FindFace Enterprise Server

Выпуск 4.0.3

NtechLab

июн. 15, 2023

Содержание:

FindFace Enterprise Server — это передовая технология распознавания лиц на базе искусственного интеллекта.

Возможности:

- Быстрое и надежное распознавание лиц на основе AI на фото- и видеоизображениях.
- Быстрая и точная идентификация и верификация лиц AI-алгоритмами.
- Возможность задания правил обработки лиц.
- AI-распознавание пола, возраста, эмоций, очков, бороды и других атрибутов лица.
- AI-детектор живых лиц (Liveness).
- Расширенный API для анализа и распознавания биометрических данных.
- Расширенный API для управления видеодетекцией лиц.
- Возможность установки в кластере. Практически неограниченная масштабируемость.
- Возможность лицензирования в открытых и закрытых системах.
- Интеграция по HTTP API.

FindFace Enterprise Server может использоваться как платформа для построения интегрированных отраслевых решений и приложений Android/iOS, решающих широкий спектр прикладных задач, таких как биометрическая идентификация и аутентификация, контроль и управление доступом, работа со списками «свой-чужой», работа с большими объемами медиа-данных, оценка контрагентов, аттестация сотрудников, предотвращение правонарушений, контроль аффилированности, расследование инцидентов, создание безопасных городов и многих других.

FindFace Enterprise Server предназначен для независимых производителей программного обеспечения, системных интеграторов, корпоративных заказчиков, а также производителей оборудования, программно-аппаратных решений и облачных сервисов. FindFace Enterprise Server будет востребован в таких областях, как розничная торговля, банковское обслуживание, индустрия развлечений, спортивные мероприятия, организация мероприятий, сервисы знакомств, видеонаблюдение, государственное управление, общественная и корпоративная безопасность, транспорт, медицина и многие другие.

Руководство предназначается для специалистов по интеграции, системных администраторов, специалистов по установке и настройке систем анализа и распознавания биометрических данных на базе FindFace Enterprise Server, а также для разработчиков приложений на его основе.

Для начала мы рекомендуем вам ознакомиться с *основными этапами интеграции*. Это даст вам общее представление о процессе развертывания FindFace Enterprise Server. Давайте приступим!

Глава 1

Приступая к интеграции

Интеграция FindFace Enterprise Server включает в себя следующие основные этапы:

1. *выбор архитектуры развертывания*;
2. подготовка физических серверов;
3. *установка ядра FindFace* и *тестирование его работы*;
4. *настройка обнаружения лиц на видео* и задание *правил* их обработки.
5. *настройка расширенных функций*;
6. *создание интерфейса обмена данными между Сервером и партнерским приложением*.

Глава 2

Архитектура

Обязательно уделите немного времени изучению архитектуры FindFace Enterprise Server. Эти знания необходимы для развертывания, интеграции, обслуживания и устранения проблем при работе программного комплекса.

В этой главе:

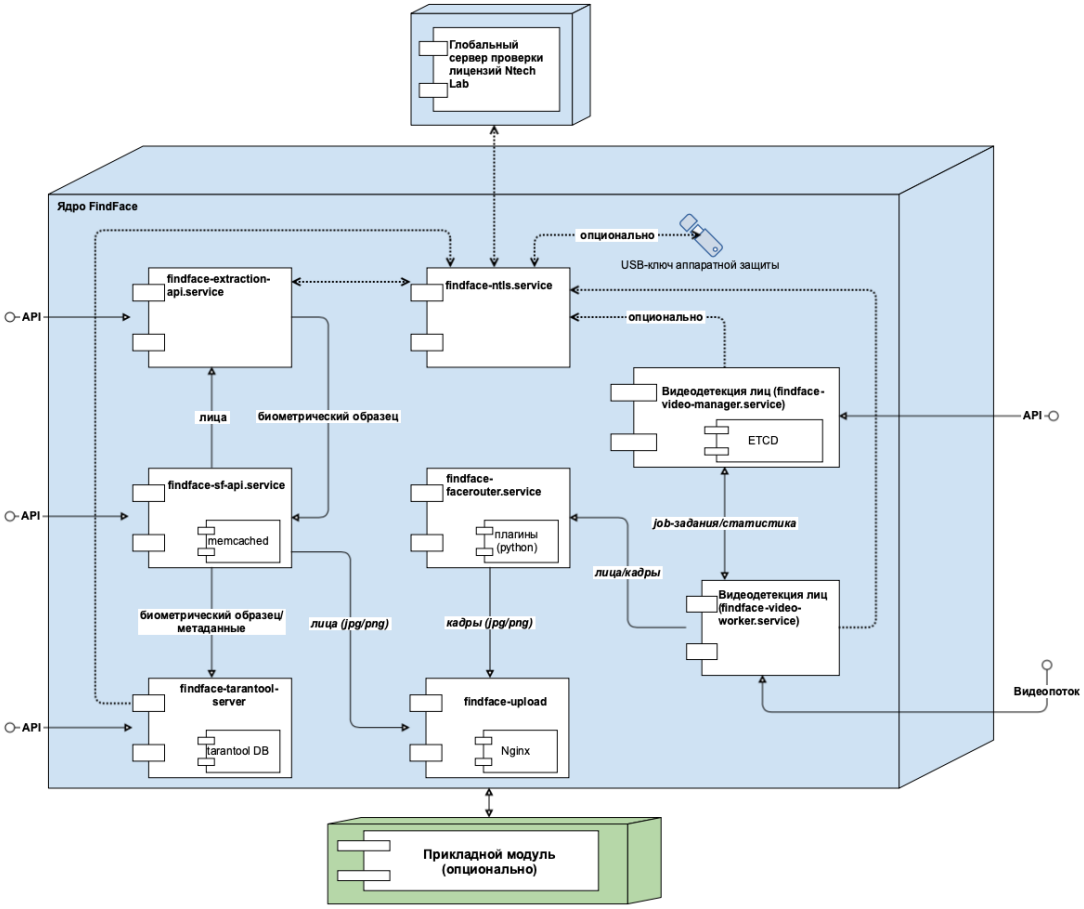
2.1 Архитектурные элементы

FindFace Enterprise Server состоит из следующих основных архитектурных элементов:

- Ядро FindFace,
- (опционально) прикладные модули.

> **Примечание:** В базовой конфигурации прикладные модули отсутствуют. Свяжитесь с нашими экспертами по адресу info@ntechlab.com, чтобы узнать больше о создании с нашей помощью приложения «под ключ».

Ядро FindFace включает в себя следующие компоненты:


Глобальный сервер проверки лицензий Ntech Lab
Ядро FindFace
опционально
USB-ключ аппаратной защиты
findface-extraction-api.service
findface-ntls.service
опционально
API
Видеодетекция лиц (findface-video-manager.service)
ETCD
API
лица
биометрический образец
findface-sf-api.service
memcached
findface-facerouter.service
плагины (python)
job-задания/статистика
API
лица/кадры
Видеодетекция лиц (findface-video-worker.service)
биометрический образец/ метаданные
лица (jpg/png)
кадры (jpg/png)
findface-tarantool-server
tarantool DB
findface-upload
Nginx
API
Видеопоток
Прикладной модуль (опционально)

Компонент	Описание	Поставщик
findface-extraction-api	Сервис, использующийся для обнаружения лиц на статических изображениях и извлечения из лиц биометрических образцов (векторов признаков), информации о поле, возрасте, эмоциях и других атрибутах лица. Работа на базе нейронных сетей. CPU- или GPU-ускорение.	Собственная разработка Ntech Lab
findface-sf-api	Сервис, реализующий HTTP API для анализа и распознания биометрических данных.	
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.	
findface-video-manager	Часть модуля видеодетекции лиц. Сервис, через который осуществляется управление детекцией лиц на видео, а именно задаются настройки и список видеопотоков для обработки.	
findface-video-worker	Часть модуля видеодетекции лиц. Сервис, который распознает лицо на видео и отправляет его нормализованное изображение, полный кадр и метаданные (такие как ID камеры и время обнаружения) в сервис `findface-facerouter` для дальнейшей обработки в соответствие с заданными правилами. CPU- или GPU-ускорение.	
findface-ntls	Сервер лицензий, который проверяет подлинность *лицензии* FindFace Enterprise Server, взаимодействуя с глобальным сервером лицензий NtechLab или USB-ключом.	
Tarantool	Стороннее программное обеспечение, на основе которого реализована биометрическая база данных, хранящая извлеченные биометрические образцы (векторы признаков).	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 компонент `findface-upload`.	nginx
memcached	Стороннее программное обеспечение, реализующее сервис кэширования данных в оперативной памяти на основе хеш-таблицы. Используется компонентом `findface-extraction-api` для временного хранения извлеченных биометрических образцов перед их записью в базу данных Tarantool.	memcached

См.также:

components

2.2 Развертывание на одиночном сервере или в кластере

В зависимости от требований к характеристикам проектируемой системы анализа и распознавания биометрических данных, FindFace Enterprise Server может быть развернут в соответствии со следующими архитектурными схемами:

Схема	Рекомендация
Развертывание на одиночном сервере	Вы можете развернуть FindFace Enterprise Server и модели нейронной сети на одном физическом сервере, если количество лиц в базе данных не превышает 1 000 000 (рекомендуемое ограничение). Используйте этот вариант, если вы только приступаете к развертыванию своей системы и вам нужно сформировать требования к ней.
Развертывание на кластере серверов	Если количество лиц в базе данных превышает 1 000 000, рекомендуется развернуть FindFace Enterprise Server в кластерной среде. В этом случае компоненты FindFace Enterprise Server распределяются по нескольким физическим серверам. Данный тип развертывания обеспечивает использование FindFace Enterprise Server в средних и крупных проектах и имеет потенциал практически неограниченного масштабирования. Он также подходит для профессиональных проектов с высокой нагрузкой и жесткими требованиями к производительности.

При развертывании в кластере возможны следующие схемы:

- Один центральный сервер FindFace Enterprise Server и несколько дополнительных серверов `findface-video-worker`.
- Компоненты FindFace Enterprise Server распределяются между несколькими физическими серверами. При необходимости настраивается балансировка нагрузки.

2.3 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 Enterprise Server 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. requirements.

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

Глава 3

Системные требования

Для расчета характеристик серверов FindFace Enterprise Server используйте приведенные ниже системные требования.

Совет: Сначала обязательно ознакомьтесь с *архитектурой* FindFace Enterprise Server.

В этой главе:

3.1 Базовая конфигурация

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

Примечание: При проектировании архитектуры системы в кластерной среде имейте в виду, что количество шардов `findface-tarantool-server` на одном физическом сервере не должно превышать количество каналов памяти CPU, умноженное на 2.

	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 Enterprise Server 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 AVX	
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 Enterprise Server 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 Enterprise Server требуется 15 Гб.	
Оперативная система	Ubuntu 16.04 только x64	

Глава 4

Принцип лицензирования

FindFace Enterprise Server лицензируется по следующим критериям:

1. Количество биометрических образцов, извлеченных из статических изображений и видео.
2. Количество камер в системе.
3. Количество используемых `findface-extraction-api` экземпляров моделей.
4. Распознавание атрибутов лица: пол/возраст/эмоции/очки/борода.
5. Распознавание живых лиц в реальном времени (Liveness).
6. Быстрый индекс.

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:

Получение информации о лицензии

Глава 5

Развертывание FindFace Enterprise Server

Для вашего удобства мы предлагаем несколько вариантов развертывания FindFace Enterprise Server:

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

После установки *проверьте* работоспособность системы и настройте поиск по *быстрому индексу*.

5.1 Развертывание из консольного инсталлятора

Для развертывания FindFace Enterprise Server используется консольный инсталлятор.

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

Выполните следующие действия:

1. Загрузите файл инсталлятора `<findface-security-and-server-xxx>.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-xxx>.run
   ```

4. Запустите файл `.run`.

   ```
   sudo ./<findface-security-and-server-xxx>.run
   ```

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

   1. Устанавливаемый продукт: `FindFace Server`.

2. Тип установки:
   - 1: установить FindFace Enterprise Server на одиночном физическом сервере.
   - 2: установить FindFace Enterprise Server 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.
   - 4: *полностью настраиваемая установка*.

     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 Enterprise Server на других серверах, не отвечая повторно на вопросы инсталлятора.

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

Сервис	Конфигурация
etcd	Устанавливается и запускается.
memcached	Устанавливается и запускается.
nginx	Устанавливается и запускается.
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-facerouter	Устанавливается и запускается.
findface-upload	Устанавливается.
findface-video-manager	Устанавливается и запускается.
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.
jq	Устанавливается. Используется для структурирования API-ответов от FindFace Enterprise Server в формате JSON.

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

Совет: Обязательно сохраните эти данные: они вам понадобятся.

```
################################################################################
#                          Installation is complete                            #
################################################################################
- upload your license to http://127.0.0.1:3185/
- FindFace SF-API address: http://172.20.77.78:18411/
- FindFace VideoManager address: http://172.20.77.78:18411/
```

5. Загрузите файл лицензии FindFace Enterprise Server через веб-интерфейс `findface-ntls` по адресу `http://<IP адрес сервера ntls>:3185`.

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.

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

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

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

5.2 Пошаговое развертывание из apt-репозитория

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

В этом разделе:

5.2.1 Установка apt-репозитория

Прежде всего установите apt-репозиторий FindFace следующим образом:

1. Загрузите файл инсталлятора `<findface-security-and-server-xxx>.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-xxx>.run
```

4. Запустите файл `.run`.

```
sudo ./<findface-security-and-server-xxx>.run
```

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

1. Устанавливаемый продукт: `FindFace Server`.
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.

5.2.2 Установка необходимого стороннего ПО

Для работы FindFace Enterprise Server необходимо стороннее программное обеспечение etcd и memcached. Выполните следующие действия:

1. Установите пакеты с указанным сторонним ПО следующим образом:

```
sudo apt update
sudo apt install -y 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 etcd.service memcached.service
sudo systemctl start etcd.service memcached.service
```

5.2.3 Обеспечение лицензирования

Вы получаете файл лицензии от своего менеджера Ntech Lab. Для лицензирования в закрытой сети вам также будет предоставлен ключ аппаратной защиты.

Лицензирование FindFace Enterprise Server обеспечивается следующим образом:

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

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

Важно: Система на базе FindFace Enterprise Server 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*.

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 веб-интерфейс `findface-ntls`, чтобы посмотреть информацию о лицензиях, обновить или продлить лицензию.

   - 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:

Устранение неполадок с лицензией и findface-ntls

5.2.4 Развертывание `findface-extraction-api`

Для развертывания компонента `findface-extraction-api` выполните следующие действия:

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

1. Установите `findface-extraction-api` следующим образом:

```
sudo apt install -y findface-extraction-api
```

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

2. Откройте файл конфигурации `findface-extraction-api`.

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

3. Укажите IP-адрес сервера `findface-ntls`, если `findface-ntls` установлен на удаленном сервере. See licensing.

```
license_ntls_server: 192.168.113.2:3133
```

4. При необходимости настройте другие параметры. Например, включите или выключите получение изображений из Интернета.

```
fetch:
  enabled: true
  size_limit: 10485760
```

5. Параметры `min_face_size` и `max_face_size` не являются фильтрами, а задают минимальный и максимальный размеры лиц, которые будут гарантированно обнаружены. Подбор данных значений нужно выполнять с осторожностью, поскольку они влияют на производительность.

```
nnd:
  min_face_size: 30
  max_face_size: .inf
```

6. Параметр `instances` означает количество используемых экземпляров `findface-extraction-api`. Укажите количество экземпляров из вашей лицензии. Значение по умолчанию (0) означает, что это число равно количеству ядер ЦП.

Примечание: Данный параметр существенно влияет на потребление оперативной памяти.

```
instances: 2
```

7. Для оценки качества лица включите `quality_estimator`. В этом случае `findface-extraction-api` будет возвращать показатель качества как значение параметра *detection_score*.

Совет: 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.

```
quality_estimator: true
```

8. 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`
	GPU	`face: face/grapefruit_480.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: ""
```

9. Добавьте сервис `findface-extraction-api` в автозагрузку Ubuntu и запустите сервис.

```
sudo systemctl enable findface-extraction-api.service && sudo systemctl start findface-
↪extraction-api.service
```

5.2.5 Развертывание `findface-tarantool-server`

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 системы на одиночном физическом сервере одного шарда, созданного по умолчанию, будет достаточно. При развертывании в кластерной среде количество шардов рассчитывается в зависимости от конфигурации оборудования и размера базы данных (см. подробности ниже).

Для развертывания компонента `findface-tarantool-server` выполните следующие действия:

1. Установите `findface-tarantool-server`

```
sudo apt update
sudo apt install -y findface-tarantool-server
```

2. Удалите тестовый сервис Tarantool из автозагрузки Ubuntu и остановите его.

