.com:554/',
         }
}
```

3. (Опционально). Если в событиях распознавания лиц требуется отображать клипы видео из Axxon Next, отредактируйте секцию `FFSECURITY_UI_CONFIG` так, как показано в примере ниже.

```
FFSECURITY_UI_CONFIG = {
 'dossier': {
   'video': True,
 }
}
```

4. Создайте камеры в FindFace Security (см. *Управление видеокамерами*). При создании камер вам потребуется ввести их URL в формате `axxon:<friendlyNameLong>`, где `friendlyNameLong` - имя камеры на сервере Axxon Next. Данное имя можно посмотреть в интерфейсе Axxon, или через API Axxon, выполнив команду:

```
curl http://user:password@127.0.0.1/video-origins/

{
   "OLOLOE-DEV/DeviceIpint.vhod_1/SourceEndpoint.video:0:0" : {
      "friendlyNameLong" : "vhod_1.Vhod_1",
      "friendlyNameShort" : "Vhod_1",
      "origin" : "OLOLOE-DEV/DeviceIpint.vhod_1/SourceEndpoint.video:0:0",
      "state" : "signal_restored"
   }
}
```

   Для единственной камеры из примера выше URL должен быть задан как `axxon:vhod_1.Vhod_1`.

На этом настройка интеграции будет завершена. Если интеграция настроена корректно, FindFace Security будет выполнять проверку наличия лиц из досье в видеопотоках Axxon Next, а в событиях распознавания лиц будут отображаться клипы видео из Axxon Next (при соответствующих настройках).

3.4 Плагины

В ходе настройки системы вы можете установить собственные правила обработки лиц, обнаруженных на видео. Для этого понадобится написать плагин на Python.

Плагины подключаются через файл конфигурации компонента `findface-facerouter`. Использование плагинов позволяет индивидуально и очень точно выполнить настройку видеодетекции лиц в том, что касается операций с обнаруженными лицами.

3.4.1 Развертывание `findface-facerouter` в FindFace Security

Для развертывания компонента `findface-facerouter` выполните следующие действия:

1. Установите `findface-facerouter` либо из *консольного инсталлятора*, либо из apt-репозитория следующим образом:

```
sudo apt update
sudo apt install -y findface-facerouter
```

2. Откройте файл конфигурации `/etc/findface-facerouter.py`.

```
sudo vi /etc/findface-facerouter.py
```

3. Если компоненты `findface-facerouter` и `findface-sf-api` установлены на разных физических серверах, раскомментируйте параметр `sfapi_url` и укажите в нем IP-адрес сервера `findface-sf-api`.

```
sfapi_url                       = 'http://localhost:18411'
```

4. Откройте файл конфигурации `/etc/ffsecurity/config.py`. В параметре `ROUTER_URL` актуализируйте IP-адрес и порт `findface-facerouter` (по умолчанию порт 18820). IP-адрес указывается внешний или внутренний в зависимости от сети, по которой `findface-video-worker` взаимодействует с `findface-facerouter`.

```
sudo vi /etc/ffsecurity/config.py

...
FFSECURITY = {
   'ROUTER_URL': 'http://172.20.77.58:18820/v0/frame?',
```

5. Откройте файл конфигурации `/etc/findface-video-manager.conf`. В параметре `router_url` укажите IP-адрес и порт сервера `findface-facerouter` для получения лиц, обнаруженных `findface-video-worker`.

```
sudo vi /etc/findface-video-manager.conf

...
router_url: http://127.0.0.1:18820/v0/frame
```

6. Добавьте сервис `findface-facerouter` в автозагрузку Ubuntu и запустите сервис.

```
sudo systemctl enable findface-facerouter.service && sudo systemctl start findface-facerouter.
↪service
```

7. Перезапустите сервис `findface-security`.

