FindFace Enterprise Server SDK Documentation

Выпуск 2.5

NtechLab

июн. 15, 2023

Содержание:

FindFace Enterprise Server SDK – это промышленная высокопроизводительная программно-технологическая платформа нового поколения, которая позволяет создавать интегрированные биометрические системы фото- и видеоидентификации любого масштаба – от небольших решений уровня предприятия до городских, федеральных и международных проектов.

Возможности:

- Быстрое и надежное обнаружение лиц на фото и видео, их биометрическая обработка и запись в базу данных. Возможность пакетной загрузки лиц.
- Интеллектуальная видеоаналитика.
- Быстрая и точная идентификация и верификация лиц алгоритмами на основе нейронных сетей.
- Распознавание пола, возраста и эмоций.
- Группировка лиц одной персоны в базе данных и распознавание «свой-чужой».
- Неограниченная масштабируемость благодаря интеграции с Tarantool.
- Удобный многофункциональный интерфейс REST API со встроенным фреймворком для тестирования.
- Возможность гибкой настройки формата API-ответа.
- Дружественный веб-интерфейс.
- Возможность лицензирования в открытых и закрытых системах.

Совет: Для ознакомления со списком изменений в версии, выполните команду:

```
$ dpkg-parsechangelog -l /usr/share/doc/findface-repo/changelog.Debian.gz --all
```

FindFace Enterprise Server SDK предназначен для независимых производителей программного обеспечения, системных интеграторов, корпоративных заказчиков, а также производителей оборудования, программно-аппаратных решений и облачных сервисов. FindFace Enterprise Server SDK 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.

FindFace Enterprise Server SDK может использоваться как платформа для построения интегрированных отраслевых решений, которые позволят заказчикам решать такие задачи, 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, мониторинг активности в социальных сетях.

Руководство предназначается для специалистов по интеграции, системных администраторов, специалистов по установке и настройке систем анализа и распознавания биометрических данных на базе FindFace Enterprise Server SDK, 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 SDK.

Давайте приступим!

Глава 1

Приступая к интеграции

Структура системы анализа и распознавания лиц на базе FindFace Enterprise Server SDK показана на следующей схеме:

FindFace Enterprise Server SDK включает в себя `Сервер анализа и распознавания биометрических данных` (далее `Сервер FindFace` или `Сервер`) и, опционально, видеодетектор для обнаружения лиц в видеопотоке (устанавливается как компонент `fkvideo_detector`) и другие *дополнительные компоненты*.

Работу Сервера в свою очередь обеспечивают следующие компоненты:

Компонент	Описание
`findface-facenapi`	Реализованный на Python сервис, обеспечивающий функционирование HTTP API. Данный сервис выполняет функцию обнаружения лиц на фотографиях, взаимодействует с базой данных MongoDB и сервисами `findface-nnapi` и `tarantool@FindFace`.
`tntapi` (`tarantool@FindFace` как шард)	Сервис, обеспечивающий взаимодействие между сервисом `findface-facenapi` и базой биометрических данных на основе Tarantool. Для увеличения скорости поиска на каждом сервере с базой данных Tarantool могут быть созданы многочисленные сегменты («шарды») `tntapi`. Их параллельное функционирование приводит к значительному увеличению производительности (в 70-100 раз).
`findface-nnapi`	Сервис, выполняющий посредством нейронных сетей извлечение вектора признаков (биометрического образца) из обнаруженных лиц, а также распознавание пола, возраста и эмоций. Требует установки пакетов с *моделями нейронных сетей* `<findface-data>.deb`.
MongoDB	База данных, в которой хранятся метаданные лиц, подробная информация о галереях, настройки, векторы признаков и т. д.
`findface-upload`	Веб-сервер на базе nginx, который принимает изображения через WebDAV. Устанавливается, если требуется хранить исходные загруженные изображения, миниатюры и нормализованные изображения лиц на Сервере.
`NTLS`	Локальный сервер лицензий с управлением через веб-интерфейс, взаимодействующий для верификации лицензий с глобальным центром лицензий NtechLab. Для закрытых систем поддерживается работа с аппаратными лицензионными ключами.

Интеграция FindFace Enterprise Server SDK включает в себя следующие основные этапы:

1. *выбор архитектуры развертывания*;
2. *подготовка физических серверов*;
3. *установка необходимого стороннего ПО*;
4. *установка лицензии и компонентов Сервера*;
5. *создание токена авторизации* и *тестирование работы сервера*;
6. *настройка обнаружения лиц на видео*;
7. увеличение производительности *балансировкой нагрузки на компоненты* и *индексированием базы данных*;
8. *настройка расширенных функций*;
9. *создание интерфейса обмена данными между Сервером и партнерским приложением*;

Глава 2

Системные требования

В этой главе:

2.1 Общие требования

2.1.1 Базовый сервер(ы)

Перед развертыванием Сервера FindFace убедитесь, что физические серверы, на которых будут установлены его компоненты, удовлетворяют следующим минимальным требованиям:

Примечание: FindFace Enterprise Server SDK может быть развернут на *одиночном физическом сервере*, если количество лиц в базе данных ориентировочно не превышает `1 000 000`. Иначе FindFace Enterprise Server SDK должен быть развернут в *кластерном окружении* с настройкой *быстрого поиска* по индексу.

Требование	Описание
Процессор	x86-64 CPU (Intel), >2.0 ГГц, >2 ядер. Для работы всех компонентов Сервера FindFace, кроме `findface-upload`, необходимо, чтобы процессор поддерживал расширение набора инструкций AVX.
Оперативная память	Потребление RAM зависит от количества лиц в базе данных. Используйте результаты эталонного теста, приведенные *ниже*, для определения необходимого вам объема памяти. Имейте в виду, что если векторы признаков (биометрические образцы) хранятся в 2-х и более галереях, вам потребуется умножить приведенное потребление памяти базами данных MongoDB и Tarantool на соответствующее количество галерей. Как правило, для хранения `10 000 000` лиц Tarantool необходимо 20 ГБ RAM. MongoDB не требует большого объема памяти, поскольку при необходимости задействует в качестве RAM пространство жесткого диска.
Жесткий диск	Хранение `10 000 000` лиц в MongoDB требует 24 ГБ, в Tarantool ~20х[кол-во снапшотов для каждого шарда] ГБ (по умолчанию, 20х3=60 ГБ).Для хранения всех загруженных изображений с помощью компонента `findface-upload`: суммарный размер всех загруженных изображений + 10%.
Операционная система	Ubuntu 16.04 LTS (только 64-битная версия).
Поддержка виртуальных машин	VMware

Ниже приведены результаты эталонного теста использования памяти компонентами FindFace Enterprise Server SDK. Используйте эти данные для вычисления нужного вам объема RAM.

Примечание: Использование памяти может незначительно флуктуировать от теста к тесту.

Примечание: В зависимости от своих нужд скорректируйте максимальное потребление памяти базой данных Tarantool в файле `/etc/tarantool/instances.enabled/FindFace.lua`.

Эталонный тест выполнен со следующими настройками:

- *Модель* биометрического образца: `apricot_320`.
- Модели для *распознавания пола, возраста и эмоций* (GAE в таблице): `fr_1_gender0`, `fr_1_age0`, `emotion_1`.

- Модели, использованные в компоненте *extraction-api*: `apricot_320`, `fr_1_gender0`, `fr_1_age0`, `emotion_1`.
- MongoDB, Tarantool: для хранения биометрических образцов используется только 1 галерея. Если в вашем случае таких галерей несколько, умножьте приведенное потребление RAM на соответствующее количество галерей.

Количество лиц	Потребление RAM по компонентам, МБ				
	MongoDB	Tarantool	nnapi	nnapi + GAE	extraction-api
0 (собственные нужды)	~70	~77	~265	~1000	~1ГБ (1 ядро)/~7ГБ (8 ядер) (до 10,5 под нагрузкой)
50000	~181	~189	~400	~1400	
100000	~294	~263	~400	~1400	
500000	~1190	~1013	~400	~1400	
1000000	~2310	~1943	~400	~1400	

2.1.2 Форматы изображений

FindFace Enterprise Server SDK поддерживает следующие форматы изображений:

- JPEG,
- PNG,
- WebP.

Максимальный размер изображения 10 МБ. Минимальное расстояние между зрачками 40 пикселей.

2.2 Работа с видео

2.2.1 Сервер под видеодетектор

Сервер, на котором установлен компонент *видеодетектор лиц*, должен отвечать следующим требованиям (считая, что видеопоток 1 x 720p (1280×720) 25 FPS):

Примечание: Требования в общем случае определяются интенсивностью движения и количеством лиц в кадре, настройками компонента, а также общей нагрузкой Сервера анализа и распознавания биометрических данных. Для выбора оптимальной конфигурации свяжитесь с нашими специалистами по адресу info@ntechlab.com.

Требование	Описание
Процессор	INTEL Core i5 6400 (CPU с 2-мя физическими ядрами). Необходима поддержка AVX.
Оперативная память	4 ГБ в режиме реального времени.
Операционная система	Ubuntu 16.04 LTS (только 64-битная версия).

2.2.2 Форматы видеофайлов и кодеков

Компонент `fkvideo_detector` поддерживает все форматы видеофайлов и кодеков, которые могут быть декодированы FFmpeg.

2.3 Поддержка веб-интерфейса

Для обработки видео в *веб-интерфейсе FindFace* физический сервер должен отвечать *требованиям к видеодетектору лиц*.

Глава 3

Выбор архитектуры развертывания

Для установки FindFace Enterprise Server SDK используются следующие установочные пакеты:

- Пакет с компонентами `<findface-repo>.deb`.
- Несколько пакетов с моделями нейронных сетей `<findface-data>.deb`. Каждая модель поставляется в отдельном пакете.

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

Схема	Рекомендация
Развертывание на одиночном сервере	FindFace Enterprise Server SDK может быть развернут на *одиночном сервере*, если количество лиц в базе биометрических данных ориентировочно не превышает `1 000 000`.
Развертывание на кластере серверов	Если база данных будет содержать более `1 000 000` лиц, FindFace Enterprise Server SDK должен быть развернут в *кластерном окружении* с настройкой быстрого поиска по индексу. Данная схема предназначена для использования в средних и крупных проектах и обладает возможностями практически неограниченного масштабирования. Подходит для профессиональных высоконагруженных проектов с жесткими требованиями к производительности.

3.1 Развертывание на одиночном сервере

Компоненты и модели нейронных сетей FindFace Enterprise Server SDK могут быть установлены на одиночном физическом сервере. Данная установка рекомендуется для использования в небольших про-

ектах уровня предприятия с числом лиц в базе данных до 1 000 000. Вы также можете использовать данный вариант установки в более крупных проектах на подготовительном этапе для формирования требований к проектируемой системе. Взаимодействие компонентов Сервера FindFace при таком варианте установки показано на схеме ниже:

Совет: В дополнение к Серверу FindFace вы также можете установить *расширенный функционал*.

Примечание: Установка на одиночном физическом сервере может быть выполнена как *пошагово*, так и из *консольного инсталлятора*.

Компонент	Описание
`findface-facenapi`	Реализованный на Python сервис, обеспечивающий функционирование HTTP API. Данный сервис выполняет функцию обнаружения лиц на фотографиях, взаимодействует с базой данных MongoDB и сервисами `findface-nnapi` и `tarantool@FindFace`.
`tntapi` (`tarantool@FindFace` как шард)	Сервис, обеспечивающий взаимодействие между сервисом `findface-facenapi` и базой биометрических данных на основе Tarantool. Для увеличения скорости поиска на каждом сервере с базой данных Tarantool могут быть созданы многочисленные сегменты («шарды») `tntapi`. Их параллельное функционирование приводит к значительному увеличению производительности (в 70-100 раз).
`findface-nnapi`	Сервис, выполняющий посредством нейронных сетей извлечение вектора признаков (биометрического образца) из обнаруженных лиц, а также распознавание пола, возраста и эмоций. Требует установки пакетов с *моделями нейронных сетей* `<findface-data>.deb`.
MongoDB	База данных, в которой хранятся метаданные лиц, подробная информация о галереях, настройки, векторы признаков и т. д.
`findface-upload`	Веб-сервер на базе nginx, который принимает изображения через WebDAV. Устанавливается, если требуется хранить исходные загруженные изображения, миниатюры и нормализованные изображения лиц на Сервере.
`NTLS`	Локальный сервер лицензий с управлением через веб-интерфейс, взаимодействующий для верификации лицензий с глобальным центром лицензий NtechLab. Для закрытых систем поддерживается работа с аппаратными лицензионными ключами.

3.2 Развертывание на кластере серверов

В средних и крупных высоконагруженных проектах с количеством лиц в базе данных более 1 000 000 FindFace Enterprise Server SDK должен быть установлен в кластерной среде. Взаимодействие компонентов Сервера FindFace при таком варианте установки показано на схеме ниже:

Совет: В дополнение к Серверу FindFace вы также можете установить *расширенный функционал*.

Компонент	Описание
`findface-facenapi`	Реализованный на Python сервис, обеспечивающий функционирование HTTP API. Данный сервис выполняет функцию обнаружения лиц на фотографиях, взаимодействует с базой данных MongoDB и сервисами `findface-nnapi` и `tarantool@FindFace`.
`tntapi` (`tarantool@FindFace` как шард)	Сервис, обеспечивающий взаимодействие между сервисом `findface-facenapi` и базой биометрических данных на основе Tarantool. Для увеличения скорости поиска на каждом сервере с базой данных Tarantool могут быть созданы многочисленные сегменты («шарды») `tntapi`. Их параллельное функционирование приводит к значительному увеличению производительности (в 70-100 раз).
`findface-nnapi`	Сервис, выполняющий посредством нейронных сетей извлечение вектора признаков (биометрического образца) из обнаруженных лиц, а также распознавание пола, возраста и эмоций. Требует установки пакетов с *моделями нейронных сетей* `<findface-data>.deb`.
MongoDB	База данных, в которой хранятся метаданные лиц, подробная информация о галереях, настройки, векторы признаков и т. д.
`findface-upload`	Веб-сервер на базе nginx, который принимает изображения через WebDAV. Устанавливается, если требуется хранить исходные загруженные изображения, миниатюры и нормализованные изображения лиц на Сервере.
`NTLS`	Локальный сервер лицензий с управлением через веб-интерфейс, взаимодействующий для верификации лицензий с глобальным центром лицензий NtechLab. Для закрытых систем поддерживается работа с аппаратными лицензионными ключами.

3.3 Список дополнительных возможностей

В дополнение к Серверу FindFace (установленному на *одном* или *нескольких* серверах) вы можете задействовать расширенный функционал, установив следующие компоненты из пакета `<findface-repo>.deb`:

Компонент	Описание
`fkvideo_detector`	Видеодетектор лиц *fkvideo_detector* обнаруживает лица «на лету» в видеопотоке или видеофайле и отправляет их через REST API на Сервер FindFace. Требуется дополнительная лицензия.
`findface-extraction-api`	С помощью *findface-extraction-api* вы можете гибко настраивать формат API-ответов для получения таких данных, как координаты рамки с лицом, нормализованное лицо, пол, возраст, эмоции, вектор признаков. Использование данной функции в системе может значительно расширить спектр аналитических задач, которые она в состоянии выполнить. Вы также можете использовать компонент как аналог компонента *findface-nnapi* для непосредственного извлечения биометрических образцов. Требуется дополнительная лицензия.
`findface-mass-enroll`	Функционал массовой загрузки лиц *findface-mass-enroll* позволяет отправлять лица в компонент `findface-facenapi` одновременно из множества изображений.
`findface-ui`	*Веб-интерфейс* FindFace Enterprise Server SDK обеспечивает удобный доступ к большинству функций, доступных через REST API, без необходимости написания кода.
`findface-tarantool-build-index`	Для галерей с количеством лиц более `1 000 000` должны быть проиндексированы с помощью компонента *findface-tarantool-build-index*.

Глава 4

Развертывание Сервера FindFace

4.1 Необходимое стороннее ПО

FindFace Enterprise Server SDK использует в работе следующее программное обеспечение сторонних производителей:

Стороннее ПО	Использование	Установка
MongoDB	База данных, обеспечивающая функционирование Сервера FindFace. В ней хранятся метаданные лиц, информация о галереях, биометрические образцы, данные внутреннего характера.	Вручную перед установкой компонентов Сервера
Tarantool	Гибко настраиваемая база данных, в которой хранятся только биометрические данные лиц (векторы признаков). Использование отдельной базы данных под биометрические образцы приводит к значительному уменьшению времени отклика системы.	Автоматически, вместе с компонентом `tntapi`.

В этом разделе:

4.1.1 MongoDB

Перед установкой компонентов Сервера необходимо установить базу данных MongoDB, которая будет служить внутренней базой данных Сервера. Вы можете установить базу данных MongoDB на одном сервере с компонентом `findface-facenapi` или на удаленном сервере. Если FindFace Enterprise Server SDK развертывается на одном физическом сервере, база данных MongoDB устанавливается там же. FindFace Enterprise Server SDK совместим с MongoDB 3.2 и более поздними версиями.

Установка базы данных MongoDB на сервере с `findface-facenapi`

Для установки последней стабильной версии MongoDB (на данный момент 3.4) на одном сервере с компонентом `findface-facenapi` выполните следующие действия:

Примечание: Для установки другой версии MongoDB ознакомьтесь с документацией по этой версии, например, с 3.2.

---

1. Импортируйте ключ подписи, используемый системой управления пакетами:

```
sudo apt-key adv --keyserver hkp://keyserver.ubuntu.com:80 --recv␣
↪0C49F3730359A14518585931BC711F9BA15703C6
```

2. Создайте список пакетов для MongoDB `/etc/apt/sources.list.d/mongodb-org-3.4.list`:

```
echo "deb [ arch=amd64,arm64 ] http://repo.mongodb.org/apt/ubuntu xenial/mongodb-org/3.4␣
↪multiverse" | sudo tee /etc/apt/sources.list.d/mongodb-org-3.4.list
```

3. Перезагрузите локальную базу данных пакетов:

```
sudo apt-get update
```

4. Установите последнюю стабильную версию MongoDB:

```
sudo apt-get install -y mongodb-org
```

5. Запустите сервис `mongod`:

```
sudo service mongod start
```

Установка базы данных MongoDB на удаленном сервере

Для установки MongoDB на удаленном сервере выполните следующие действия:

1. На удаленном сервере установите последнюю стабильную версию MongoDB по аналогии с *установкой на сервере с findface-facenapi*.
2. Откройте для редактирования файл конфигурации базы данных MongoDB:

```
sudo vi /etc/mongod.conf
```

3. Для того чтобы разрешить входящие подключения к базе данных со всех IP-адресов, закомментируйте строку `bind_ip = 127.0.0.1`. Убедитесь, что доступ к серверу с установленной базой данных возможен только из локальной сети.

```
#bind_ip = 127.0.0.1
```

4. Перезапустите сервис `mongod`:

```
sudo service mongod restart
```

Подключение к существующей базе данных MongoDB

При подключении к существующей базе данных MongoDB необходимо указать IP-адрес соответствующего сервера в настройках конфигурации.

4.1.2 Tarantool

FindFace Enterprise Server SDK совместим только с базой данных Tarantool версии `1.7.3.673.g23cc4dc-1`. Данная версия устанавливается автоматически вместе с компонентом `tntapi`.

4.2 Установка Сервера FindFace

Для вашего удобства мы предлагаем несколько вариантов установки FindFace Enterprise Server SDK. Выберите наиболее подходящий вариант в зависимости от выбранной вами архитектуры развертывания:

- Развертывание на кластере серверов может быть только *пошаговым*.
- Развертывание на одиночном сервере может быть выполнено *пошагово*, из консольного *инсталлятора*, а также в виде преднастроенного *образа виртуальной машины*.

> **Предупреждение:** Для высоконагруженных проектов установка в виде виртуальной машины не рекомендуется даже в тестовых целях.

4.2.1 Пошаговая установка

Данный раздел содержит сведения о пошаговой настройке лицензирования и установке компонентов Сервера FindFace. Выполните приведенные ниже инструкции, придерживаясь заданного порядка.

Совет: Установка на одиночном физическом сервере также может быть выполнена с помощью *консольного инсталлятора* или в виде полностью настроенного готового к использованию *образа виртуальной машины*.

В этом разделе:

Подготовка deb-пакетов к установке

Для получения установочных пакетов FindFace Enterprise Server SDK свяжитесь со своим менеджером компании NtechLab. Для того чтобы подготовить пакеты к установке, выполните следующие действия:

Предупреждение: На данном этапе будет автоматически создан пользователь `ntech`. Во избежание конфликта, убедитесь, что пользователь с таким именем отсутствует в системе.

1. Распакуйте пакет с компонентами на каждом из физических серверов развертывания.

```
sudo dpkg -i <findface-repo>.deb
```

2. Добавьте ключ подписи на каждом из серверов развертывания.

```
sudo apt-key add /var/findface-repo/public.key
sudo apt-get update
```

3. Распакуйте пакеты с *моделями нейронных сетей* (биометрия лица, пол, возраст, эмоции). Если FindFace Enterprise Server SDK развертывается на кластере серверов, модели устанавливаются только на серверах `findface-nnapi`.

```
sudo dpkg -i findface-data*
```

Лицензирование

Вы получаете файл лицензии вместе с установочными пакетами FindFace Enterprise Server SDK от менеджера NtechLab. Если вы выберите лицензирование в закрытой сети, вам также будет отправлен ключ аппаратной защиты Guardant. Принцип лицензирования FindFace Enterprise Server SDK показан на схеме ниже:

Этапы лицензирования:

1. Установите и настройте локальный сервер лицензий `NTLS`.
2. Если лицензируемые компоненты (`findface-nnapi`, `tntapi`, `fkvideo_detector` и `extraction-api`) установлены на удаленных серверах, укажите IP-адрес сервера NTLS в их файлах конфигурации.