```
sudo systemctl disable tarantool@example && sudo systemctl stop tarantool@example
```

3. Откройте файл конфигурации:

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

4. Отредактируйте максимальное использование оперативной памяти. Значение задается в байтах в зависимости от количества лиц, обрабатываемых шардом, исходя из соотношения примерно 1280 байт на 1 лицо. Например, значение 1.2*1024*1024*1024 соответствует 1 000 000 лиц:

```
memtx_memory = 1.2 * 1024 * 1024 * 1024,
```

5. Укажите IP-адрес сервера `findface-ntls`, если `findface-ntls` установлен на удаленном сервере:

```
FindFace.start(''127.0.0.1'', 8001, {license_ntls_server=''192.168.113.2:3133''})
```

6. База данных Tarantool по умолчанию доступна только локально (`127.0.0.1`). Если необходимо открыть доступ к базе данных Tarantool с удаленного сервера, в разделе `FindFace.start` укажите IP-адрес определенного сервера, либо измените `127.0.0.1` на `0.0.0.0`, чтобы разрешить доступ к базе данных Tarantool со всех IP-адресов.

Совет: Для того чтобы настроить доступ только с определенного IP-адреса (`192.168.113.10` в примере), настройте следующим образом:

```
FindFace.start("192.168.113.10", 8001, {license_ntls_server=''192.168.113.2:3133''})
```

Для доступа с любого IP-адреса вместо конкретного значения введите `0.0.0.0`:

```
FindFace.start("0.0.0.0", 8001, {license_ntls_server=''192.168.113.2:3133''})
```

---

7. Создайте структуру базы данных для хранения результатов распознавания лиц в параметре `meta_scheme`. Структура создается в виде набора 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 больше '1e14 – 1', то его следует записывать в виде строки, т. е. `''123123..''` вместо `123123...`

```
box.cfg{
    listen = '127.0.0.1:33001',

    vinyl_dir = '/opt/ntech/var/lib/tarantool/name',
    work_dir = '/opt/ntech/var/lib/tarantool/name',
    memtx_dir = '/opt/ntech/var/lib/tarantool/name/snapshots',
    wal_dir = '/opt/ntech/var/lib/tarantool/name/xlogs',

    memtx_memory = 16 * 1024 * 1024 * 1024,

    checkpoint_interval = 3600*4,
    checkpoint_count = 3,

    -- force_recovery = true,
}

pcall(function() box.schema.user.grant('guest', 'execute,read,write', 'universe') end)
```

(continues on next page)

(продолжение с предыдущей страницы)

```
FindFace = require("FindFace")
FindFace.start(
    "0.0.0.0",
    8001,
    {
        license_ntls_server="127.0.0.1:3133",
        facen_size=480,
        meta_scheme  = {

            {
                id = 1,
                name = 'm:timestamp',
                field_type = 'unsigned',
                default = 0
            },

            {
                id = 2,
                name = 'feat',
                field_type = 'string',
                default = ""
            },

            {
                id = 3,
                name = 'normalized_id',
                field_type = 'string',
                default = ""
            },

            {
                id = 4,
                name = 'm:camera',
                field_type = 'string',
                default = ""
            },

            {
                id = 5,
                name = 'm:photo',
                field_type = 'string',
                default = ""
            },

            {
                id = 6,
                name = 'm:thumbnail',
                field_type = 'string',
                default = ""
            },

            {
                id = 7,
                name = 'm:score',
                field_type = 'unsigned',
```

(continues on next page)

(продолжение с предыдущей страницы)

```
                default = "10000000000000000000"
            },

            {
                id = 8,
                name = 'm:bbox',
                field_type = 'string',
                default = ""
            },

            {
                id = 9,
                name = 'm:labels',
                field_type = 'set[string]',
                default = {}
            },

            {
                id = 10,
                name = 'm:is_friend',
                field_type = 'unsigned',
                default = 0
            },
            }
    }
)
```

8. (Опционально) При обслуживании одним шардом более 10,000,000 лиц поиск среди них может занимать продолжительное время. Для реализации крупных проектов рекомендуется создание дополнительных шардов `findface-tarantool-server`. Руководствуйтесь следующими правилами:

   - Один шард может обрабатывать приблизительно 10 000 000 лиц.
   - Количество шардов на одном сервере не должно превышать число каналов памяти процессора, умноженное на 2. Помните об этом при разработке архитектуры системы в кластерной среде.

   Для того чтобы создать дополнительные шарды, создайте аналогичное количество файлов конфигурации на основе файла по умолчанию `/etc/tarantool/instances.enabled/FindFace.lua` и внесите актуальные значения IP-адресов и портов. Для этого напишите bash-скрипт (`shard.sh`, например), который автоматически создаст файлы конфигурации для всех шардов на определенном сервере. Используйте приведенный ниже скрипт как основу для своего кода. Примерный скрипт создает 4 шарда, использующие порты `findface-tarantool-server` 33001..33004 и HTTP 8001..8004.

```
#!/bin/sh
set -e

for I in `seq 1 4`; do
       TNT_PORT=$((33000+$I)) &&
       HTTP_PORT=$((8000+$I)) &&
       sed "
                s#/opt/ntech/var/lib/tarantool/default#/opt/ntech/var/lib/tarantool/shard_$I#g;
                s/listen = .*$/listen = '127.0.0.1:$TNT_PORT',/;
                s/\"127.0.0.1\", 8001,/\"0.0.0.0\", $HTTP_PORT,/;
```

(continues on next page)

(продолжение с предыдущей страницы)

```
      " /etc/tarantool/instances.enabled/FindFace.lua > /etc/tarantool/instances.enabled/
↪FindFace_shard_$I.lua;

      mkdir -p /opt/ntech/var/lib/tarantool/shard_$I/snapshots
      mkdir -p /opt/ntech/var/lib/tarantool/shard_$I/xlogs
      mkdir -p /opt/ntech/var/lib/tarantool/shard_$I/index
      chown -R tarantool:tarantool /opt/ntech/var/lib/tarantool/shard_$I
      echo "Shard #$I inited"
done;
```

Совет: Загрузите `пример скрипта`.

Запустите скрипт из домашней директории.

```
sudo sh ~/shard.sh
```

Проверьте созданные файлы конфигурации.

```
ls /etc/tarantool/instances.enabled/

##example.lua FindFace.lua FindFace_shard_1.lua FindFace_shard_2.lua FindFace_shard_3.lua␣
↪FindFace_shard_4.lua
```

9. Добавьте шард `findface-tarantool-server` в автозагрузку Ubuntu и запустите его.

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

В случае нескольких шардов вы можете сделать это аналогично примеру запуска 4-х шардов:

```
for I in `seq 1 4`; do sudo systemctl enable tarantool@FindFace_shard_$I; done;
for I in `seq 1 4`; do sudo systemctl start tarantool@FindFace_shard_$I; done;
```

5.2.6 Развертывание `findface-upload`

Для хранения всех исходных изображений, которые когда-либо отправлялись в систему для обработки, а также таких артефактов работы ядра FindFace, как миниатюры и нормализованные изображения лиц, разверните сервис `findface-upload`.

Совет: Пропустите данный шаг, если вы не собираетесь хранить указанные данные. В этом случае на Сервере будут храниться только биометрические образцы в базе данных Tarantool.

Установите `findface-upload` следующим образом:

```
sudo apt update
sudo apt install -y findface-upload
```

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

5.2.7 Развертывание findface-sf-api

Для развертывания компонента findface-sf-api выполните следующие действия:

1. Установите findface-sf-api следующим образом:

```
sudo apt update
sudo apt install -y findface-sf-api
```

2. Откройте файл конфигурации /etc/findface-sf-api.ini.

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

3. Если FindFace Enterprise Server развертывается в кластерной конфигурации, укажите IP-адреса и порты серверов с компонентами findface-extraction-api (параметр extraction-api), findface-tarantool-server (параметр storage-api, в формате http://IP_address:port/v2/), findface-upload (параметр upload_url).

```
extraction-api:
  extraction-api: http://10.220.85.120:18666
storage-api:
 shards:
 - master: http://10.200.85.115:8003/v2/
 - master: http://10.200.85.120:8004/v2/
 - master: http://10.200.85.120:8005/v2/
 - master: http://10.200.85.120:8006/v2/
   slave: ''
upload_url: http://127.0.0.1:3333/uploads/
```

4. Добавьте сервис findface-sf-api в автозагрузку Ubuntu и запустите сервис.

```
sudo systemctl enable findface-sf-api.service && sudo systemctl start findface-sf-api.service
```

5.2.8 Развертывание findface-facerouter

Для развертывания компонента findface-facerouter выполните следующие действия:

1. Установите findface-facerouter 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. Добавьте сервис findface-facerouter в автозагрузку Ubuntu и запустите сервис.

```
sudo systemctl enable findface-facerouter.service && sudo systemctl start findface-facerouter.
↪service
```

5.2.9 Настройка видеодетекции лиц

Работа видеодетектора лиц обеспечивается взаимодействием компонентов `findface-video-manager` и `findface-video-worker`.

Для развертывания компонента `findface-video-manager` выполните следующие действия:

1. Установите `findface-video-manager`:

```
sudo apt install -y findface-video-manager
```

2. Откройте файл конфигурации `/etc/findface-video-manager.conf`.

```
sudo vi /etc/findface-video-manager.conf
```

3. В параметре `router_url` укажите IP-адрес и порт компонента `findface-facerouter`, на который компонент `findface-video_worker` будет отправлять обнаруженные лица.

```
router_url: http://127.0.0.1:18820/v0/frame
```

4. В параметре `ntls -> url` укажите IP-адрес сервера `findface-ntls`, если `findface-ntls` установлен на удаленном сервере.

```
ntls:
    url: http://127.0.0.1:3185/
```

5. При необходимости настройте параметры, общие для всех видеопотоков.

Совет: Данный шаг может быть пропущен: при создании job-заданий вы сможете отдельно задавать настройки для каждого обрабатываемого видеопотока (см. *HTTP API для управления видеодетекцией лиц*).

---

6. Добавьте сервис `findface-video-manager` в автозагрузку Ubuntu и запустите сервис.

```
sudo systemctl enable findface-video-manager.service && sudo systemctl start findface-video-
↪manager.service
```

Для развертывания компонента `findface-video-worker` выполните следующие действия:

1. Установите `findface-video-worker`:

```
sudo apt update
sudo apt install -y findface-video-worker-cpu
```

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*.

---

2. Откройте файл конфигурации `findface-video-worker`.

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

3. В параметре `ntls-addr` укажите IP-адрес сервера `findface-ntls`, если `findface-ntls` установлен на удаленном сервере.

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

4. В параметре `mgr-static` укажите IP-адрес сервера с установленным компонентом `findface-video-manager`, у которого компонент `findface-video-worker` будет запрашивать настройки и список видеопотоков.

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

5. В параметре `capacity` укажите максимальное количество видеопотоков, обрабатываемых компонентом `findface-video-worker`.

```
capacity=10
```

6. Добавьте сервис `findface-video-worker` в автозагрузку Ubuntu и запустите сервис.

```
sudo systemctl enable findface-video-worker-cpu.service && sudo systemctl start findface-
↪video-worker-cpu.service
```

5.3 Тестовые запросы

Перед тем как приступить к программированию и использованию распознавания лиц в своем приложении, убедитесь, что компоненты Сервера FindFace работают надлежащим образом. Для этого выполните по порядку приведенные ниже тестовые запросы. Для того чтобы структурировать текст ответов на запросы, используйте обработчик JSON `jq`.

В этом разделе:

5.3.1 Структурирование ответов на запросы

Используйте обработчик `jq`, чтобы структурировать данные в формате JSON в ответах на запросы. Обработчик `jq` устанавливается автоматически из консольного инсталлятора.

Совет: Если это не так, установите `jq` следующим образом:

```
sudo apt install jq
```

Примечание: Поскольку обработчик `jq` аппроксимирует целые числа, большие 2^53 (например, для `''id'':12107867323949968228` в результате получится `''id'':12107867323949967000` и т. д.), вместо него может быть целесообразным использовать `json_pp`.

5.3.2 Создание галереи

Данный запрос создает новую галерею `galleryname`. Соответствующий метод HTTP API для анализа и распознавания лиц: `/galleries/<gallery> POST`.

Запрос

```
curl -s -X POST http://localhost:18411/v2/galleries/galleryname | jq
```

Ответ

```
{}
```

5.3.3 Получение списка галерей

Данный запрос возвращает имя единственной на данный момент галереи (`galleryname`). Соответствующий метод HTTP API для анализа и распознавания лиц: `/galleries GET`.

Запрос

```
curl -s http://localhost:18411/v2/galleries | jq
```

Ответ

```
{
  "galleries": [
    {
      "name": "galleryname",
      "faces": 0
    }
  ]
}
```

5.3.4 Обнаружение лица на фотографии

Первый запрос обнаруживает лицо на тестовых изображениях, размещенных в сети Интернет, и возвращает координаты рамок вокруг лиц и их ориентацию. Соответствующий метод HTTP API для анализа и распознавания лиц: `/detect POST`.

Запрос №1

```
curl -s -H 'Content-Type: text/x-url' -d https://static.findface.pro/sample.jpg -X POST http://
↪localhost:18411/v2/detect | jq
```

Ответ


```
{
  "faces": [
    {
      "bbox": {
        "left": 595,
        "top": 127,
        "right": 812,
        "bottom": 344
      },
      "features": {
        "score": 0.9999999
      }
    }
  ],
  "orientation": 1
}
```


Если `facen=on`, возвращенный результат будет сохранен в `memcached`. В следующем запросе изображение то же самое, но на этот раз `facen=on`, а также запрашиваются такие параметры, как пол, возраст и эмоции.

Совет: Для извлечения результата распознавания из `memcached` примените метод `/detect GET`.

Запрос №2

```
curl -s -H 'Content-Type: text/x-url' -d https://static.findface.pro/sample.jpg -X POST 'http://
↪localhost:18411/v2/detect?facen=on&gender=on&age=on&emotions=on' | jq
```

Ответ


```
{
  "faces": [
    {
      "id": "bhse5elubdg0ajgm2nkg",
      "bbox": {
        "left": 595,
```


(continues on next page)

(продолжение с предыдущей страницы)


```
        "top": 127,
        "right": 812,
        "bottom": 344
      },
      "features": {
        "gender": {
          "gender": "FEMALE",
          "score": -2.6415923
        },
        "age": 26.04833,
        "score": 0.9999999,
        "emotions": [
          {
            "emotion": "neutral",
            "score": 0.99958
          },
          {
            "emotion": "sad",
            "score": 0.0004020398
          },
          {
            "emotion": "happy",
            "score": 8.603504e-06
          },
          {
            "emotion": "surprise",
            "score": 8.076798e-06
          },
          {
            "emotion": "disgust",
            "score": 6.653509e-07
          },
          {
            "emotion": "angry",
            "score": 6.14346e-07
          },
          {
            "emotion": "fear",
            "score": 7.33713e-10
          }
        ]
      }
    }
  ],
  "orientation": 1
}
```


В следующем запросе детектируется лицо с другого изображения (с целью последующего наполнения базы данных). Результат будет записан в `memcached` (`facen=on`).

Запрос №3

```
curl -s -H 'Content-Type: text/x-url' -d https://static.findface.pro/sample2.jpg -X POST 'http://
↪localhost:18411/v2/detect?facen=on
```

```
{
  "faces": [
    {
      "id": "bhse45dubdg0ajgm2nk0",
      "bbox": {
        "left": 515,
        "top": 121,
        "right": 821,
        "bottom": 427
      },
      "features": {
        "score": 0.9999982
      }
    }
  ],
  "orientation": 1
}
```


5.3.5 Извлечение результата детекции из memcached

Данный запрос извлекает из `memcached` результат детекции, полученный в одном из предыдущих запросов, по его id. Соответствующий метод HTTP API для анализа и распознавания лиц: `/detect GET`.

Примечание: `bhse5elubdg0ajgm2nkg` – id лица в `memcached`, полученный в ответе детектора. При выполнении тестовых запросов вам потребуется заменить его на актуальное полученное значение.

Важно: Прежде чем продолжить, откройте файл конфигурации `findface-sf-api` и убедитесь, что параметр `allow-return-facen` имеет значение `on`.

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

allow-return-facen: on
```

Запрос №1

```
curl -s 'http://localhost:18411/v2/detect/bhse5elubdg0ajgm2nkg'
```

Ответ


```
{
  "id": "bhse5elubdg0ajgm2nkg",
  "bbox": {
    "left": 595,
    "top": 127,
    "right": 812,
    "bottom": 344
  },
  "features": {
```


(continues on next page)

(продолжение с предыдущей страницы)

```
    "score": 0.9999999
  }
}
```

Для того чтобы получать вместе с результатом детекции вектор признаков лица (facen), откройте файл конфигурации `/etc/findface-sf-api.ini` и установите `allow-return-facen: true`. Перезапустите `findface-sf-api` и добавьте параметр строки `return_facen = on` к предыдущей команде:

Запрос №2

```
curl -s 'http://localhost:18411/v2/detect/bhse5elubdg0ajgm2nkg?return_facen=on' | jq
```

Ответ

```
{
  "id": "bhse5elubdg0ajgm2nkg",
  "bbox": {
    "left": 595,
    "top": 127,
    "right": 812,
    "bottom": 344
  },
  "features": {
    "score": 0.9999999
  },
  "facen": "1ji...Vr3TEQg8"
}
```

5.3.6 Добавление лица из `memcached` в галерею

Данные запросы извлекают результаты детекции из `memcached` по id и добавляют их в галерею `galleryname` под другими, пользовательскими id. Соответствующий метод HTTP API для анализа и распознавания лиц: `/v2/galleries/<gal>/faces/<id>` .

Запрос №1

```
curl -s -X POST -H 'Content-Type: application/json' --data '{"from":"detection:bd2blott8f63g8nbhi50
↪"}' http://localhost:18411/v2/galleries/galleryname/faces/1 | jq
```

Ответ

```
{
  "id": {
    "gallery": "galleryname",
    "face": 1
  },
  "features": {
    "gender": {
```

(continues on next page)

(продолжение с предыдущей страницы)


```
      "gender": "FEMALE",
      "score": -2.6415923
    },
    "age": 26.04833,
    "score": 0.9999999,
    "emotions": [
      {
        "emotion": "neutral",
        "score": 0.99958
      },
      {
        "emotion": "sad",
        "score": 0.0004020398
      },
      {
        "emotion": "happy",
        "score": 8.603504e-06
      },
      {
        "emotion": "surprise",
        "score": 8.076798e-06
      },
      {
        "emotion": "disgust",
        "score": 6.653509e-07
      },
      {
        "emotion": "angry",
        "score": 6.14346e-07
      },
      {
        "emotion": "fear",
        "score": 7.33713e-10
      }
    ]
  },
  "meta": {},
  "normalized_id": "3_bd323i5t8f66ph0eafq0.png"
}
```


Запрос №2

```
curl -s -X POST -H 'Content-Type: application/json' --data '{"from":"detection:␣
↪bd44p6dt8f66ph0eahkg "}' http://localhost:18411/v2/galleries/galleryname/faces/2 | jq
```

5.3.7 Получение списка лиц в галерее

Данный запрос выводит список всех лиц, добавленных в галерею `galleryname`. Соответствующий метод HTTP API для анализа и распознавания лиц: `/galleries/<gallery>/faces` с включенным фильтром по максимальному количеству возвращаемых лиц (`limit=`).