```
sudo systemctl restart findface-security.service
```

3.4.2 Настройка `findface-facerouter` на использование плагинов

Важно: Обязательно предварительно *измените* структуру биометрической базы Tarantool в соответствие с правилами обработки, заданными в плагинах.

Важно: Компонент `findface-facerouter` должен быть *установлен и настроен*.

Для настройки компонента `findface-facerouter` на использование плагинов выполните следующие действия:

1. Поместите плагин в каталог по вашему выбору. Для хранения нескольких плагинов можно использовать разные каталоги.
2. Откройте файл конфигурации `findface-facerouter`.

   ```
   sudo vi /etc/findface-facerouter.py
   ```

   Предупреждение: Содержимое `findface-facerouter.py` должно представлять собой синтаксически корректный код Python.

3. Раскомментируйте параметр `plugins_dirs` и через запятую укажите список каталогов с плагинами.

   ```
   plugins_dirs                    = '/etc/findface/plugins/video, /etc/findface/plugins/html'
   ```

4. Сохраните файл конфигурации.

3.4.3 Принципы написания плагина

В этом разделе:

Архитектура плагина

После того как компонент `findface-video-worker` обнаруживает лицо, он отправляет его в компонент `findface-facerouter` в виде HTTP API запроса. Для обработки запроса каждый плагин должен экспортировать функцию `activate(app, ctx, plugin_name, plugin_source)`.

Параметры функции `activate`:

- `app`: сущность `tornado.web.Application` компонента `findface-facerouter`.
- `ctx`: контекст, передаваемый плагину при активации.
- `plugin_name`: имя активируемого плагина.
- `plugin_source`: объект источника, из которого загружается плагин.

При активации плагину передается следующий контекст:

1. `request.ctx.sfapi`: настроенный экземпляр `ntech.sfapi_client.Client`, к которому можно обращаться напрямую для обработки результата видеодетекции (создание новой галереи, добавление лица в галерею и т. д.).
2. `plugins`: `OrderedDict` со всеми плагинами (`key`: имя плагина, `value`: результат, возвращенный функцией `activate`).
3. `idgen`: генератор id, который может вызываться как `ctx.idgen()`.

Функция `activate(app, ctx, plugin_name, plugin_source)` должна вернуть объект со следующими методами:

1. `preprocess`,
2. `process`,
3. `shutdown` (опционально).

Метод `preprocess`

В данном методе плагин решает, интересует ли его полученное лицо, и если да, возвращает кортеж или список, содержащий одну или несколько строк: `'facen'`, `'gender'`, `'age'`, `'emotions'`, что соответственно означает, что нужно извлечь вектор признаков, распознать пол, возраст и/или эмоции. Если возвращенные кортеж или список непусты, компонент `findface-facerouter` перенаправляет обнаруженное лицо в компонент `findface-sf-api` в запросе `/detect POST` с соответствующими параметрами в query string (`facen=on`, `gender=on`, `age=on`, `emotions=on`).

Синтаксис базового метода `preprocess`, от которого следует наследоваться (см. класс `Plugin`):

`preprocess`(*self, request: FrHTTPRequest, labels: typing.Mapping[str, str]*) → typing.Tuple[str]

Параметры

- `FrHTTPRequest` (*tornado.httpserver.HTTPRequest*) – HTTP API запрос, который включает в себя дополнительный аргумент `params`
- `labels` (*dictionary*) – пользовательский набор меток кадра, который задается в параметрах задания для компонента `findface-video-worker` и затем присваивается кадру

Аргумент `params` FrHTTPRequest содержит следующие поля:

Параметры

- `photo` (*bytes*) – кадр с лицом в формате JPEG
- `face0` (*bytes*) – нормализованное изображение лица

- `bbox` (list of integers [[x1,y1,x2,y2]], where `x1`: x coordinate of the top-left corner, `y1`: y coordinate of the top-left corner, `x2`: x coordinate of the bottom-right corner, `y2`: y coordinate of the bottom-right corner) – координаты рамки с лицом в кадре
- `cam_id` (*string*) – id видеокамеры
- `timestamp` (*datetime.datetime*) – временная метка кадра
- `detectorParams` (*dictionary*) – словарь со служебно-отладочной информацией от детектора
- `bs_type` (*string*) – режим поиска лучшего кадра. Доступные опции: overall (буферный режим: на сервер отправляется кадр, который был лучшим за весь период нахождения лица в поле зрения видеокамеры), realtime (режим реального времени: на сервер отправляются кадры, считающиеся лучшими в последовательных интервалах).
- `labels` (*dictionary*) – (дублирует params.labels) пользовательский набор меток кадра, который задается в параметрах задания для компонента `findface-video-worker` и затем присваивается кадру

Решение об обработке лица принимается на основании данных из `request.params`, в том числе пользовательского набора меток, а также из любых других соображений.

Метод `process`

Данный метод вызывается, если метод `preprocess` вернул непустой кортеж или список ('facen', 'gender', 'age', 'emotions'). После того, как компонент `findface-sf-api` вернул ответ с результатом детекции (см. запрос `/detect POST`) со всеми запрошенными параметрами лица, компонент `findface-facerouter` вызывает метод `` `process `` плагина для выполнения собственно обработки лица.

Для выполнения обработки лица плагин использует `request.ctx.sfapi`.

Синтаксис базового метода `process`, от которого следует наследоваться (см. класс `Plugin`):

`process`(*self, request: FrHTTPRequest, photo: bytes, bbox: typing.List[int], event_id: int, detection: DetectFace*)

Метод `shutdown`

Данный метод вызывается только перед завершением работы компонента `findface-facerouter`.

Синтаксис базового метода `shutdown`, от которого следует наследоваться (см. класс `Plugin`):

`shutdown`(*self*)

3.4.4 Классы и методы

В этом разделе:

- *Фильтры для поиска по базе данных*
- *Отображение сообщений об ошибках*

Базовые классы

class facerouter.plugin.Plugin