Для того чтобы установить и настроить NTLS, выполните следующие действия:

1. Установите компонент NTLS:

```
sudo apt-get update
sudo apt-get install ntls
```

Совет: В файле конфигурации NTLS вы можете изменить папку для хранения файла лицензии и при необходимости указать IP-адрес прокси-сервера для доступа в Интернет. Вы также можете настроить удаленный доступ к веб-интерфейсу NTLS, используемому для управления лицензией. Для того чтобы открыть файл конфигурации NTLS, выполните команду:

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

При необходимости укажите в параметре `license-dir` другую папку для хранения файла лицензии. По умолчанию файл лицензии хранится в папке `/ntech/license`:

```
license-dir = /ntech/license
```

При необходимости раскомментируйте строку `proxy` и укажите IP-адрес прокси-сервера:

```
proxy = http://192.168.1.1:12345
```

По умолчанию доступ в веб-интерфейс NTLS возможен с любого удаленного сервера (`ui = 0.0.0.0:3185`). Для того чтобы обеспечить доступ к веб-интерфейсу NTLS только с определенного IP-адреса, отредактируйте параметр `ui`:

```
ui = 127.0.0.1:3185
```

2. Добавьте сервис NTLS в автозагрузку и запустите сервис:

```
sudo systemctl enable ntls && sudo systemctl start ntls
```

3. Загрузите файл лицензии в веб-интерфейсе NTLS по адресу `http://<IP-адрес NTLS>:3185/#/`. Вы также можете использовать веб-интерфейс NTLS для продления и апгрейда лицензии.

4. Для лицензирования в закрытой сети вставьте ключ Guardant в USB-порт.

Важно: Если лицензируемые компоненты (`findface-nnapi`, `tntapi`, `fkvideo_detector` и `extraction-api`) будут установлены на удаленных серверах, обязательно укажите IP-адрес сервера NTLS в их файлах конфигурации после установки.

См.также:

Устранение неполадок с лицензией и NTLS

Установка компонентов Сервера

После того как deb-пакеты FindFace Enterprise Server SDK подготовлены, а лицензирование настроено, установите компоненты Сервера в соответствии с архитектурной схемой.

Установка findface-facenapi

Установите и настройте компонент `findface-facenapi` следующим образом:

1. Установите компонент.

```
sudo apt-get update
sudo apt-get install python3-facenapi
```

2. Если база данных MongoDB установлена на удаленном сервере, укажите его IP адрес в файле конфигурации `findface-facenapi`.

```
sudo vi /etc/findface-facenapi.ini

mongo_host = '192.168.113.1'
```

3. Проверьте работоспособность компонента. Для этого вызовите приложение `findface-facenapi`, выполнив приведенную ниже команду. После вызова подождите минуту. Если ошибок не обнаружено, верните контроль над командной строкой, нажав сочетание клавиш `Ctrl+C`.

   Если база данных MongoDB установлена на одном сервере с компонентом `findface-facenapi`, выполните:

```
findface-facenapi
```

   Если база данных MongoDB установлена на удаленном сервере, выполните:

```
sudo findface-facenapi --config=/etc/findface-facenapi.ini
```

4. Проверьте, не добавлен ли сервис `findface-facenapi` в автозагрузку Ubuntu.

```
systemctl list-unit-files | grep findface-facenapi
```

5. Добавьте сервис в автозагрузку и запустите его.

```
sudo systemctl enable findface-facenapi.service && sudo service findface-facenapi start
```

6. Убедитесь, что сервис активен. Команда вернет описание сервиса, его статус (должен быть Активен), путь и длительность текущей сессии.

```
sudo service findface-facenapi status
```

Совет: Для того чтобы отобразить *логи* `findface-facenapi`, выполните команду:

```
sudo tail -f /var/log/syslog | grep facenapi
```

Установка findface-nnapi

Установите и настройте компонент `findface-nnapi` следующим образом:

Совет: Возможно, вам также захочется узнать больше о *распознавании пола, возраста и эмоций*.

1. Установите компонент.

```
sudo apt-get update
sudo apt-get install findface-nnapi
```

2. Если NTLS установлен на удаленном сервере, укажите его IP адрес в файле конфигурации `findface-nnapi`.

```
sudo vi /etc/findface-nnapi.ini

license_ntls_server = 192.168.113.2:3133
```

3. Проверьте работоспособность компонента. Для этого вызовите приложение `findface-nnapi`, выполнив приведенную ниже команду. После вызова подождите минуту. Если ошибок не обнаружено, верните контроль над командной строкой, нажав сочетание клавиш `Ctrl+C`.

```
findface-nnapi
```

4. Проверьте, не добавлен ли сервис `findface-nnapi` в автозагрузку Ubuntu.

```
systemctl list-unit-files | grep findface-nnapi
```

5. Добавьте сервис в автозагрузку и запустите его.

```
sudo systemctl enable findface-nnapi.service && sudo service findface-nnapi start
```

6. Убедитесь, что сервис активен. Команда вернет описание сервиса, его статус (должен быть Активен), путь и длительность текущей сессии.

```
sudo service findface-nnapi status
```

Совет: Для того чтобы отобразить *логи* `findface-nnapi`, выполните команду:

```
sudo tail -f /var/log/syslog | grep nnapi
```

Установка findface-upload

Для хранения на Сервере FindFace всех обработанных исходных изображений, а также таких артефактов Сервера, как миниатюры и нормализованные изображения лиц, установите и настройте компонент `findface-upload`.

Совет: Пропустите данный шаг, если вы не собираетесь хранить эти данные. В этом случае на Сервере будут храниться только биометрические образцы (в базах данных `MongoDB` и `Tarantool`).

Выполните следующие действия:

1. Установите компонент:

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

2. По умолчанию исходные изображения, миниатюры и нормализованные изображения лиц хранятся в папке `/var/lib/ffupload/uploads/`. Вы можете просмотреть содержимое папки прямо в браузере по адресу `http://127.0.0.1:3333/uploads/`. Убедитесь, что этот адрес доступен.

```
curl -I http://127.0.0.1:3333/uploads/
##HTTP/1.1 200 OK
```

Важно: Вам потребуется указать его в настройках сети.

См.также:

Устранение неполадок при работе с папкой Uploads

Установка tntapi

Компонент `tntapi` соединяет базу данных Tarantool и компонент `findface-facenapi`, передавая результаты поиска от базы данных в `findface-facenapi` для дальнейшей обработки. Для увеличения скорости поиска на каждом сервере с базой данных `Tarantool` могут быть созданы многочисленные сегменты (шарды) `tntapi`. Их параллельное функционирование приводит к значительному увеличению производительности. Каждый шард может обрабатывать приблизительно до `10 000 000` лиц. Если FindFace Enterprise Server SDK развертывается на одном сервере, одного шарда, созданного по умолчанию, будет достаточно. При развертывании в кластерной среде количество шардов рассчитывается в зависимости от нескольких параметров (см. ниже).

Установка tntapi на одном сервере

Установите и настройте компонент `findface-nnapi` следующим образом:

1. Установите компонент `tntapi`. База данных Tarantool будет установлена автоматически вместе с ним.

```
sudo apt-get update
sudo apt-get install 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. Поскольку один шард может обрабатывать до `10 000 000` лиц, для обслуживания небольших проектов до `1 000 000` лиц будет достаточно задействовать шард `tntapi`, созданный по умолчанию (`tarantool@FindFace`). Настройки конфигурации данного шарда задаются в файле `/etc/tarantool/instances.enabled/FindFace.lua`. Настоятельно рекомендуется ничего не добавлять и не редактировать в данном файле, за исключением максимального использования оперативной памяти (`memtx_memory`), IP-адреса локального сервера лицензий NTLS, необходимого для лицензирования `tntapi`, а также настроек удаленного доступа. Максимальное использование памяти задается в байтах в зависимости от количества лиц, обрабатываемых шардом, исходя из соотношения примерно 1280 байт на 1 лицо.

   Откройте файл конфигурации:

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

   Отредактируйте значение в зависимости от количества лиц, обрабатываемых шардом. Значение `1.2*1024*1024*1024` соответствует `1 000 000` лиц:

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

   Укажите IP-адрес сервера NTLS, если он удаленный:

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

Если Сервер FindFace развертывается 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, 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. В примере ниже доступ возможен только с IP-адреса `192.168.113.10`:

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

Здесь доступ возможен с любого IP-адреса:

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

4. Добавьте шард `tntapi` в автозагрузку Ubuntu и запустите шард.

```
sudo systemctl enable tarantool@FindFace && sudo systemctl start tarantool@FindFace
```

5. Убедитесь, что шард активен. Команда вернет описание сервиса, его статус (должен быть Активен), путь и длительность текущей сессии.

```
sudo systemctl status tarantool@FindFace
```

Совет: Для того чтобы отобразить *логи* `tntapi`, выполните команду:

```
sudo tail -f /var/log/tarantool/FindFace.log
```

---

6. Файл `tntapi.json` с описанием параметров шарда автоматически устанавливается вместе с компонентом `tntapi` в папку `/etc`.

Важно: Вам потребуется раскомментировать путь к данному файлу в настройках сети.

Установка tntapi на кластере серверов

Установите и настройте компонент `findface-nnapi` следующим образом:

1. Установите компонент `tntapi` на выделенных серверах. База данных Tarantool будет установлена автоматически вместе с ним.

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

2. Создайте шарды `tntapi` на каждом сервере. Процедура шардинга будет рассмотрена на примере кластерной среды, содержащей 4 сервера под базу данных, на каждом из которых нужно создать по 4 шарда.

Важно: При создании шардов для крупных проектов соблюдайте следующие правила:

1. Один шард может обрабатывать приблизительно `10 000 000` лиц.

2. Количество шардов на одном сервере не должно превышать число физических ядер процессора минус 1.

3. На каждом из 4-х серверов удалите тестовый сервис Tarantool из автозагрузки Ubuntu и остановите его.

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

4. На каждом из 4-х серверов отключите созданный по умолчанию шард.

```
sudo systemctl disable tarantool@FindFace
```

5. Напишите bash-скрипт `shard.sh`, который автоматически создаст шарды и файлы конфигурации для каждого из них. Скрипт должен быть написан для каждого из 4-х серверов. Используйте приведенный ниже скрипт как основу для своего кода. Данный скрипт создает на сервере 4 шарда, использующие порты: tntapi `33001..33004` и http `8001..8004`.

Важно: Скрипт создает файлы конфигурации на основе настроек по умолчанию, хранящихся в файле `/etc/tarantool/instances.enabled/FindFace.lua`. Настоятельно рекомендуется ничего не добавлять и не редактировать в этом файле, за исключением максимального использования оперативной памяти (`memtx_memory`) и IP-адреса локального сервера лицензий NTLS, необходимого для лицензирования `tntapi`. Максимальное использование памяти задается для каждого шарда в байтах в зависимости от количества лиц, обрабатываемых шардом, исходя из соотношения примерно 1280 байт на 1 лицо.

Откройте файл конфигурации:

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

Отредактируйте значение в зависимости от количества лиц, обрабатываемых шардом. Значение `1.2*1024*1024*1024` соответствует `1 000 000` лиц:

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

Укажите IP-адрес сервера NTLS, если он удаленный:

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

```
#!/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\", $HTTP_PORT,/;
       " /etc/tarantool/instances.enabled/FindFace.lua > /etc/tarantool/instances.enabled/
↪FindFace_shard_$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
```

(continues on next page)

(продолжение с предыдущей страницы)

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

Совет: Загрузите пример скрипта.

6. Запустите скрипт из домашней директории.

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

7. Проверьте, созданы ли 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
```

8. Запустите все 4 шарда. Это нужно сделать на каждом сервере.

```
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;
```

9. Проверьте статус шардов.

```
sudo systemctl status tarantool@FindFace*
```

Вы должны получить следующие сообщения:

```
tarantool@FindFace_shard_3.service - Tarantool Database Server
Loaded: loaded (/lib/systemd/system/tarantool@.service; disabled; vendor preset: enabled)
Active: active (running) since Tue 2017-01-10 16:22:07 MSK; 32s ago
...
tarantool@FindFace_shard_2.service - Tarantool Database Server
Loaded: loaded (/lib/systemd/system/tarantool@.service; disabled; vendor preset: enabled)
Active: active (running) since Tue 2017-01-10 16:22:07 MSK; 32s ago
...
tarantool@FindFace_shard_1.service - Tarantool Database Server
Loaded: loaded (/lib/systemd/system/tarantool@.service; disabled; vendor preset: enabled)
Active: active (running) since Tue 2017-01-10 16:22:07 MSK; 32s ago
...
tarantool@FindFace_shard_4.service - Tarantool Database Server
Loaded: loaded (/lib/systemd/system/tarantool@.service; disabled; vendor preset: enabled)
Active: active (running) since Tue 2017-01-10 16:22:07 MSK; 32s ago
...
```

Совет: Для того чтобы отобразить *логи* tntapi, выполните команду:

```
sudo tail -f /var/log/tarantool/FindFace_shard_{1,2,3,4}.log
```

10. На сервере с компонентом findface-facenapi создайте файл tntapi_cluster.json, содержащий адреса и порты всех шардов. Распределите доступные шарды равномерно по 1024 ячейкам в одной строке. Нажмите здесь, чтобы посмотреть, как выглядит файл для 4 серверов с 4 шардами на каждом.

Совет: Вы можете создать файл `tntapi_cluster.json` следующим образом:

1. Создайте файл с линейным списком всех шардов, каждый шард с новой строки (нажмите здесь, чтобы посмотреть пример).

```
sudo vi s.txt
```

2. Выполните приведенный ниже скрипт. В результате будет создан новый файл `tntapi_cluster.json`, содержащий список адресов и портов всех шардов, равномерно распределенных по 1024 ячейкам.

```
cat s.txt | perl -lane 'push(@s,$_); END{$m=1024; $t=scalar @s;for($i=0;$i<
↪$m;$i++){$k=int($i*$t/$m); push(@r,"\"".$s[$k]."\"")} print "[[".join(",
↪",@r)."]]"; }' > tntapi_cluster.json
```

11. Переместите файл `tntapi_cluster.json` в папку `/etc`.

Важно: Вам потребуется раскомментировать и указать путь к данному файлу в настройках сети.

Сетевые настройки

После установки компонентов Сервера FindFace настройте их взаимодействие друг с другом. Выполните следующие действия:

1. Откройте для редактирования файл конфигурации `findface-facenapi.ini`:

```
sudo vi /etc/findface-facenapi.ini
```

2. Отредактируйте настройки в соответствии с фактическим распределением компонентов:

```
ffupload_url = 'http://127.0.0.1:3333'
mongo_host = '127.0.0.1'
nnapi_url = 'http://127.0.0.1:18088'
tntapi_servers_file = '/etc/tntapi.json'
```

Предупреждение: Содержимое файла `findface-facenapi.ini` должно представлять собой синтаксически верный код Python.

Примечание: Не меняйте значение параметра `ffupload_url`, если компонент `findface-upload` не установлен.

3. По умолчанию, если один или несколько шардов `tntapi` будут недоступны во время идентификации лица, компонент `findface-facenapi` вернет ошибку. При необходимости раскомментируйте параметр `tntapi_ignore_search_error` и присвойте ему значение `True`. В этом случае `findface-facenapi` будет использовать для идентификации только доступные шарды `tntapi` и укажет в ответе отношение количества доступных шардов `tntapi` к их общему количеству.

```
tntapi_ignore_search_errors = True
```

4. Перезапустите все сервисы Сервера FindFace, а также сервис nginx (если компонент `findface-upload` установлен) на соответствующих серверах.

```
sudo service 'findface*' restart
sudo service nginx restart
```

5. Проверьте статус сервисов. Команда вернет описание сервисов, их статус (должен быть Активен), путь и длительность текущей сессии.

```
sudo service 'findface*' status
sudo service nginx status
```

4.2.2 Установка из консольного инсталлятора

Для развертывания FindFace Enterprise Server SDK на одиночном сервере можно использовать консольный инсталлятор.

Предупреждение: Инсталлятор не предназначен для обновления FindFace Enterprise Server SDK с версии 2.3 или более ранних версий.

См.также:

- *Пошаговая установка*
- *Установка в виде преднастроенной виртуальной машины*

Выполните следующие действия:

1. Загрузите файл инсталлятора `<findface-server-xxx>.run`.
2. Поместите файл `.run` в любую папку на сервере установки (например, `/home/username`).
3. Из данной папки сделайте файл `.run` в исполняемым.

```
chmod +x <findface-server-xxx>.run
```

4. Запустите файл `.run`.

   Предупреждение: На данном этапе будет автоматически создан пользователь `ntech`. Во избежание конфликта, убедитесь, что пользователь с таким именем отсутствует в системе.

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

   Инсталлятор проверит, соответствует ли сервер системным требованиям. После этого компоненты FindFace Enterprise Server SDK будут автоматически установлены, настроены и запущены в соответствии со следующей конфигурацией:

Компонент	Особенности установки
`findface-facenapi`	Устанавливается и запускается с включенной и настроенной *группировкой лиц одного человека* в базе данных и *распознаванием «свой-чужой»*.
`findface-nnapi`	Устанавливается и запускается с количеством экземпляров: `N = min(cores, RAM/2Gb)/2` с включенным и настроенным *распознаванием пола, возраста и эмоций*.
`findface-server-tarantool` (tntapi)	Устанавливается и запускается с количеством шардов tntapi: `N = min(cores, RAM/2Gb)/2`
`findface-tarantool-index`	Устанавливается. Перед использованием ознакомьтесь с *документацией по компоненту*.
`ffupload`	Устанавливается и запускается.
`fkvideo_detector`	Устанавливается. Для ручного запуска используйте командную строку или веб-интерфейс FindFace Web UI. Перед использованием ознакомьтесь с *документацией по компоненту*.
Extraction API	Устанавливается. Только для опытных пользователей. Перед использованием обязательно ознакомьтесь с *документацией по компоненту*.
`NTLS`	Устанавливается и запускается.
Веб-интерфейс FindFace	Устанавливается и запускается.
`findface-mass-enroll`	Устанавливается. Для работы с компонентом используйте командную строку. Перед использованием ознакомьтесь с *документацией по компоненту*.
nginx	Устанавливается и запускается.
База данных MongoDB	Устанавливается и запускается.
База данных Tarantool	Устанавливается и запускается.
jq	Устанавливается. Используется для структурирования API-ответов от FindFace Enterprise Server SDK в формате JSON.

5. По завершении установки в консоль будет выведена информация, необходимая для использования FindFace Enterprise Server SDK.

Совет: Обязательно сохраните эти данные: они вам понадобятся.

```
################################################
#          Installation is complete            #
################################################
- upload your license to http://172.16.213.249:3185/
  login:          admin
  password:       fZh9-zZDX
- user interface: http://172.16.213.249:8000/
- token for UI:   fZh9-zZDX
- documentation:  http://172.16.213.249:8000/v1/docs/v1/overview.html
Should you forget your password, recover it by executing
  findface-facenapi.token
 user@ubuntu:~$
```

6. Загрузите файл лицензии через веб-интерфейс NTLS `http://<IP_адрес_сервера>:3185/#/`. Для доступа в веб-интерфейс NTLS используйте логин и пароль, выведенные в консоли.

Примечание: IP-адрес сервера в ссылках на веб-интерфейсы FindFace имеет вид 127.0.0.1 или

<IP_адрес_в_сети>, в зависимости от того, принадлежит ли сервер к сети.

4.2.3 Установка в виде преднастроенной виртуальной машины

Вы можете развернуть FindFace Enterprise Server SDK в виде полностью настроенного готового к использованию образа виртуальной машины, работающего в среде виртуализации на любой операционной системе. Данный тип установки является самым простым и требует минимальных навыков.

Важно: Данный тип установки подходит только для развертывания на *одиночном сервере*.

Предупреждение: Для высоконагруженных проектов установка в виде виртуальной машины не рекомендуется даже в тестовых целях.

См.также:

- *Пошаговая установка*
- *Установка из консольного инсталлятора*

Важно: Мы официально поддерживаем только среды виртуализации на базе продуктов `VMware`. Установите необходимое программное обеспечение, перед тем как приступить к выполнению настоящей инструкции.

Совет: Для получения образа виртуальной машины обратитесь к своему менеджеру NtechLab по адресу info@ntechlab.com. Вам будут предоставлены файлы `ffserver-*.ovf` и `disk-*.vmdk` (по отдельности или в архиве).