Запрос

```
curl -s 'http://localhost:18411/v2/galleries/galleryname/faces?limit=2' | jq
```


```
{
  "faces": [
    {
      "id": {
        "gallery": "galleryname",
        "face": 1
      },
      "features": {
        "gender": {
          "gender": "FEMALE",
          "score": -2.6415923
        },
        "age": 26.04833,
        "score": 0.9999999,
        "emotions": [
          {
            "emotion": "neutral",
            "score": 0.99958
          },
          {
            "emotion": "sad",
            "score": 0.0004020398
          },
          {
            "emotion": "happy",
            "score": 8.603504e-06
          },
          {
            "emotion": "surprise",
            "score": 8.076798e-06
          },
          {
            "emotion": "disgust",
            "score": 6.653509e-07
          },
          {
            "emotion": "angry",
            "score": 6.14346e-07
          },
          {
            "emotion": "fear",
            "score": 7.33713e-10
          }
        ]
      },
      "meta": {},
      "normalized_id": "1_bd321tlt8f66ph0eaflg.png"
    },
    {
      "id": {
        "gallery": "galleryname",
        "face": 2
      },
```


(continues on next page)

(продолжение с предыдущей страницы)


```
      "features": {
        "gender": {
          "gender": "FEMALE",
          "score": -2.6415923
        },
        "age": 26.04833,
        "score": 0.9999999,
        "emotions": [
          {
            "emotion": "neutral",
            "score": 0.99958
          },
          {
            "emotion": "sad",
            "score": 0.0004020398
          },
          {
            "emotion": "happy",
            "score": 8.603504e-06
          },
          {
            "emotion": "surprise",
            "score": 8.076798e-06
          },
          {
            "emotion": "disgust",
            "score": 6.653509e-07
          },
          {
            "emotion": "angry",
            "score": 6.14346e-07
          },
          {
            "emotion": "fear",
            "score": 7.33713e-10
          }
        ]
      },
      "meta": {},
      "normalized_id": "2_bd323f5t8f66ph0eafp0.png"
    }
  ],
  "next_page": "3"
}
```


5.3.8 Поиск лица в галерее

Следующий запрос выполняет поиск обнаруженного лица (результата детекции) в галерее `galleryname` с пороговой схожестью, равной 0.5. Соответствующий метод HTTP API для анализа и распознавания лиц: `/galleries/<gallery>/faces` с включенными фильтрами по `detection:id` (биометрическому образцу) и `similarity` (схожести) лиц.

Запрос

```
curl -s 'http://localhost:18411/v2/galleries/galleryname/faces?detection:bd3hv4tt8f66ph0eag1g=0.5&
↪limit=1' | jq
```

Ответ


```
{
  "faces": [
    {
      "id": {
        "gallery": "galleryname",
        "face": 2
      },
      "features": {
        "gender": {
          "gender": "FEMALE",
          "score": -2.6415923
        },
        "age": 26.04833,
        "score": 0.9999999,
        "emotions": [
          {
            "emotion": "neutral",
            "score": 0.99958
          },
          {
            "emotion": "sad",
            "score": 0.0004020398
          },
          {
            "emotion": "happy",
            "score": 8.603504e-06
          },
          {
            "emotion": "surprise",
            "score": 8.076798e-06
          },
          {
            "emotion": "disgust",
            "score": 6.653509e-07
          },
          {
            "emotion": "angry",
            "score": 6.14346e-07
          },
          {
            "emotion": "fear",
            "score": 7.33713e-10
          }
        ]
      },
      "meta": {},
      "normalized_id": "2_bd323f5t8f66ph0eafp0.png",
      "confidence": 0.9999
    }
```


(continues on next page)

(продолжение с предыдущей страницы)

```
  ],
  "next_page": "There are more than 1 results, but pagination is not supported when filtering by␣
↪FaceN"
}
```

Следующий запрос ищет в галерее `galleryname` лица, похожие на заданное лицо в той же галерее с пороговым сходством, равным 0,5. Соответствующий метод HTTP API для анализа и распознавания лиц: `/galleries/<gallery>/faces` с включенными фильтрами `face:<gallery>/<db_id>` и `similarity`.

```
curl -s 'http://localhost:18411/v2/galleries/galleryname/faces?face:galleryname/1=0.1&limit=1' | jq
```

```
{
  "faces": [
    {
      "id": {
        "gallery": "galleryname",
        "face": 2
      },
      "features": null,
      "meta": {},
      "confidence": 0.999
    }
  ],
  "next_page": "There are more than 1 results, but pagination is not supported when filtering by␣
↪FaceN"
}
```

5.3.9 Сравнение лиц

Следующие запросы выполняют сравнение 2 лиц и возвращают результат их проверки на идентичность. Соответствующий метод HTTP API для анализа и распознавания лиц: `/verify POST`.

Первый запрос сравнивает 2 лица, представляющие собой результаты детекции (метод `/detect POST`), хранящиеся в `memcached`.

Запрос №1

```
curl -s 'http://localhost:18411/v2/verify?face1=detection:bd3hv4tt8f66ph0eag1g&
↪face2=detection:bd3hv8dt8f66ph0eag2g' | jq
```

Ответ

```
{
  "confidence": 0.92764723
}
```

Второй запрос сравнивает результат детекции в memcached и лицо в галерее.

Запрос

```
curl -s 'http://localhost:18411/v2/verify?face1=detection:bd3hv4tt8f66ph0eag1g&
↪face2=face:galleryname/2' | jq
```

Ответ

```
{
  "confidence": 0.999996
}
```

5.4 Индексирование для быстрого поиска по базе данных

Для ускорения поиска каждая галерея должна быть проиндексирована. Для подготовки быстрого индекса вам понадобится утилита `findface-tarantool-build-index` (устанавливается из консольного инсталлятора). Для работы данной утилиты не требуется компонент `findface-tarantool-server`, поэтому она может использоваться как на сервере базы данных Tarantool, так и на удаленном сервере с доступом к базе данных.

Для подготовки быстрого индекса выполните следующие действия:

1. Если вы установили ядро FindFace *пошагово*, установите утилиту `findface-tarantool-build-index`.

   ```
   sudo apt install findface-tarantool-build-index
   ```

2. Создайте быстрый индекс для вашей галереи (`testgal` в примерах ниже). Сначала подключитесь к консоли базы данных `Tarantool`.

   Важно: Галерея не должна быть пустой. Для наполнения галереи лицами см. *Прямые API-запросы к базе данных Tarantool*.

   Примечание: Создание быстрого индекса производится на каждом шарде `findface-tarantool-server`.

   ```
   tarantoolctl connect 127.0.0.1:33001
   ```

3. Выполните метод `prepare_preindex`. В результате все элементы из пространства `linear` в данной галерее будут перемещены в пространство `preindex`:

   ```
   127.0.0.1:33001> FindFace.Gallery.new("testgal"):prepare_preindex()
   ---
   ...
   ```

4. Сохраните пространство `preindex` в файл, который будет использован для генерации индекса:

   ```
   127.0.0.1:33001> FindFace.Gallery.new("testgal"):save_preindex("/tmp/preindex.bin")
   ---
   ...
   ```

5. Запустите генерацию индекса на основе файла `preindex.bin` с помощью утилиты `findface-tarantool-build-index` (вызовите `--help` для ознакомления с дополнительными опциями). В зависимости от количества элементов, данный процесс может занимать до нескольких часов. В этом случае индексирование лучше выполнять на отдельной, более мощной машине (для больших галерей рекомендуется использовать c4.8xlarge amazon, например, spot-instance).

```
sudo findface-build-index --input /tmp/preindex.bin --output /opt/ntech/var/lib/tarantool/
↪default/index/testgal.idx --facen_size 320
Config values:
.input = /tmp/preindex.bin
.output = /opt/ntech/var/lib/tarantool/default/index/testgal.idx
.facen_size = 320
.param_m = 12
.param_ef = 500
.limit = 4294967295

Building index: [XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX] 100% ; 3 / 3
Index saved at /opt/ntech/var/lib/tarantool/default/index/testgal.idx
```

6. Удалите файл `preindex.bin`.

```
sudo rm /tmp/preindex.bin
```

7. Включите быстрый индекс для галереи.

Примечание: Если Tarantool функционирует как *набор реплик*, скопируйте по такому же пути на сервер-реплику файл индекса (`.idx`) и только после этого включите быстрый индекс на мастере (`:use_index`).

Совет: Рекомендуется удалять все файлы индекса на реплике, кроме последнего, во избежание промежуточных обновлений индекса в случае сильного отставания реплики от мастера.

```
127.0.0.1:33001> FindFace.Gallery.new("testgal"):preindex_to_index()
---
...
127.0.0.1:33001> FindFace.Gallery.new("testgal"):use_index("/opt/ntech/var/lib/tarantool/
↪default/index/testgal.idx")
---
...
```

8. После включения быстрого индекса поиск по галерее должен стать значительно быстрее. Информация об индексе остается в служебном пространстве Tarantool, поэтому когда вы перезапускаете Tarantool, индекс также подгружается.

Предупреждение: Не перемещайте файл индекса!

5.5 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-xxx>.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-xxx>.run
```

4. Запустите файл `.run`.

```
sudo ./<findface-security-and-server-xxx>.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 сервера FindFace Enterprise Server.

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. Подробнее см. *Файл с параметрами установки*.

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

Примечание: Если `findface-ntls` и/или `findface-video-manager` будут установлены на серверах, отличных от сервера `findface-sf-api`, 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
```

5.6 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-xxx>.run`.

   ```
   sudo ./<findface-security-and-server-xxx>.run
   ```

2. Тип установки: `Fully customized installation`.
3. Выберите устанавливаемый компонент : `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_160.cpu.fnk  grapefruit_160.gpu.fnk  grapefruit_480.cpu.fnk  grapefruit_480.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
```

5.7 Типичная установка в кластере

Данный раздел посвящен развертыванию FindFace Enterprise Server 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 Enterprise Server 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.

- 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-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 Enterprise Server 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*

5.7.1 Развертывание центрального сервера

Для развертывания центрального сервера FindFace Enterprise Server выполните следующие действия:

1. На выделенном физическом сервере установите FindFace Enterprise Server из инсталлятора следующим образом:
   - Устанавливаемый продукт: `FindFace Server`.
   - Тип установки: `Single server, multiple video workers`. В этом случае FindFace Enterprise Server 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:

Совет: Обязательно сохраните эти данные: они вам понадобятся.

```
################################################################################
#                        Installation is complete                              #
################################################################################
- upload your license to http://127.0.0.1:3185/
- FindFace SF-API address: http://172.20.77.78:18411/
- FindFace VideoManager address: http://172.20.77.78:18411/
```

2. Загрузите файл лицензии FindFace Enterprise Server через веб-интерфейс `findface-ntls` по адресу `http://<IP адрес сервера ntls>:3185`.

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.

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
```

5.7.2 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 Enterprise Server 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-xxx>.run -f /tmp/<findface-installer-*>.json
```

Примечание: После установки укажите IP-адреса компонентов `findface-ntls` и `findface-video-manager` 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
```

5.7.3 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 Server`.
- Тип установки: `Fully customized installation`.
- Устанавливаемые компоненты FindFace Enterprise Server: `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-160-cpu_3.0.0_amd64.deb, findface-data-grapefruit-480-cpu_3.0.0_amd64.deb
	GPU	findface-data-grapefruit-160-gpu_3.0.0_amd64.deb, findface-data-grapefruit-480-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-xxx>.run -f /tmp/<findface-installer-*>.json
```

После развертывания биометрических серверов распределите между ними нагрузку.

5.7.4 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 Enterprise Server (`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 настройки балансировки нагрузки выполните следующие действия:

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

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

   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` укажите как значение параметра `listen` номер слушающего порта сервера шлюза. Вам потребуется указать данный порт при настройке *распределенной сети* FindFace Enterprise Server.

```
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. Обязательно укажите актуальный IP-адрес и слушающий порт сервера шлюза при настройке *распределенной сети* FindFace Enterprise Server.

5.7.5 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 Server`.
- Тип установки: `Fully customized installation`.
- Устанавливаемые компоненты FindFace Enterprise Server: `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-xxx>.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 Enterprise Server. 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.

5.7.6 Настройка сетевого взаимодействия

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

1. Откройте файл конфигурации `findface-sf-api`:

   ```
   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 `findface-facerouter`. Укажите IP-адрес сервера `findface-sf-api`.

```
sudo vi /etc/findface-facerouter.py

sfapi_url                       = 'http://localhost:18411'
```

3. Откройте файл конфигурации `findface-video-manager`. 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
```

На этом установка FindFace Enterprise Server в кластерной среде будет завершена.

5.8 Полностью настраиваемая установка

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

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

- Устанавливаемый продукт: `FindFace Server`.
- Тип установки: `Fully customized installation`.
- Устанавливаемые компоненты FindFace Enterprise Server: 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.

Глава 6

HTTP API для анализа и распознавания биометрических данных

6.1 Как пользоваться API для анализа и распознавания биометрических данных

В этом разделе:

6.1.1 Точка доступа

Все запросы HTTP API для анализа и распознавания биометрических данных нужно отправлять на адрес http://<IP адрес findface-sf-api>:18411/. Запросы выполняются компонентом `findface-sf-api`.

6.1.2 Версия API

Версия API обновляется каждый раз при больших изменениях функционала. Версия API указывается в пути запроса (например, `v2` в `/v2/detect/`).

На данный момент самой последней версией API является `v2`.

Совет: При запуске нового проекта должна всегда использоваться последняя стабильная версия API.

6.1.3 Лицо как объект API

API для анализа и распознавания биометрических данных оперирует объектом `лицо`, представляющим собой изображение человеческого лица.

Примечание: На фотографии может быть несколько лиц и, следовательно, несколько ассоциированных с ней объектов `лицо`.

Примечание: Разные изображения одного и того же человека рассматриваются как разные объекты `лицо`.

Объект `лицо` имеет следующие атрибуты:

- `''id'' (uint64)`: (только если лицо добавлено в биометрическую базу данных) пользовательский идентификатор лица (uint64), задается в API-запросе при добавлении лица из `memcached` в базу данных Tarantool. Идентификатор передается как <id> в методе `/galleries/<gallery>/faces/<id> POST`.
- `"facen" (байты)`: биометрический образец, вектор признаков, извлеченный из лица.
- `"meta" (строка)`: набор строк с метаданными, который может быть использован для хранения любой связанной с лицом информации (например, имени человека, id видеокамеры, даты и времени обнаружения и пр.).
- `"features" (словарь)`: словарь, ключом в котором выступает строка, а значение может быть любого типа. Используется для хранения таких биометрических параметров лица, как пол, возраст, эмоции и др.

6.1.4 Формат параметров

Существует два способа передать фотографию в систему:

- как публичный адрес в сети Интернет,
- в виде файла.

Существуют следующие способы передачи параметров в методы HTTP API для анализа и распознавания биометрических данных:

- image/jpeg, image/png, image/webp: для передачи изображения в виде файла,
- text/x-url: для передачи изображения в виде URL,
- query string: параметры добавляются в конец URI запроса.

Все ответы отправляются в формате JSON и кодировке UTF-8.

6.1.5 Как пользоваться примерами

Примеры в описаниях методов иллюстрируют возможные запросы и ответы метода. Для проверки примеров без написания кода используйте встроенный API-фреймворк. Для перехода в него введите в адресной строке браузера `http://<findface-sf-api_ip>:18411/v2/docs/v2/overview.html` для API версии /v2.

6.1.6 Ограничения

Для FindFace Enterprise Server актуальны следующие ограничения.

Параметр	Значение
Формат изображения	JPG, PNG, WEBP, BMP
Максимальный размер файла	Настраивается в файле конфигурации компонента `findface-sf-api`.
Минимальный размер лица	50x50 пикселей
Максимальное количество обнаруженных лиц на 1 изображение	Не ограничено

Важно: В дополнение к этому, URL изображения должен быть публичным (не требующим авторизации) и прямым (без перенаправлений).

6.1.7 Сообщения об ошибках

Если метод выполнить не удается, Сервер возвращает ответ с кодом HTTP, отличном от 200, а также тело ответа в формате JSON, содержащее описание ошибки. Тело ответа всегда содержит хотя бы 2 поля — `code` и `desc`.

- `code` — это код ошибки в виде `CAPS_AND_UNDERSCORES`, который может быть использован для автоматического преобразования.
- `desc` — это описание ошибки, предназначенное для прочтения человеком.

Полный список ошибок

Код ошибки	Описание	Код HTTP
UNKNOWN_ERROR	Ошибка неизвестного происхождения.	500
BAD_PARAM	Запрос может быть прочитан, однако некоторые параметры метода недействительны. Данный тип ответа содержит дополнительные атрибуты `param` и `value` для описания ошибочных параметров.	400
CONFLICT	Конфликт.	409
EXTRACTION_ERROR	Ошибка при извлечении из лица вектора признаков.	503
LICENSE_ERROR	Конфигурация системы не соответствует лицензии.	503
MALFORMED_REQUEST	Запрос неправильно сформирован и не может быть прочитан.	400
OVER_CAPACITY	Превышен размер очередей в компоненте `findface-extraction-api`.	429
SOURCE_NOT_FOUND	В параметре `from` задано несуществующее лицо.	400
SOURCE_GALLERY_NOT_FOUND	В параметре `from` задана несуществующая галерея.	400
STORAGE_ERROR	Биометрическая база данных недоступна.	503
CACHE_ERROR	Хранилище memcached недоступно.	503
NOT_FOUND	Подходящие лица не найдены.	404
NOT_IMPLEMENTED	Функционал не реализован.	501
GALLERY_NOT_FOUND	Подходящие галереи не найдены.	404

6.2 Методы для анализа и распознавания биометрических данных

В этом разделе:

6.2.1 Обнаружение лица на изображении