Данный класс предоставляет базовые методы для написания плагина, описанные в разделе *Принципы написания плагина*. Пользовательский класс, выполняющий роль оболочки для плагина, должен наследовать от класса `Plugin`.

`preprocess`(*self, request: FrHTTPRequest, labels: typing.Mapping[str, str]*) → typing.Tuple[str]

Возвращает кортеж, включающий в себя одну или несколько строк: `'facen'`, `'gender'`, `'age'`, `'emotions'`, что соответственно означает, что компонент `findface-facerouter` должен запросить у компонента `findface-sf-api` извлечение вектора признаков, распознать пол, возраст и/или эмоции.

Параметры

- `FrHTTPRequest` (*tornado.httpserver.HTTPRequest*) – HTTP API запрос, который включает в себя дополнительный аргумент `params`
- `labels` (*dictionary*) – пользовательские метки из `request.params`

Результат одна или несколько строк `'facen'`, `'gender'`, `'age'`, `'emotions'`

Тип результата tuple

Аргумент `params` FrHTTPRequest содержит следующие поля:

Параметры

- `photo` (*bytes*) – кадр с лицом в формате JPEG
- `face0` (*bytes*) – нормализованное изображение лица
- `bbox` (list of integers [[x1,y1,x2,y2]], where `x1`: x coordinate of the top-left corner, `y1`: y coordinate of the top-left corner, `x2`: x coordinate of the bottom-right corner, `y2`: y coordinate of the bottom-right corner) – координаты рамки с лицом в кадре
- `cam_id` (*string*) – id камеры
- `timestamp` (*datetime.datetime*) – временная метка кадра
- `detectorParams` (*dictionary*) – словарь со служебно-отладочной информацией от детектора
- `bs_type` (*string*) – режим поиска лучшего кадра. Доступные опции: overall (буферный режим: на сервер отправляется кадр, который был лучшим за весь период нахождения лица в поле зрения видеокамеры), realtime (режим реального времени: на сервер отправляются кадры, считающиеся лучшими в последовательных интервалах).
- `labels` (*dictionary*) – (дублирует params.labels) пользовательский набор меток кадра, который задается в параметрах задания для компонента `findface-video-worker` и затем присваивается кадру

`process`(*self, request: FrHTTPRequest, photo: bytes, bbox: typing.List[int], event_id: int, detection: DetectFace*)

Принимает атрибуты обнаруженного лица.

Параметры

- `request` (*tornado.httpserver.HTTPRequest*) – HTTP API-запрос от `findface-video-worker`
- `photo` (*bytes*) – кадр с лицом в формате JPEG из `request.params`
- `bbox` (list of integers [[x1,y1,x2,y2]], where `x1`: x coordinate of the top-left corner, `y1`: y coordinate of the top-left corner, `x2`: x coordinate of the bottom-right corner, `y2`: y coordinate of the bottom-right corner) – координаты рамки с лицом в кадре из `request.params`
- `event_id` (*uint64*) – id обнаруженного на видео лица (автоматически задается компонентом `findface-facerouter` при получении лица от `findface-video-worker`). Может использоваться в качестве пользовательского идентификатора лица в базе данных.
- `detection` (`objects.DetectFace`) – результат детекции, полученный от компонента `findface-sf-api`, включающий в себя запрошенные параметры лица, такие как вектор признаков, пол, возраст, эмоции.

Результат н/п

Тип результата n/a

`shutdown(self)`

Данный метод вызывается только перед завершением работы компонента `findface-facerouter`.

Параметры н/п

Результат н/п

Классы объектов

`class objects.BBox`

Представляет собой координаты рамки с лицом.

`class objects.DetectFace`

Представляет собой результат детекции со следующими полями:

Параметры

- `id` (*string*) – id результата детекции в memcached
- `bbox` (*objects.Bbox*) – координаты рамки с лицом
- `features` (*dictionary*) – информация о поле (gender), возрасте (age) и эмоциях (emotions) (опционально)

`class objects.DetectResponse`

Представляет собой список объектов `objects.DetectionFace` с дополнительным полем `orientation`, содержащим информацию об ориентации EXIF лица.

Параметры `orientation` (*EXIF orientation*) – ориентация обнаруженного лица

`class objects.FaceId(`*namedtuple('FaceId', ('gallery', 'face'))*`)`

Представляет собой объект пользовательского идентификатора лица в галерее.

Параметры

- `gallery` (*string*) – имя галереи
- `face` (*integer*) – пользовательский идентификатор лица в галерее

class objects.Face
Представляет собой результат поиска лица в базе данных по биометрическому образцу

Параметры

- id (objects.FaceId) – объект Faceid
- features (*dictionary*) – информация о поле, возрасте и эмоциях
- meta (*dictionary*) – метаданные лица
- confidence (*float*) – степень схожести лица с заданным биометрическим образцом

class objects.ListResponse
Представляет собой список объектов objects.Face (т. е. список результатов поиска по биометрическому образцу) с дополнительным полем next_page, содержащим информацию о следующей странице с результатами.

Параметры next_page (*string*) – курсор следующей страницы с результатами поиска

Обнаружение лица и работа с галереями

class ntech.sfapi_client.client.Client
Предоставляет базовые методы для обнаружения лиц на изображении и работы с галереями.

detect(*self*, ***, *url=None*, *image=None*, *facen=False*, *gender=False*, *age=False*, *emotions=False*, *return_facen=False*, *autorotate=False*, *detector: str = None*, *timeout=None*) → DetectResponse
Обнаруживает лица на изображении и возвращает обнаруженные лица.

Параметры

- url (*URL*) – URL изображения, если вы передаете общедоступное изображение из интернета
- image (*bytes*) – файл PNG/JPG/WEBP, если вы передаете изображение в виде файла
- facen (*boolean*) – извлечь вектор признаков из обнаруженного лица. Для сохранения результата детекции в memcached передайте facen=True.
- gender (*boolean*) – извлечь и вернуть информацию о поле
- age (*boolean*) – извлечь и вернуть информацию о возрасте
- emotions (*boolean*) – извлечь и вернуть информацию об эмоциях
- return_facen (*boolean*) – вернуть вектор признаков в результате работы метода