Образ виртуальной машины содержит следующее предустановленное программное обеспечение:

- Ubuntu Sever 16.04 LTS x64 без графического интерфейса пользователя
- FindFace Enterprise Server SDK в следующей конфигурации:

Компонент	Особенности установки
findface-facenapi	Устанавливается и запускается с включенной и настроенной *группировкой лиц одного человека* в базе данных и *распознаванием «свой-чужой»*.
findface-nnapi	Устанавливается и запускается (в 1 экземпляре) с включенным и настроенным *распознаванием пола, возраста и эмоций*. Может потребоваться *балансировка нагрузки*.
findface-server-tarantool (tntapi)	Устанавливается и запускается (1 шард). Может потребоваться шардинг.
findface-tarantool-build- index	Устанавливается. Перед использованием ознакомьтесь с *документацией по компоненту*.
ffupload	Устанавливается и запускается.
fkvideo_detector	Устанавливается. Для ручного запуска используйте командную строку или веб-интерфейс FindFace Web UI. Перед использованием ознакомьтесь с *документацией по компоненту*.
Extraction API	Устанавливается. Только для опытных пользователей. Перед использованием обязательно ознакомьтесь с *документацией по компоненту*.
NTLS	Устанавливается и запускается.
Веб-интерфейс FindFace	Устанавливается и запускается.
findface-mass-enroll	Устанавливается. Для работы с компонентом используйте командную строку. Перед использованием ознакомьтесь с *документацией по компоненту*.
nginx	Устанавливается и запускается.
MongoDB	Устанавливается и запускается.
Tarantool Database	Устанавливается и запускается.
jq	Устанавливается. Используется для структурирования API-ответов от FindFace Enterprise Server SDK в формате JSON.

Для развертывания FindFace Enteprise Server SDK в виде виртуальной машины выполните следующие действия:

1. Поместите файлы виртуальной машины `ffserver-*.ovf` и `disk-*.vmdk` в общий каталог.
2. Запустите среду виртуализации. Нажмите *Open a Virtual Machine* и выберите файл `ffserver-*.ovf`. По запросу конвертируйте файл в формат VMware. Это может занять некоторое время.
3. По завершении импорта виртуальной машины в среду виртуализации откройте настройки ее аппаратного обеспечения: *Edit virtual machine settings* → *Hardware*.

Совет: Ознакомьтесь с официальной документацией VMware.

- Выберите тип сетевого подключения с учетом сетевой конфигурации хоста.
- По умолчанию аппаратное обеспечение виртуальной машины уже настроено таким образом, чтобы обеспечить оптимальную производительность в большинстве систем со средней нагрузкой. Убедитесь, что оно удовлетворяет требованиям и вашего проекта. Если вы собираетесь одновременно обрабатывать несколько видеопотоков или работать с большим объемом

данных, увеличьте RAM виртуальной машины и количество ядер процессора. Сохраните настройки.

Важно: Для еще большего увеличения производительности создайте дополнительные шарды `tntapi` и настройте *балансировку нагрузки* `findface-nnapi` после выполнения данной инструкции (через консоль виртуальной машины).

---

4. Включите виртуальную машину, нажав *Power On*. Дождитесь окончания загрузки Ubuntu.
5. Для входа в систему введите логин `user` и пароль `ntechlab`.
6. Определите IP-адрес основного сетевого интерфейса виртуальной машины (`192.168.112.144` в примере).

```
ifconfig

ens33 Link encap:Ethernet HWaddr 00:0c:29:8f:db:d5
inet addr:192.168.112.144 Bcast:192.168.112.255 Mask:255.255.255.0
inet6 addr: fe80::20c:29ff:fe8f:dbd5/64 Scope:Link
UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1
RX packets:37751 errors:0 dropped:0 overruns:0 frame:0
TX packets:36205 errors:0 dropped:0 overruns:0 carrier:0
collisions:0 txqueuelen:1000
RX bytes:5621377 (5.6 MB) TX bytes:39193951 (39.1 MB)

lo Link encap:Local Loopback
inet addr:127.0.0.1 Mask:255.0.0.0
inet6 addr: ::1/128 Scope:Host
UP LOOPBACK RUNNING MTU:65536 Metric:1
RX packets:152521 errors:0 dropped:0 overruns:0 frame:0
TX packets:152521 errors:0 dropped:0 overruns:0 carrier:0
collisions:0 txqueuelen:1000
RX bytes:24549909 (24.5 MB) TX bytes:24549909 (24.5 MB)
```

7. Присвойте IP-адрес основного сетевого интерфейса параметру `ffupload_url` в файле конфигурации `findface-facenapi`.

```
sudo vi /etc/findface-facenapi.ini

ffupload_url = 'http://192.168.112.144:3333'
```

Предупреждение: Содержимое файла `findface-facenapi.ini` должно представлять собой синтаксически верный код Python.

8. Перезапустите сервисы FindFace Enterprise Server SDK.

```
sudo service 'findface*' restart
```

9. Сделайте IP-адрес виртуальной машины статическим. Для этого откройте файл `etc/network/interfaces` и измените текущую запись для основного сетевого интерфейса так, как показано в примере ниже. Замените адреса в примере на актуальные с учетом настроек сети.

Важно: С осторожностью редактируйте файл `etc/network/interfaces`. Перед тем как присту-

пить к редактированию, ознакомьтесь с инструкцией по настройке сетей Ubuntu.

```
sudo vi /etc/network/interfaces

# The primary network interface
iface eth0 inet static
address 192.168.112.144
netmask 255.255.255.0
gateway 192.168.112.254
dns-nameservers 192.168.112.254
```

10. Перезапустите сетевые интерфейсы.

```
sudo service networking restart
```

11. Загрузите файл лицензии FindFace Enterprise Server SDK через веб-интерфейс NTLS `http://<IP_адрес>:3185/#/` (`http://192.168.112.144:3185/#/` в примере).
12. Создайте *токен авторизации*. Используйте его для доступа к *веб-интерфейсу FindFace* по адресу `http://<IP_адрес>:8000/`.

4.3 Создание токена авторизации

После того как установка компонентов Сервера FindFace завершена, сгенерируйте токен авторизации в длинном или коротком формате, в зависимости от предпочтений. Используйте созданный токен для авторизации вашего экземпляра FindFace Enterprise Server SDK в API-запросах.

Для создания длинного токена выполните команду:

```
findface-facenapi.token

##0123456789_abcdefghijklmnopqrstuvw
```

Для создания короткого токена выполните команду:

```
findface-facenapi.token --short
##A0B1-C2D3
```

Если база данных MongoDB установлена на удаленном сервере, в команде нужно указать путь к файлу конфигурации `findface-facenapi.ini`.

```
sudo findface-facenapi.token --config=/etc/findface-facenapi.ini
```

4.4 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`.

Примечание: Сообщения запросов приведены в качестве примера. Вам потребуется заменить токен авторизации в запросах на *актуальный*.

Совет: Вы можете найти примеры кода на C#, PHP, Java и Python на нашем ресурсе GitHub.

В этом разделе:

4.4.1 Структурирование ответов на запросы

Используйте обработчик `jq`, чтобы структурировать данные в формате JSON в ответах на запросы. Для того чтобы установить `jq`, выполните команду:

```
sudo apt-get install jq
```

4.4.2 Получение списка галерей

Данный запрос возвращает имя единственной на данный момент галереи (создана по умолчанию). Соответствующий метод REST API: */galleries GET*.

Запрос

```
curl -H "Authorization: Token t3WGNhZbyaE_GFyQaywYllFoR2QkHXi-" http://localhost:8000/v0/galleries␣
↪| jq
```

Ответ

```
{
  "results": [
    "default"
  ]
}
```

4.4.3 Создание галереи

Данный запрос создает новую галерею `testgal`. Соответствующий метод REST API: */galleries/new POST*.

Запрос

```
curl -H "Authorization: Token t3WGNhZbyaE_GFyQaywYllFoR2QkHXi-" -X POST http://localhost:8000/v0/
↪galleries/testgal | jq
```

Ответ

```
{
  "name": "testgal"
}
```

4.4.4 Обнаружение лица на фотографии

Данный запрос обнаруживает лицо на тестовом изображении, размещенном в сети Интернет, и возвращает координаты рамки вокруг лица (*bbox*). Соответствующий метод REST API: */detect POST*.

Запрос

```
curl -H "Authorization: Token t3WGNhZbyaE_GFyQaywYllFoR2QkHXi-" -F "photo=http://static.findface.
↪pro/sample.jpg" http://localhost:8000/v0/detect | jq
```

Ответ

```
{
  "faces": [
    {
      "x1": 595,
      "x2": 812,
      "y1": 127,
      "y2": 344
    }
  ],
  "orientation": 1
}
```

4.4.5 Добавление лица в галерею

Данный запрос обрабатывает тестовое изображение из предыдущего запроса, обнаруживает лицо и добавляет его с уникальной меткой в галерею по умолчанию. Соответствующий метод REST API: */face POST*.

Запрос

```
curl -H "Authorization: Token t3WGNhZbyaE_GFyQaywYllFoR2QkHXi-" -F "photo=http://static.findface.
↪pro/sample.jpg" -F "meta=Sam Berry" http://localhost:8000/v0/face | jq
```

Ответ


```
{
  "results": [
    {
      "friend": false,
      "galleries": [
        "default"
      ],
      "id": 3827229391220303,
      "meta": "Sam Berry",
      "normalized": "http://192.168.113.88:3333/uploads//20170517/1495011480937809.jpeg",
      "person_id": 5,
      "photo": "http://192.168.113.88:3333/uploads//20170517/14950114809306293.jpeg",
      "photo_hash": "53477c4a72f52c6efc951d9c7ece42bc",
      "thumbnail": "http://192.168.113.88:3333/uploads//20170517/149501148093593.jpeg",
      "timestamp": "2017-05-17T08:58:00.930572",
      "x1": 595,
      "x2": 812,
      "y1": 127,
      "y2": 344
    }
  ]
}
```


Следующий запрос также добавляет лицо в галерею, но на этот раз лицо должно быть обнаружено на локальном изображении, а галерея является пользовательской (`testgal`).

Запрос

```
curl -H "Authorization: Token t3WGNhZbyaE_GFyQaywYllFoR2QkHXi-" -F "photo=@sample.jpg" -F
↪"meta=sample" -F "galleries=testgal" http://localhost:8000/v0/face | jq
```

Ответ


```
{
  "results": [
    {
      "friend": false,
      "galleries": [
```


(continues on next page)

(продолжение с предыдущей страницы)


```
        "default",
        "testgal"
      ],
      "id": 3827229578000564,
      "meta": "sample",
      "normalized": "http://192.168.113.88:3333/uploads//20170517/14950115538997407.jpeg",
      "person_id": 5,
      "photo": "http://192.168.113.88:3333/uploads//20170517/14950115538939695.jpeg",
      "photo_hash": "53477c4a72f52c6efc951d9c7ece42bc",
      "thumbnail": "http://192.168.113.88:3333/uploads//20170517/14950115538985784.jpeg",
      "timestamp": "2017-05-17T08:59:13.893921",
      "x1": 595,
      "x2": 812,
      "y1": 127,
      "y2": 344
    }
  ]
}
```


4.4.6 Поиск лица в галерее

Следующие 2 запроса обрабатывают изображение в сети Интернет (№1) и локальное изображение (№2), обнаруживают на них лицо и сравнивают его с лицами, хранящимися в галерее по умолчанию. Возвращают данные о наиболее схожих лицах и процент схожести. Соответствующий метод REST API: */identify POST*.

Запрос №1

```
curl -H "Authorization: Token t3WGNhZbyaE_GFyQaywYllFoR2QkHXi-" -F "photo=http://static.findface.
↪pro/sample2.jpg" http://localhost:8000/v0/identify | jq
```

Ответ


```
{
  "results": {
    "[515, 121, 821, 427]": [
      {
        "confidence": 0.9373,
        "face": {
          "age": 26.0483455657959,
          "emotions": [
            "neutral",
            "sad"
          ],
          "friend": false,
          "galleries": [
            "default"
          ],
          "gender": "female",
          "id": 3827062458772442,
          "meta": "Sam Berry",
```


(continues on next page)

(продолжение с предыдущей страницы)


```
          "normalized": "http://192.168.113.88:3333/uploads//20170516/1494946272949371.jpeg",
          "person_id": 5,
          "photo": "http://192.168.113.88:3333/uploads//20170516/14949462729435823.jpeg",
          "photo_hash": "53477c4a72f52c6efc951d9c7ece42bc",
          "thumbnail": "http://192.168.113.88:3333/uploads//20170516/14949462729480093.jpeg",
          "timestamp": "2017-05-16T14:51:12.943000",
          "x1": 595,
          "x2": 812,
          "y1": 127,
          "y2": 344
        }
      }
    ]
  }
}
```


Запрос №2

```
curl -H "Authorization: Token t3WGNhZbyaE_GFyQaywYllFoR2QkHXi-" -F "photo=@Pictures/sample.jpg"␣
↪http://localhost:8000/v0/identify | jq
```

Ответ


```
{
  "results": {
    "[595, 127, 812, 344]": [
      {
        "confidence": 0.9999,
        "face": {
          "age": 26.0483455657959,
          "emotions": [
            "neutral",
            "sad"
          ],
          "friend": false,
          "galleries": [
            "default"
          ],
          "gender": "female",
          "id": 3827062458772442,
          "meta": "Sam Berry",
          "normalized": "http://192.168.113.88:3333/uploads//20170516/1494946272949371.jpeg",
          "person_id": 5,
          "photo": "http://192.168.113.88:3333/uploads//20170516/14949462729435823.jpeg",
          "photo_hash": "53477c4a72f52c6efc951d9c7ece42bc",
          "thumbnail": "http://192.168.113.88:3333/uploads//20170516/14949462729480093.jpeg",
          "timestamp": "2017-05-16T14:51:12.943000",
          "x1": 595,
          "x2": 812,
          "y1": 127,
          "y2": 344
        }
      }
```


(continues on next page)

(продолжение с предыдущей страницы)

```
    ]
  }
}
```

4.4.7 Сравнение двух лиц

Данный запрос сравнивает лицо из локального изображения и лицо из сети Интернет и возвращает результат проверки лиц на идентичность. Соответствующий метод REST API: */verify POST*.

Запрос

```
curl -H "Authorization: Token t3WGNhZbyaE_GFyQaywYllFoR2QkHXi-" -F "photo1=@Pictures/sample.jpg" -
↪F "photo2=http://static.findface.pro/sample2.jpg" http://localhost:8000/v0/verify | jq
```

Ответ

```
{
  "results": [
    {
      "bbox1": {
        "x1": 595,
        "x2": 812,
        "y1": 127,
        "y2": 344
      },
      "bbox2": {
        "x1": 515,
        "x2": 821,
        "y1": 121,
        "y2": 427
      },
      "confidence": 0.9373794198036194,
      "verified": true
    }
  ],
  "verified": true
}
```

4.4.8 Получение списка лиц в галереях

Следующие запросы возвращают список лиц, хранящихся в галереях: в обеих (№1) и только в пользовательской (№2). Соответствующий метод REST API: */faces GET*.

Запрос №1

```
curl -H "Authorization: Token t3WGNhZbyaE_GFyQaywYllFoR2QkHXi-" http://localhost:8000/v0/faces | jq
```

Ответ


```
{
  "next_page": "/v0/faces?max_id=3827058103081960",
  "prev_page": null,
  "results": [
    {
      "friend": false,
      "galleries": [
        "default",
        "testgal"
      ],
      "id": 3827229578000564,
      "meta": "sample",
      "normalized": "http://192.168.113.88:3333/uploads//20170517/14950115538997407.jpeg",
      "person_id": 5,
      "photo": "http://192.168.113.88:3333/uploads//20170517/14950115538939695.jpeg",
      "photo_hash": "53477c4a72f52c6efc951d9c7ece42bc",
      "thumbnail": "http://192.168.113.88:3333/uploads//20170517/14950115538985784.jpeg",
      "timestamp": "2017-05-17T08:59:13.893000",
      "x1": 595,
      "x2": 812,
      "y1": 127,
      "y2": 344
    },
    {
      "friend": false,
      "galleries": [
        "default"
      ],
      "id": 3827229391220303,
      "meta": "Sam Berry",
      "normalized": "http://192.168.113.88:3333/uploads//20170517/1495011480937809.jpeg",
      "person_id": 5,
      "photo": "http://192.168.113.88:3333/uploads//20170517/14950114809306293.jpeg",
      "photo_hash": "53477c4a72f52c6efc951d9c7ece42bc",
      "thumbnail": "http://192.168.113.88:3333/uploads//20170517/149501148093593.jpeg",
      "timestamp": "2017-05-17T08:58:00.930000",
      "x1": 595,
      "x2": 812,
      "y1": 127,
      "y2": 344
    },
    {
      "age": 26.0483455657959,
      "emotions": [
        "neutral",
        "sad"
      ],
      "friend": false,
      "galleries": [
        "default"
      ],
      "gender": "female",
      "id": 3827227793957831,
      "meta": "Sam Berry",
      "normalized": "http://192.168.113.88:3333/uploads//20170517/14950108570078573.jpeg",
```


(continues on next page)

(продолжение с предыдущей страницы)


```
      "person_id": 5,
      "photo": "http://192.168.113.88:3333/uploads//20170517/14950108570022256.jpeg",
      "photo_hash": "53477c4a72f52c6efc951d9c7ece42bc",
      "thumbnail": "http://192.168.113.88:3333/uploads//20170517/14950108570066717.jpeg",
      "timestamp": "2017-05-17T08:47:37.002000",
      "x1": 595,
      "x2": 812,
      "y1": 127,
      "y2": 344
    }
  ]
}
```


Запрос №2

```
curl -H "Authorization: Token t3WGNhZbyaE_GFyQaywYllFoR2QkHXi-" http://localhost:8000/v0/faces/
↪gallery/testgal | jq
```

Ответ


```
{
  "next_page": "/v0/faces/gallery/testgal?max_id=3827059994026334",
  "prev_page": null,
  "results": [
    {
      "friend": false,
      "galleries": [
        "default",
        "testgal"
      ],
      "id": 3827229578000564,
      "meta": "sample",
      "normalized": "http://192.168.113.88:3333/uploads//20170517/14950115538997407.jpeg",
      "person_id": 5,
      "photo": "http://192.168.113.88:3333/uploads//20170517/14950115538939695.jpeg",
      "photo_hash": "53477c4a72f52c6efc951d9c7ece42bc",
      "thumbnail": "http://192.168.113.88:3333/uploads//20170517/14950115538985784.jpeg",
      "timestamp": "2017-05-17T08:59:13.893000",
      "x1": 595,
      "x2": 812,
      "y1": 127,
      "y2": 344
    },
    {
      "galleries": [
        "default",
        "testgal"
      ],
      "id": 3827059994026334,
      "meta": "sample",
      "normalized": "http://127.0.0.1:3333/uploads//20170516/14949453101653092.jpeg",
      "photo": "http://127.0.0.1:3333/uploads//20170516/14949453101581762.jpeg",
      "photo_hash": "53477c4a72f52c6efc951d9c7ece42bc",
```


(continues on next page)

(продолжение с предыдущей страницы)


```
      "thumbnail": "http://127.0.0.1:3333/uploads//20170516/14949453101640306.jpeg",
      "timestamp": "2017-05-16T14:35:10.158000",
      "x1": 595,
      "x2": 812,
      "y1": 127,
      "y2": 344
    }
  ]
}
```


4.4.9 Распознавание пола, возраста и эмоций

Данный запрос обнаруживает лицо на тестовом изображении, размещенном в сети Интернет, и возвращает координаты рамки вокруг лица вместе с информацией о поле, возрасте и эмоциях. Соответствующий метод REST API: */detect POST*.

Примечание: Распознавание пола, возраста и эмоций должно быть *настроено*.

Запрос

```
curl -H "Authorization: Token t3WGNhZbyaE_GFyQaywYllFoR2QkHXi-" -F 'photo=https://static.findface.
↪pro/sample2.jpg' -F 'gender=true' -F 'emotions=true' -F 'age=true' http://localhost:8000/v1/
↪detect | jq
```

Ответ


```
{
  "faces": [
    {
      "age": 29.057680130004883,
      "emotions": [
        "neutral",
        "happy"
      ],
      "gender": "female",
      "x1": 515,
      "x2": 821,
      "y1": 121,
      "y2": 427
    }
  ],
  "orientation": 1
}
```

Глава 5

Настройка обнаружения лиц на видео

5.1 О видеодетекторе лиц

Для распознавания лиц на видео вам потребуется компонент `fkvideo_detector`. Этот компонент обнаруживает лица «на лету» в видеопотоке или видеофайле и отправляет их на Сервер FindFace по API. Компонент поддерживает все форматы видеофайлов и кодеков, которые могут быть декодированы FFmpeg.

В этом разделе:

5.1.1 Установка

Установите `fkvideo_detector` из пакета с компонентами `<findface-repo>.deb` на одном из серверов FindFace или на отдельном сервере:

Совет: Нажмите здесь для перехода к инструкции по подготовке deb-пакета.

```
sudo apt-get update
sudo apt-get install fkvideo-detector
```

5.1.2 Принцип работы

Детектор движения и трекер лиц

При обработке видео `fkvideo_detector` последовательно использует следующие алгоритмы:

- `Детектор движения`. Данный алгоритм позволяет снизить потребление ресурсов, поскольку трекер лиц включается только по движению в кадре.
- `Трекер лиц`. Алгоритм детектирует, отслеживает и захватывает лица на видео и отправляет их на Сервер FindFace. Может работать одновременно с несколькими лицами в кадре.