```
/detect POST
```

Данный метод обнаруживает лицо на предоставленном изображении и возвращает координаты прямоугольной рамки вокруг лица (т. н. bbox) и ориентацию лица.

Примечание: Обнаружение лица выполняется компонентом `findface-extraction-api`. Таким образом, компонент `findface-sf-api` форматирует ваш исходный запрос и перенаправляет его в компонент `findface-extraction-api` для непосредственной обработки.

Важно: Обязательно передайте в query string метода активированный параметр `facen=on`, чтобы возвращенный результат был сохранен в `memcached`. Для извлечения результата распознавания из `memcached` используйте метод `/detect GET`.

Совет: Для того чтобы активировать параметр типа Boolean (`gender`, `age` и т. д.), подойдет любая из строк `1`, `yes`, `true` или `on`, регистр не имеет значения.

Важно: Чтобы включить распознавание атрибутов лица, можно использовать как новые (предпочтительно), так и старые параметры API (подробное описание вы найдете в параметрах query string). Старый API позволяет распознавать пол, возраст, эмоции и страну, в то время как новый API обеспечивает распознавание пола, возраста, эмоций, страны, бороды и очков. Каждый атрибут лица (пол, возраст, эмоции, страна, борода или очки) может быть упомянут только один раз в запросе, в новом или старом формате API.

Параметры query string:

- `''detector''`: детектор лиц, который будет применен к изображению. Может принимать значение `nnd` (обычный детектор) или `normalized` (детектор принимает нормализованное изображение лица, при этом стадия обнаружения лица пропускается).
- `"gender"`: логический тип, включает распознавание пола (старый API).
- `"age"`: логический тип, включает распознавание возраста (старый API).
- `''emotions''`: логический тип, включает распознавание эмоций (старый API).
- `''facen''`: логический тип, в запросе к `findface-extraction-api` будут включены параметры `need_facen` (извлечь вектор признаков) и `need_normalized` (получить нормализованное изображение лица). При этом полный ответ `findface-extraction-api` будет сохранен в `memcached` с временным UUID, который будет записан в поле `id` ответа.
- `''countries47''`: логический тип, включает распознавание страны (старый API).
- `"autorotate"`: логический тип, автоматически поворачивает исходное изображение в 4-х различных направлениях и возвращает лица, обнаруженные в каждом из них.
- `"return_facen"`: логический тип, возвращает в ответе вектор признаков лица. Для этого в файле конфигурации `findface-sf-api` также должна быть включена опция `allow-return-facen`.

- `"attribute"`: массив строк в формате `["gender", "age", "emotions", "countries47", "beard", "glasses3"]`, включает распознавание атрибутов лица, переданных в массиве (новый API).

Параметры в теле запроса:

Изображение в виде файла с mime-типом `image/jpeg`, `image/png`, `image/webp` или ссылка на интернет-изображение с mime-типом `text/x-url`.

Возвращает:

- список координат рамок вокруг обнаруженных лиц;
- временный UUID результата детекции (`id`, если был передан активный параметр `facen`);

> **Важно:** При написании кода необходимо предусмотреть проверку данного id на актуальность при обращении к нему впоследствии, поскольку со временем он становится неактуальным. В последнем случае повторите детекцию лица.

- вектор признаков (если активирован `return_facen`);
- пол (если активирован `gender`): `male` или `female`, со значением уверенности алгоритма в результате (`"score"`);
- возраст (если активирован `age`): количество лет;
- эмоции (если активирован‘‘emotions‘‘): 6 базовых эмоций + нейтральная эмоция (`angry`, `disgust`, `fear`, `happy`, `sad`, `surprise`, `neutral`) со значением уверенности алгоритма в выражении каждой эмоции;
- страны (если активирован `countries47`): вероятные страны происхождения со значением уверенности алгоритма в результате;
- атрибуты (если переданы): пол (`male` или `female`), возраст (количество лет), эмоции (преобладающая эмоция), вероятные страны происхождения, борода (`beard` или `none`), очки (`sun`, `eye` или `none`), вместе со значением уверенности алгоритма в результате;
- ориентация.

Примеры

Запрос №1. Одновременное использование старого и нового API

```
curl -i -X POST 'http://127.0.0.1:18411/v2/detect?facen=on&age=on&gender=on&emotions=on&
↪attribute=glasses3' -H 'Content-Type: image/jpeg' --data-binary @sample.jpg
HTTP/1.1 100 Continue
```

Ответ

```
HTTP/1.1 200 OK
Content-Type: application/json
X-Request-Id: SF:BpLnfgDs
Date: Thu, 23 May 2019 12:00:22 GMT
Content-Length: 713

{
    "faces": [
        {
            "id": "bjj8mlhjisgjrk6hj1v0",
            "bbox": { "left": 595, "top": 127, "right": 812, "bottom": 344 },
            "features": {
                "gender": { "gender": "FEMALE", "score": 0.9998938 },
                "age": 25,
                "score": -0.000696103,
                "emotions": [
                    { "emotion": "neutral", "score": 0.99958 },
                    { "emotion": "sad", "score": 0.0004020365 },
                    { "emotion": "happy", "score": 8.603454e-06 },
                    { "emotion": "surprise", "score": 8.076766e-06 },
                    { "emotion": "disgust", "score": 6.6535216e-07 },
                    { "emotion": "angry", "score": 6.1434775e-07 },
                    { "emotion": "fear", "score": 7.3372125e-10 }
                ],
                "attributes": {
                    "glasses3": {
                        "attribute": "glasses3",
                        "model": "glasses3.v0",
                        "result": [
                            { "confidence": 0.99958307, "name": "none" },
                            { "confidence": 0.00033243417, "name": "eye" },
                            { "confidence": 8.4465064e-05, "name": "sun" }
                        ]
                    }
                }
            }
        }
    ],
    "orientation": 1
}
```

Запрос №2. Использование нового API

```
curl -s -X POST 'http://master:18411/v2/detect?attribute=countries47&attribute=gender' -H 'Content-
↪Type: image/jpeg' --data-binary @pasha.jpg | jq
{
  "faces": [
    {
      "bbox": {
        "left": 1019,
        "top": 1138,
        "right": 1666,
        "bottom": 2041
      },
      "features": {
```

(continues on next page)

(продолжение с предыдущей страницы)


```
        "score": -0.00035252835,
        "attributes": {
          "countries47": {
            "extractor": "countries47",
            "model": "countries47.v1",
            "result": [
              {
                "confidence": 0.24693502,
                "name": "ASTL"
              },
              {
                "confidence": 0.0913519,
                "name": "POL"
              },
              {
                "confidence": 0.05358066,
                "name": "ARG"
              },
              {
                "confidence": 0.05128677,
                "name": "UKR"
              },
              {
                "confidence": 0.043531597,
                "name": "GER"
              },
              {
                "confidence": 0.03273358,
                "name": "CZEC"
              },
              {
                "confidence": 0.03272725,
                "name": "LEBN"
              },
              {
                "confidence": 0.02950912,
                "name": "CHIL"
              },
              {
                "confidence": 0.027654657,
                "name": "RUS"
              },
              {
                "confidence": 0.026015146,
                "name": "SAFR"
              },
              {
                "confidence": 0.025724495,
                "name": "GRBR"
              },
              {
                "confidence": 0.025504896,
                "name": "CSTR"
              },
              {
                "confidence": 0.023839278,
```


(continues on next page)

(продолжение с предыдущей страницы)


```
      "name": "KOR"
    },
    {
      "confidence": 0.022917742,
      "name": "ISRL"
    },
    {
      "confidence": 0.015377127,
      "name": "PKST"
    },
    {
      "confidence": 0.015136372,
      "name": "TRKY"
    },
    {
      "confidence": 0.014596664,
      "name": "ROM"
    },
    {
      "confidence": 0.014106703,
      "name": "ELSL"
    },
    {
      "confidence": 0.013242814,
      "name": "IND"
    },
    {
      "confidence": 0.012512706,
      "name": "IDSA"
    },
    {
      "confidence": 0.010475861,
      "name": "SARB"
    },
    {
      "confidence": 0.010281001,
      "name": "GUAT"
    },
    {
      "confidence": 0.009767002,
      "name": "GRC"
    },
    {
      "confidence": 0.009712666,
      "name": "VENZ"
    },
    {
      "confidence": 0.0096269725,
      "name": "JAM"
    },
    {
      "confidence": 0.009401413,
      "name": "HAT"
    },
    {
      "confidence": 0.008189098,
```


(continues on next page)

(продолжение с предыдущей страницы)


```
            "name": "CUBA"
          },
          {
            "confidence": 0.007926449,
            "name": "GHAN"
          },
          {
            "confidence": 0.0077127796,
            "name": "HOND"
          },
          {
            "confidence": 0.007496704,
            "name": "NRA"
          },
          {
            "confidence": 0.0072590904,
            "name": "ECUA"
          },
          {
            "confidence": 0.007247953,
            "name": "CHIN"
          },
          {
            "confidence": 0.0072472636,
            "name": "PHIL"
          },
          {
            "confidence": 0.006719906,
            "name": "JPN"
          },
          {
            "confidence": 0.0062700314,
            "name": "DOMR"
          },
          {
            "confidence": 0.005724779,
            "name": "MEX"
          },
          {
            "confidence": 0.005676317,
            "name": "PERU"
          },
          {
            "confidence": 0.0055375276,
            "name": "BRZL"
          },
          {
            "confidence": 0.005422363,
            "name": "EGYP"
          },
          {
            "confidence": 0.005258461,
            "name": "MLAS"
          },
          {
            "confidence": 0.004999833,
```


(continues on next page)

(продолжение с предыдущей страницы)


```
              "name": "KENY"
            },
            {
              "confidence": 0.004916244,
              "name": "TWAN"
            },
            {
              "confidence": 0.0045134826,
              "name": "VTNM"
            },
            {
              "confidence": 0.004322668,
              "name": "HNK"
            },
            {
              "confidence": 0.004172176,
              "name": "THAI"
            },
            {
              "confidence": 0.0031889428,
              "name": "TRIN"
            },
            {
              "confidence": 0.0026484467,
              "name": "NEP"
            }
          ]
        },
        "gender": {
          "extractor": "gender",
          "model": "gender.v2",
          "result": [
            {
              "confidence": 0.9999999,
              "name": "male"
            },
            {
              "confidence": 7.935678e-08,
              "name": "female"
            }
          ]
        },
        "quality": {
          "extractor": "quality",
          "model": "quality.v0",
          "result": 0.00035252835
        }
      }
    }
  }
 ],
 "orientation": 1
}
```

6.2.2 Извлечение результата детекции из memcached

```
/detect/:id GET
```

Данный метод служит для получения параметров обнаруженного лица, включая его вектор признаков, по временному UUID's в memcached.

Параметры в сегментах пути:

- :id: временный UUID обнаруженного лица в memcached.

Возвращает:

Представление JSON результата детекции.

Пример

Запрос

```
curl -i -X GET 'http://127.0.0.1:18411/v2/detect/bg2gu31jisghl6pee09g'
```

Ответ:

```
{
    "bbox": { "bottom": 343, "left": 593, "right": 824, "top": 112 },
    "features": {
        "age": 26.096783,
        "emotions": [
            { "emotion": "neutral", "score": 0.9094986 },
            { "emotion": "happy", "score": 0.11464329 },
            { "emotion": "sad", "score": 0.005675929 },
            { "emotion": "surprise", "score": -0.02556022 },
            { "emotion": "fear", "score": -0.14499822 },
            { "emotion": "angry", "score": -0.19491306 },
            { "emotion": "disgust", "score": -0.31617728 }
        ],
        "gender": { "gender": "FEMALE", "score": -2.7309942 },
        "score": -0.000696103
    },
    "id": "bg2gu31jisghl6pee09g"
}
```

6.2.3 Создание результата детекции из ответа findface-extraction-api

```
/detect/:id POST
```

Данный метод создает результат детекции из ответа findface-extraction-api.

Параметры в сегментах пути:

- `:id`: задайте UUID, под которым в memcached будет хранится результат детекции.

Возвращает:

Пустой JSON в случае успеха.

Пример

Запрос

```
$ curl -i -X POST 'http://127.0.0.1:18411/v2/detect/bg2gu31jisghl6peea9g' -H 'Content-Type:␣
↪application/json' --data-binary '@extapi-face.json'
```

Ответ:

```
HTTP/1.1 200 OK
Content-Type: application/json
X-Request-Id: jFSBuSPm
Date: Wed, 05 Dec 2018 08:08:56 GMT
Content-Length: 2

{}
```

6.2.4 Получение списка галерей

```
/galleries GET
```

Данный метод возвращает список всех существующих галерей в биометрической базе данных.

Параметры:

Отсутствуют.

Возвращает:

Словарь JSON со списком имен (`name`) галерей.

Пример

Запрос

```
GET /v2/galleries HTTP/1.1
Host: 172.17.47.19:18411
```

Ответ

```
HTTP/1.1 200 OK
Content-Type: application/json; charset=utf-8
Date: Fri, 02 Feb 2018 10:11:43 GMT
Content-Length: 35

{"galleries":[{"name":"sandbox"}]}
```

6.2.5 Создание галереи в базе данных

```
/galleries/:gallery POST
```

Данный метод создает новую галерею с заданным именем.

Параметры в сегментах пути:

`:gallery`: имя новой галереи. Может содержать латинские буквы, числа, знак подчеркивания и минус (`[a-zA-Z0-9_-]+`) и не может быть длиннее 48 символов.

Возвращает:

- Пустой JSON в случае успеха.
- JSON с описанием ошибки при сбое.

Пример

Запрос

```
POST /v2/galleries/newone HTTP/1.1
Host: 172.17.47.19:18411
```

Ответ

```
HTTP/1.1 200 OK
Content-Type: application/json; charset=utf-8
Date: Fri, 02 Feb 2018 10:18:01 GMT
Content-Length: 2

{}
```

6.2.6 Получение информации о галерее

```
/galleries/:gallery GET
```

Данный метод проверяет, существует ли галерея, и возвращает количество лиц в ней.

Параметры в сегментах пути:

`:gallery`: имя галереи.

Возвращает:

- Словарь JSON с количеством лиц и именем галереи в случае успеха.
- JSON с описанием ошибки при сбое.

Пример

Запрос

```
curl -i -X GET 'http://127.0.0.1:18411/v2/galleries/hello'
```

Ответ

```
HTTP/1.1 200 OK
Content-Type: application/json
X-Request-Id: Ard3exjn
Date: Wed, 05 Dec 2018 08:17:54 GMT
Content-Length: 29

{ "faces": 123, "name": "hello" }
```

6.2.7 Удаление галереи

```
/galleries/:gallery DELETE
```

Данный метод удаляет заданную галерею и все лица в ней.

Параметры в сегментах пути:

`:gallery`: имя галереи, подлежащей удалению.

Возвращает:

- Пустой JSON в случае успеха.
- JSON с описанием ошибки при сбое.

Пример

Запрос

```
DELETE /v2/galleries/newone HTTP/1.1
Host: 172.17.47.19:18411
```

Ответ

```
HTTP/1.1 200 OK
Content-Type: application/json; charset=utf-8
Date: Fri, 02 Feb 2018 10:18:01 GMT
Content-Length: 2

{}
```

6.2.8 Добавление лица в галерею

```
/galleries/:gallery/faces/:id POST
```

Данный метод берет обнаруженное лицо из `memcached` или лицо из галереи, чтобы затем добавить его вместе с вектором признаков в указанную в параметрах query string галерею базы биометрических данных с присвоением указанного пользовательского id. Вместе с лицом вы также можете добавить метаданные, которые являются уникальными для его обладателя, например, идентификатор или имя.

Параметры в сегментах пути:

- `:gallery`: имя галереи, в которую добавляется лицо.
- `:id`: постоянный идентификатор лица в галерее, uint64.

Параметры в теле запроса:

- `"from"`: временный UUID обнаруженного лица в `memcached` (`"from": "detection:<id>"`) или id лица в галерее (`"from: "face:<gallery>/<id>"`).
- `"meta"` [опционально]: метаданные, такие как имя обладателя лица, сведения об исходном изображении, дата и время детекции и т. д., словарь.

Возвращает:

- Представление JSON добавленного лица в случае успеха.
- Ошибка в случае неудачной попытки.

Пример

Запрос

```
curl -i -X POST http://127.0.0.1:18411/v2/galleries/hello/faces/123/ -H 'Content-Type: application/
↪json' --data-binary '@-' <<EOF
{
    "from": "detection:bg2gu31jisghl6pee09g",
    "meta": {
        "camera": "openspace",
        "labels": ["foo", "bar"],
        "timestamp": "1543837276"
    }
```

(continues on next page)

(продолжение с предыдущей страницы)

```
}
EOF
```

Ответ

```
HTTP/1.1 200 OK
Content-Type: application/json
X-Request-Id: SF:OSSKbJg3
Date: Wed, 05 Dec 2018 08:27:59 GMT
Content-Length: 555

{
    "features": {
        "age": 26.096783,
        "emotions": [
            { "emotion": "neutral", "score": 0.9094986 },
            { "emotion": "happy", "score": 0.11464329 },
            { "emotion": "sad", "score": 0.005675929 },
            { "emotion": "surprise", "score": -0.02556022 },
            { "emotion": "fear", "score": -0.14499822 },
            { "emotion": "angry", "score": -0.19491306 },
            { "emotion": "disgust", "score": -0.31617728 }
        ],
        "gender": { "gender": "FEMALE", "score": -2.7309942 },
        "score": -0.000696103
    },
    "id": { "face": 123, "gallery": "hello" },
    "meta": {
        "camera": "openspace",
        "labels": ["foo", "bar"],
        "timestamp": "1543837276"
    },
    "normalized_id": "123_bg2hcupjisghl6pee0ag.png"
}
```

6.2.9 Получение информации о лице из галереи

```
/galleries/:gallery/faces/:id GET
```

Возвращает лицо из базы данных по id.

Параметры в сегментах пути:

- `:gallery`: имя галереи, в которой хранится лицо.
- `:id`: идентификатор лица в галерее, uint64.

Возвращает:

- Представление лица в JSON в случае успеха.
- Ошибка в случае неудачной попытки.

Пример

Запрос

```
curl -s 'http://172.17.47.19:18411/v2/galleries/galleryname/faces/2' | jq
```

Ответ


```
{
  "id": {
    "gallery": "galleryname",
    "face": 2
  },
  "features": {
    "gender": {
      "gender": "FEMALE",
      "score": -2.6415923
    },
    "age": 26.04833,
    "score": 0.9999999,
    "emotions": [
      {
        "emotion": "neutral",
        "score": 0.99958
      },
      {
        "emotion": "sad",
        "score": 0.0004020398
      },
      {
        "emotion": "happy",
        "score": 8.603504e-06
      },
      {
        "emotion": "surprise",
        "score": 8.076798e-06
      },
      {
        "emotion": "disgust",
        "score": 6.653509e-07
      },
      {
        "emotion": "angry",
        "score": 6.14346e-07
      },
      {
        "emotion": "fear",
        "score": 7.33713e-10
      }
    ]
  },
  "meta": {
    "timestamp": 2
  },
  "normalized_id": "2_bd323f5t8f66ph0eafp0.png"
}
```

6.2.10 Удаление лица из галереи