- autorotate (*boolean*) – автоматически повернуть изображение в 4-х разных ориентациях для обнаружения лиц в каждой их них. Пересекающиеся направления с IOU > 0.5 будут объединены
- detector (*boolean*) – может принимать значение nnd или normalized. Детектор normalized используется для обработки нормализованных изображений, например, поступающих от видеодетектора лиц
- timeout (*number*) – максимальное время ожидания ответа от ядра FindFace в секундах (если none, используется время ожидания ответа, заданное по умолчанию)

Результат Результат детекции

Тип результата Объект DetectorResponse

`gallery(self, name)`
Возвращает объект `sfapi_client.Gallery` для последующей с ним работы (например, получения списка лиц).

Параметры `name` (*string*) – имя галереи

Результат объект типа «галерея»

Тип результата sfapi_client.Gallery

`list_galleries(self, timeout=None):`
Возвращает список галерей.

Параметры `timeout` (*number*) – максимальное время ожидания ответа от ядра FindFace в секундах (если `none`, используется время ожидания ответа, заданное по умолчанию)

Результат список галерей со свойствами `name` (имя галереи, строка) и `number` (количество лиц в галереи, число)

Тип результата list of GalleryListItem

`class ntech.sfapi_client.gallery.Gallery`
Предоставляет методы для работы с галереями и лицами в них.

`list(self, *, filters: typing.Iterable[filters.Filter] = None, limit: int = 1000, sort: str = '', page=None, ignore_errors=False, timeout=None)` → ListResponse
Возвращает объект типа список с лицами из галереи, удовлетворяющими заданным фильтрам. Возвращаемый объект типа список содержит дополнительное свойство `next_page`, которое может использоваться как значение параметра `page` в последующих запросах.

Параметры

- `filters` (`sfapi_client.filters.Filter`) – список фильтров
- `limit` (*integer*) – максимальное количество лиц в ответе
- `sort` (*string*) – метод сортировки лиц. Возможные значения: `id` (по возрастанию id), `-id` (по убыванию id), `-confidence` (по убыванию степени схожести лиц). Сортировка по id возможна только при отключенном фильтре `facen`, который задает вектор признаков для поиска в базе данных (т.н. идентификация лица). Наоборот, сортировка по степени схожести лиц (`confidence`) возможна только при включенном фильтре `facen`. По умолчанию метод использует сортировку по возрастанию id (вектор признаков не задан) и по убыванию степени схожести лиц (вектор признаков задан).
- `page` – вернуть результаты, начиная с указанной страницы. Номер следующей страницы с результатами возвращается в ответе сервера в виде `next_page`.
- `ignore_errors` (*boolean*) – Игнорировать ошибку обращения к базе данных, если поиск по галерее выполняется, когда некоторые сервера базы данных недоступны. В этом случае поиск будет выполнен с использованием доступных серверов.
- `timeout` (*number*) – максимальное время ожидания ответа от ядра FindFace в секундах (если `none`, используется время ожидания ответа, заданное по умолчанию)

Результат список с лицами из галереи, удовлетворяющими заданным фильтрам.

Тип результата ListResponse object

add(*self, new_id: typing.Union[int, typing.Callable], source: typing.Union[DetectFace, Face, str], *, meta: typing.Dict[str, typing.Union[int, str, typing.List[str]]] = None, regenerate_attempts=None, timeout=None*) → Face
Создает лицо в галерее.

Параметры

- `new_id` (*integer or callable*) – пользовательский идентификатор лица в базе данных. Может быть (async) callable, который возвращает id. Для генерации id может использоваться функция `ctx.idgen()` из контекста.
- `source` (*sfapi_client.DetectFace, sfapi_client.Face, sfapi_client.FaceId, or string*) – источник, из которого лицо добавляется в базу данных, может представлять собой лицо в базе данных или результат детекции.
- `meta` (*dictionary*) – метаданные лица. Ключи могут быть строками, а значения – целыми числами, строками или списками строк. Перед добавлением метаданных в базе данных должна быть создана соответствующая структура.
- `regenerate_attempts` – количество попыток генерации уникального id функцией `ctx.idgen()`, если new_id callable.
- `timeout` (*number*) – максимальное время ожидания ответа от ядра FindFace в секундах (если `none`, используется время ожидания ответа, заданное по умолчанию)

Результат представление созданного лица

Тип результата Face object

delete(*self, face: typing.Union[Face, int], timeout=None*) → None
Удаляет лицо из галереи.

Параметры

- `face` (*sfapi_client.Face, sfapi_client.FaceId or id in integer*) – лицо, которое нужно удалить из базы данных
- `timeout` (*number*) – максимальное время ожидания ответа от ядра FindFace в секундах (если `none`, используется время ожидания ответа, заданное по умолчанию)

Результат None

get(*self, face: typing.Union[Face, int], timeout=None*) → Face
Возвращает лицо из галереи.

Параметры

- `face` (*sfapi_client.Face, sfapi_client.FaceId or id in integer*) – лицо, которое нужно извлечь из базы данных
- `timeout` (*number*) – максимальное время ожидания ответа от ядра FindFace в секундах (если `none`, используется время ожидания ответа, заданное по умолчанию)

Результат представление лица

Тип результата Face object

`create`(*self*, *timeout=None*) → None
Создает галерею в `findface-sf-api` в виде объекта `sfapi_client.Gallery`. Объект `sfapi_client.Gallery` представляет собой промежуточный объект, и для работы с ним не требуется наличии галереи на сервере.

Параметры `timeout` (*number*) – максимальное время ожидания ответа от ядра FindFace в секундах (если `none`, используется время ожидания ответа, заданное по умолчанию)

Результат None

`drop`(*self*, *timeout=None*) → None:
Удаляет галерею из `findface-sf-api`.