  Совет: Настройте максимальное количество активных лиц в *файле конфигурации* `fkvideo_detector`.

  Трекер лиц отправляет на Сервер область изображения с обнаруженным лицом (рамку с лицом) через запрос `POST /face` или `POST /identify` (зависит от *настроек видеодетектора*). Если одновременно отслеживается несколько лиц, трекер отправит на Сервер такое же количество запросов с рамкой.

Поиск лучшего изображения лица

Перед тем как отправить лицо на Сервер FindFace, трекер лиц подбирает его лучшее изображение.

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

- в реальном времени,
- в буфере.

Режим реального времени

В режиме реального времени трекер лиц начинает отправлять на Сервер изображения лица сразу после его появления в поле зрения видеокамеры. В этом режиме трекер лиц подбирает лучшее изображение лица динамически:

1. Сначала оценивается качество изображения лица. Если оно превышает некое предустановленное пороговое значение, то лицо отправляется на Сервер.
2. Порог повышается после каждой отправки изображения лица на Сервер. Каждый раз, когда трекер лиц получает изображение того же лица лучшего качества, оно отправляется.
3. При исчезновении лица из поля зрения видеокамеры снова устанавливается пороговое значение по умолчанию.

Буферный режим

В буферном режиме трекер лиц использует меньший объем дискового пространства по сравнению с режимом реального времени, поскольку для каждого лица отправляет на Сервер только одно изображение, но наивысшего качества. При работе в буферном режиме трекер лиц сохраняет в буфере фрагмент видео с лицом, после чего выбирает из этого фрагмента лучшее изображение лица и отправляет его.

5.1.3 Настройка и использование

Параметры конфигурации видеодетектора задаются одним из следующих способов:

- непосредственно в интерфейсе командной строки как опции команды запуска видеодетектора;

```
fkvideo_detector [options]
```

- в файле конфигурации.

Предупреждение: По умолчанию в качестве файла конфигурации fkvideo_detector используется файл `/etc/fkvideo.ini`. Не редактируйте файл `/etc/fkvideo.ini`, особенно если fkvideo_detector и *веб-интерфейс FindFace* установлены на одном физическом сервере, т. к. веб-интерфейс тоже использует данный файл. Вместо этого скопируйте файл, отредактируйте копию и укажите ее в опции `-c` при запуске fkvideo_detector.

```
sudo cp /etc/fkvideo.ini /etc/fkvideo_example.ini

fkvideo_detector -c /etc/fkvideo_example.ini
```

См. полный список параметров конфигурации в *Параметры конфигурации*.

5.1.4 Управление видеопотоками

Вы можете задать видеопотоки для обработки одним из следующих способов:

- Единственный поток может быть задан непосредственно с помощью параметров `--camid` и `--source` при настройке `fkvideo_detector`.
- Список видеопотоков должен быть сначала зарегистрирован на Сервере FindFace. Для этого для каждого потока из списка сформируйте и отправьте на Сервер запрос */camera POST* с одинаковым пользовательским идентификатором `detector`. Данный идентификатор должен быть затем указан в параметре `--detector-name` при настройке `fkvideo_detector`. В этом случае `fkvideo_detector` запросит список потоков от Сервера FindFace на основании их `detector-name` и начнет по отдельности обрабатывать каждый из них. Он также будет периодически проверять список на наличие обновлений с интервалом, определяемым параметром `reload-timeout`.

5.2 Настройка и запуск видеодетектора

Для развертывания обнаружения лиц на видео выполните по порядку приведенные ниже инструкции.

Примечание: Компонент `fkvideo_detector` должен быть *установлен*.

В этом разделе:

5.2.1 Задание видеопотоков

Для того чтобы задать видеопотоки для обработки, выполните следующие действия:

1. Скопируйте файл конфигурации `/etc/fkvideo.ini`. Откройте копию для редактирования.

```
sudo cp /etc/fkvideo.ini /etc/fk_local_config.ini
sudo vi /etc/fk_local_config.ini
```

2. Если у вас только одна видеокамера, вы можете добавить ее через файл конфигурации.

```
[General]
; Host settings
api-host=127.0.0.1
; Put your token here
api-token=RczGgVEMizR1njHHQegNH_g9mwGl6-A1
api-port=8000

; Camera params
; If params doesn't set detector ask cameras list from server by key
; Key for receiving cameras list
;detector-name=detec1
; Camera ID
camid=local
; Stream path
; Example: rtsp:// - network stream; /dev/video0 - webcam; file@FPS:PATH - file with␣
↪configurable FPS
source=rtsp://admin:qwert1234@192.168.104.199:554/Streaming/Channels/1
; Maximum cameras
detectors-max=20

; Motion detector scale coefficient for best performance
scale=0.3

; In realtime mode detector posts many frames wih increasing quality
; Else it sends only best frame
realtime=1

; URL that will receive frames
request-url=/v1/face/
; You can add custom head and body params to HTML POST request
head=
body=mf_selector=all,meta=User Meta
;

; Address of ntls server
license-ntls-server=127.0.0.1:3133
```

Совет: Пример файла конфигурации см. здесь.

3. Если у вас несколько видеокамер, храните список видеокамер на Сервере FindFace. Для этого добавьте каждую видеокамеру с помощью запроса `POST v1/camera` в виртуальный детектор, соответствующий определенному списку камер. Например, для того чтобы добавить видеокамеру в детектор `detec1`, выполните следующие действия:

Запрос

```
curl -H 'Authorization: Token 1234567890qwertyuiop' -F "detector=detec1" -F "url=rtsp://
↪user:pass@192.168.1.1:554/Streaming/Channels/1" -F "meta=test" http://localhost:8000/v1/
↪camera
```

Ответ

```
{"detector": "detec1", "id": "0e663c00-b945-4676-bb0e-032c1dcf353a", "meta": "test", "url":
↪"rtsp:// user:pass@192.168.1.1:554/Streaming/Channels/1"}
```

Теперь отредактируйте файл конфигурации. Запущенный с приведенным ниже файлом конфигурации видеодетектор лиц подключится к Серверу и запросит список камер из виртуального детектора `detec1`.

```
[General]
; Host settings
api-host=127.0.0.1
; Put your token here
api-token=RczGgVEMizR1njHHQegNH_g9mwGl6-A1
api-port=8000

; Camera params
; If params doesn't set detector ask cameras list from server by key
; Key for receiving cameras list
detector-name=detec1
; Camera ID
;camid=
; Stream path
; Example: rtsp:// - network stream; /dev/video0 - webcam; file@FPS:PATH - file with␣
↪configurable FPS
;source=
; Maximum cameras
detectors-max=20

; Motion detector scale coefficient for best performance
scale=0.3

; In realtime mode detector posts many frames wih increasing quality
; Else it sends only best frame
realtime=1

; URL that will receive frames
request-url=/v1/face/
; You can add custom head and body params to HTML POST request
```

(continues on next page)

(продолжение с предыдущей страницы)

```
head=
body=mf_selector=all,,meta=UserMeta
;

; Address of ntls server
license-ntls-server=127.0.0.1:3133
```

Совет: Пример файла конфигурации см. здесь.

5.2.2 Запуск компонента как приложения

Вы можете запустить видеодетектор лиц как приложение, используя следующую команду:

```
fkvideo_detector -c /etc/fk_local_config.ini
```

Используйте данный метод для тестирования компонента.

5.2.3 Запуск компонента как сервиса

Для запуска видеодетектора лиц как сервиса выполните следующие действия:

1. Выполните команду:

   ```
   sudo service fkvideo_detector@fk_local_config start
   ```

2. Убедитесь, что сервис активен. Команда вернет описание сервиса, его статус (должен быть Активен), путь и длительность текущей сессии.

   ```
   sudo service fkvideo_detector@fk_local_config status
   ```

   Примечание: Вы можете отобразить список всех камер, отправив запрос:

   ```
   curl -H 'Authorization: Token 1234567890qwertyuiop' http://localhost:8000/v1/camera | jq
   ```

5.3 Параметры конфигурации

Параметры конфигурации видеодетектора задаются одним из следующих способов:

- непосредственно в интерфейсе командной строки как опции команды запуска видеодетектора;

  ```
  fkvideo_detector [options]
  ```

- в файле конфигурации.

Предупреждение: По умолчанию в качестве файла конфигурации fkvideo_detector используется файл `/etc/fkvideo.ini`. Не редактируйте файл `/etc/fkvideo.ini`, особенно если fkvideo_detector и *веб-интерфейс FindFace* установлены на одном физическом сервере, т. к. веб-интерфейс тоже использует данный файл. Вместо этого скопируйте файл, отредактируйте копию и укажите ее в опции `-c` при запуске fkvideo_detector.

```
sudo cp /etc/fkvideo.ini /etc/fkvideo_example.ini

fkvideo_detector -c /etc/fkvideo_example.ini
```

В этом разделе:

5.3.1 Аргументы в командной строке

Использование:

```
fkvideo_detector [options]
```

Опции:

Предупреждение: Опции `api-host`, `api-port`, `api-token`, `--license-ntls-server` являются обязательными.

Опция	Описание	Аргумент	Пример
-c [--config] arg	Запускает видеодетектор лиц с заданным файлом конфигурации `.ini`. Если параметр задан как в командной строке, так и в файле конфигурации, значение в командной строке будет иметь приоритет.	Путь к файлу конфигурации `.ini`. Если имя файла указывается без полного пути, видеодетектор лиц ищет файл в папке своей установки. По умолчанию файл конфигурации fkvideo_detector `/etc/fkvideo.ini`. Если fkvideo_detector и *веб-интерфейс FindFace* установлены на одном физическом сервере, не редактируйте файл `/etc/fkvideo.ini`, поскольку он также используется веб-интерфейсом FindFace. Вместо этого скопируйте данный файл, отредактируйте копию и укажите ее в опции `-c` при запуске fkvideo_detector.	$ fkvideo_detector -c /etc/fkvideo_example.ini
--license-ntls-server arg	Обязательная опция. Определяет IP-адрес и порт локального сервера лицензий NTLS. Редактируется только в случае удаленного NTLS.	IP-адрес:порт	--license-ntls-server 192.168.10.1:3133
-n [--detector-name] arg	Применяет видеодетектор к заданному списку видеокамер.	Уникальный идентификатор виртуального детектора (по умолчанию имя локального сервера), соответствующего определенному списку видеокамер на Сервере FindFace.	--detector-name detec1
-d [--detectors-max] arg	Определяет максимальное количество видеопотоков, обрабатываемых видеодетектором лиц.	Максимальное количество видеопотоков, одновременно обрабатываемых видеодетектором лиц (по умолчанию 5).	--detectors-max 7
-t [--reload-timeout] arg	Определяет интервал в секундах между 2-мя последовательными запросами, отправляемыми видеодетектором лиц на Сервер для обновления списка видеокамер.	Интервал в секундах между 2-мя последовательными обновлениями списка видеокамер (по умолчанию 15 с).	-t 20

Продолжается на следующей странице

Таблица 1 – продолжение с предыдущей страницы

Опция	Описание	Аргумент	Пример
--camid arg	Задает видеопоток для отслеживания как ID видеокамеры (см. также `--source`). Если видеопоток не задан, видеодетектор лиц запрашивает *список видеокамер* у Сервера FindFace с интервалом `reload-timeout`.	ID видеокамеры.	--camid b28a898b-6334
--api-host arg	Обязательная опция. Определяет физический сервер `findface-facenapi` в составе Сервера FindFace, на который видеодетектор лиц отправляет API-запросы.	IP-адрес сервера FindFace.	--api-host 127.0.0.1
--api-port arg	Обязательная опция. Определяет порт физического сервера `findface-facenapi` для API-запросов.	Номер порта.	--api-port 8000
--api-token arg	Обязательная опция. Определяет токен авторизации для Сервера FindFace.	*Токен авторизация*.	--api-token c9FsRNDAt
-S [--source] arg	Задает видеопоток для отслеживания как адрес видеокамеры (см. также `--camid`). Если видеопоток не задан, видеодетектор лиц запрашивает *список видеокамер* у Сервера FindFace с интервалом `reload-timeout`.	Адрес видеокамеры: rtsp://... — сетевой видеопоток; /dev/video0 — вебкамера; file@FPS:PATH - видеофайл с настраиваемой частотой кадров в секунду (FPS).	--source rtsp://192.168.120.55:500
--source-params arg	Задает опции ffmeg для видеопотока.	Список опций ffmeg с присвоенными значениями.	--source-params rtsp_transport=tcp, rtsp_flags=prefer, timeout=-1
--md-threshold arg	Определяет минимальную интенсивность движения, которая будет регистрироваться детектором движения. Пороговое значение определяется эмпирически.	Интенсивность движения в эмпирических единицах (ноль и положительные рациональные числа). Реперные точки: 0 = детектор выключен, 0.002 = значение по умолчанию, 0.05 = минимальная интенсивность слишком высока, чтобы зарегистрировать движение.	--md-threshold 0.003

Продолжается на следующей странице

Таблица 1 – продолжение с предыдущей страницы

Опция	Описание	Аргумент	Пример
--scale arg	Изменяет размер кадра для детектора движения. 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`.	Коэффициент изменения размера видеокадра.	--scale 0.3
--request-url arg	Определяет метод API-запросов, которые отправляюся видеодетектором лиц на Сервер FindFace при обнаружении лица.	/v0/face/ или /v0/identify/.	--request-url /v0/identify
--camera-url arg	Определяет метод API-запросов, которые отправляются видеодетектором лиц на Сервер FindFace для получения списка камер.	/v0/camera (по умолчанию) или /v1/camera.	--camera-url /v1/camera
--img-arg arg	Определяет имя аргумента с изображением лица, отправляемым в API-запросе.	Имя аргумента (по умолчанию photo).	--img-arg picture
--req-timeout arg	Определяет время ожидания ответа от Сервера FindFace на API-запрос видеодетектора лиц.	Время ожидания API-ответа в секундах (по умолчанию 3 с).	--req-timeout 2
--headers arg	Создает дополнительный заголовок в POST-запросе с изображением лица.	Дополнительный заголовок (заголовки) в POST-запросе.	--headers xxx = yyy --headers kkk = ppp
--body arg	Создает дополнительное поле в POST-запросе с изображением лица.	Дополнительное поле (поля).	--body galleries=testgal --body gender=true --body age=true --body emotions=true --body meta=video.mp4
--bbox-scale	Изменяет размер изображения лица.	Коэффициент изменения размера лица (по умолчанию 1).	--bbox-scale 1.3

Продолжается на следующей странице

Таблица 1 – продолжение с предыдущей страницы

Опция	Описание	Аргумент	Пример
--post-uniq arg	Включает отправку на Сервер только нескольких лиц, принадлежащих одному человеку, из множества захваченных в течение определенного промежутка времени. В этом случае, если видеодетектор лиц отправляет лицо на Сервер 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`, видеодетектор лиц 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`, видеодетектор отправляет на Сервер второе лицо. Иначе, второе лицо не отправляется.	Логический: 1 = только определенное количество лиц из множества принадлежащих одному человеку отправляется на Сервер, 0 = все захваченные лица отправляются на Сервер.	--post-uniq 1
--uc-max-time-diff arg	Только для `--post-uniq=1`. Определяет максимальный период времени, в течение которого схожие лица рассматриваются как лица одного человека.	Максимальный период времени в секундах.	--uc-max-time-diff 1
--uc-max-dup arg	Только для `--post-uniq=1`. Определяет максимальное количество лиц в течение периода времени `uc-max-time-diff`, которое отправляется на Сервер для одного человека.	Максимальное количество лиц.	--uc-max-dup 3
--uc-max-avg-shift arg	Только для `--post-uniq=1`. Определяет максимальное расстояние, на котором схожие лица еще рассматриваются как лица одного человека.	Расстояние в пикселях.	--uc-max-avg-shift 10
-r [--realtime] [=arg(=1)]	Включает режим *реального времени* видеодетектора.	Режим работы fkvideo_detector: 1 = реального времени, 0 = буферный режим. Записи -r и -r 1 идентичны.	-r или -r 1, -r 0

Продолжается на следующей странице

Таблица 1 – продолжение с предыдущей страницы

Опция	Описание	Аргумент	Пример
--min-score arg	Определяет пороговое значение качества изображения лица. Лицо отправляется на Сервер, если его изображение лучшего качества. Пороговое значение определяется эмпирически.	Пороговое значение качества изображения лица в эмпирических единицах (отрицательные рациональные числа до нуля). Реперные точки: 0 = высокое качество изображения, -1 = хорошее качество, -2 = удовлетворительное качество, -5 = распознавание лица может быть неэффективным. Значение по умолчанию -7.	--min-score -1.5
--min-dir-score arg	Определяет максимальное отклонение лица от положения анфас. Лицо отправляется на Сервер FindFace, если отклонение не превышает заданного значения (определяется эмпирически).	Максимальное отклонение лица от положения анфас в эмпирических единицах (отрицательные рациональные числа до нуля). Реперные точки: -3.5 = допускаются лица, повернутые под большим углом, распознавание может быть неэффективно, -2.5 = удовлетворительное качество, -0.05 = близко к фронтальной позиции, 0 = анфас. Значение по умолчанию -1000.	--min-dir-score -1
--rt-refresh arg	Только для режима реального времени. Определяет временной интервал между 2-мя последовательными обнулениями счетчика при динамическом поиске лучшего изображения лица.	Временной интервал в миллисекундах. Значение по умолчанию 0 (обнуление отключено).	--rt-refresh 10
--rt-score-step arg	Только для режима реального времени. Определяет шаг увеличения порогового значения качества при динамическом поиске лучшего изображения лица.	Шаг увеличения порогового изображения (положительные рациональные числа).	--rt-score-step 3.4
--rt-delay arg	Только для режима реального времени. Определяет максимальный период времени между двумя последовательными отправками одного и того же лица, но в улучшенном качестве.	Период времени в миллисекундах между отправками одного и того же лица.	--rt-delay 100

Продолжается на следующей странице

Таблица 1 – продолжение с предыдущей страницы

Опция	Описание	Аргумент	Пример
--rot arg	Включает детектирование и отслеживание лиц только внутри заданной прямоугольной области. Используйте данную опцию, чтобы уменьшить нагрузку на fkvideo_detector.	Прямоугольник: WxH+X+Y	--rot 150x123+300+155
--roi arg	Включает отправку на Сервер FindFace лиц, обнаруженных только внутри интересующей области.	Интересующая область: WxH+X+Y.	--roi 123x122+159+220
--draw-track [=arg(=1)]	Включает рисование в bbox следа от движения лица.	Логический: 1 = след рисуется, 0 = рисование следа отключено. Записи –draw-track и –draw-track 1 тождественны.	--draw-track
--min-face-size arg	Определяет минимальный размер лица. Лица меньшего размера на Сервер не отправляются.	Минимальный размер меньшей стороны прямоугольника с лицом, в пикселях.	--min-face-size 50
--max-face-size arg	Определяет максимальный размер лица. Лица большего размера на Сервер не отправляются.	Максимальный размер большей стороны прямоугольника с лицом, в пикселях.	--max-face-size 120
--max-persons arg	Определяет максимальное количество лиц, одновременно отслеживаемых трекером лиц. Данный параметр существенно влияет на производительность.	Максимальное количество одновременно отслеживаемых лиц.	--max-persons 4
--single-pass [=arg(=1)]	Отключает периодические обновления списка видеокамер. Используйте эту опцию, если нужно обработать видеофайл. В этом случае видеодетектор лиц запросит список камер только один раз.	Логический: 1 = обновления отключены, 0 = обновления включены. Записи – single-pass и –single-pass 1 тождественны.	--single-pass 0
--start-ts arg	Добавляет время обнаружения лица в API-запрос к Серверу FindFace.	Логический: 1 = метка времени добавляется, 0 = добавление метки времени отключено.	--start-ts 1
--disable-drops [=arg(=1)]	Включает отправку на Сервер FindFace всех подходящих лиц без пропусков. По умолчанию, если fkvideo_detector не обладает достаточными ресурсами для обработки всех кадров с лицами, он отбрасывает некоторые из них. Если данная опция активна, fkvideo_detector помещает лишние кадры в очередь, чтобы обработать их впоследствии.	Логический: 1 = лишние кадры не отбрасываются, 0 = лишние кадры отбрасываются. Записи –disable-drops и –disable-drops 1 тождественны.	--disable-drops

Продолжается на следующей странице

Таблица 1 – продолжение с предыдущей страницы

Опция	Описание	Аргумент	Пример
--sink-url arg	Только в случае если fkvideo_detector обрабатывает одну видеокамеру (указанную в файле конфигурации или в командной строке). Определяет IP-адрес видео-сервера nginx для выходного видеопотока из fkvideo_detector (видео-сервер далее перенаправляет поток в *веб-интерфейс FindFace*).	IP-адрес и порт видео-сервера nginx.	--sink-url 192.168.15.1:3222
--sink-res arg	Определяет разрешение выходного видеопотока.	Разрешение WxH	--sink-res 1280x720
--tracker-threads arg	Определяет количество тредов отслеживания для трекера лиц. Данное значение должно быть меньше или равно значению параметра `max-persons`. Оптимально, когда они равны. Если количество тредов отслеживания меньше, чем максимальное количество отслеживаемых лиц, потребление ресурсов уменьшается, однако также уменьшается и скорость отслеживания.	Количество тредов отслеживания.	--tracker-threads 4
-h [--help]	Выводит справку по опциям видеодетектора лиц.	–	–

5.3.2 Формат файла конфигурации

```
[General]
| long-arg=option ; long-arg from command line arguments
| ...

| license-ntls-server=192.168.10.1:3133
| source-params=rtsp_transport=tcp,rtsp_flags=prefer,timeout=-1
| body=galleries=testgal,gender=true,age=true,emotions=true,meta=video.mp4
```

5.4 Визуализация ответов Сервера

Видеодетектор лиц не обрабатывает ответы Сервера на свои API запросы, связанные с идентификацией лиц и управлением видеокамерами. При необходимости вам потребуется написать собственный прокси-скрипт, управляющий взаимодействием видеодетектора лиц и Сервера и перенаправляющий API-ответы в приложение, которое может их обработать и визуализировать. Пример использования прокси-скрипта между Сервером и видеодетектором показан на схеме ниже:

При написании прокси-скрипта придерживайтесь следующей логики:

1. Запрос от видеодетектора передается на Сервер FindFace без изменений в следующем формате:

```
curl -X POST -H 'Authorization: Token ntech' -F "gender=true" -F "emotions=true" -F "age=true
↪" -F "cam_id=1b19a189-26b9-42e5-8cd8-6cabde79dc7e" -F "timestamp=2017-08-25T13:09:54" -F
↪"bbox=[[620,380,1383,1143]]" -F "photo=@15036665986531599.jpeg" -F
↪"face0=@15036665986766284_norm.png" -F 'detectorParams={"score": -0.000911839, "direction_
↪score": -0.568228}' http://192.168.104.184:8000/v1/face
```

2. Ответ от Сервера, отправленный видеодетектору лиц, перенаправляется в пользовательское приложение для обработки.