```
/galleries/:gallery/faces/:id DELETE
```

Удаляет лицо из базы данных по id.

Параметры в сегментах пути:

- `:gallery`: имя галереи, из которой удаляется лицо.
- `:id`: идентификатор лица в галерее, uint64.

Возвращает:

- Пустой JSON в случае успеха.
- Ошибка в случае неудачной попытки.

Пример

Запрос

```
curl -s -X DELETE 'http://172.17.47.19:18411/v2/galleries/galleryname/faces/1' | jq
```

Ответ

```
{}
```

6.2.11 Обновление метаданных лица в галерее

```
/galleries/:gallery/faces/:id PATCH
```

Заменяет параметры лица в базе данных по id.

Параметры в сегментах пути:

- `:gallery`: имя галереи.
- `:id`: идентификатор лица в галерее, uint64.

Параметры в теле запроса

- `"meta"`: новые метаданные лица, словарь.

Возвращает:

- Представление данных измененного лица в JSON в случае успеха.
- Ошибка в случае неудачной попытки.

Пример

Запрос

```
curl -s -X PATCH -H 'Content-Type: application/json' --data '{"meta":{"timestamp":2}}' 'http://172.
↪17.47.19:18411/v2/galleries/galleryname/faces/2' | jq
```

Ответ


```
{
  "id": {
    "gallery": "galleryname",
    "face": 2
  },
  "features": {
    "gender": {
      "gender": "FEMALE",
      "score": -2.6415923
    },
    "age": 26.04833,
    "score": 0.9999999,
    "emotions": [
      {
        "emotion": "neutral",
        "score": 0.99958
      },
      {
        "emotion": "sad",
        "score": 0.0004020398
      },
      {
        "emotion": "happy",
        "score": 8.603504e-06
      },
      {
        "emotion": "surprise",
        "score": 8.076798e-06
      },
      {
        "emotion": "disgust",
        "score": 6.653509e-07
      },
      {
        "emotion": "angry",
        "score": 6.14346e-07
      },
      {
        "emotion": "fear",
        "score": 7.33713e-10
      }
    ]
  },
  "meta": {
    "timestamp": 2
  },
```


(continues on next page)

(продолжение с предыдущей страницы)

```
  "normalized_id": "2_bd323f5t8f66ph0eafp0.png"
}
```

6.2.12 Сравнение лиц

```
/verify POST
```

Данный метод сравнивает два лица, проверяя, принадлежит ли изображенное на них лицо одному и тому же человеку, и возвращает степень уверенности алгоритма в совпадении лиц (степень схожести).

Параметры query string:

- `"face1"`: первое лицо, результат работы `/detect POST` (хранится в `memcached`) или сохраненное в базе данных.
- `"face2"`: второе лицо, результат работы /detect POST (хранится в memcached) или сохраненное в базе данных.

Возвращает:

Степень уверенности алгоритма в совпадении лиц.

Пример

Запрос №1. Сравнение 2-х результатов детекции

```
curl -s 'http://172.17.47.19:18411/v2/verify?face1=detection:bd3hv4tt8f66ph0eag1g&
↪face2=detection:bd3hv8dt8f66ph0eag2g' | jq
```

Ответ

```
{
  "confidence": 0.92764723
}
```

Запрос №2. Сравнение результата детекции с лицом из галереи

```
curl -s 'http://172.17.47.19:18411/v2/verify?face1=detection:bd3hv4tt8f66ph0eag1g&
↪face2=face:galleryname/2' | jq
```

Ответ

```
{
  "confidence": 0.999996
}
```

6.2.13 Извлечение данных из галереи с использованием фильтров. Поиск лиц

`/v2/galleries/:gallery/faces GET`

Данный метод выполняет поиск по галерее с использованием фильтров, заданных в query string.

Параметры в сегментах пути:

`:gallery`: имя галереи, по которой выполняется поиск.

Параметры query string:

- `?limit=`: (обязательный) максимальное количество лиц в ответе.
- `?sort=`: порядок сортировки. Передайте одно из следующих значений: `id`: в порядке возрастания id, `-id`: в порядке убывания id, `-confidence`: в порядке убывания схожести лиц (только в случае поиска по вектору признаков).
- `?page=<page>`: показать следующую страницу с результатами поиска. Значение `<page>` возвращается в ответе вместе с предыдущей страницей как `next_page`.
- `?ignore_errors`: 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.
- `?meta:in:meta1=val1&meta:in:meta1=val2&...`: выбрать лицо, если строка `meta1` в метаданных имеет одно из значений `val1`/`val2`/ ... и т. д. (*uint64*, *string*).
- `?meta:gte:meta1=val1`: выбрать все лица, у которых строка `meta1` в метаданных больше или равна значению `val1` (*uint64*).
- `?meta:lte:meta1=val1`: выбрать все лица, у которых строка `meta1` в метаданных меньше или равна значению `val1` (*uint64*).
- `?id:in=value_id`: выбрать все лица с `id`, равным `value_id`.
- `?id:gte=value_id`: выбрать все лица с `id`, большим или равным `value_id`.
- `?id:lte=value_id`: выбрать все лица с `id`, меньшим или равным `value_id`.
- `?meta:subset:meta1=val1&meta:subset:meta1=val2&...`: выбрать лицо, если строка `meta1` в метаданных включает в себя все значения `val1`, `val2`, ... и т. д. (*[]string*).
- `?<id>=<confidence>`: задает вектор признаков для поиска по базе данных через параметр `<id>`, а также устанавливает пороговую схожесть лиц в результатах поиска `<confidence>`. Параметр `<id>` может быть представлен как id лица в базе данных (`face:<gallery>/<db_id>`), так и id лица, являющегося результатом детекции (метод `/detect POST`) и хранящегося в `memcached` (`detection:<memcached_id>`). Значение `<confidence>` от 0 до 1.

Возвращает:

Массив лиц в представлении JSON. Лица в ответе по умолчанию отсортированы по id, а в случае задания вектора признаков – по убыванию степени схожести.

Формат ответа:

```
{
    "faces": [
      {
        ... face 1 data ...
        "confidence": 0.123 // if you search for a feature vector
      },
      {
        ... face 2 data ...
        "confidence": 0.123 // if you search for a feature vector
      },
      ...
    ],
    "next_page": "vgszk2bkexbl" // next page cursor
}
```

Параметр `next_page` – URL-safe строка, которую нужно передать в `?page=` в следующем запросе, чтобы получить следующую страницу результатов. Разбиение по страницам доступно только при выключенной фильтрации по вектору признаков.

Запрос №1. Идентификация лица (поиск лица в галерее)

```
curl -s 'http://172.17.47.19:18411/v2/galleries/galleryname/faces?detection:bd3hv4tt8f66ph0eag1g=0.
↪5&limit=1' | jq
```

Ответ

```
{
  "faces": [
    {
      "id": {
        "gallery": "galleryname",
        "face": 2
      },
      "features": {
        "gender": {
          "gender": "FEMALE",
          "score": -2.6415923
        },
        "age": 26.04833,
        "score": 0.9999999,
        "emotions": [
          {
            "emotion": "neutral",
            "score": 0.99958
          },
          {
            "emotion": "sad",
            "score": 0.0004020398
          },
          {
            "emotion": "happy",
            "score": 8.603504e-06
          },
          {
```

(continues on next page)

(продолжение с предыдущей страницы)


```
            "emotion": "surprise",
            "score": 8.076798e-06
          },
          {
            "emotion": "disgust",
            "score": 6.653509e-07
          },
          {
            "emotion": "angry",
            "score": 6.14346e-07
          },
          {
            "emotion": "fear",
            "score": 7.33713e-10
          }
        ]
       },
      "meta": {},
      "normalized_id": "2_bd323f5t8f66ph0eafp0.png",
      "confidence": 0.9999
    }
  ],
  "next_page": "There are more than 1 results, but pagination is not supported when filtering by␣
↪FaceN"
}
```


Запрос №2. Получение списка лиц в галерее

```
curl -s 'http://172.17.47.19:18411/v2/galleries/galleryname/faces?limit=2' | jq
```

Ответ


```
{
  "faces": [
    {
      "id": {
        "gallery": "galleryname",
        "face": 1
      },
      "features": {
        "gender": {
          "gender": "FEMALE",
          "score": -2.6415923
        },
        "age": 26.04833,
        "score": 0.9999999,
        "emotions": [
          {
            "emotion": "neutral",
            "score": 0.99958
          },
          {
            "emotion": "sad",
```


(continues on next page)

(продолжение с предыдущей страницы)


```
          "score": 0.0004020398
        },
        {
          "emotion": "happy",
          "score": 8.603504e-06
        },
        {
          "emotion": "surprise",
          "score": 8.076798e-06
        },
        {
          "emotion": "disgust",
          "score": 6.653509e-07
        },
        {
          "emotion": "angry",
          "score": 6.14346e-07
        },
        {
          "emotion": "fear",
          "score": 7.33713e-10
        }
      ]
    },
    "meta": {},
    "normalized_id": "1_bd321tlt8f66ph0eaflg.png"
  },
  {
    "id": {
      "gallery": "galleryname",
      "face": 2
    },
    "features": {
      "gender": {
        "gender": "FEMALE",
        "score": -2.6415923
      },
      "age": 26.04833,
      "score": 0.9999999,
      "emotions": [
        {
          "emotion": "neutral",
          "score": 0.99958
        },
        {
          "emotion": "sad",
          "score": 0.0004020398
        },
        {
          "emotion": "happy",
          "score": 8.603504e-06
        },
        {
          "emotion": "surprise",
          "score": 8.076798e-06
        },
```


(continues on next page)

(продолжение с предыдущей страницы)


```
          {
            "emotion": "disgust",
            "score": 6.653509e-07
          },
          {
            "emotion": "angry",
            "score": 6.14346e-07
          },
          {
            "emotion": "fear",
            "score": 7.33713e-10
          }
        ]
      },
      "meta": {},
      "normalized_id": "2_bd323f5t8f66ph0eafp0.png"
    }
  ],
  "next_page": "3"
}
```


Запрос №3. Расширенная идентификация лица

```
curl -i -X GET http://127.0.0.1:18411/v2/galleries/history/faces/?limit=5&meta:in:camera=openspace&
↪meta:in:camera=entrance&meta:lte:timestamp=1543845934&meta:gte:timestamp=1514801655&
↪detection:bg2gu31jisghl6pee09g=0.4 | jq
```

Ответ


```
HTTP/1.1 200 OK
Content-Type: application/json
X-Request-Id: SF:ibKVYpcb
Date: Wed, 05 Dec 2018 08:37:33 GMT
Transfer-Encoding: chunked

{
    "faces": [
        {
            "confidence": 0.6026,
            "features": { "score": 1 },
            "id": { "face": 4141715030051545133, "gallery": "history" },
            "meta": {
                "bbox": "[607, 802, 738, 933]",
                "camera": "openspace",
                "is_friend": 0,
                "labels": [],
                "score": 9999999999998079040,
                "timestamp": 1542909082
            },
            "normalized_id": "4141715030051545133_bfrep71jisghl6pedvk0.png"
        },
        {
            "confidence": 0.5325,
```


(continues on next page)

(продолжение с предыдущей страницы)


```
            "features": { "score": 1 },
            "id": { "face": 4141715086422990894, "gallery": "history" },
            "meta": {
                "bbox": "[741, 905, 953, 1117]",
                "camera": "openspace",
                "is_friend": 0,
                "labels": [],
                "score": 9999999999993877300,
                "timestamp": 1542909103
            },
            "normalized_id": "4141715086422990894_bfrepc9jisghl6pedvl0.png"
        },
        {
            "confidence": 0.531,
            "features": {
                "age": 41.2622,
                "gender": { "gender": "FEMALE", "score": -0.880698 },
                "score": 1
            },
            "id": { "face": 4141716493024780347, "gallery": "history" },
            "meta": {
                "bbox": "[90, 869, 166, 945]",
                "camera": "openspace",
                "is_friend": 0,
                "labels": [],
                "score": 10000000000000000013,
                "timestamp": 1542909627
            },
            "normalized_id": "4141716493024780347_bfretf9jisghl6pee020.png"
        },
        {
            "confidence": 0.5236,
            "features": {
                "age": 48.949913,
                "gender": { "gender": "FEMALE", "score": -0.7653318 },
                "score": 1
            },
            "id": { "face": 4141716498393489468, "gallery": "history" },
            "meta": {
                "bbox": "[56, 853, 125, 923]",
                "camera": "openspace",
                "is_friend": 0,
                "labels": [],
                "score": 9999999999999999053,
                "timestamp": 1542909629
            },
            "normalized_id": "4141716498393489468_bfretg1jisghl6pee030.png"
        },
        {
            "confidence": 0.5212,
            "features": {
                "age": 33.3112,
                "gender": { "gender": "MALE", "score": 1.9504981 },
                "score": 1
            },
            "id": { "face": 4141715338752319538, "gallery": "history" },
```


(continues on next page)

(продолжение с предыдущей страницы)


```
            "meta": {
                "bbox": "[-36, 862, 60, 958]",
                "camera": "openspace",
                "is_friend": 0,
                "labels": [],
                "score": 9999999999999999425,
                "timestamp": 1542909197
            },
            "normalized_id": "4141715338752319538_bfreq4pjisghl6pedvp0.png"
        }
    ],
    "next_page": "There are more than 5 results, but pagination is not supported when filtering by␣
↪FaceN"
}
```


Совет: Вы также можете найти документацию по API для анализа и распознавания биометрических данных на нашем вебсайте и по адресу `http://<findface-sf-api_ip>:18411/v2/docs`.

Глава 7

HTTP API для управления видеодетекцией лиц

7.1 Как пользоваться HTTP API для управления видеодетекцией лиц

В этом разделе:

7.1.1 Точка доступа

Все запросы HTTP API для управления видеодетекцией лиц нужно отправлять на адрес http://<findface-video-manager IP address>:18810/. Запросы обрабатываются компонентом `findface-video-manager`.

7.1.2 Объект типа job (задание)

Объект `job` представляет собой задание на обработку видеопотока, выдаваемое компонентом `findface-video-manager` компоненту `findface-video-worker`.

Объект `job` имеет следующие атрибуты:

- `id`: id job-задания, установленный пользователем.
- `stream_url`: URL/адрес видеопотока или файла для обработки.

- `labels`: метки, по которым будет осуществляться обработка обнаруженных лиц в компоненте `findface-facerouter`.
- `single_pass`: если true (по умолчанию false), то не перезапускать обработку потока в случае ошибки.
- `router_url`: IP-адрес и порт компонента `findface-facerouter`, на который компонент `findface-video_worker` будет отправлять обнаруженные лица для обработки.
- `status`: статус job-задания.
- `status_msg`: дополнительная информация о статусе job-задания.
- `statistic`: статистика выполнения задания (продолжительность использования задания, количество отправленных лиц).
- `worker_id`: id экземпляра `findface-video-worker`, выполняющего job-задание.

7.1.3 Сообщения об ошибках

Если метод выполнить не удается, Сервер возвращает ответ с кодом HTTP, отличном от 200, а также тело ответа в формате JSON, содержащее описание ошибки. Тело ответа всегда содержит хотя бы 2 поля — `code` и `desc`.

- `code` — это код ошибки в виде `CAPS_AND_UNDERSCORES`, который может быть использован для автоматического преобразования.
- `desc` — это описание ошибки, предназначенное для прочтения человеком.

Полный список ошибок

Код ошибки	Описание	Код HTTP
`UNKNOWN_ERROR`	Ошибка неизвестного происхождения.	500
`BAD_REQUEST`	Запрос не может быть прочитан, или некоторые параметры метода неверно заданы.	400
`CONFLICT`	Конфликт.	409
`NOT_FOUND`	Job-задание не найдено.	404
`DELETING`	Выполняется запрошенное ранее удаление задания.	423

7.2 Методы для управления видеодетекцией лиц

В этом разделе:

- *Перезапуск задания*

7.2.1 Создание job-задания

```
POST /job/:id
```

Создает задание на обработку видеопотока (job) для компонента `findface-video-worker`.

Параметры в сегментах пути

`id`: id job-задания

Параметры в теле запроса:

- `stream_url`: URL/адрес видеопотока или файла для обработки.
- `labels`: метки, по которым будет осуществляться обработка обнаруженных лиц в компоненте `findface-facerouter`.
- `single_pass`: если true (по умолчанию false), то не перезапускать обработку потока в случае ошибки.
- Параметры конфигурации компонента `findface-video-manager`, которые должны быть уникальны для обрабатываемого видеоизображения по сравнению со значениями, заданными по умолчанию в файле конфигурации `findface-video-manager`.

Возвращает:

Объект `job`: все, что было передано в запросе, а также параметры объекта, предназначенные только для чтения.

Пример

Запрос

```
curl -s 'http://localhost:18810/job/myid-123' --data '{"stream_url":"http://1.2.3.4/stream.mp4",
↪"labels": {"district": "SVAO"}}' | jq
```

Ответ

```
{
  "id": "myid-123",
  "stream_url": "http://1.2.3.4/stream.mp4",
  "labels": {
    "district": "SVAO"
  },
  "router_url": "http://localhost:1514/",
  "single_pass": false,
  "status": "AWAITING",
```

(continues on next page)

(продолжение с предыдущей страницы)

```
  "status_msg": "",
  "statistic": {
    "processed_duration": 0,
    "faces_posted": 0
  },
  "worker_id": ""
}
```

7.2.2 Получение списка всех заданий

```
GET /jobs
```

Данный метод возвращает список существующих job-заданий.

Параметры:

Отсутствуют.

Возвращает:

Список всех заданий в представлении JSON.

Пример

Запрос

```
curl -s 'http://localhost:18810/jobs' | jq
```

Ответ

```
[
    {
        "id": "b9c73bhg74hnekpaa0o0",
        "stream_url": "http://1.2.3.4/stream.mp4",
        "labels": {
            "district": "SVAO"
        },
        "router_url": "http://localhost:1514/",
        "single_pass": false,
        "status": "AWAITING",
        "status_msg": "",
        "worker_id": ""
    },
    {
        "id": "b9c73rhg74hnekpaa0og",
        "stream_url": "http://xxx.ru/stream.mp4",
        "labels": {
            "district": "ZAO"
```

(continues on next page)

(продолжение с предыдущей страницы)