Параметры `timeout` (*number*) – максимальное время ожидания ответа от ядра FindFace в секундах (если `none`, используется время ожидания ответа, заданное по умолчанию)

Результат None

`update`(*self, face: typing.Union[Face, str], *, meta: typing.Dict[str, typing.Union[int, str, typing.List[str]]] = None, timeout=None*) → Face
Редактирует метаданные лица в галерее.

Параметры

- `face` (*sfapi_client.Face , sfapi_client.FaceId or id in integer*) – лицо, метаданные которого нужно заменить в базе данных
- `meta` (*dictionary*) – метаданные лица, которые нужно заменить. Ключи могут быть строками, а значения – целыми числами, строками или списками строк. Если поля meta не передаются или null, они не изменяются в базе данных.
- `timeout` (*number*) – максимальное время ожидания ответа от ядра FindFace в секундах (если `none`, используется время ожидания ответа, заданное по умолчанию)

Результат представление измененного лица

Тип результата Face object

Фильтры для поиска по базе данных

`class ntech.sfapi_client.filters.Filter`
Общий класс. Представляет собой сводный список фильтров (со значениями), которые должны быть применены к содержимому галереи.

`serialize`(*self*)
Метод для передачи списка фильтров со значениями в компонент `findface-sf-api`.

Результат имена и значения заданных фильтров

Тип результата tuple („filtername“, [«value1», «value2»])

`class ntech.sfapi_client.filters.Id`
Предоставляет методы для фильтрации содержимого галереи по id. Для использования фильтра нужно напрямую вызвать соответствующий classmethod, не создавая экземпляр класса.

`classmethod lte`(*cls*, *value: int*) → Filter
Фильтр LTE. Выбрать все лица с `id`, меньшим или равным указанному.

Параметры `value` (*integer*) – значение id

Результат имя фильтра (LTE) и его значение.

Тип результата объект класса Filter.

Пример: `Id.lte(1234)` выбирает лица с id, меньшим или равным `1234`.

`classmethod gte(`*cls, value: int*`)` → Filter
Фильтр GTE. Выбрать все лица с `id`, большим или равным указанному.

Параметры `value` (*integer*) – значение id

Результат имя фильтра (GTE) и его значение.

Тип результата объект класса Filter.

Пример: `Id.gte(1234)` выбирает лица с id, большим или равным `1234`.

`classmethod oneof(`*cls, *value: typing.Union[int]*`)` → Filter
Фильтр IN. Выбрать лица с id из заданной последовательности.

Параметры `value` (*list of integers*) – список значений id

Результат имя фильтра (IN) и его значение.

Тип результата объект класса Filter.

Пример: `Id.oneof(1234, 5678)` выбирает лицо с id `1234` и/или `5678`.

`class ntech.sfapi_client.filters.Meta`
Предоставляет методы для фильтрации содержимого галереи по метаданным. Для использования фильтра нужно напрямую вызвать соответствующий метод, не создавая экземпляр класса.

`classmethod lte(`*self, value: typing.Union[str, int]*`)` → Filter
Фильтр LTE. Выбрать все лица, у которых определенная строка в метаданных меньше или равна указанному значению.

Параметры `value` (*string or integer*) – значение строки с метаданными

Результат имя фильтра (LTE) и его значение.

Тип результата объект класса Filter.

Пример: `Meta('foo').lte(1234)` выбирает лица с мета-строкой `foo`, имеющей значение меньшее или равное `1234`.

`classmethod gte(`*self, value: typing.Union[str, int]*`)` → Filter
Фильтр GTE. Выбрать все лица, у которых определенная строка в метаданных больше или равна указанному значению.

Параметры `value` (*string or integer*) – значение строки с метаданными

Результат имя фильтра (GTE) и его значение.

Тип результата объект класса Filter.

Пример: `Meta('foo').gte(1234)` выбирает лица с мета-строкой `foo`, имеющей значение большее или равное `1234`.

`classmethod oneof(`*self, *value: typing.Union[str, int]*`)` → Filter
Фильтр IN. Выбрать все лица, у которых определенная строка в метаданных совпадает с одним из значений из заданной последовательности.

Параметры `value` (*list of strings or integers*) – список строк с метаданными

Результат имя фильтра (IN) и его значение.

Тип результата объект класса Filter.

Пример: `Meta.oneof(1234, 5678)` выбирает лица с мета-строкой, имеющей значение `1234` и/или `5678`.

`classmethod subset(self, *value: str)` → Filter

Фильтр SUBSET. Выбрать все лица, у которых определенная строка содержит все значения из указанной последовательности.

Параметры `value` (*list of strings or integers*) – список строк с метаданными

Результат имя фильтра (SUBSET) и его значение.

Тип результата объект класса Filter.

Пример: `Meta('foo').subset("male", "angry")` выбирает лица с мета-строкой `foo`, содержащей все значения из последовательности ["male", "angry"].

`class ntech.sfapi_client.filters.Detection(Filter)`

Предоставляет метод для идентификации (поиска похожих лиц в базе данных) обнаруженного лица.

`__init__(self, id: typing.Union[str, objects.DetectFace], threshold: float)`

Параметры

- `id` (`objects.DetectFace` or temporary face id in memcached returned by sfapi_client.Client.detect(), string) – лицо (результат детекции), которое нужно найти в базе данных
- `threshold` (*float*) – минимальная степень схожести лиц от 0 до 1

Пример: `Detection(det1, 0.77)` выбирает лица, похожие на результат детекции `det1` со степенью схожести, большей или равной `0.77`.

`class ntech.sfapi_client.filters.Face(Filter)`

Предоставляет метод для поиска лиц в базе данных, похожих на лицо из галереи.

`__init__(self, id: typing.Union[str, objects.Face], threshold: float)`

Параметры

- `id` (`objects.Face`, `objects.FaceId` or custom face id in the gallery, string) – лицо из базы данных, которое нужно найти
- `threshold` (*float*) – минимальная степень схожести лиц от 0 до 1

Пример: `Detection(FaceId("gal1", 1234), 0.77)` выбирает лица, похожие на лицо с пользовательским идентификатором face `1234` из галереи `gal1` со степенью схожести, большей или равной `0.77`.

Пример использования нескольких фильтров