Примечание: Сервер отвечает на запросы одинаково, независимо от того, были ли они отправлены напрямую или пришли от видеодетектора лиц. В связи с этим ответы Сервера на запросы видеодетектора лиц могут содержать ссылку на миниатюру лица и другие данные, которые можно проанализировать в пользовательском приложении.

Глава 6

Увеличение производительности

6.1 Балансировка нагрузки с помощью NginX

Для увеличения производительности и уменьшения времени обработки запросов в высоконагруженных системах с повышенными требованиями к оптимизации ресурсов рекомендуется настроить балансировку нагрузки с помощью nginx.

Если нагрузка балансируется, входящий поток запросов вместо того, чтобы направиться на обработку в единственный экземпляр компонента, проходит через прокси-сервер и распределяется между несколькими экземплярами этого компонента в режиме round-robin (циклически). Результатом является значительное уменьшение времени ожидания обработки запроса, а также улучшение общей производительности, масштабируемости и надежности системы.

Нагрузка может быть сбалансирована для следующих компонентов:

Компонент	Рекомендуемое количество экземпляров на одном физическом сервере
`findface-facenapi`	Один экземпляра на сервер обычно бывает достаточно. Применительно к компоненту `findface-facenapi` нагрузка обычно балансируется только в кластерной среде между несколькими физическими серверами `findface-facenapi`.
`findface-nnapi`	Количество ядер процессора минус 1. Дает значительный выигрыш в производительности.
`extraction-api`	1 (нагрузка балансируется автоматически). Балансировка нагрузки через nginx применяется только к экземплярам `extraction-api`, установленным на различных физических серверах.

Приведенная ниже пошаговая инструкция демонстрирует настройку балансировки нагрузки через nginx для 2-х экземпляров `findface-nnapi` на одном физическом сервере. Нагрузка остальных компонентов может быть сбалансирована по аналогии.

Выполните следующие действия:

1. При необходимости установите nginx на серверы с `findface-nnapi` (nginx устанавливается автоматически вместе с компонентом `findface-upload`).

```
sudo apt-get install nginx
```

2. Скопируйте содержимое файла `/lib/systemd/system/findface-nnapi.service` в новый файл `/etc/systemd/system/findface-nnapi@.service`.

```
sudo cp /lib/systemd/system/findface-nnapi.service /etc/systemd/system/findface-nnapi@.service
```

3. Остановите все сервисы `findface-nnapi` и удалите их из автозагрузки. Отредактируйте новый файл `findface-nnapi@.service`, добавив в конец строки `ExecStart` опцию `--listen 127.0.0.1:%i`.

```
sudo service findface-nnapi stop && sudo systemctl disable findface-nnapi

sudo vi /etc/systemd/system/findface-nnapi@.service

ExecStart=/usr/bin/findface-nnapi -c /etc/findface-nnapi.ini --listen 127.0.0.1:%i
```

4. Создайте новый файл конфигурации nginx.

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

5. Вставьте следующий текст в файл конфигурации. В тексте замените номера портов, предложенные для экземпляров `findface-nnapi` (`upstream nnapibackends`), и номер слушающего порта (`listen`) на актуальные значения. Номера портов должны быть уникальны для каждого компонента на сервере.

```
upstream nnapibackends {
        server 127.0.0.1:18090;
        server 127.0.0.1:18091;
}
server {
        listen 18088;
        server_name nnapi;
        client_max_body_size 64m;
        location / {
                proxy_pass http://nnapibackends;
                proxy_next_upstream error;
        }
        access_log /var/log/nginx/nnapi.access_log;
        error_log /var/log/nginx/nnapi.error_log;
}
```

6. Включите балансировщик нагрузки в nginx.

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

7. Перезапустите nginx.

```
sudo service nginx restart
```

8. Добавьте каждый экземпляр `findface-nnapi` в автозагрузку.

```
sudo systemctl enable findface-nnapi@18090
sudo systemctl enable findface-nnapi@18091
```

9. Запустите экземпляры `findface-nnapi`.

```
sudo systemctl start findface-nnapi@18090
sudo systemctl start findface-nnapi@18091
```

10. Теперь запросы, отправленные в `findface-nnapi`, будут распределяться между двумя экземплярами `findface-nnapi` в режиме round-robin. Распределение запросов наглядно отображается в логе `findface-nnapi` `/var/log/syslog` (обратите внимание на различные значения id процессов).

```
sudo tail -f /var/log/syslog | grep nnapi
Jul  7 03:53:05 ubuntu findface-nnapi[49606]: (2017-07-07 10:53:05) [INFO    ] Request: 127.0.
↪0.1:34494 0x7fb100000960 HTTP/1.0 POST /facen
Jul  7 03:53:06 ubuntu findface-nnapi[49606]: (2017-07-07 10:53:06) [INFO    ] Response:␣
↪0x7fb100000960 /facen?x2=0&y1=0&x1=0&y2=0 200 0
Jul  7 03:53:06 ubuntu findface-nnapi[49624]: (2017-07-07 10:53:06) [INFO    ] Request: 127.0.
↪0.1:52960 0x7f9cf8000960 HTTP/1.0 POST /facen
Jul  7 03:53:06 ubuntu findface-nnapi[49624]: (2017-07-07 10:53:06) [INFO    ] Response:␣
↪0x7f9cf8000960 /facen?x2=0&y1=0&x1=0&y2=0 200 0
Jul  7 03:53:32 ubuntu findface-nnapi[49606]: (2017-07-07 10:53:32) [INFO    ] Request: 127.0.
↪0.1:34502 0x7fb100001ec0 HTTP/1.0 POST /facen
Jul  7 03:53:32 ubuntu findface-nnapi[49606]: (2017-07-07 10:53:32) [INFO    ] Response:␣
↪0x7fb100001ec0 /facen?x2=0&y1=0&x1=0&y2=0 200 0
Jul  7 03:53:32 ubuntu findface-nnapi[49624]: (2017-07-07 10:53:32) [INFO    ] Request: 127.0.
↪0.1:52968 0x7f9cf8001ec0 HTTP/1.0 POST /facen
Jul  7 03:53:33 ubuntu findface-nnapi[49624]: (2017-07-07 10:53:33) [INFO    ] Response:␣
↪0x7f9cf8001ec0 /facen?x2=0&y1=0&x1=0&y2=0 200 0
```

Совет: Вы можете использовать данный метод для балансировки нагрузки между экземплярами компонента на различных физических серверах.

6.2 Индексирование для быстрого поиска по базе данных

Для ускорения поиска галереи с количеством лиц более `1 000 000` должны быть проиндексированы. Для подготовки быстрого индекса вам понадобится утилита `findface-tarantool-build-index` из вашего дистрибутивного пакета. Для работы данной утилиты не требуется компонент `tntapi`, поэтому она может быть установлена как на сервере базы данных `Tarantool`, так и на удаленном сервере с доступом к базе данных.

Для подготовки быстрого индекса выполните следующие действия:

1. Установите утилиту `findface-tarantool-build-index`.

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

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

Примечание: Создание быстрого индекса производится на каждом шарде `tntapi`.

```
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-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 --facen_size 320 --out /opt/ntech/var/lib/
↪tarantool/default/index/testgal.idx

0% 10 20 30 40 50 60 70 80 90 100%
|----|----|----|----|----|----|----|----|----|----|
**************************************************
0% 10 20 30 40 50 60 70 80 90 100%
|----|----|----|----|----|----|----|----|----|----|
**************************************************

[Benchmark] create_index took 29.994ms
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 (в 70-100 раз). Информация об индексе остается в служебном пространстве Tarantool, поэтому когда вы перезапускаете Tarantool, индекс также подгружается.

Предупреждение: Не перемещайте файл индекса!

Глава 7

Веб-интерфейс FindFace

Веб-интерфейс FindFace Enterprise Server SDK обеспечивает удобный доступ к большинству функций REST API без необходимости написания кода.

Для того чтобы установить веб-интерфейс, выполните на сервере `findface-facenapi` следующие команды:

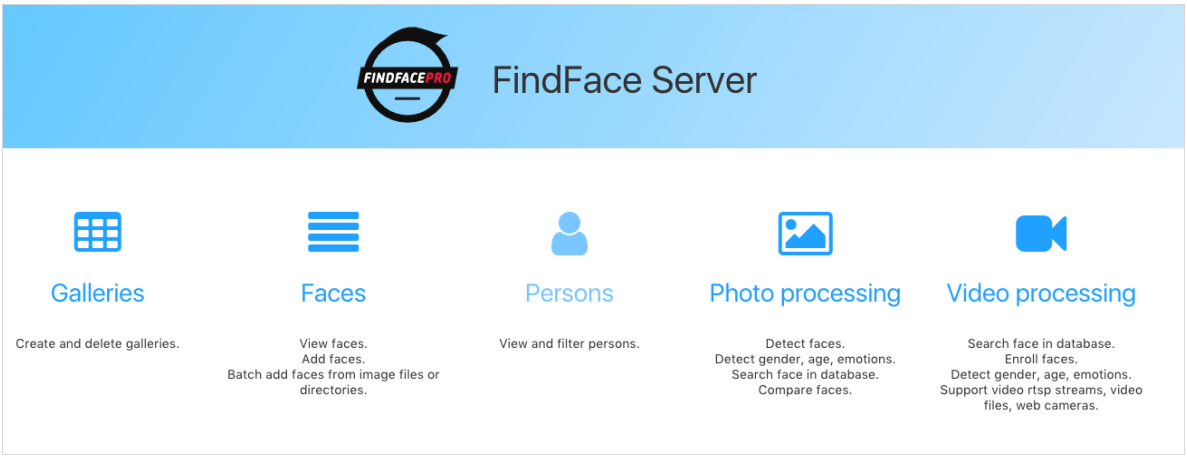
Примечание: Для работы веб-интерфейса вам понадобится nginx. Если он не установлен, это можно сделать следующим образом:

```
sudo apt-get install nginx
```

```
sudo apt-get install findface-ui
```

Для того чтобы отобразить веб-интерфейс, выполните следующие действия:

1. В адресной строке браузера введите `http://<facenapi_ip>:8000/#/`.
2. Для того чтобы войти в веб-интерфейс, укажите *токен авторизации* своего экземпляра FindFace Enterprise Server SDK. В результате отобразится домашняя страница веб-интерфейса.



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

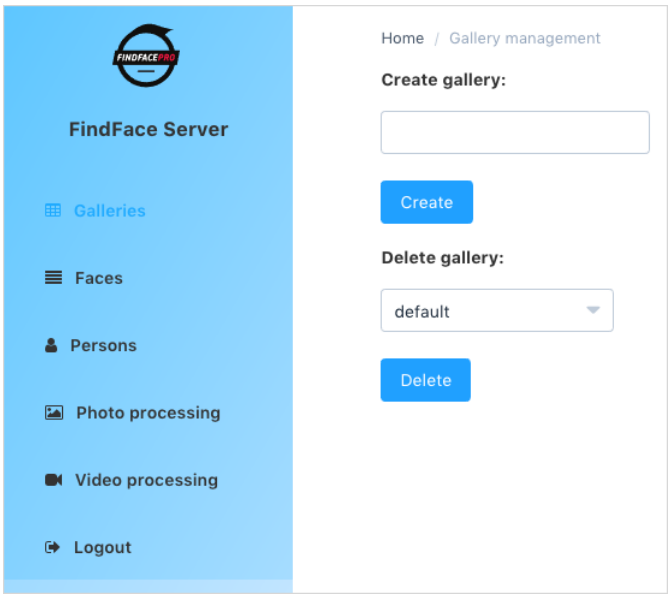
Примечание: Для работы с распознаванием пола, возраста и эмоций (GAE) в веб-интерфейсе данную функцию нужно *настроить*.

Примечание: Для работы с фотографиями необходим настроенный компонент *findface-upload*.

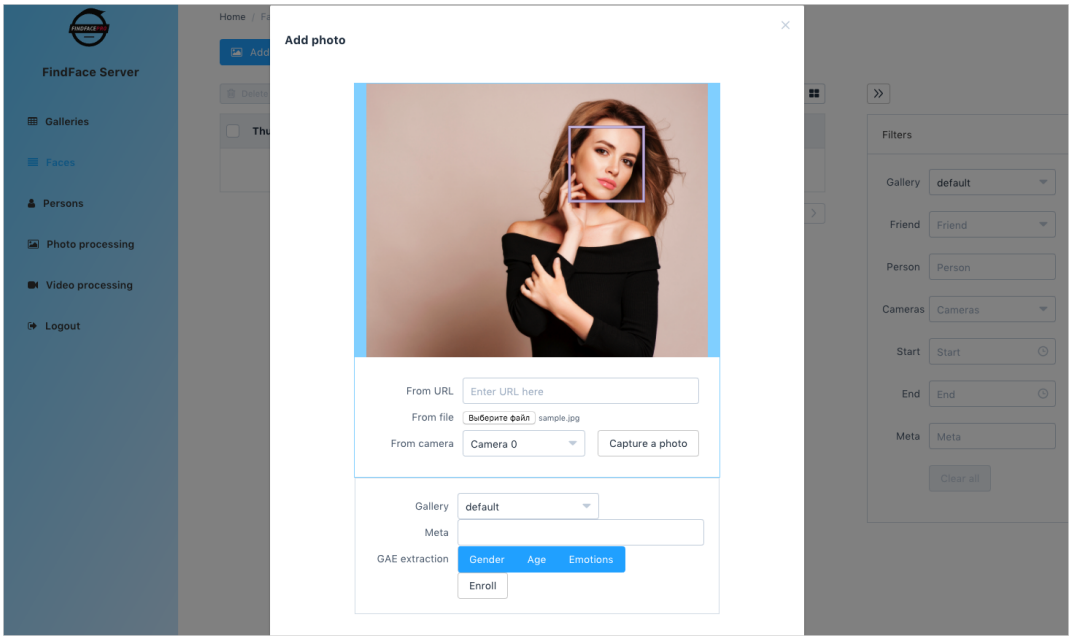
Примечание: Для работы с персонами необходимо настроить *группировку лиц персоны*.

Примечание: Для того чтобы разрешить веб-интерфейсу проигрывать Flash в браузере `Chrome`, добавьте IP-адрес веб-интерфейса в список разрешенных веб-сайтов: *Настройки* → *Дополнительно* → *Настройки контента* → *Flash* → *Разрешить* → *Добавить сайт* `http://<facenapi_ip>:8000/#/`. Перезапустите `Chrome`.

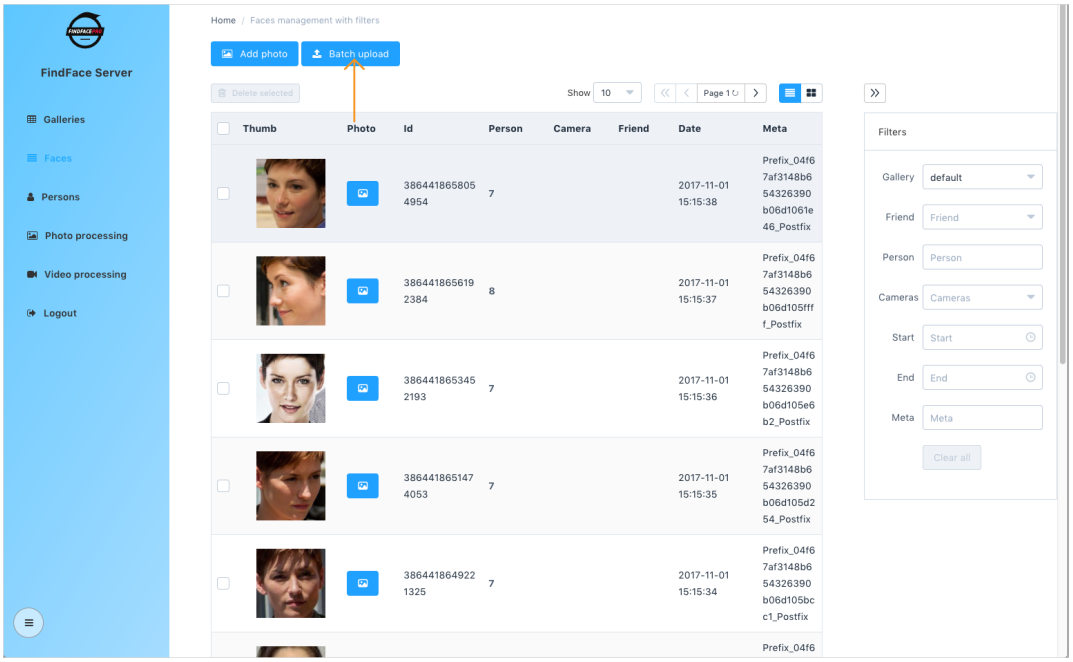
- *Galleries* (Галереи). В данном разделе можно добавлять и удалять галереи.



- *Faces* (Лица). В данном разделе можно просматривать, добавлять и удалять лица из галерей.

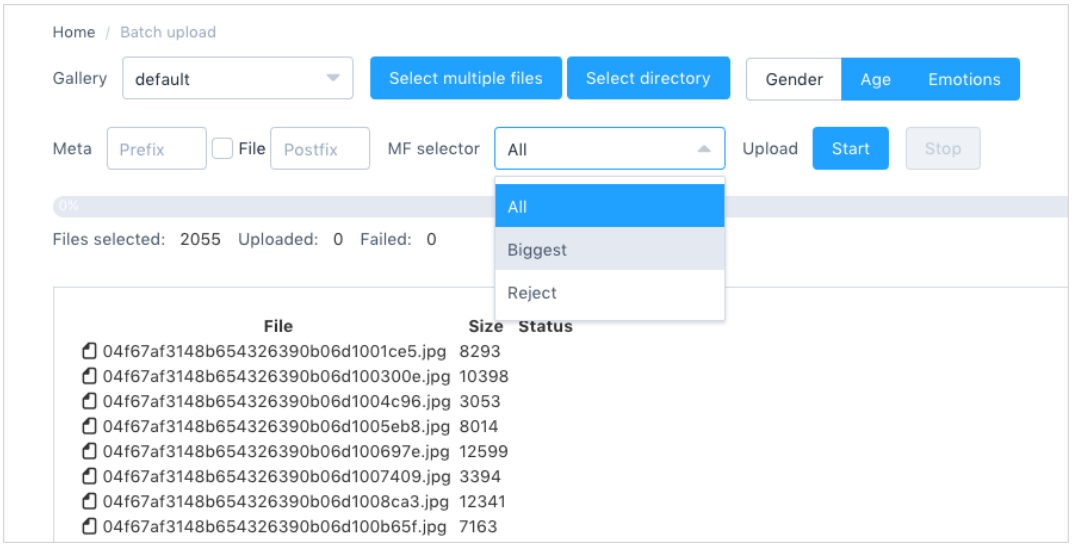


Используйте опцию *Batch upload*, чтобы добавить лица пакетно.



Совет: Возможно, вы также захотите воспользоваться ее *консольной альтернативой*.

Выберите несколько изображений по отдельности или укажите папку с изображениями. Затем настройте правило формирования метаданных для загружаемых лиц. Используйте опцию *MF selector*, чтобы задать поведение системы на случай обнаружения нескольких лиц на фото: добавить все лица, самое большое, вернуть ошибку.

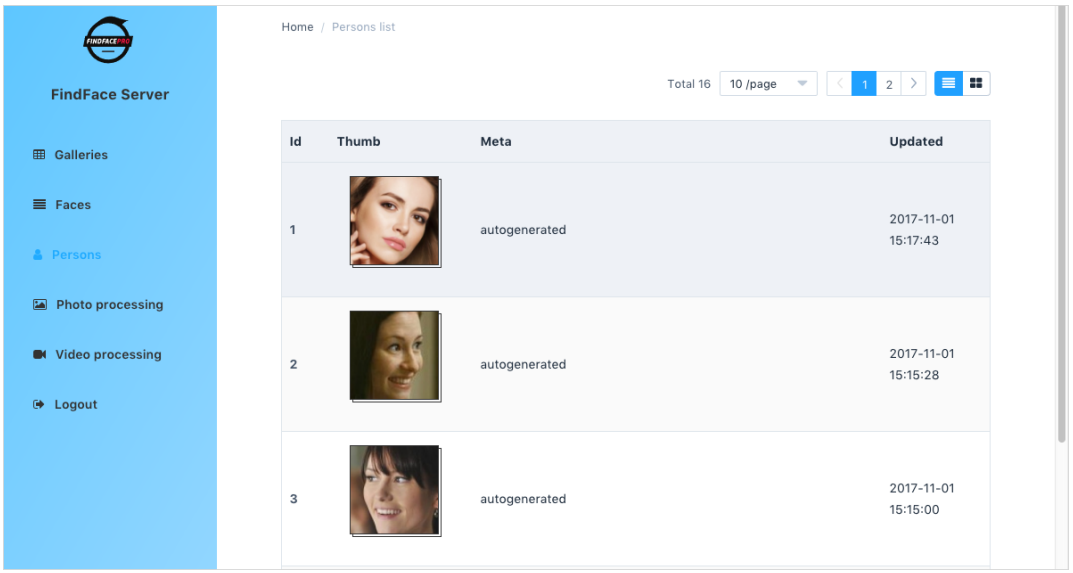


Совет: Вы можете настроить правило формирования метаданных, добавив пользовательский префикс и/или постфикс к имени файла изображения. Во избежание слияние 3-х слов в одно, используйте символ подчеркивания в префиксе и постфиксе.

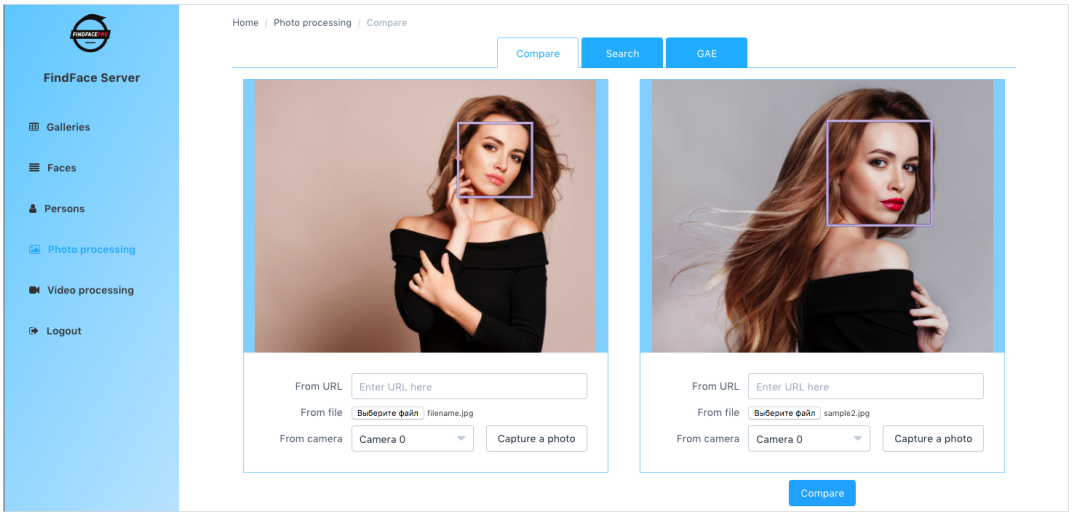
Совет: Для того чтобы выбрать фотографии в режиме *icons*, щелкайте по ним при нажатой

клавише CTRL.

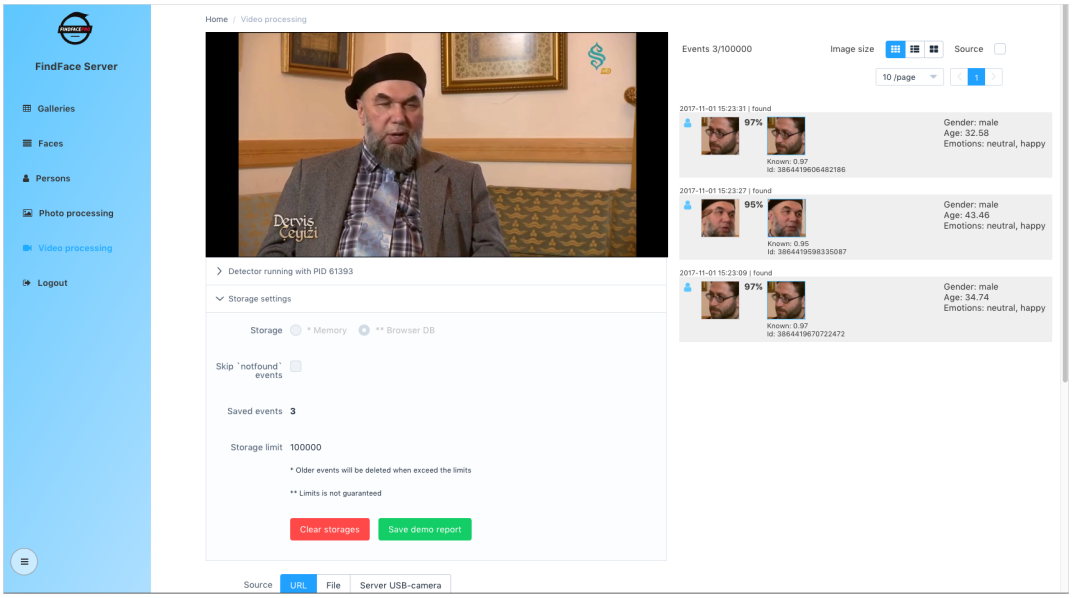
- *Persons* (Персоны). Здесь можно просматривать и сортировать персоны.

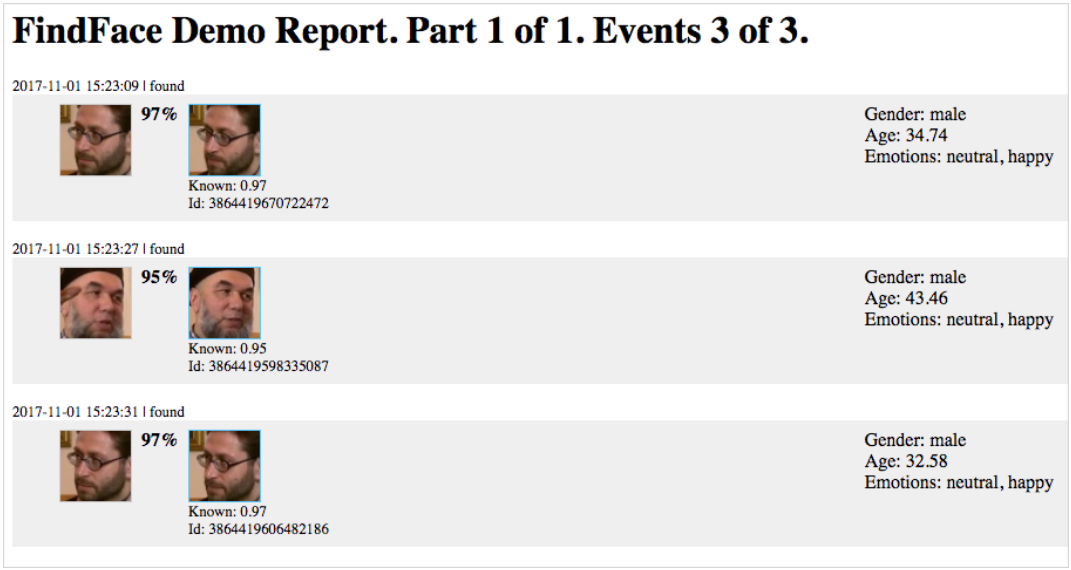


- *Photo processing* (Обработка фото). Выберите данный раздел для обнаружения лиц на статических изображениях, распознавания пола, возраста и эмоций, поиска лица в базе данных (идентификации) и сравнения двух лиц (верификации).



- *Video processing* (Обработка видео). В данном разделе вы можете обработать видеопотоки с rtsp- и web-камер, а также видеофайлы. Возможности: обнаружение лиц на видео и добавление их в базу данных, идентификация и распознавание пола, возраста и эмоций. Для генерации отчета о добавленных или идентифицированных лицах в формате HTML нажмите кнопку *Save demo report*.





Примечание: Раздел обработки видео в веб-интерфейсе предназначен для тестирования Сервера. В рабочем режиме используйте компонент *fkvideo_ detector*.

Глава 8

Расширенный функционал

8.1 Распознавание пола, возраста и эмоций

В этом разделе:

8.1.1 Настройка распознавания пола, возраста и эмоций

Примечание: Для распознавания пола, возраста и эмоций необходимы 2 ГБ оперативной памяти сверх *общих требований* к Серверу FindFace.

Для настройки распознавания пола, возраста и эмоций выполните следующие действия:

1. Включите распознавание пола, возраста и эмоций, раскомментировав и отредактировав строку `gae = False` в файле конфигурации `findface-facenapi`. Перезапустите сервис `findface-facenapi`.

   Предупреждение: Содержимое файла `findface-facenapi.ini` должно представлять собой синтаксически верный код Python.