```
        },
        "router_url": "http://localhost:1514/",
        "single_pass": false,
        "status": "AWAITING",
        "status_msg": "",
        "worker_id": ""
    }
]
```

7.2.3 Получение задания по id

```
GET /job/:id
```

Данный метод возвращает параметры задания по id.

Параметры в сегментах пути:

`id`: id job-задания.

Возвращает:

Представление JSON объекта `job`.

Пример

Запрос

```
curl -s 'http://localhost:18810/job/b9c73rhg74hnekpaa0og' | jq
```

Ответ

```
{
    "id": "b9c73rhg74hnekpaa0og",
    "stream_url": "http://xxx.ru/stream.mp4",
    "labels": {
        "district": "ZAO"
    },
    "router_url": "http://localhost:1514/",
    "single_pass": false,
    "status": "AWAITING",
    "status_msg": "",
    "worker_id": ""
}
```

7.2.4 Удаление задания по id

```
DELETE /job/:id
```

Данный метод удаляет задание по id.

Параметры в сегментах пути:

`id`: id job-задания.

Возвращает:

Представление JSON удаленного объекта `job`.

Пример

Запрос

```
curl -s 'http://localhost:18810/job/myid-123' -X DELETE | jq
```

Ответ

```
{
  "id": "myid-123",
  "stream_url": "http://1.2.3.4/stream.mp4",
  "labels": {
    "district": "SVAO"
  },
  "router_url": "http://myrouter",
  "single_pass": false,
  "status": "DELETED",
  "status_msg": "",
  "statistic": {
    "processed_duration": 0,
    "faces_posted": 0
  },
  "worker_id": "b9kqns1g74hm6mbmhbqg"
}
```

7.2.5 Изменение задания

```
PATCH /job/:id
```

Изменяет параметры задания по id.

Параметры в сегментах пути:

`id`: id job-задания.

Параметры в теле запроса:

- `id`: id job-задания.
- `stream_url`: URL/адрес видеопотока или файла для обработки.
- `labels`: метки, по которым будет осуществляться обработка обнаруженных лиц в компоненте `findface-facerouter`.
- `single_pass`: если true (по умолчанию false), то не перезапускать обработку потока в случае ошибки.
- `router_url`: IP-адрес и порт компонента `findface-facerouter`, на который компонент `findface-video_worker` будет отправлять обнаруженные лица для обработки.
- `status`: статус job-задания.
- `status_msg`: дополнительная информация о статусе job-задания.
- `statistic`: статистика выполнения задания (продолжительность использования задания, количество отправленных лиц).
- `worker_id`: id экземпляра `findface-video-worker`, выполняющего job-задание.
- Параметры конфигурации компонента `find-video-manager` с новыми значениями. Данные значения имеют приоритет по сравнению с заданными в файле конфигурации `find-video-manager`.

Возвращает:

Представление JSON измененного объекта `job`.

Пример

Запрос

```
curl -s 'http://localhost:18810/job/myid-123' -X PATCH --data '{"router_url":"http://myrouter"}' |␣
↪jq
```


```
{
  "id": "myid-123",
  "stream_url": "http://1.2.3.4/stream.mp4",
  "labels": {
    "district": "SVAO"
  },
  "router_url": "http://myrouter",
  "single_pass": false,
  "status": "INPROGRESS",
  "status_msg": "",
  "statistic": {
    "processed_duration": 0,
    "faces_posted": 0
  },
  "worker_id": "b9kqns1g74hm6mbmhbqg"
}
```

7.2.6 Перезапуск задания

```
RESTART /job/:id
```

Данный метод перезапускает задание по id.

Параметры в сегментах пути:

`id`: id job-задания.

Возвращает:

HTTP/1.1 200 OK в случае успеха.

Пример

Запрос

```
curl -s -D - -X RESTART http://localhost:18810/job/1
```

Ответ

```
HTTP/1.1 200 OK
Content-Type: application/json
X-Request-Id: VbhV3vC5
Date: Tue, 24 Apr 2018 15:23:19 GMT
Content-Length: 0
3
```

Совет: Вы также можете найти документацию по API для управления видеодетекцией лиц по адресу `http://<findface-video-manager_ip>:18810/docs`.

Глава 8

Задание правил обработки лиц

В ходе настройки системы вы можете установить собственные правила обработки обнаруженных на видео лиц. Для этого понадобится написать плагин(ы) 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.

8.1 Настройка `findface-facerouter` на использование плагинов

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.

8.2 Принципы написания плагина

В этом разделе:

8.2.1 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).

8.2.2 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.

8.2.3 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*)

8.2.4 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)
```

8.3 Классы и методы

В этом разделе:

8.3.1 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 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

`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

8.3.2 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

8.3.3 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 `findface-video-worker`.
- `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` – 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).
- `sort` – string
- `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

8.3.4 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.

param value список строк с метаданными

Значение список строк или целых чисел

return имя фильтра (IN) и его значение.

rtype 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')]
```

8.3.5 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` и `desc`.

- `code` — это код ошибки в виде `CAPS_AND_UNDERSCORES`, который может быть использован для автоматического преобразования.
- `desc` — 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`.

8.4 Примеры

В следующих примерах наглядно продемонстрированы описанные выше принципы написания плагина, а также использование классов и методов.

1. В данном примере плагин запрашивает извлечение вектора признаков и информации об эмоциях, если обнаруженное лицо содержит метку `'emo'`, и записывает все полученную информацию в лог.

```
import logging

from ntech import sfapi_client

from facerouter.plugin import Plugin

logger = logging.getLogger(__name__)


class LogEmoPlugin(Plugin):
    async def preprocess(self, request, labels):
        if labels.get('emo'):
            return ('facen', 'emotions')

    async def process(self, request, photo, bbox, event_id, detection: sfapi_client.
↪DetectFace):
        logger.info('%r: %r', bbox, detection.features.get('emotions')[0]['emotion'])
        logger.info('%r: params: ', bbox)
        for param in request.params._fields:
            param_repr = repr(getattr(request.params, param))
            if len(param_repr) > 100:
                param_repr = param_repr[:97] + "..."
            logger.info("%r: %s", param, param_repr)


def activate(app, ctx, plugin_name, plugin_source):
    return LogEmoPlugin(ctx=ctx)
```

2. В данном примере плагин запрашивает извлечение вектора признаков, после этого сохраняет лицо в галерею `'ppl'` Tarantool. Если такой галереи не существует, она будет создана.

```
import logging
import PIL.Image
import time
from io import BytesIO

from ntech import sfapi_client

from facerouter.plugin import Plugin

logger = logging.getLogger(__name__)


class EnrollPlugin(Plugin):
    async def preprocess(self, request, labels):
        if labels.get('lol') == 'kek':
            return ('facen',)

    async def process(self, request, photo, bbox, event_id, detection: sfapi_client.
↪DetectFace):
```

(continues on next page)

(продолжение с предыдущей страницы)

```
        img = PIL.Image.open(BytesIO(photo))
        thumb = img.crop(bbox)
        fname = '/tmp/%x.jpeg' % (event_id,)
        thumb.save(fname)
        while True:
            try:
                await self.ctx.sfapi['ppl'].add(event_id, detection, meta={
                    'timestamp': int(time.time()),
                    'photo_hash': fname,
                })
            except sfapi_client.SFApiRemoteError as e:
                if e.code == "GALLERY_NOT_FOUND":
                    await self.ctx.sfapi['ppl'].create()
                else:
                    raise
            else:
                break
        logger.info('%r: %r %r', bbox, event_id, fname)


def activate(app, ctx, plugin_name, plugin_source):
    return EnrollPlugin(ctx=ctx)
```

Глава 9

Расширенный функционал

9.1 Прямые API-запросы к `findface-extraction-api`

Вы можете использовать HTTP API для извлечения данных напрямую из компонента `findface-extraction-api`.

Примечание: Будучи аналогом компонента `findface-sf-api` в том, что касается извлечения данных через API, компонент `findface-extraction-api` является более ресурсоемким. Данный компонент не может служить полноценной заменой `findface-sf-api`, поскольку не позволяет добавлять лица и работать с базой данных.

Совет: Нормализованные изображения, полученные с помощью компонента `findface-extraction-api`, подходят для обработки в компоненте `findface-sf-api`.

В этом разделе:

9.1.1 API-запросы

Компонент `findface-extraction-api` принимает POST-запросы по адресу `http://127.0.0.1:18666/`.

Существует 2 способа настроить формат тела запроса:

- `application/json`: тело запроса целиком состоит из данных в формате JSON.
- `multipart/form-data`: тело запроса включает в себя часть в формате JSON, содержащую собственно запрос, остальные части тела используются для передачи изображения.

Часть в формате JSON тела запроса представляет собой набор запросов:

```
{
    "requests": [request1, request2, .., requestN]
    "include_timings": true|false // include face processing timing in response, false by default
}
```

Каждый запрос в наборе имеет отношение к определенному изображению или области на изображении и принимает следующие параметры:

Важно: Чтобы включить распознавание атрибутов лица, можно использовать как новые (предпочтительно), так и старые параметры API. Старый API позволяет распознавать пол, возраст и эмоции, в то время как новый API обеспечивает распознавание пола, возраста, эмоций, страны, бороды и очков. Каждый атрибут лица (пол, возраст, эмоции, страна, борода или очки) может быть упомянут только один раз в запросе, в новом или старом формате API.

- `"image"`: загруженное изображение (используйте формат `multipart:часть` для указания соответствующей `части` тела запроса) или публичный URL с изображением (`http:`, `https:`).
- `"roi"`: интересующая область на изображении. Если данная область не задана, обрабатывается все изображение.
- `"detector"`: детектор лиц, который будет применен к изображению (`legacy`, `nnd` или `prenormalized`). Детектор `prenormalized` принимает нормализованные изображения лиц, при этом стадия обнаружения лица пропускается. Используйте `nnd`, если вам нужна оценка качества лиц (`"quality_estimator": true`).
- `"need_facen"`: если `true`, запрос возвращает в ответе вектор признаков.
- `"need_gender"`: возвращает пол (старый API).
- `"need_emotions"`: возвращает эмоции (старый API).
- `"need_age"`: возвращает возраст (старый API).
- `"need_normalized"`: возвращает нормализованное изображение лица в формате `base64`. Нормализованное изображение может быть снова отправлено в `findface-extraction-api` в режиме детектора `prenormalized`.
- `"auto_rotate"`: если `true`, автоматически поворачивает исходное изображение в 4-х различных ориентациях и возвращает лица, найденные в каждой из них. Работает, если `"detector": "nnd"` и `"quality_estimator": true`.
- `"attributes"`: массив строк в формате `["gender", "age", "emotions", "countries47", "beard", "glasses3"]`, включает распознавание атрибутов лица, переданных в массиве (новый API).

```
{
    "image": "http://static.findface.pro/sample.jpg",
    "roi": {"left": 0, "right": 1000, "top": 0, "bottom": 1000},
    "detector": "nnd",
    "need_facen": true,
    "need_gender": true,
    "need_emotions": true,
```

(continues on next page)

(продолжение с предыдущей страницы)

```
    "need_age": true,
    "need_normalized": true,
    "auto_rotate": true
}
```

9.1.2 Формат API-ответа

Типичный ответ компонента `findface-extraction-api` представляет собой набор ответов на соответствующие запросы, вложенные в основной API-запрос:

```
{
    "response": [response1, response2, .., responseN]
}
```

Каждый ответ в наборе содержит следующие данные в формате JSON:

- `"faces"`: набор лиц, обнаруженных на предоставленном изображении или в интересующей области изображения.
- `"error"`: ошибка обработки запроса (если имела место). Тело ошибки содержит код ошибки, который может быть интерпретирован автоматически (`"code"`) и описание ошибки, удобное для восприятия человеком (`"desc"`).
- `"facen_model"`: модель, использованная для извлечения биометрического образца, если `"need_facen": true`.
- `"timings"`: время обработки, если `"include_timings": true`.

```
{
    "faces": [face1, face2, .., faceN],
    "error": {
        "code": "IMAGE_DECODING_FAILED",
        "desc": "Failed to decode: reason"
    }
    "facen_model": "elderberry_576",
    "timings": ...

}
```

Для каждого лица в наборе указываются следующие данные:

- `"bbox"`: координаты рамки с лицом.
- `"detection_score"`: показатель качества лица или точность обнаружения лица (в зависимости от того, включена ли опция `quality_estimator` в файле `/etc/findface-extraction-api.ini`). 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.
- `"facen"`: вектор признаков.
- `"gender"`: информация о поле (MALE или FEMALE) с указанием точности оценки, если запрашивалась (старый API).
- `"age"`: оценка возраста, если запрашивалась (старый API).

- `"emotions"`: все доступные эмоции в порядке убывания вероятности, если запрашивались (старый API).
- `"countries47"`: вероятные страны происхождения вместе с уверенностью алгоритма в результате, если запрашивались (старый API).
- `"attributes"`: пол (`male` или `female`), возраст (число лет), эмоции (преобладающая эмоция), вероятные страны происхождения, борода (`beard` или `none`), очки (`sun`, `eye`, или `none`), вместе с уверенностью алгоритма в результате, если запрашивались (новый API).
- `"normalized"`: нормализованное изображение лица в формате base64, если запрашивалось.
- `''timings''`: время обработки лиц, если запрашивалась.

```
{
 "bbox": { "left": 1, "right": 2, "top": 3, "bottom": 4},
 "detection_score": 0.99,
 "facen": "...",
 "gender": {
     "gender": "MALE",
     "score": "1.123"
 },
 "age": 23.59,
 "emotions": [
     { "emotion": "neutral", "score": 0.95 },
     { "emotion": "angry", "score": 0.55 },
     ...
 ],
 "normalized": "...",
 "attributes": {
   "age": {
       "attribute": "age",
       "model": "age.v1",
       "result": 25
   },
   "beard": {
       "attribute": "beard",
       "model": "beard.v0",
       "result": [
           { "confidence": 0.015328666, "name": "beard" }
       ]
   },
   "countries47": {
       "attribute": "countries47",
       "model": "countries47.v1",
       "result": [
           { "confidence": 0.90330666, "name": "UKR" },
           { "confidence": 0.013165677, "name": "RUS" },
           { "confidence": 0.009136979, "name": "POL" },
           ...
       ]
   },
   "emotions": {
       "attribute": "emotions",
       "model": "emotions.v1",
       "result": [
           { "confidence": 0.99959123, "name": "neutral" },
           { "confidence": 0.00039093022, "name": "sad" },
           { "confidence": 8.647058e-06, "name": "happy" },
```

(continues on next page)

(продолжение с предыдущей страницы)

```
            { "confidence": 7.994732e-06, "name": "surprise" },
            { "confidence": 6.495376e-07, "name": "disgust" },
            { "confidence": 6.063106e-07, "name": "angry" },
            { "confidence": 7.077886e-10, "name": "fear"                }
        ]
    },
    "gender": {
        "attribute": "gender",
        "model": "gender.v2",
        "result": [
            { "confidence": 0.999894, "name": "female" },
            { "confidence": 0.00010597264, "name": "male" }
        ]
    },
    "glasses3": {
        "attribute": "glasses3",
        "model": "glasses3.v0",
        "result": [
            { "confidence": 0.9995815, "name": "none" },
            { "confidence": 0.0003348241, "name": "eye" },
            { "confidence": 8.363914e-05, "name": "sun" }
        ]
    }
  }
  "timings": ...
}
```

9.1.3 Примеры

Запрос №1

```
curl -X POST -F sample=@sample.jpg -F 'request={"requests":[{"image":"multipart:sample","detector":
↪"nnd", "need_gender":true, "need_normalized": true, "need_facen": true}]}' http://127.0.0.
↪1:18666/| jq
```

Ответ

```
{
  "responses": [
    {
      "faces": [
        {
          "bbox": {
            "left": 595,
            "top": 127,
            "right": 812,
            "bottom": 344
          },
          "detection_score": -0.0012599,
          "facen": "qErDPTE...vd4oMr0=",
          "gender": {
            "gender": "FEMALE",
```

(continues on next page)

(продолжение с предыдущей страницы)


```
            "score": -2.6415858
          },
          "normalized": "iVBORw0KGgoAAAANSUhE...79CIbv"
        }
      ]
    }
  ]
}
```


Запрос №2

```
curl -X POST  -F 'request={"requests": [{"need_age": true, "need_gender": true, "detector": "nnd",
↪"roi": {"left": -2975, "top": -635, "right": 4060, "bottom": 1720}, "image": "https://static.
↪findface.pro/sample.jpg", "need_emotions": true}]}' http://127.0.0.1:18666/ |jq
```

Ответ


```
{
  "responses": [
    {
      "faces": [
        {
          "bbox": {
            "left": 595,
            "top": 127,
            "right": 812,
            "bottom": 344
          },
          "detection_score": 0.9999999,
          "gender": {
            "gender": "FEMALE",
            "score": -2.6415858
          },
          "age": 26.048346,
          "emotions": [
            {
              "emotion": "neutral",
              "score": 0.90854686
            },
            {
              "emotion": "sad",
              "score": 0.051211596
            },
            {
              "emotion": "happy",
              "score": 0.045291856
            },
            {
              "emotion": "surprise",
              "score": -0.024765536
            },
            {
              "emotion": "fear",
```


(continues on next page)

(продолжение с предыдущей страницы)


```
            "score": -0.11788454
          },
          {
            "emotion": "angry",
            "score": -0.1723868
          },
          {
            "emotion": "disgust",
            "score": -0.35445923
          }
        ]
      }
    ]
  }
]
}
```


Запрос №3. Автоповорот

```
curl -s -F 'sample=@/path/to/your/photo.png' -F 'request={"requests":[{"image":"multipart:sample",
↪"detector":"nnd", "auto_rotate": true, "need_normalized": true }]}' http://192.168.113.79:18666/
```

Ответ


```
{
 "responses": [
   {
     "faces": [
       {
         "bbox": {
           "left": 96,
           "top": 99,
           "right": 196,
           "bottom": 198
         },
         "detection_score": -0.00019264,
         "normalized": "iVBORw0KGgoAAAANSUhE....quWKAAC"
       },
       {
         "bbox": {
           "left": 205,
           "top": 91,
           "right": 336,
           "bottom": 223
         },
         "detection_score": -0.00041600747,
         "normalized": "iVBORw0KGgoAAAANSUhEUgAA....AByquWKAACAAElEQVR4nKy96XYybybIdnF"
       }
     ]
   }
 ]
}
```

Запрос №4. Использование нового API (атрибутов: «beard», «emotions», «age», «gender», «glasses3», «face»)