```
filters=[filters.Id.gte(123456), filters.Meta('age').gte(45), filters.Meta('camera').oneof('abc',
↪'def')]
```

Отображение сообщений об ошибках

class sfapi_client.SFApiRemoteError
Данное сообщение об ошибке появляется, если ошибка произошла по причине, отличной от сетевого сбоя.

Сообщение об ошибке содержит как минимум два поля:

- `code` — это код ошибки в виде `CAPS_AND_UNDERSCORES`, который может быть использован для автоматического преобразования.
- `reason` — это описание ошибки, предназначенное для прочтения человеком.

Полный список ошибок

Код ошибки	Описание
`UNKNOWN_ERROR`	Ошибка неизвестного происхождения.
`BAD_PARAM`	Запрос может быть прочитан, однако некоторые параметры метода недействительны. Данный тип ответа содержит дополнительные атрибуты `param` и `value` для описания ошибочных параметров.
`CONFLICT`	Конфликт.
`EXTRACTION_ERROR`	Ошибка при извлечении из лица вектора признаков.
`LICENSE_ERROR`	Конфигурация системы не соответствует лицензии.
`MALFORMED_REQUEST`	Запрос неправильно сформирован и не может быть прочитан.
`OVER_CAPACITY`	Превышен размер очередей в компоненте `findface-extraction-api`.
`SOURCE_NOT_FOUND`	В параметре `from` задано несуществующее лицо.
`SOURCE_GALLERY_NOT_FOUND`	В параметре `from` задана несуществующая галерея.
`STORAGE_ERROR`	Биометрическая база данных недоступна.
`CACHE_ERROR`	Хранилище memcached недоступно.
`NOT_FOUND`	Подходящие лица не найдены.
`NOT_IMPLEMENTED`	Функционал не реализован.
`GALLERY_NOT_FOUND`	Подходящие галереи не найдены.

class sfapi_client.SFApiMalformedResponseError
Это сообщение об ошибке появляется, если ошибка произошла из-за сбоя в сети, или если Клиент не смог прочитать API-ответ от `findface-sf-api`.

3.4.5 Пример

Следующий пример иллюстрирует основы написания плагина и использование классов и методов. Данный плагин запрашивает атрибуты лица у `findface-sf-api`, а затем отправляет запрос в `<FFSEC_URL>/video-detector/process` для создания события с данными, полученными из `findface-sf-api`.

Вы можете найти данный плагин в каталоге `/opt/ffsecurity/fr_plugin/ffsec_fr_plugin.py`. *Настройте* систему на его использование и попробуйте.

Важно: Убедитесь, что в переменной `FFSEC_URL` указаны актуальный IP-адрес и порт сервера `findface-security`.