```
sudo vi /etc/findface-facenapi.ini

      → gae = True
```

(continues on next page)

(продолжение с предыдущей страницы)

```
sudo service findface-facenapi restart
```

2. Включите соответствующие *модели* распознавания, раскомментировав строки `model_*` в файле конфигурации `findface-nnapi`. Перезапустите сервис `findface-nnapi`.

```
sudo vi /etc/findface-nnapi.ini

      → model_emotions = emotion_1
      → model_age = fr_1_age0
      → model_gender = fr_1_gender0

sudo service findface-nnapi restart
```

8.1.2 API для распознавания пола, возраста и эмоций

Данный запрос обнаруживает лицо на изображении и возвращает координаты рамки вокруг лица вместе с информацией о поле, возрасте и эмоциях.

Запрос №1

```
POST /v1/detect/ HTTP/1.1
Host: 192.168.113.76:8000
Connection:close
Authorization: Token BpdNA6eaUlN9bPhXVSK1r92_SFOODPOU
Content-Type:   application/json
Content-Length: 108

{
    "photo": "https://static.findface.pro/sample.jpg",
    "emotions": true,
    "gender": true,
    "age": true
}
```

Ответ

```
HTTP/1.1 200 OK
Date: Thu, 06 Apr 2017 12:38:40 GMT
Server: TornadoServer/4.4.2
Content-Length: 120
Content-Type: application/json; charset=UTF-8


{
  "faces": [
    {
     "age": 26,
     "emotions": [
       "neutral",
       "sad"
```

(continues on next page)

(продолжение с предыдущей страницы)

```
    ],
    "gender": "female",
    "x1": 595,
    "x2": 812,
    "y1": 127,
    "y2": 344
   }
  ]
}
```

Для добавления лица в базу данных вместе с информацией о поле, возрасте и эмоциях, отправьте POST-запрос на адрес `v1/face`.

Запрос №2

```
POST /v1/face/ HTTP/1.1
Host: 127.0.0.1
Authorization: Token e93437ccdae66d57a45a5c6d9aa7602e
Content-Type: application/json
Content-Length: [length]

{
  "meta": "Jane Berry",
  "photo": "http://static.findface.pro/sample.jpg",
  "galleries": ["gal1", "niceppl"],
  "emotions": true,
  "gender": true,
  "age": true
}
```

Ответ

```
HTTP/1.1 200 OK
Date: Mon, 13 Jun 2016 06:04:02 GMT
Content-Type: application/json; charset=UTF-8
Content-Length: [length]

{
  "results": [
    {
      "galleries": ["default", "gal1", "niceppl"]
      "id": 2334,
      "meta": "Jane Berry",
      "photo": "http://static.findface.pro/sample.jpg",
      "photo_hash": "dc7ac54590729669ca869a18d92cd05e",
      "timestamp": "2016-06-13T11:11:29.425339",
      "age": 26,
      "emotions": [
       "neutral",
       "sad"
      ],
      "gender": "female",
      "x1": 225,
```

(continues on next page)

(продолжение с предыдущей страницы)

```
      "x2": 307,
      "y1": 345,
      "y2": 428
    }
  ]
}
```

8.2 Группировка лиц персоны в базе данных

Вы можете интегрировать FindFace Enterprise Server SDK в системы видеонаблюдения и видеоаналитики. Для этого задействуйте функцию группировки лиц, принадлежащих одному человеку (персоне).

В этом разделе:

8.2.1 Принцип работы

- Содержащее лицо изображение (например, захваченное в кадре видеодетектором лиц) отправляется на Сервер FindFace для идентификации через запрос `/face POST`.
- При идентификации персоны Сервер использует свойство лица `person_id`. Для каждой персоны в базе данных значение данного свойства должно быть уникально.
- Сервер FindFace получает новое лицо и находит наиболее похожее на него в базе данных (так называемое опорное лицо). В случае если схожесть лиц равна или превышает порог, указанный в файле конфигурации `findface-facenapi.ini` (`person_identify_threshold`), новое лицо добавляется в базу данных с присвоением того же значения параметра `person_id`, что и у опорного лица. Оно также наследует некоторые другие свойства опорного лица (например, метаданные). Это означает, что новое лицо было распознано как принадлежащее существующей персоне.
- Если схожесть между лицами не превышает заданного порогового значения, Сервер рассматривает новое лицо как нераспознанное. В этом случае новое лицо добавляется в базу данных как принадлежащее новой персоне, с новым значением `person_id`.
- В обоих случаях Сервер возвращает ответ, содержащий значение `person_id`, присвоенное добавленному лицу. Данное значение может быть далее использовано в аналитике.

Предупреждение: Похожие лица, принадлежащие разным людям, могут быть распознаны как лица одной персоны и получить одинаковые значения параметра `person_id`. Для того чтобы избежать подобной ситуации, рекомендуется периодически просматривать базу данных, вручную разрешая каждый конфликт идентичности.

8.2.2 Включение группировки лиц

По умолчанию лица персоны не группируются. Это означает, что всем вновь добавленным лицам не назначается свойство `person_id`, а разные люди Сервером не различаются.

Для того чтобы включить группировку лиц персоны, выполните следующие действия:

1. Откройте для редактирования файл конфигурации `findface-facenapi.ini`.

```
sudo vi /etc/findface-facenapi.ini
```

2. Отредактируйте настройки.

   Предупреждение: Содержимое файла `findface-facenapi.ini` должно представлять собой синтаксически верный код Python.

   Раскомментируйте и отредактируйте строку `person_identify = False`, чтобы включить группировку лиц.

```
→ person_identify = True
```

   По умолчанию группировка лиц выполняется отдельно по каждой видеокамере. Для слияния результатов группировки по всем видеокамерам раскомментируйте и отредактируйте строку `person_identify_global = False`. Данную опцию можно использовать в небольших системах до 5 камер. Если камер больше, оставьте опцию выключенной.

```
→ person_identify_global = True
```

   Раскомментируйте строку и установите порог для идентификации персоны от 0 до 1.

```
→ person_identify_threshold = 0.75
```

3. Перезапустите сервис `findface-facenapi`.

```
sudo service findface-facenapi restart
```

8.2.3 Последовательность REST API для анализа данных персоны

Существует множество способов задействовать группировку лиц персон в аналитике. Типичная последовательность REST API для идентификации персоны и последующей работы с ее данными представлена ниже:

№ п/п	Метод	Описание
1	`/face POST`	Добавьте лицо в базу данных и получите информацию о добавленном лице в формате JSON, в том числе значение параметра `person_id`.
2	`/history/search POST`	Найдите все события по видеокамерам, связанные с персоной, чей `person_id` вы получили в ответе на запрос `/face POST`.

Метод /face POST

Запрос

```
POST /v0/face/ HTTP/1.1
Host: 127.0.0.1
Authorization: Token e93437ccdae66d57a45a5c6d9aa7602e
Content-Type: application/json
Content-Length: [length]

{
    "photo": "http://static.findface.pro/sample.jpg"
}
```

Ответ

```
HTTP/1.1 200 OK
Date: Mon, 13 Jun 2016 12:23:56 GMT
Content-Type: application/json
Content-Length: [length]

{
  "results": {
    "[595, 127, 812, 344]": [
      {
        "confidence": 1,
        "face": {
          "friend": false,
          "galleries": [
            "default"
          ],
          "id": 2,
          "meta": "Jack Smith",
          "normalized": "http://192.168.113.76:3333/uploads/20170418/1492509569217098.jpeg",
          "person_id": 2,
          "photo": "http://192.168.113.76:3333/uploads/20170418/14925095692111893.jpeg",
          "photo_hash": "53477c4a72f52c6efc951d9c7ece42bc",
          "thumbnail": "http://192.168.113.76:3333/uploads/20170418/14925095692159095.jpeg",
          "timestamp": "2017-04-18T09:59:29.211000",
          "x1": 595,
          "x2": 812,
          "y1": 127,
          "y2": 344
        }
      }
    ]
```

(continues on next page)

(продолжение с предыдущей страницы)

```
  }
}
```

Метод /history/search POST

Запрос

```
POST /v0/history/search    HTTP/1.1
Host:   127.0.0.1
Authorization:  Token   e93437ccdae66d57a45a5c6d9aa7602e
Content-Type:   application/json
Content-Length: [length]
{
    "person_id": 2,
}
```

Ответ

```
HTTP/1.1   200 OK
Date:   Mon, 13 Jun 2016 12:23:56 GMT
Content-Type:   application/json
Content-Length: [length]
{
   "next_page": "/v0/history/search?max_id=4",
   "results":[
      {
         "friend":false,
         "meta":"Jack Smith",
         "photo_hash":"9fda49f2444f93c33ad8aa914e20e53b",
         "cam_id":"12345678123456781234567812345678",
         "person_id":2,
         "timesamp":"2016-10-11T14:36:27.450000",
         "photo":"http://192.168.113.76:3333/uploads/20170418/149250956922566.jpeg",
         "id":20146,
         "y1":77,
         "x1":285,
         "x2":552,
         "y2":345
      },
      {
         "friend":false,
         "meta":"Jack Smith",
         "photo_hash":"dc7ac54590729669ca869a18d92cd05e",
         "cam_id":"12345678123456781234567812345678",
         "person_id":2,
         "timesamp":"2016-10-12T12:57:07.509000",
         "photo":"http://192.168.113.76:3333/uploads/20170418/14925095692111596.jpeg",
         "id":20147,
         "x1":236,
         "y1":345,
         "x2":311,
         "y2":419
```

(continues on next page)

(продолжение с предыдущей страницы)

```
      }
   ]
}
```

8.2.4 Другие методы API для работы с персоной в базе данных

Добавление и изменение person_id

Для добавления или изменения значения параметра person_id определенного лица используйте метод PUT /face/id/<face_id>.

Предупреждение: Поскольку свойство person_id назначается только вновь добавляемым лицам, прежние лица в базе данных не участвуют в идентификации персоны. Используйте метод PUT /face/id/<face_id> для решения данной проблемы.

Запрос

```
PUT /v0/face/id/5/ HTTP/1.1
Host: 127.0.0.1
Authorization: Token e93437ccdae66d57a45a5c6d9aa7602e
Content-Type: application/json
Content-Length: [length]

{
  "person_id": "4"
}
```

Ответ

```
HTTP/1.1 200 OK
Date: Mon, 13 Jun 2016 12:23:56 GMT
Content-Type: application/json
Content-Length: [length]

{
  "id": 5,
  "meta": "Jane Richardson",
  "person_id": "4",
  "photo": "http://static.findface.pro/sample2.jpg",
  "photo_hash": "dc7ac54590729669ca869a18d92cd05e",
  "timestamp": "2016-06-13T11:06:42.075754",
  "x1": 225,
  "x2": 307,
  "y1": 345,
  "y2": 428
}
```

Поиск событий, связанных с персоной

Для поиска в истории видеокамер всех событий, связанных с персоной с заданным `person_id`, используйте метод `GET /person/history/id/<person_id>` (наравне с методом `/history/search POST`).

Запрос

```
GET    v0/person/history/id/2001   HTTP/1.1
Host:   127.0.0.1
Authorization:  Token   e93437ccdae66d57a45a5c6d9aa7602e
Content-Type:   application/json
Content-Length: [length]
{
    "cam_ids":    [1, 25, 26, 27],
    "start":  "2016-06-13T11:00:00.000000",
    "end":    "2016-06-14T11:00:00.000000"
}
```

Ответ

```
HTTP/1.1   200 OK
Date:   Mon,    13  Jun 2016    12:23:56    GMT
Content-Type:   application/json
Content-Length: [length]
{
    "results":
    [
        {
            "person_id":  2001,
            "face_id":    240344,
            "cam_id": 25,
            "meta":   "Sam   Berry",
    "screenshot":"https://static.findface.pro/57726179d6946f02f3763824/
↪dc7ac54590729669ca869a18d92cd05e_thumb.j
pg",
            "timestamp":  "2016-06-13T11:06:42.075754",
        },
        {
            "person_id":  2001,
            "face_id":    240422,
            "cam_id": 25,
            "meta":   "Sam   Berry",
            "screenshot": "https://static.findface.pro/57726179
d6946f02f3763824/dc7ac54590729669ca869a18d92cd05e_thumb.j
pg",
            "timestamp":  "2016-06-13T11:08:44.073452",
        }
    ]
}
```

Получение списка персон

Для получения списка всех существующих персон в базе данных, используйте метод `GET /persons`.

Запрос

```
GET /v0/persons HTTP/1.1
Host: 127.0.0.1
Authorization: Token e93437ccdae66d57a45a5c6d9aa7602e
```

Ответ

```
HTTP/1.1   200 OK
Date:   Mon,    13  Jun 2016    12:23:56    GMT
Content-Type:   application/json
Content-Length: [length]

{
  "results": [
    {
      "id": 2,
      "meta": ""
    }
  ]
}
```

8.3 Распознавание «свой-чужой»

Для расширения функционала видеоаналитики после настройки *Группировка лиц персоны в базе данных* можно активировать распознавание «свой-чужой».

В этом разделе:

8.3.1 О своих и чужих

Система распознавания «свой-чужой» Сервера FindFace может положительно определять только «своих», не «чужих». «Свой» — это человек, чье лицо появлялось в поле зрения одной и той же видеокамеры определенное количество дней в течение определенного промежутка времени. Во всех других случаях человек рассматривается как «не свой».

8.3.2 Включение распознавания «свой-чужой»

Для включения распознавания «Свой-чужой» выполните следующие действия:

1. Настройте и выполните отладку функции *группировки лиц персоны*.

2. Откройте для редактирования файл конфигурации `findface-facenapi.ini`.