```
curl -s -F photo=@sample.jpg -Frequest='{"requests": [{"image":"multipart:photo", "detector": "nnd
↪", "attributes": ["beard", "emotions", "age", "gender", "glasses3", "face"]}]}' http://127.0.0.
↪1:18666 | jq
```

Ответ


```
{
  "responses": [
    {
      "faces": [
        {
          "bbox": {
            "left": 595,
            "top": 127,
            "right": 812,
            "bottom": 344
          },
          "detection_score": -0.00067401276,
          "rotation_angle": 0,
          "attributes": {
            "age": {
              "attribute": "age",
              "model": "age.v1",
              "result": 25
            },
            "beard": {
              "attribute": "beard",
              "model": "beard.v0",
              "result": [
                {
                  "confidence": 0.015324414,
                  "name": "beard"
                }
              ]
            },
            "emotions": {
              "attribute": "emotions",
              "model": "emotions.v1",
              "result": [
                {
                  "confidence": 0.99958,
                  "name": "neutral"
                },
                {
                  "confidence": 0.0004020365,
                  "name": "sad"
                },
                {
                  "confidence": 8.603454e-06,
                  "name": "happy"
                },
                {
```


(continues on next page)

(продолжение с предыдущей страницы)


```
            "confidence": 8.076766e-06,
            "name": "surprise"
          },
          {
            "confidence": 6.6535216e-07,
            "name": "disgust"
          },
          {
            "confidence": 6.1434775e-07,
            "name": "angry"
          },
          {
            "confidence": 7.3372125e-10,
            "name": "fear"
          }
        ]
      },
      "face": {
        "attribute": "face",
        "model": "elderberry_576",
        "result": "KjiHu6cWh70ppqa9l"
      },
      "gender": {
        "attribute": "gender",
        "model": "gender.v2",
        "result": [
          {
            "confidence": 0.9998938,
            "name": "female"
          },
          {
            "confidence": 0.000106243206,
            "name": "male"
          }
        ]
      },
      "glasses3": {
        "attribute": "glasses3",
        "model": "glasses3.v0",
        "result": [
          {
            "confidence": 0.99958307,
            "name": "none"
          },
          {
            "confidence": 0.00033243417,
            "name": "eye"
          },
          {
            "confidence": 8.4465064e-05,
            "name": "sun"
          }
        ]
      }
    }
  }
```


(continues on next page)

(продолжение с предыдущей страницы)

```
      ],
      "orientation": 1
    }
  ]
}
```

9.2 Статистика по галереям шарда

Вы можете отображать прямо в браузере статистику по галереям того или иного шарда `tntapi` в JSON-формате. Данный функционал может быть использован в системах мониторинга.

Примечание: В случае если FindFace Enterprise Server 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`.

В этом разделе:

9.2.1 Получение списка галерей

Для того чтобы отобразить список всех галерей, относящихся к шарду, введите в строке адреса браузера:

```
http://<tarantool_host_ip:shard_port>/stat/list/:start/:limit
```

`:start`: номер галереи, с которой начинается список.
`:limit`: максимальное количество галерей в списке.

Ответ будет содержать JSON со следующими полями:

- `next`: курсор для получения следующей страницы с результатами; передайте его как значение параметра *:start_id* в следующем запросе
- `total`: общее количество галерей на шарде
- `galleries`: список галерей со следующими данными:
  - `id`: id галереи
  - `name`: имя галереи

- `cnt_linear`: количество лиц в пространстве `linear` (лиц без быстрого индекса)
- `cnt_preindex`: количество лиц в пространстве `preindex` (в промежуточной стадии создания быстрого индекса)
- `cnt_indexed`: количество лиц в пространстве `indexed` (лиц с быстрым индексом)

Пример

Запрос

```
http://127.0.0.1:8001/stat/list/1/99
or
curl http://127.0.0.1:8001/stat/list/1/99 \| jq
```

Ответ

```
HTTP/1.1 200 Ok
Content-length: 170
Server: Tarantool http (tarantool v1.7.3-673-g23cc4dc)
Connection: keep-alive

{"next":3,"galleries":[{"cnt_indexed":3,"id":1,"cnt_preindex":0,"name":"a","cnt_linear":0},{"cnt_
↪indexed":1,"id":2,"cnt_preindex":0,"name":"b","cnt_linear":1}],"total":5}
```

9.2.2 Получение информации по галерее

Для получения информации по галерее, введите в адресной строке браузера:

```
http://<tarantool_host_ip:shard_port>/stat/info/:name
```

`:name`: имя галереи.

Ответ будет содержать JSON со следующими полями:

- `id`: id галереи
- `name`: имя галереи
- `cptr`: адрес uint64_t объекта галереи в памяти
- `cnt_linear`: количество лиц в пространстве `linear`
- `cnt_preindex`: количество лиц в пространстве `preindex`
- `cnt_preindex_deleted`: количество лиц, удаленных из пространства `preindex`, которые физически еще присутствуют в Tarantool
- `cnt_indexed`: количество лиц в пространстве `indexed`
- `cnt_indexed_deleted`: количество лиц в пространстве `indexed`, которые физически еще присутствуют в Tarantool
- `index_file`: путь к файлу быстрого индекса
- `index_loaded`: индикатор того, был ли загружен быстрый индекс

Пример

Запрос

```
http://127.0.0.1:8001/stat/info/my_gal
or
curl http://127.0.0.1:8001/stat/info/my_gal | jq
```

Ответ

```
HTTP/1.1 200 Ok
Content-length: 196
Server: Tarantool http (tarantool v1.7.3-673-g23cc4dc)
Connection: keep-alive

{"cnt_indexed":2464344,"cnt_preindex_deleted":139,"index_file":"none","index_loaded":false,"cnt_
↪preindex":8310,"cnt_linear":959,"cptr":29253696,"id":1,"name":"my_gal","cnt_indexed_deleted
↪":78811}
```

9.3 Прямые API-запросы к базе данных Tarantool

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

В этом разделе:

9.3.1 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 Enterprise Server 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 (подробнее см. *Получение списка галерей*):

```
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.

9.3.2 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",
"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
```

9.3.3 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
```

9.3.4 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]
                },
                "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
    }
  ]
}
```

9.3.5 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, ... }
```

9.3.6 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
    }
  ]
}
```

9.3.7 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}
```

9.3.8 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}}
```

9.3.9 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
```

9.4 Дополнительные возможности `findface-tarantool-server`

В этом разделе:

9.4.1 Дополнительные параметры конфигурации

Для того чтобы настроить взаимодействие между компонентом `findface-sf-api` и Tarantool, укажите дополнительные параметры в 3-м аргументе раздела `FindFace.start` в файле конфигурации `findface-tarantool-server`:

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

FindFace.start("127.0.0.1", 8001, {license_ntls_server="127.0.0.1:3133", additional parameter 1, ..
↪., additional parameter N})

## Example:
FindFace.start("127.0.0.1", 8001, {license_ntls_server="127.0.0.1:3133", facen_size = 576, log_
↪requests = false})
```

Дополнительные параметры:

Параметр	Значение по умолчанию	Описание
log_requests	true	Включает запись запросов в лог (`/var/log/tarantool/FindFace.log`).
facen_size	576	Размер вектора признаков, зависит от используемой модели нейронной сети. Перед редактированием данного параметра обязательно проконсультируйтесь у наших экспертов по адресу support@ntechlab.com.
search_threads	1	Количество тредов, используемых для поиска по быстрому индексу.
replication	nil	Только для реплики базы данных Tarantool. IP-адрес ведущего сервера (мастера).
soft_delete_mode	false	Включает режим «мягкого удаления», при котором лица не удаляются из быстрого индекса, а скрываются в результатах поиска.

9.4.2 Режим «мягкого удаления»

Tarantool поддерживает так называемый режим «мягкого удаления», при котором лица физически не удаляются из быстрого индекса, а скрываются в результатах поиска. Данный режим рекомендуется активировать в связи со следующими преимуществами:

- Время запуска Tarantool линейно зависит от количества лиц, удаленных из пространства `Indexed` (т. е. из быстрого индекса). Если режим «мягкого удаления» включен, лица физически не удаляются из быстрого индекса. Таким образом, удаление лиц не влияет на время запуска Tarantool.
- Качество поиска по быстрому индексу также зависит от количества физически удаленных лиц. Оно не снижается в режиме «мягкого удаления».

Для активации режима «мягкого удаления» отредактируйте раздел `FindFace.start` следующим образом:

```
FindFace.start("127.0.0.1", 8001, {license_ntls_server="127.0.0.1:3133", soft_delete_mode = true})
```

9.4.3 Репликация базы данных Tarantool

Репликация позволяет множественным экземплярам Tarantool работать с копиями одной и той же базы данных. Копии базы данных синхронизируются, так как каждый экземпляр Tarantool может передавать изменения в остальные. Tarantool поддерживает репликацию master-slave (ведущий-ведомые). Вы можете добавлять или удалять данные только через ведущий экземпляр (мастер). Ведомые экземпляры (реплики) могут быть использованы только для поиска и просмотра данных.

Для того чтобы развернуть набор реплик Tarantool, обратитесь к официальной документации Tarantool.

Для первого запуска созданной реплики выполните следующие действия:

1. Запустите мастер.
2. В файле конфигурации реплики укажите IP-адрес и слушающий порт мастера.

   ```
   FindFace.start("127.0.0.1", 48001, {replication = "127.0.0.1:33001"})
   ```

3. Скопируйте последний снимок (snapshot) мастера (`*.snap`) в папку, указанную в параметре `memtx_dir` реплики.

   ```
   --Directory to store data
      memtx_dir = '/opt/ntech/var/lib/tarantool/default/snapshots'
   ```

4. Скопируйте логи мастера в папку, указанную в параметре `wal_dir` реплики.

   ```
   --Directory to store data
        wal_dir = '/opt/ntech/var/lib/tarantool/default/xlogs'
   ```

5. Запустите реплику. Вы можете запустить любое количество реплик под одним мастером.

Важно: Перед включением *быстрого индекса* на мастере (`:use_index("/путь/к/<index>.idx")`) скопируйте файл индекса (`<index>.idx`) на реплику по тому же пути. Затем выполните `use_index` на мастере.

Совет: Рекомендуется удалять все файлы индекса на реплике, кроме последнего, во избежание промежуточных обновлений индекса в случае сильного отставания реплики от мастера.

Совет: Для того чтобы ускорить синхронизацию мастера и реплики, вы также можете скопировать последний снимок (snapshot) мастера в реплику.

9.5 Распознавание живых лиц в реальном времени (Liveness)

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.

Для включения Liveness-детектора выполните следующие действия:

1. Откройте файл конфигурации `findface-video-worker`. В параметре `liveness` –> `fnk` укажите путь к модели liveness-детектора, как показано ниже.

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

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


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

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

2. Перезапустите `findface-video-worker`.

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

После активации liveness-детектора сервис `findface-video-worker` будет отправлять оценку liveness лица в компонент `findface-facerouter` в ключе `liveness` словаря `detectorParams`. Для обработки лица в соответствии с его liveness *создайте плагин*.

9.6 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:

9.6.1 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)
```

(continues on next page)

(продолжение с предыдущей страницы)

```
  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
  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
```

9.7 Распознавание атрибутов лица

FindFace Enterprise Server 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.

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
```

Глава 10

Обслуживание и устранение неисправностей

10.1 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-facerouter`	sudo systemctl status findface-facerouter.service
`findface-tarantool-server`	sudo systemctl status tarantool@FindFace.service
`findface-video-manager`	sudo systemctl status findface-video-manager.service
`findface-video-worker-cpu`	sudo systemctl status findface-video-worker-cpu.service
`findface-video-worker-gpu`	sudo systemctl status findface-video-worker-gpu.service
`findface-ntls`	sudo systemctl status findface-ntls
`etcd`	sudo systemctl status etcd.service
`NginX`	sudo systemctl status nginx.service
`memcached`	sudo systemctl status memcached.service

10.2 Анализ логов

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-facerouter`	sudo tail -f /var/log/syslog | grep facerouter
`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-ntls`	sudo tail -f /var/log/syslog | grep ntls
`etcd`	sudo tail -f /var/log/syslog | grep etcd

10.3 Устранение неполадок с лицензией и `findface-ntls`

При устранении неполадок с лицензией и `findface-ntls` (см. licensing) первым шагом является получение информации о лицензии и статусе сервера `findface-ntls`. 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 info@ntechlab.com.

10.3.1 Получение информации о лицензии

Для получения информации о лицензии FindFace Enterprise Server 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": 1520844897,
  "type": "offline (extended)",
  "license_id": "001278983",
```

(continues on next page)

(продолжение с предыдущей страницы)


```
  "generated": 487568400,
  "last_updated": 4,
  "valid": {
    "value": true,
    "description": ""
  },
  "source": "/ntech/license/001278983.lic",
  "limits": [
    {
      "type": "time",
      "name": "end",
      "value": 25343
    },
    {
      "type": "number",
      "name": "faces",
      "value": 90071,
      "current": 230258
    },
    {
      "type": "number",
      "name": "cameras",
      "value": 9007,
      "current": 3
    },
    {
      "type": "number",
      "name": "extraction_api",
      "value": 900,
      "current": 8
    },
    {
      "type": "boolean",
      "name": "gender",
      "value": true
    },
    {
      "type": "boolean",
      "name": "age",
      "value": true
    },
    {
      "type": "boolean",
      "name": "emotions",
      "value": true
    },
    {
      "type": "boolean",
      "name": "fast-index",
      "value": true
    }
  ],
  "services": [
    {
      "name": "video-worker",
      "ip": "127.0.0.1:58970"
```


(continues on next page)

(продолжение с предыдущей страницы)

```
    },
    {
      "name": "FindFace-tarantool",
      "ip": "127.0.0.1:58978"
    },
    {
      "name": "findface-extraction-api",
      "ip": "127.0.0.1:52376"
    }
  ]
}
```

10.4 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:

Примечание: Приведенные действия нужно повторить для каждого шарда Tarantool.

1. Откройте файл конфигурации шарда.

   ```
   sudo vi /etc/tarantool/instances.enabled/<shard_001>.lua
   ```

2. Раскомментируйте строку `force_recovery = true`.

   ```
   box.cfg{

       force_recovery = true,
   }
   ```

3. Перезапустите шард.

   ```
   sudo systemctl restart tarantool@<shard_001>.service
   ```

Глава 11

Приложения

11.1 Модели нейронных сетей

В этом разделе вы найдете сводную информацию по моделям нейронных сетей, созданным в нашей лаборатории и используемым в FindFace Enterprise Server.

Примечание: Конфигурация для теста производительности на CPU и GPU:

- CPU: Intel® Core™ i7-5930K CPU @ 3.50GHz × 12
- GPU: GeForce GTX 1080

Тип нейронной сети	Имя	Размер вектора признаков, байты	CPU, FPS	GPU, FPS
Биометрия лица	face/elderberry_160	160	5.78	204.98
	face/elderberry_576.r2	576	1.86	60.62
	face/grapefruit_160	160	6.01	191.43
	face/grapefruit_480	480	1.93	64.97
Распознавание пола	faceattr/gender.v2	N/A	15.01	523.22
Распознавание возраста	faceattr/age.v1	N/A	14.99	529.35
Распознавание эмоций	faceattr/emotions.v1	N/A	10.99	235.59
Распознавание страны	faceattr/countries47.v1	N/A	14.97	532.53

11.2 Подробно о компонентах

11.2.1 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`).

Совет: Вы можете использовать *HTTP API* для прямого доступа к компоненту `findface-extraction-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 пола/возраста/эмоций/страны.

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:

```
allow_cors: false
detector_instances: 0
dlib:
  model: /usr/share/findface-data/normalizer.dat
  options:
    adjust_threshold: 0
    upsample_times: 1
extractors:
  instances: 1
  max_batch_size: 16
  models:
    age: ""
    beard: ""
    emotions: ""
    face: face/grapefruit_480.cpu.fnk
    gender: ""
    glasses3: ""
    liveness: ""
  models_root: /usr/share/findface-data/models
```

(continues on next page)

(продолжение с предыдущей страницы)

```
fetch:
  enabled: true
  size_limit: 10485760
license_ntls_server: 127.0.0.1:3133
listen: 127.0.0.1:18666
max_dimension: 6000
nnd:
  model: /usr/share/nnd/nnd.dat
  options:
    max_face_size: .inf
    min_face_size: 30
    o_net_thresh: 0.9
    p_net_max_results: 0
    p_net_thresh: 0.5
    r_net_thresh: 0.5
    scale_factor: 0.79
  quality_estimator: true
  quality_estimator_model: /usr/share/nnd/quality_estimator_v2.dat
ticker_interval: 5000
```

Файл конфигурации сервиса на GPU:

```
listen: 127.0.0.1:18666
dlib:
  model: /usr/share/findface-data/normalizer.dat
  options:
    adjust_threshold: 0
    upsample_times: 1
nnd:
  model: /usr/share/nnd/nnd.dat
  quality_estimator: true
  quality_estimator_model: /usr/share/nnd/quality_estimator_v2.dat
  options:
    min_face_size: 30
    max_face_size: .inf
    scale_factor: 0.7900000214576721
    p_net_thresh: 0.5
    r_net_thresh: 0.5
    o_net_thresh: 0.8999999761581421
    p_net_max_results: 0
detector_instances: 0
extractors:
  models_root: /usr/share/findface-data/models
  max_batch_size: 3
  instances: 2
  models:
    age: ""
    beard: ""
    emotions: ""
    face: face/grapefruit_480.gpu.fnk
    gender: ""
    glasses3: ""
    liveness: ""
  cache_dir:
  gpu_device: 0
```

(continues on next page)

(продолжение с предыдущей страницы)

```
license_ntls_server: 172.17.46.26:3133
fetch:
  enabled: true
  size_limit: 10485760
max_dimension: 6000
allow_cors: false
ticker_interval: 5000
```

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

Параметр	Описание
`nnd -> quality_estimator`	Включает/отключает оценку качества лица. В этом случае компонент `findface-extraction-api` будет возвращать показатель качества лица в параметре `detection.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.
`nnd -> min_face_size`	Минимальный размер лица, которое будет гарантированно найдено. Определяется размером рамки с лицом (bbox). Чем больше значение, тем менее ресурсоемок процесс обнаружения лиц.
`nnd -> max_face_size`	Максимальный размер лица, которое будет гарантированно найдено. Определяется размером рамки с лицом (bbox).
`license_ntls_server`	IP-адрес и порт сервера лицензирования `findface-ntls`.
`gpu_device`	(Только для GPU) Номер GPU-устройства, используемого `findface-extraction-api-gpu`.
№ экземпляров	Количество экземпляров нейронных сетей (и, следовательно, одновременно обрабатываемых запросов), загружаемых в RAM компонентом `findface-extraction-api`. Укажите количество экземпляров из лицензии. Значение по умолчанию (0) означает, что данное количество равно числу ядер процессора. Данный параметр существенно влияет на потребление памяти.
`ntls -> enabled`	Включает получение изображение из Интернета.
`fetch -> size_limit`	Максимальный размер получаемого из Интернета изображения.