```
import datetime
import logging
import aiohttp
from dateutil.tz import tzutc
from facerouter.plugin import Plugin
from ntech import sfapi_client
from ntech.asyncio_utils import wrap_future
from ntech.asyncio_utils.noop_cookie import NoopCookieJar
from ntech.tornado_utils import asyncio_to_tornado
# change this if your ffsecurity is located on another host or listens on a non-default port
FFSEC_URL = 'http://127.0.0.1:8002'
logger = logging.getLogger(__name__)
class FFSecurityPlugin(Plugin):
    def __init__(self, ctx, ffsec_url):
        super().__init__(ctx)
        self.ffsec_url = ffsec_url.rstrip('/')
        self.session = aiohttp.ClientSession(cookie_jar=NoopCookieJar())
        self.future_wrapper = asyncio_to_tornado
    def deactivate(self, *args):
        self.session.close()
    def request_headers(self, request):
        return {
            "Authorization": request.headers['Authorization'],
            'X-Request-ID': request.request_id,
        }
    @wrap_futures
    async def preprocess(self, request, labels):
        # somewhat hacky way to pass data between preprocess and process:
        request.ffsec_reception_timestamp = datetime.datetime.now(tzutc())
        headers = self.request_headers(request)
        async with self.session.post(self.ffsec_url + '/video-detector/preprocess',␣
↪headers=headers) as resp:
            resp.raise_for_status()
            resp_json = await resp.json()
            logger.debug("request_id=%r preprocess: ffsecurity response: %r", request.request_id,␣
↪resp_json)
            plugin_wants = resp_json['plugin_wants']
            request.ffsec_plugin_wants = plugin_wants
            logger.info("request_id=%r preprocess: ffsecurity requested features: %r", request.
↪request_id, plugin_wants)
        return plugin_wants
    @wrap_futures
    async def process(self, request, photo, bbox, event_id, detection: sfapi_client.DetectFace):
        headers = self.request_headers(request)
        with aiohttp.MultipartWriter('form-data') as mpwriter:
            part = aiohttp.payload.BytesPayload(request.params.photo)
            part.set_content_disposition('form-data', name='photo', filename='photo.jpg')
            mpwriter.append(part)
            part = aiohttp.payload.BytesPayload(b'')
            part.set_content_disposition('form-data', name='face0', filename='norm.png')
            mpwriter.append(part)
            part = aiohttp.payload.JsonPayload(request.params.detectorParams)
            part.set_content_disposition('form-data', name='detectorParams')
            mpwriter.append(part)
            part = aiohttp.payload.JsonPayload([list(bbox)])
            part.set_content_disposition('form-data', name='bbox')
            mpwriter.append(part)
```

(continues on next page)

(продолжение с предыдущей страницы)