```
sudo vi /etc/findface-facenapi.ini
```

3. Отредактируйте настройки.

> **Предупреждение:** Содержимое файла `findface-facenapi.ini` должно представлять собой синтаксически верный код Python.

Своим человек считается, если появляется в поле зрения одной и той же видеокамеры определенное количество дней в течение интервала `[now() - $interval ; now()]`. Раскомментируйте строку и отредактируйте количество дней появления человека в поле зрения видеокамеры.

```
→ friend_count = 5
```

Раскомментируйте и задайте интервал в секундах, в течение которого человек должен попадать в поле зрения видеокамеры определенное количество дней (по умолчанию 1 неделя):

```
→ friend_interval = (3600*24*7)
```

4. Перезапустите сервис `findface-facenapi`.

```
sudo service findface-facenapi restart
```

8.3.3 Распознавание «свой-чужой» в REST API

Ниже приведен пример запроса `POST /face` и соответствующего ответа, содержащего параметр `friend` («свой») (`"friend": true` или `"friend": false`).

Запрос

```
POST /v0/face/ HTTP/1.1
Host: 127.0.0.1
Authorization: Token e93437ccdae66d57a45a5c6d9aa7602e
Content-Type: application/json
Content-Length: [length]

{
    "photo": "http://static.findface.pro/sample.jpg"
}
```

Ответ

```
HTTP/1.1 200 OK
Date: Mon, 13 Jun 2016 12:23:56 GMT
Content-Type: application/json
Content-Length: [length]

{
  "results": {
```

(continues on next page)

(продолжение с предыдущей страницы)


```
    "[595, 127, 812, 344]": [
      {
        "confidence": 1,
        "face": {
          "friend": true,
          "galleries": [
            "default"
          ],
          "id": 2,
          "meta": "Jack Smith",
          "normalized": "http://192.168.113.76:3333/uploads/20170418/1492509569217098.jpeg",
          "person_id": 2,
          "photo": "http://192.168.113.76:3333/uploads/20170418/14925095692111893.jpeg",
          "photo_hash": "53477c4a72f52c6efc951d9c7ece42bc",
          "thumbnail": "http://192.168.113.76:3333/uploads/20170418/14925095692159095.jpeg",
          "timestamp": "2017-04-18T09:59:29.211000",
          "x1": 595,
          "x2": 812,
          "y1": 127,
          "y2": 344
        }
      }
    ]
  }
}
```


8.4 Компонент Extraction API

С компонентом `Extraction API` вы можете гибко настраивать формат API-ответов для извлечения таких данных, как координаты рамки с лицом, нормализованное лицо, пол, возраст, эмоции, вектор признаков. Использование данной функции в вашей системе может значительно расширить спектр аналитических задач, которые она в состоянии выполнить.

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

Примечание: Вы также можете *использовать* `Extraction API` для непосредственного извлечения биометрических образцов, т. е. как альтернативу `findface-nnapi`.

Совет: Нормализованные изображения в формате `base64`, полученные с помощью компонента `Extraction API`, подходят для обработки в компоненте `findface-facenapi`.

Важно: Для того чтобы использовать такие функции `Extraction API`, как *оценка качества лица* и *автоповорот* исходных изображений, активируйте модуль `Face Quality` (Качество лица). Узнайте больше у наших специалистов техподдержки по адресу info@ntechlab.com.

В этом разделе:

8.4.1 Развертывание Extraction API

Для развертывания компонента `Extraction API` выполните следующие действия:

Примечание: Для работы `Extraction API` необходимы пакеты с *моделями нейронных сетей* `<findface-data>.deb`. Убедитесь, что они установлены.

1. Установите компонент.

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

2. Откройте для редактирования файл конфигурации `findface-extraction-api.ini`.

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

3. В случае если локальный сервер лицензий NTLS является удаленным, укажите его IP-адрес.

```
license_ntls_server: 192.168.113.2:3133
```

4. При необходимости настройте другие параметры. Например, включите/отключите обработку интернет-изображений.

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

5. Параметры `min_face_size` и `max_face_size` не являются фильтрами. Скорее они обозначают интервал гарантированного обнаружения лица. Данные параметры влияют на производительность, поэтому значения для них нужно подбирать с осторожностью.

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

6. Параметр `model_instances` задает количество экземпляров каждого активированного детектора лиц (`nnd`, `legacy` или `prenormalized`) и каждой активированной модели нейронной сети (вектор признаков, пол, возраст, эмоции), которое может выполняться параллельно. Значение по умолчанию (0) означает, что данное количество равно числу ядер процессора. Если количество запущенных экземпляров потребляет чересчур много памяти (что проявляется в том, что компонент Extraction API не запускается или самопроизвольно останавливается), отредактируйте значение данного параметра.

```
model_instances: 2
```

7. Для оценки качества лица включите опцию `quality_estimator`. В этом случае компонент `extraction-api` будет возвращать показатель качества в лица в параметре *detection_score*.

Совет: Показатель качества можно затем 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
```

Важно: Данная функция доступна только после активации модуля `Face Quality` (Качество лица). Узнайте больше у наших специалистов техподдержки по адресу info@ntechlab.com.

---

8. Добавьте сервис `Extraction API` в автозагрузку Ubuntu и запустите его.

```
sudo systemctl enable findface-extraction-api && sudo systemctl start findface-extraction-api
```

8.4.2 API-запросы

Компонент `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]
}
```

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

- `"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.
- `"need_emotions"`: возвращает эмоции.

- "need_age": возвращает возраст.
- "need_normalized": возвращает нормализованное изображение лица в формате base64. Нормализованное изображение может быть снова отправлено в 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.

Важно: Данная функция доступна только после активации модуля Face Quality (Качество лица). Узнайте больше у наших специалистов техподдержки по адресу info@ntechlab.com.

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

8.4.3 Формат API-ответа

Типичный ответ компонента 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").

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

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, если запрашивался.
- `"gender"`: информация о поле (MALE или FEMALE) с указанием точности оценки, если запрашивалась.
- `"age"`: оценка возраста, если запрашивалась.
- `"emotions"`: все доступные эмоции в порядке убывания вероятности, если запрашивались.
- `"normalized"`: нормализованное изображение лица в формате base64, если запрашивалось.

```
{
    "bbox": { "left": 1, "right": 2, "top": 3, "bottom": 4},
    "detection_score": -0.0004299,
    "facen": "...",
    "gender": {
        "gender": "MALE",
        "score": "1.123"
    },
    "age": 23.59,
    "emotions": [
        { "emotion": "neutral", "score": 0.95 },
        { "emotion": "angry", "score": 0.55 },
        ...
    ],
    "normalized": "...",
}
```

8.4.4 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
```

(continues on next page)

(продолжение с предыдущей страницы)


```
          },
          "detection_score": -0.0012599,
          "facen": "qErDPTE...vd4oMr0=",
          "gender": {
            "gender": "FEMALE",
            "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
            },
            {
```


(continues on next page)

(продолжение с предыдущей страницы)


```
            "emotion": "surprise",
            "score": -0.024765536
          },
          {
            "emotion": "fear",
            "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"
       }
```


(continues on next page)

(продолжение с предыдущей страницы)

```
    ]
  }
 ]
}
```

8.4.5 Извлечение биометрических образцов

По умолчанию, компонент `findface-facenapi` обнаруживает лица на фото и отправляет их в компонент `findface-nnapi` для извлечения биометрического образца (вектора признаков). Затем `findface-facenapi` сохраняет полученные образцы в базах данных MongoDB и Tarantool. В качестве лучшей альтернативы `findface-nnapi` в этой последовательности можно использовать `Extraction API`.

Основным преимуществом `Extraction API` по сравнению с `findface-nnapi` является его способность к распараллеливанию и автоматическому распределению нагрузки между параллельными экземплярами, в то время как для `findface-nnapi` балансировку нагрузки нужно *настраивать* с помощью NginX вручную.

Для того чтобы извлекать биометрические образцы с помощью `Extraction API`, выполните следующие действия:

1. Откройте для редактирования файл конфигурации `findface-facenapi.ini`:

   ```
   sudo vi /etc/findface-facenapi.ini
   ```

2. Раскомментируйте и отредактируйте следующим образом параметр `extractor`:

   ```
   extractor = 'extraction-api'
   ```

   Предупреждение: Содержимое файла `findface-facenapi.ini` должно представлять собой синтаксически верный код Python.

3. Раскомментируйте и/или отредактируйте `extraction_api_url` в соответствии с фактическим расположением сервера `Extraction API`:

   ```
   extraction_api_url = 'http://localhost:18666'
   ```

4. Запустите сервис `Extraction API` и добавьте его в автозагрузку Ubuntu.

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

5. Перезапустите сервис `findface-facenapi`.

   ```
   sudo service findface-facenapi restart
   ```

6. Остановите сервис `findface-nnapi` и удалите его из автозагрузки.

   ```
   sudo service findface-nnapi stop && sudo systemctl disable findface-nnapi
   ```

7. Проверьте статус сервисов. Команда вернет описание сервисов, их статус (должен быть Активен), путь и длительность текущей сессии.

```
sudo service 'findface*' status
```

8.5 Пакетная загрузка лиц

Функция пакетного добавления лиц позволяет отправлять лица в компонент `findface-facenapi` одновременно из множества изображений.

В этом разделе:

8.5.1 Общие сведения

Вы можете пакетно добавлять лица одним из следующих способов:

- из изображений в текущей папке,
- из изображений в заданной дочерней папке,
- из изображений во всех дочерних папках.

Для того чтобы установить компонент для пакетного добавления лиц, выполните команду:

```
sudo apt-get install findface-mass-enroll
```

Для того чтобы отобразить справку по компоненту, выполните команду:

```
findface-mass-enroll --help
```

```
## $ findface-mass-enroll --help
Usage: findface-mass-enroll [OPTIONS] COMMAND [ARGS]...

Options:
  --job PATH  Job file (default: ffmassenroll.job)
  --help      Show this message and exit.

Commands:
  prepare  Prepare upload job
  print    Print contents of job file as JSON
  run      Run upload job

$ findface-mass-enroll prepare --help
Usage: findface-mass-enroll prepare [OPTIONS] [IMAGES]...

  This subcommand is used to prepare one or more job files for subsequent
  runs.

  Examples:
```

(continues on next page)

(продолжение с предыдущей страницы)

```
  Enrolling all *.jpg files in current directory with meta 'Phillip J. Fry':

  $ ls
  photo1.jpg photo2.jpg photo3.jpg
  $ findface-mass-enroll prepare --meta-const='Phillip J. Fry' '*.jpg'

  Enrolling all JPEGs and PNGs from a subdirectory with meta from accompanying TXT files:

  $ ls subdir
  photo1.jpg photo1.txt photo2.png photo2.txt photo3.jpeg photo3.txt
  $ findface-mass-enroll prepare --meta-companion='txt' 'subdir/*.jpg' 'subdir/*.png' 'subdir/*.
↪jpeg'

  Enrolling JPEGs from all subdirectories with meta from CSV file:

  $ cat meta.csv
  "Phillip J. Fry","dir1/photo1.jpg"
  "Phillip J. Fry","dir1/photo2.jpg"
  "Phillip J. Fry","dir1/photo3.jpg"
  "Turanga Leela","dir2/photo1.jpg"
  "Turanga Leela","dir2/photo2.jpg"
  "Turanga Leela","dir2/photo3.jpg"
  $ ls -R
  .:
  meta.csv

  ./dir1:
  photo1.jpg photo2.jpg photo3.jpg

  ./dir2:
  photo1.jpg photo2.jpg photo3.jpg
  $ findface-mass-enroll prepare --meta-csv=meta.csv '**/*.jpg' '**/*.jpeg'

Options:
  --meta-const TEXT      Shared metadata string
  --meta-companion TEXT  Extension of metadata files accompanying the images
                         (e.g. txt)
  --meta-csv PATH        Name of the CSV file containing metadata
  --meta-filename        Use file name (without extension) as metadata string
  --split INTEGER        Split job file into N parts (default: don't split)
  --help                 Show this message and exit.

## $ findface-mass-enroll print --help
Usage: findface-mass-enroll print [OPTIONS]

  Print contents of job file as JSON

Options:
  --failed  Show only failed images
  --help    Show this message and exit.

## $ findface-mass-enroll run --help
Usage: findface-mass-enroll run [OPTIONS]

  Run upload job
```

(continues on next page)

(продолжение с предыдущей страницы)

```
Options:
  --parallel INTEGER              Number of enroll threads (default: 10)
  --api TEXT                      API url (default: http://127.0.0.1:8000/)
                                  [required]
  --token TEXT                    API token  [required]
  --gallery TEXT                  Enroll faces into specified gallery
                                  (default: default)
  --failed                        Include failed images
  --mf-selector [all|biggest|reject]
                                  mf_selector (biggest,all,reject)
  --gender                        Extract gender
  --age                           Extract age
  --emotions                      Extract emotions
  --stats-interval INTEGER        Output stats after every STATS_INTERVAL
                                  seconds (default: 1)
  --help                          Show this message and exit.
```

Для пакетного добавления лиц выполните следующие действия:

1. Подготовьте файл задания (job-файл), содержащий список изображений с метаданными (метод `prepare`). Если все изображения имеют одинаковую строку с метаданными, вы можете указать ее прямо в командной строке при подготовке job-файла (опция `--meta-const`). Если каждому изображению должна соответствовать уникальная строка с метаданными, задайте соответствие метаданных изображениям в CSV-файле (опция `--meta-csv`).

   Примечание: В качестве источника метаданных CSV-файл должен иметь следующий формат: `метаданные | изображение`. Если некоторые изображения не были перечислены в CSV-файле, они будут добавлены с пустыми метаданными.

   Совет: Для того чтобы записать список изображений в CSV-файл, можно использовать приведенную ниже команду. В этом случае для каждого изображения в списке в качестве строки с метаданными будет создана строка с указанием полного пути к изображению (в формате `метаданные | изображение`).

   ```
   find /home/user/sample | grep -E 'jpg|png' |awk '{print $0","$0}' > list.csv
   ```

2. При необходимости отобразите содержимое job-файла (метод `print`).
3. Отправьте лица в компонент findface-facenapi для дальнейшей обработки (метод `run`).

   Примечание: В случае неудачи при обработке job-файла исправьте ошибку и повторите попытку с опцией `--failed` (см. пример ниже).

8.5.2 Пример

Добавьте лица из всех файлов `.jpg` в папке `/home/user/images/` с общими метаданными `Phillip J. Fry`:

Для отображения списка всех файлов в папке выполните команду:

```
ls /home/user/images/
photo1.jpg photo2.jpg photo3.jpg ...
```

Подготовьте job-файл:

```
findface-mass-enroll prepare --meta-const='Phillip J. Fry' '/home/user/images/*'

Looking for images matching '*.jpg'
2055 files prepared for upload
2055 files in job file samplejob
```

Выполните job-файл:

```
findface-mass-enroll run --token 'RczGgVEMizR1njHHQegNH_g9mwGl6-A1' --api http://127.0.0.1:8000/ --
↪gender --age --emotions --mf-selector=all

[33/2055] faces processed (4 succeeded, 9 failed, 10 skipped). 2.14 rps. [00:00:17/00:16:04]

---------------------------------------- Summary ----------------------------------------

Found 2055 images in job file
Skipped 0 already processed images
Successfully processed 2000 images
Failed to process 55 images
```

В случае неудачи при обработке job-файла исправьте ошибку и повторите попытку с опцией `--failed`.

```
findface-mass-enroll run --token 'RczGgVEMizR1njHHQegNH_g9mwGl6-A1' --api http://127.0.0.1:8000/ --
↪gender --age --emotions --mf-selector=all --failed
```

8.6 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 SDK 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 `tntapi` во время установки компонента.

В этом разделе:

8.6.1 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.

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

```
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100  6700  100  6700    0     0  45812      0 --:--:-- --:--:-- --:--:-- 45890
{
  "galleries": [
    {
      "cnt_indexed": 0,
      "id": 1,
      "cnt_preindex": 0,
      "name": "591b0cdfb0d5bd7058ef0968_default",
      "cnt_linear": 35268
    },
    {
      "cnt_indexed": 0,
      "id": 2,
      "cnt_preindex": 0,
      "name": "591b0cdfb0d5bd7058ef0968_lublino",
      "cnt_linear": 1818
    },
    {
      "cnt_indexed": 0,
      "id": 3,
      "cnt_preindex": 0,
      "name": "591b0cdfb0d5bd7058ef0968_gifs",
      "cnt_linear": 297
    }
  ],
  "total": 3
}
```

8.6.2 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.

Пример

Запрос

```
curl http://127.0.0.1:8001/stat/info/5968bda4a2a4bb6018bee2b2_cam_cam1 | jq
```

Ответ

```
 % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100   210  100   210    0     0  17654      0 --:--:-- --:--:-- --:--:-- 19090
{
  "cnt_indexed": 0,
  "cnt_preindex_deleted": 0,
  "index_file": "none",
  "index_loaded": false,
  "cnt_preindex": 0,
  "cnt_linear": 85011,
  "cptr": 29556448,
  "id": 34,
  "name": "5968bda4a2a4bb6018bee2b2_cam_cam1",
  "cnt_indexed_deleted": 0
}
```

8.7 Прямые API-запросы к базе данных Tarantool

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

В этом разделе:

8.7.1 Общие сведения

API-запросы к базе данных Tarantool следует 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 SDK 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 `tntapi` во время установки компонента.

Каждый API-запрос к Tarantool содержит следующие параметры:

- `:ver`: версия API (v1 на данный момент).
- `:name`: имя галереи.
- `:id`: id лица.

8.7.2 Добавление лица

Запрос

```
POST /:ver/:name/add/:id
```

`Body`: необработанный вектор признаков (facen).

Возвращает:

- HTTP 200 и пустое тело в случае успеха.
- HTTP 409, если лицо с таким id уже есть в галерее.
- HTTP с отличным от 200 статусом и описание ошибки в случае неудачи.

Пример

```
curl -s -D - 'http://localhost:8001/v1/my_gal/add/1234' --data-binary @3827305024709134.facen

HTTP/1.1 200 Ok
Content-length: 0
Server: Tarantool http (tarantool v1.7.3-673-g23cc4dc)
Connection: keep-alive
```

8.7.3 Получение вектора признаков

Запрос

```
GET /:ver/:name/get/:id
```

Возвращает:

- Представление лица в формате JSON, содержащее его id и вектор признаков в формате base64 в случае успеха.
- HTTP 404, если лицо с заданным id не найдено в галерее.
- HTTP с отличным от 200 статусом и описание ошибки в случае неудачи.