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
```

(continues on next page)

(продолжение с предыдущей страницы)

```
  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: ""
```

11.2.2 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 изображении вам нужно отправить его в *API-запросе* в компонент `findface-sf-api` в виде файла или URL. Компонент `findface-sf-api` затем перенаправит запрос в `findface-extraction-api` для обнаружения и распознавания лица.

Совет: Вы можете *напрямую* обращаться к компоненту `findface-extraction-api`.

При наличии в системе настроенного видеодетектора лиц компонент `findface-sf-api` также взаимодействует с сервисом `findface-facerouter`. Он 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).

Полный список функций:

- реализация *HTTP API* (методы для обнаружения и распознавания лиц, выполняемые с использованием компонента `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.

```
cache:
  inmemory:
    size: 16384
   memcache:
    nodes:
    - 127.0.0.1:11211
    timeout: 100ms
  redis:
    addr: localhost:6379
    db: 0
    network: tcp
    password: ''
    timeout: 5s
  type: memcache
extraction-api:
  extraction-api: http://127.0.0.1:18666
  timeouts:
    connect: 5s
    idle_connection: 10s
    overall: 35s
    response_header: 30s
limits:
  allow-return-facen: false
  body-image-length: 33554432
  deny-networks: 127.0.0.0/8,192.168.0.0/16,10.0.0.0/8,::1/128,fe00::/8
  url-length: 4096
listen: 127.0.0.1:18411
normalized-storage:
  enabled: true
  s3:
    access-key: ''
    bucket-name: ''
    endpoint: ''
    operation-timeout: 30
    public-url: ''
    region: ''
    secret-access-key: ''
    secure: true
  type: webdav
  webdav:
    timeouts:
      connect: 5s
      idle_connection: 10s
      overall: 35s
```

(continues on next page)

(продолжение с предыдущей страницы)

```
      response_header: 30s
    upload-url: http://127.0.0.1:3333/uploads/
storage-api:
  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: ''
  timeouts:
    connect: 5s
    idle_connection: 10s
    overall: 35s
    response_header: 30s
```

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.
`upload_url`	Путь в WebDAV nginx, по которому в компонент `findface-upload` будут отправляться исходные изображения, миниатюры и нормализованные изображения лиц.

11.2.3 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/<*>.lua`. 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 = '127.0.0.1:33001',

    --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_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 сервер 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.

Стандартная структура базы данных передается из файла `/etc/ffsecurity/tnt_schema.lua` в параметр `meta_scheme`, если FindFace Enterprise Server автоматически устанавливается из инсталлятора. Если установка выполняется из apt-репозитория, вам потребуется *вручную задать* структуру через файл конфигурации.

11.2.4 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.

11.2.5 findface-facerouter

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
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   = ''
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. |
```

11.2.6 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:

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

Функции `findface-video-worker`

Компонент `findface-video-worker` (на CPU/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.

Полный список функций:

- 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: 127.0.0.1:18810
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: 127.0.0.1:18811
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
```

(continues on next page)

(продолжение с предыдущей страницы)

```
  uc_max_dup: 3
  uc_max_avg_shift: 10
  det_period: 8
  realtime: false
  npersons: 4
  disable_drops: false
  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: true
  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: 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 от findface-video-worker. 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 определенного потока, переданные в job-задании, имеют приоритет над настройками в файле конфигурации (см. *Job-задания*).

Опция CPU	Опция GPU	Описание
additional_router_body	router_body	Массив дополнительных полей в POST-запросе с изображением лица в формате ["ключ=значение"]. По умолчанию дополнительные поля не передаются.
additional_router_headers	router_headers	Массив дополнительных заголовков в POST-запросе с изображением лица в формате ["ключ=значение"]. По умолчанию дополнительные заголовки не передаются.
api_ssl_verify	router_verify_ssl	Включает/отключает проверку подписи SSL-сертификата при взаимодействии по https findface-video-worker с компонентом findface-facerouter. Значение по умолчанию: true. Если false, может быть принят самоподписанный сертификат.
api_timeout	router_timeout_ms	Время ожидания в миллисекундах ответа от компонента findface-facerouter на API-запрос компонента findface-video-worker. Если время ожидания истекло, регистрируется ошибка. Значение по умолчанию: 15000.
disable_drops	disable_drops	Включает/отключает отправку в компонент findface-facerouter всех подходящих лиц без пропусков. По умолчанию, если findface-video-worker не обладает достаточными ресурсами для обработки всех кадров с лицами, он отбрасывает некоторые из них. Если данная опция активна, findface-video-worker помещает лишние кадры в очередь, чтобы обработать их впоследствии. Значение по умолчанию: false.
draw_track	Н/п	Включает рисование в bbox следа от движения лица. Значение по умолчанию: false.

Продолжается на следующей странице

Таблица 1 – продолжение с предыдущей страницы

Опция CPU	Опция GPU	Описание
`fd_frame_height`	Н/п	Размер кадра для детектора лиц в пикселях. Кадр должен быть уменьшен при больших разрешениях камеры, отображении лиц крупным планом, а также при чрезмерной загрузке процессора — для снижения потребления системных ресурсов. Убедитесь, что размер лиц после уменьшения превышает значение параметра `min-face-size`. Значение по умолчанию: -1 (отрицательные значения соответствуют исходному размеру). Оптимальные значения для уменьшения нагрузки: 640-720.
`ffmpeg_format`	`ffmpeg_format`	Передает формат FFMPEG (mxg, flv и т. д.), если он не может быть автоматически определен.
`ffmpeg_params`	`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.
`image_arg`	Н/п	Определяет имя аргумента с изображением лица, отправляемым в API-запросе. Значение по умолчанию: `photo`.
`jpeg_quality`	`jpeg_quality`	Качество сжатия исходного кадра в JPEG. Значение по умолчанию: 95 % от исходного размера.
`max_face_size`	`filter_max_face_size`	Определяет максимальный размер лица в пикселях. Лица большего размера не отправляются. Значение по умолчанию: 0 (фильтр выключен).
`md_scale`	Н/п	Размер кадра для детектора движения относительно исходного размера от 0 до 1. Кадр должен быть уменьшен при больших разрешениях камеры, отображении лиц крупным планом, а также при чрезмерной загрузке процессора — для снижения потребления системных ресурсов. Убедитесь, что размер кадра после масштабирования больше значения `min-face-size`. Значение по умолчанию: 1 (исходный размер).
`md_threshold`	`motion_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.
`min_score`	`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.
`min_face_size`	`filter_min_face_size`	Определяет минимальный размер лица в пикселях. Лица меньшего размера не отправляются. Значение по умолчанию: 0 (фильтр выключен).
`min_d_score`	Н/п	Максимальное отклонение отправляемого лица от положения анфас (определяется эмпирически: отрицательные числа до 0). Реперные точки: -3.5 = слишком большие углы поворота, распознавание лиц может быть неэффективным, -2.5 = удовлетворительное отклонение, -0.05 = близко к положению анфас, 0 = анфас. Значение по умолчанию: -1000.

Продолжается на следующей странице

Таблица 1 – продолжение с предыдущей страницы

Опция CPU	Опция GPU	Описание
npersons	Н/п	Максимальное количество лиц, одновременно отслеживаемых детектором лиц. Влияет на производительность. Значение по умолчанию: 4.
only_norm	normalized_only	Включает/отключает отправку только нормализованных лиц без исходных кадров. Значение по умолчанию: false.
overall	overall_only	Буферный режим. Отправлять для лица один кадр наилучшего качества. Значение по умолчанию: true.
Н/п	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.
post_uniq	Н/п	Включает функцию дедупликации лиц, т. е. отправку только нескольких лиц, принадлежащих одному человеку, из множества захваченных в течение определенного промежутка времени. В этом случае, если findface-video-worker отправляет лицо и затем захватывает еще одно в течение периода времени uc-max-time-diff, и если расстояние между лицами не превышает значение uc-max-avg-shift, findface-video-worker оценивает их схожесть. Если лица схожи и общее количество схожих лиц в течение периода времени uc-max-time-diff не превышает число uc-max-dup, findface-video-worker отправляет второе лицо. Иначе, второе лицо не отправляется. Значение по умолчанию: true.
realtime	Н/п	Включает/отключает режим реального времени детектора лиц. Значение по умолчанию: false.
realtime_dly	realtime_post_limit	Тайм-интервал для 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 миллисекундах между двумя последовательными отправками одного и того же лица, но в улучшенном качестве. Значение по умолчанию: 500.
realtime_post_perm	realtime_post_interval	Тайм-интервал реального времени. Включает отправку лучшего кадра в течение периода времени realtime_dly. Если false, лучший кадр ищется динамически и realtime_dly представляет собой максимальный период времени между двумя последовательными отправками одного и того же лица, но в улучшенном качестве. Значение по умолчанию: false.
roi	roi	Включает отправку на Сервер лиц, обнаруженных только внутри интересующей области WxH+X+Y. По умолчанию область не задана.
rot	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.
send_track	Н/п	Включает отправку в компонент findface-facerouter вместе с bbox следа от движения лица в виде массива координат точек центра bbox. Значение по умолчанию: 0 (массив координат не отправляется). Параметр принимает целое положительное число – количество точек в следе.

Продолжается на следующей странице

Таблица 1 – продолжение с предыдущей страницы

Опция CPU	Опция GPU	Описание
Н/п	start_stream_timestamp	Прибавить указанное количество секунд к временным меткам из потока.
tracker_threads	Н/д	Количество тредов отслеживания для детектора лиц. Данное значение должно быть меньше или равно значению параметра npersons. Оптимально, когда они равны. Если количество тредов отслеживания меньше, чем максимальное количество отслеживаемых лиц, потребление ресурсов уменьшается, однако также уменьшается и скорость отслеживания. Значение по умолчанию: 1.
uc_max_time_diff	Н/д	Только если post-uniq: true (включена функция дедупликации лиц). Максимальный период времени в секундах, в течение которого схожие лица рассматриваются как лица одного человека. Значение по умолчанию: 30.
uc_max_dup	Н/п	Только если post-uniq: true (включена функция дедупликации лиц). Максимальное количество лиц в течение периода времени uc-max-time-diff, которое отправляется в компонент findface-facerouter для одного человека. Значение по умолчанию: 3.
uc_max_avg_shift	Н/д	Только если post-uniq: true (включена функция дедупликации лиц). Определяет максимальное расстояние в пикселях, на котором схожие лица еще рассматриваются как лица одного человека. Значение по умолчанию 10.0.
Н/п	use_stream_timestamp	Если true, отправлять на сервер временные метки полученные из потока. Если false, отправлять текущие дату и время.

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
```

(continues on next page)

(продолжение с предыдущей страницы)

```
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 server port, 0=disabled
port = 18999

## streamer url - how to access this worker on streamer_port
url = ws://127.0.0.1:18999/stream/

#------------------------------
[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
```

(continues on next page)

(продолжение с предыдущей страницы)

```
## 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 =

## 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
```

(continues on next page)

(продолжение с предыдущей страницы)

```
## 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 server port, 0=disabled
port = 18999

## streamer url - how to access this worker on streamer_port
url = ws://172.17.46.17:18999/stream/

#------------------------------
[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
```

(continues on next page)

(продолжение с предыдущей страницы)

```
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, по которым определяются правила обработки обнаруженных лиц компонентом `findface-facerouter`.
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,
    "realtime": false,
```

(continues on next page)

(продолжение с предыдущей страницы)


```
    "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
  },
  "status": "INPROGRESS",
  "status_msg": "",
  "statistic": {
```


(continues on next page)

(продолжение с предыдущей страницы)


```
    "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`.
- `single_pass`: если true (по умолчанию false), то не перезапускать обработку потока в случае ошибки.
- `router_url`: IP-адрес и порт компонента `findface-facerouter`, на который компонент `findface-video_worker` будет отправлять обнаруженные лица для обработки.
- `stream_settings`, `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.

11.2.7 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`".

11.3 Файл с параметрами установки

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

Совет: Подробная информация об инсталляторе FindFace Enterprise Server приведена в разделе installer.

Важно: Обязательно удалите поля `*.config`, `exp_ip` и `int_ip` перед установкой FindFace Enterprise Server на сервере с другим IP-адресом.

```
{
  "ignore_lowmem": true,
  "findface-security.config": {
    "EXTERNAL_ADDRESS": "http://172.20.77.78"
  },
  "inter_ip.bind": "127.0.0.1",
  "memcached.config": {
    "listen_host": "127.0.0.1",
    "max_memory": 1024,
    "item_size": 16
  },
  "findface-video-worker.config": {
    "FKVD_WRK_CAP": "10",
    "FKVD_NTLS_ADDR": "127.0.0.1:3133",
    "streamer": [
      "port = 18999",
      "url = ws://127.0.0.1:18999/stream/"
    ],
    "FKVD_MGR_ADDR": "127.0.0.1:18811"
  },
  "ext_ip.bind": "0.0.0.0",
  "findface-data.models": [
    "./findface-data-age.v1-cpu_3.0.0_amd64.deb",
    "./findface-data-age.v1-gpu_3.0.0_amd64.deb",
    "./findface-data-beard.v0-cpu_3.0.0_amd64.deb",
    "./findface-data-beard.v0-gpu_3.0.0_amd64.deb",
    "./findface-data-grapefruit-160-cpu_3.0.0_amd64.deb",
    "./findface-data-grapefruit-160-gpu_3.0.0_amd64.deb",
```

(continues on next page)

(продолжение с предыдущей страницы)


```
    "./findface-data-grapefruit-480-cpu_3.0.0_amd64.deb",
    "./findface-data-grapefruit-480-gpu_3.0.0_amd64.deb",
    "./findface-data-emotions.v1-cpu_3.0.0_amd64.deb",
    "./findface-data-emotions.v1-gpu_3.0.0_amd64.deb",
    "./findface-data-gender.v2-cpu_3.0.0_amd64.deb",
    "./findface-data-gender.v2-gpu_3.0.0_amd64.deb",
    "./findface-data-glasses3.v0-cpu_3.0.0_amd64.deb",
    "./findface-data-glasses3.v0-gpu_3.0.0_amd64.deb",
    "./findface-data-liveness.v1-gpu_3.0.0_amd64.deb"
  ],
  "findface-video-worker.variant": "cpu",
  "inter_ip.advertised": "127.0.0.1",
  "product": "security",
  "findface-ntls.config": {
    "NTLS_LISTEN": "127.0.0.1:3133",
    "NTLS_LICENSE_DIR": "/opt/ntech/license",
    "NTLS_LISTEN_UI": "127.0.0.1:3185"
  },
  "ext_ip.advertised": "172.20.77.78",
  "tnt_instances": 2,
  "findface-facerouter.config": {
    "port": "18820",
    "host": "127.0.0.1",
    "plugin_source": "dir",
    "plugin_dir": "/etc/findface-facerouter-plugins",
    "sfapi_url": "http://127.0.0.1:18411"
  },
  "findface-sf-api.config": {
    "storage-api": {
      "shards": [
        {
          "slave": "",
          "master": "http://127.0.0.1:8101/v2/"
        },
        {
          "slave": "",
          "master": "http://127.0.0.1:8102/v2/"
        }
      ]
    },
    "listen": "127.0.0.1:18411",
    "extraction-api": {
      "extraction-api": "http://127.0.0.1:18666"
    }
  },
  "type": "stand-alone",
  "findface-extraction-api.variant": "cpu",
  "findface-video-manager.config": {
    "rpc": {
      "listen": "127.0.0.1:18811"
    },
    "listen": "127.0.0.1:18810",
    "master": {
      "self_url": "127.0.0.1:18811",
      "self_url_http": "127.0.0.1:18811"
    },
```


(continues on next page)

(продолжение с предыдущей страницы)


```
    "ntls": {
      "url": "http://127.0.0.1:3185/",
      "enabled": false
    }
  },
  "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"
  ],
  "findface-tarantool-server.config": {
    "shard-002": {
      "TNT_LISTEN": "127.0.0.1:33002",
      "TNT_DATA_DIR": "/opt/ntech/var/lib/tarantool/shard-002",
      "TNT_META_SCHEME": "meta_scheme",
      "TNT_FF_LISTEN_PORT": "8102",
      "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"
    },
    "shard-001": {
      "TNT_LISTEN": "127.0.0.1:33001",
      "TNT_DATA_DIR": "/opt/ntech/var/lib/tarantool/shard-001",
      "TNT_META_SCHEME": "meta_scheme",
      "TNT_FF_LISTEN_PORT": "8101",
      "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"
    }
  },
  "findface-extraction-api.config": {
    "extractors": {
      "instances": 1,
      "models": {
        "emotions": "",
        "age": "",
        "gender": "",
        "face": "face/grapefruit_480.cpu.fnk"
      }
    },
    "nnd": {
      "quality_estimator": true
    },
    "listen": "127.0.0.1:18666",
    "license_nils_server": "127.0.0.1:3133"
  }
}
```

Для того чтобы автоматически установить FindFace Enterprise Server на другом физическом сервере, не отвечая на вопросы инсталлятора, используйте следующую команду:

```
sudo ./findface-security-2.1.0-server-3.1.0.run -f /tmp/<findface-installer-*>.json
```

Содержание модулей Python

Алфавитный указатель

Символы

A

B

C

D

F

G

I

L

M

N

O

P

S

U