```
            part = aiohttp.payload.StringPayload(request.params.cam_id)
            part.set_content_disposition('form-data', name='cam_id')
            mpwriter.append(part)
            part = aiohttp.payload.StringPayload(request.params.timestamp.isoformat())
            part.set_content_disposition('form-data', name='timestamp')
            mpwriter.append(part)
            part = aiohttp.payload.StringPayload(request.ffsec_reception_timestamp.isoformat())
            part.set_content_disposition('form-data', name='reception_timestamp')
            mpwriter.append(part)
            part = aiohttp.payload.JsonPayload(request.ffsec_plugin_wants)
            part.set_content_disposition('form-data', name='plugin_wants')
            mpwriter.append(part)
            if request.params.bs_type is not None:
                part = aiohttp.payload.StringPayload(request.params.bs_type)
                part.set_content_disposition('form-data', name='bs_type')
                mpwriter.append(part)
            part = aiohttp.payload.JsonPayload({
                'id': getattr(detection, 'id', None),
                'features': detection.features,
                'bbox': detection.bbox._asdict(),
                'facen': getattr(detection, 'facen', None),
                'attributes': detection.attributes,
            })
            part.set_content_disposition('form-data', name='detection')
            mpwriter.append(part)
            async with self.session.post(
                    self.ffsec_url + '/video-detector/process',
                    data=mpwriter,
                    headers=headers
            ) as resp:
                await resp.read()
                resp.raise_for_status()
        logger.info("request_id=%r process: ffsecurity accepted event", request.request_id)
async def activate(app, ctx, plugin_name, plugin_source):
    plugin = FFSecurityPlugin(ctx=ctx, ffsec_url=FFSEC_URL)
    return plugin
```

Содержание модулей Python

f

n

o

Алфавитный указатель

Символы

A

B

C

D

F

G

I

L

M

N

O

P

S

U