Совет: Для преобразования вектора признаков из формата `base64` в бинарный файл используйте команду:

```
echo 'facen in base64' |base64 -d> facen
```

Пример

```
curl -s -D - 'http://localhost:8001/v1/my_gal/get/1234' HTTP/1.1 200 Ok Content-length: 1754␣
↪Server: Tarantool http (tarantool v1.7.3-673-g23cc4dc) Connection: keep-alive {"facen":
↪"BFa9PWNlS7215fI98ETQvJkxML2hUFY9cF\/Tu9ZjnLx\/
↪uVc9EzWSPQTsR7zoysI8+4PSPIsjnr2GV1M8eFMKvfn9mjsPPjA8ZXoNvTEsSr0rJkM9MR0IPINXSj3Em0s9awm5Oos5SD380a693GroPBz6nzxQM
↪rTyEDNm8Ti\/0ve4Trr1rnQA+Yc\/KvJzqnbzOPSG998CKPBFpAr77kFO9BonDvK9B0buvjAq9Q7A\/
↪u6awnTw0lvy80QZcvRFQAz0BdH498hF6vQKRcDy77c08mGRkvQ305DomnBM9XSqwvN54GT0ClFO9a+kWvhp7iT3uqqU9v1+\/
↪vYhzm7uREt091douuyDKRr2PcIG9Uc8xPVJnvzt5T309NicxPD9SAr3f6sO8UmlhvRMI67wlTte880wYvUF8o7xg4\/
↪g8aqNQu\/
↪AAWD2z59C9CQCrPepF7Dy8qUa9iCczPfKv+Dy+bRo9KhyYPZfY0b1xtbY7nKXLuvYFbr0g8rM86o0QPRCKOj1a7rU9bd+3Pbqs7LslJcO9bBh+vVY
↪Guv2beTy56wg7p\/hTPdxQgr0jxQQ9Ud0CPZcx\/
↪LtRLiU9bECQvUnvszpMVcM8b3OovURPET3JdHs9LyQUPsc9JzvW1ZQ7y2ySPdN4Xb0xi9c8X7UevRqjVL0MLpE9PoQpvFxxjD2NCDO81jH\/
↪PF1KFTzc3pc7qpaFPXxuPb2tjsY9iA5lPR1NoT1+Uuu7G6gpu727wTwo6ii8iaH+PI1WY72D9QG+8lhAPUegx71VsFs8ajQLvOdekrzGqAg+zhPLv
↪O1307D1ZMSk9IxqGvYCvFb1bE429hZF4vewikzwDbfG8wwYNPiQn4L2NV6Q9VKrvPTjwTr3dlGO5jck+vZ\/
↪KID1+n8Y8qpvnvOJjBj2P4+w8IJGgvROAfz1S4ve8QEouvQ5CkDu0OTI8\/v\/
↪pvFrK5b3bkIO82LVBPcf2Yr0aGaU9RArUvEecJz1r8zk87U4vvC65ljz6kRS956U2PH6JMT5nfAg7KX7qPBz7Ejy60vk9\/
↪iEPPYw8pT3Mfvk8UQYyPUCG+TyD5CO90c6nvSVLvDwRJSW9C3udvDORMz3zqtU8yd+0PXrubj3u9pQ9cGZIPVjlqTz6eIs8Z4wsPIjEIT3gnqI9kj
↪iVvZjEhT3W0B69IRojvQGUVj2J6vQ9FiDhPNRUO7Obcum9fOOvPKA\/y7yB9wq9ntsBPYL6XL0wgkw7nLu6O\/\/
↪USz1EoUg9JKE9PLDzNL0Pns49fPVyPJfZaj2g6pi8MuZePV0xQLxkR4W9pEe7vYTv7jytv567nakpPcCHZbsfjx89jPENPW0x87vr3Wi84L9mvSGe
↪8JL02Vh0879v\/O6weQjxpD7k85Kj2PGb0ej0V6xS8\/4EvPXmv3z0=","id":1234}
```

8.7.4 Удаление лица

Предупреждение: Удаление лица из Tarantool не удалит его из MongoDB.

Запрос

```
DELETE /:ver/:name/del/:id
```

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 случае неудачи.

Пример

```
curl -s -D - -X DELETE 'http://localhost:8001/v1/my_gal/del/1234'

HTTP/1.1 200 Ok
Content-length: 0
Server: Tarantool http (tarantool v1.7.3-673-g23cc4dc)
Connection: keep-alive
```

8.7.5 Поиск лица

Запрос

```
POST /:ver/:name/search/:limit/:threshold?linear_search
```

`:limit`: максимальное количество лиц в ответе.
`:threshold`: минимальная степень схожести лиц в ответе (от 0 до 1).
`linear_search` (опционально): установите `linear_search=1` (true), чтобы при поиске лиц использовать только пространство `linear`. Данная настройка имеет приоритет над настройкой `only_index` (`/etc/tarantool/instances.enabled/FindFace.lua`).
`body`: необработанный вектор признаков.

Возвращает:

- Массив JSON с лицами, содержащий поля `conf` и `id` в теле в случае успеха. Значение в заголовке `X-search-stat` показывает, был ли использован быстрый индекс для поиска: `with_index` или `without_index`.
- HTTP с отличным от 200 статусом и описание ошибки в случае неудачи.

Пример

```
curl -s -D - 'http://localhost:8001/v1/my_gal/search/1/0.65?linear_search=1' --data-binary␣
↪@3827305024709134.facen

HTTP/1.1 200 Ok
Content-length: 22
X-search-stat: without_index
Server: Tarantool http (tarantool v1.7.3-673-g23cc4dc)
Connection: keep-alive

[{"conf":1,"id":1234}]
```

8.7.6 Получение списка лиц

Запрос

```
GET /:ver/:name/list/:start_id/:count
```

`:start_id`: минимальное значение id лица в ответе.

`:count`: максимальное количество лиц в ответе.

Возвращает:

- Массив JSON с лицами и URL следующей страницы результатов в случае успеха. Для каждого лица указывается его `id`, вектор признаков в формате `base64` и имя пространства Tarantool, в котором хранится лицо (`linear`, `preindex`, или `indexed`). URL следующей страницы должен быть передан в другом запросе как параметр `:start_id` для получения следующей страницы результатов.
- HTTP с отличным от 200 статусом и описание ошибки в случае неудачи.

Пример

```
curl -s -D - 'http://localhost:8001/v1/my_gal/list/0/1' HTTP/1.1 200 Ok Content-length: 1795␣
↪Server: Tarantool http (tarantool v1.7.3-673-g23cc4dc) Connection: keep-alive {"faces":[{"id
↪":1234,"space":"linear","facen":"BFa9PWNlS7215fI98ETQvJkxML2hUFY9cF\/Tu9ZjnLx\/
↪uVc9EzWSPQTsR7zoysI8+4PSPIsjnr2GV1M8eFMKvfn9mjsPPjA8ZXoNvTEsSr0rJkM9MR0IPINXSj3Em0s9awm5Oos5SD380a693GroPBz6nzxQM
↪jPd7QhDxUIzC+g90sPUWUDLwjk7U9cpWkPZ83rTyEDNm8Ti\/0ve4Trr1rnQA+Yc\/
↪KvJzqnbzOPSG998CKPBFpAr77kFO9BonDvK9BObuvjAq9Q7A\/
↪u6awnTw0lvy80QZcvRFQAz0BdH498hF6vQKRcDy77c08mGRkvQ305DomnBM9XSqwvN54GT0ClFO9a+kWvhp7iT3uqqU9v1+\/
↪vYhzm7uREt091douuyDKRr2PcIG9Uc8xPVJnvzt5T309NicxPD9SAr3f6sO8UmlhvRMI67wlTte880wYvUF8o7xg4\/
↪g8aqNQu\/
↪AAWD2z59C9CQCrPepF7Dy8qUa9iCczPfKv+Dy+bRo9KhyYPZfY0b1xtbY7nKXLuvYFbr0g8rM86o0QPRCKOj1a7rU9bd+3Pbqs7LslJcO9bBh+vVY
↪Guv2beTy56wg7p\/hTPdxQgr0jxQQ9Ud0CPZcx\/
↪LtRLiU9bECQvUnvszpMVcM8b3OovURPET3JdHs9LyQUPsc9JzvW1ZQ7y2ySPdN4Xb0xi9c8X7UevRqjVL0MLpE9PoQpvFxxjD2NCDO81jH\/
↪PF1KFTzc3pc7qpaFPXxuPb2tjsY9iA5lPR1NoT1+Uuu7G6gpu727wTwo6ii8iaH+PI1WY72D9QG+8lhAPUegx71VsFs8ajQLvOdekrzGqAg+zhPLv
↪O1307D1ZMSk9IxqGvYCvFb1bE429hZF4vewikzwDbfG8wwYNPiQn4L2NV6Q9VKrvPTjwTr3dlGO5jck+vZ\/
↪KID1+n8Y8qpvnvOJjBj2P4+w8IJGgvROAfz1S4ve8QEouvQ5CkDu0OTI8\/v\/
↪pvFrK5b3bkIO82LVBPcf2Yr0aGaU9RArUvEecJz1r8zk87U4vvC65ljz6kRS956U2PH6JMT5nfAg7KX7qPBz7Ejy60vk9\/
↪iEPPYw8pT3Mfvk8UQYyPUCG+TyD5CO90c6nvSVLvDwRJSW9C3udvDORMz3zqtU8yd+0PXrubj3u9pQ9cGZIPVjlqTz6eIs8Z4wsPIjEIT3gnqI9kj
↪iVvZjEhT3WOB69IRojvQGUVj2J6vQ9FiDhPNRUO7Obcum9fOOvPKA\/y7yB9wq9ntsBPYL6XLOwgkw7aIu6Q\/\/
↪USz1EoUg9JKE9PLDzNL0Pns49fPVyPJfZaj2g6pi8MuZePV0xQLxkR4W9pEe7vYTv7jytv567nakpPcCHZbsfjx89jPENPW0x87vr3Wi84L9mvSGe
↪8JL02Vh0879v\/O6weQjxpD7k85Kj2PGbOejOV6xS8\/4EvPXmv3z0="}],"next":5678}
```

(continues on next page)

(продолжение с предыдущей страницы)

8.8 Дополнительные возможности tntapi

В этом разделе:

8.8.1 Дополнительные параметры конфигурации

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

```
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 = 320, log_
↪requests = false})
```

Дополнительные параметры:

Параметры:	Значение по умолчанию	Описание
log_requests	true	Включает запись запросов в лог (`/var/log/tarantool/FindFace.log`).
facen_size	320	Размер вектора признаков (facen). Перед тем как редактировать этот параметр, обязательно проконсультируйтесь с нашими специалистами.
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.

8.8.2 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})
```

8.8.3 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.

Глава 9

REST API

9.1 Как пользоваться REST API

В этом разделе:

9.1.1 Точка доступа

Все запросы REST API должны быть отправлены на адрес `http://<facenapi_ip>:8000/v1/`.

9.1.2 Версия API

Версия API обновляется каждый раз при больших изменениях функционала, при этом новые версии, как правило, совместимы с прежними версиями FindFace Enterprise Server SDK. Версия API указывается в пути запроса (например, v1 в `/v1/detect/`).

На данный момент самой последней версией API является v1, обеспечивающая такие передовые функции, как распознавание пола, возраста и эмоций.

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

9.1.3 Авторизация

Для всех методов API должна быть выполнена простая HTTP-авторизация на базе токена. Для того чтобы авторизовать запрос, после заголовка `Authorization` нужно добавить метку `Token` и ввести через пробел свой токен авторизации:

Примечание: См. *Создание токена авторизации*.

```
Authorization: Token yfT8ftheVqnDLS3QOyCiTH3E8YY_cm4p
```

Если запрос не содержит действительный токен, результатом его выполнения будет являться ответ `HTTP 401 Unauthorized`.

9.1.4 Распространенные типы объектов

Лицо

Представляет собой изображение человеческого лица. Имейте в виду, что на одной фотографии может быть несколько лиц. Разные изображения одного и того же человека также рассматриваются как разные лица.

- `"id" (число)`: уникальный идентификатор лица, генерируемый Сервером.
- `"timestamp"` (строка): время создания объекта лицо в формате ISO8601.
- `"photo"` (строка): URL или имя файла с изображением, которое было использовано для создания объекта лицо.
- `"photo_hash"` (строка): хэш исходной фотографии. Имейте в виду, что хэш идентичных фотографий будет одинаков, в то время как разные фотографии с большой вероятностью будут иметь разный хэш. Данный параметр не подлежит интерпретированию.
- `"thumbnail"` (строка): URL миниатюры лица, хранящейся на Сервере.
- `"x1" (число)`: координата x левого верхнего угла прямоугольника с лицом (рамки) на исходной фотографии.
- `"y1" (число)`: координата y левого верхнего угла прямоугольника с лицом (рамки) на исходной фотографии.
- `"x2"` (число): координата x нижнего правого угла прямоугольника с лицом (рамки) на исходной фотографии.

- `"y2"` (число): координата y нижнего правого угла прямоугольника с лицом (рамки) на исходной фотографии.
- `"meta"` (строка): строка с метаданными, которая может быть использована для хранения любой связанной с лицом информации (например, имени человека).
- `"galleries" (string[])`: массив имен галерей, в которых хранится лицо.

Рамка

Представляет собой прямоугольник на фотографии, как правило, содержащий лицо. Может быть задан одним из следующих способов:

- **Список координат со значениями:**
  - «x1» (число): координата x левого верхнего угла прямоугольника с лицом;
  - «y1» (число): координата y левого верхнего угла прямоугольника с лицом;
  - «x2» (число): координата x нижнего правого угла прямоугольника с лицом;
  - «y2» (число): координата y нижнего правого угла прямоугольника с лицом.
- Линейно: [x1, y1, x2, y2].

В API-запросах можно использовать оба формата, однако независимо от того, какой формат был использован, запрос вернет координаты рамки в представлении JSON.

Имейте в виду, что рамка может выходить за физические границы фотографии, включая отрицательные значения координат.

9.1.5 Формат Параметров

Существуют следующие способы передачи параметров в методы API:

- `application/json`: в формате JSON в теле запроса.
- `application/x-www-form-urlencoded`: с использованием имен полей формы и значений этих полей.
- `multipart/form-data`: параметры кодируются несколькими частями. Данный способ поддерживает загрузку фотографии в том же запросе.
- `query string`: параметры добавляются в конец URI запроса. При передаче параметров этим способом сложные структуры, такие как координаты рамок, должны быть представлены в формате JSON.

Существует два способа задать файл с изображением:

- как публичный адрес в сети Интернет (прямой, без переадресации),
- включить в запрос в составе multipart/form-data.

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

9.1.6 Использование примеров из документации

Примеры в описаниях методов иллюстрируют возможные запросы с этими методами и соответствующие ответы. Для проверки примеров нет необходимости писать код. Достаточно использовать встроенный в FindFace Enterprise Server SDK API-фреймворк. Для доступа к фреймворку введите в адресной строке браузера: `http://<facenapi_ip>:8000/v1/docs/v1/overview.html` for the API version /v1.

9.1.7 Пороговая схожесть лиц при верификации и идентификации

Для некоторых методов необходимо задать пороговую степень схожести лиц. При верификации данное значение используется для принятия решения о совпадении или несовпадении лиц. Чем выше порог, тем меньше шансов на положительную верификацию нелегитимного человека, однако некоторые подходящие фотографии могут также не пройти верификацию. При идентификации в результаты поиска попадают только те лица, чья степень схожести с искомым лицом превышает пороговую.

Существует 4 предустановленных порога:

- `Strict (0.7834)`: используется в проектах, в которых вероятность ложного пропуска (положительной верификации) должна быть минимизирована. Данный уровень соответствует вероятности ложного пропуска 1e-5 в системе False Accept Rate (FAR) при тестировании на нашей базе данных.
- `Medium (0.6616)`: низкая вероятность как ложного пропуска, так и ложного отказа в доступе. Соответствует вероятности 1e-3 FAR.
- `Low (0.5690)`: используется в проектах, в которых важно уменьшить ложный отказ в доступе, а ложный пропуск не несет за собой серьезных последствий. Соответствует вероятности 1e-1 FAR.
- `None (0)`: используется, когда нужно рассчитать степень схожести разных людей, а не проверить идентичность.

Вы также можете задать собственное пороговое значение в зависимости от ваших нужд.

Примечание: Если параметр `threshold` не передан, его значение по умолчанию устанавливается равным `0.75`.

9.1.8 Разбиение по страницам

Некоторые методы (такие как `GET /faces/` и `GET /meta/`) могут возвращать тысячи, если не сотни тысяч результатов. Для того чтобы избежать связанных с этим проблем, Сервер возвращает результаты с разбиением по страницам.

Методы, поддерживающие разбиение по страницам, возвращают в дополнение к результатам еще 2 параметра:

- `prev_page`: адрес предыдущей страницы (содержит только путь метода и указатель предыдущей порции результатов).
- `next_page`: адрес следующей страницы (содержит только путь метода и указатель следующей порции результатов).

Таким образом, если метод `GET http://<facenapi_ip>:8000/v0/faces/` вернул параметр `next_page` со значением `/v0/faces/?max_id=12345`, для получения следующей порции результатов необходим запрос `GET http://<facenapi_ip:8000/v0/faces/?max_id=12345`.

9.1.9 Ограничения

Для FindFace Enterprise Server SDK актуальны следующие ограничения.

Параметр	Значение
Формат изображения	JPEG, PNG, WEBP
Максимальный размер файла	10 МБ
Минимальный размер лица	50x50 пикселей
Максимальное количество лиц на изображении	Не ограничено

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

9.1.10 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 `status`:

- `code` — это код ошибки, который может быть использован для автоматического преобразования;
- `reason` — это описание ошибки, предназначенное для прочтения человеком, а не для автоматического преобразования.

Распространенные коды ошибок представлены в таблице ниже:

Код ошибки	Описание
`AUTH_FAILED`	Неверный токен авторизации или токен не был предоставлен.
`BAD_PARAM`	Некоторые параметры метода недействительны. Данный тип ответа содержит дополнительные атрибуты `параметр` и `значение` для описания ошибочных параметров.
`MALFORMED_JSON`	Тело запроса содержит неверно сформированный JSON.
`SERVICE_UNAVAILABLE`	Запрос не может быть обработан из-за недоступности некоторых компонентов Сервера.

9.2 Общие методы

В этом разделе:

- *Метод /faces GET*
- *Метод /faces/gallery/<gallery> GET*
- *Метод /meta GET*
- *Метод /galleries GET*
- *Метод /galleries/<gallery> POST*
- *Метод /galleries/<gallery> DELETE*
- *Метод /docs GET*
- *Метод /docs/<version> GET*
- *Метод /person/id/<id> GET*
- *Метод /history/search POST*

9.2.1 Метод /detect POST

Данный метод служит для обнаружения лица на изображении. Вы можете задать изображение в виде файла с составным содержимым (multipart/form-data) или предоставить ссылку на изображение в сети Интернет.

Параметры:

- `photo`: загруженная фотография или публичный URL фотографии.
- `gender`: если true, возвращает информацию о поле
- `age`: если true, возвращает информацию о возрасте
- `emotions`: если true, возвращает инфомацию об эмоциях

Возвращает:

- Список координат рамок вокруг обнаруженных лиц.

Пример

Запрос

```
POST /v1/detect/ HTTP/1.1
Host: 192.168.113.76:8000
Connection:close
Authorization: Token BpdNA6eaUlN9bPhXVSK1r92_SFOODPOU
Content-Type:   application/json
Content-Length: 108

{
    "photo": "https://static.findface.pro/sample.jpg",
    "emotions": true,
    "gender": true,
```

(continues on next page)

(продолжение с предыдущей страницы)

```
    "age": true
}
```

Ответ

```
HTTP/1.1 200 OK
Date: Thu, 06 Apr 2017 12:38:40 GMT
Server: TornadoServer/4.4.2
Content-Length: 120
Content-Type: application/json; charset=UTF-8


{
  "faces": [
    {
     "age": 26,
     "emotions": [
       "neutral",
       "sad"
     ],
     "gender": "female",
     "x1": 595,
     "x2": 812,
     "y1": 127,
     "y2": 344
    }
  ]
}
```

9.2.2 Метод /verify POST

Данный метод сравнивает две фотографии, проверяя, принадлежит ли изображенное на них лицо одному и тому же человеку, и возвращает степень схожести двух лиц. Дополнительно степень схожести может сравниваться с заданным пороговым значением для принятия однозначного решения о совпадении лиц.

В случае если требуется однозначное решение о совпадении лиц, на Сервер необходимо передать ненулевое значение параметра `threshold`. Вы можете использовать одно из 3-х *предустановленных значений*: `strict`, `medium` или `low`. Вы также можете задать собственное пороговое значение.

В случае если нужно просто вычислить степень схожести 2-х лиц, а не верифицировать идентичность, передайте параметр `threshold` со значением `none`.

Примечание: Если параметр `threshold` не передан, его значение по умолчанию устанавливается равным `0.75`.

Совет: При необходимости свяжитесь с нашими специалистами для подбора оптимального порогового значения для вашей системы.

Параметры:

- `photo1`: первая загруженная фотография или публичный URL фотографии.
- `photo2`: вторая загруженная фотография или публичный URL фотографии.
- `bbox1` [опционально]: массив рамок с лицами на первом фото, которые нужно сравнить.
- `bbox2` [опционально]: массив рамок с лицами на втором фото, которые нужно сравнить.
- `threshold` [опционально]: пороговая степень схожести лиц для принятия положительного решения о совпадении. Задайте значение от 0 до 1 или используйте предустановленный порог: «strict» (высокий), «medium» (средний), «low» (низкий, установлен по умолчанию), «none» (не задан). По умолчанию сравнение выполняется при 0.75.
- `mf_selector` [опционально]: задает поведение Сервера при наличии нескольких лиц на фотографии. Возможные значения `mf_selector`:
  - `"reject"`: возвращает ошибку, если хотя бы на одной из фотографий присутствует более одного лица.
  - `"biggest"` [по умолчанию]: проверяет самое большое лицо на фотографии.
  - `"all"`: проверяет схожесть всех лиц, обнаруженных на обеих фотографиях.

Примечание: Параметры `bbox1` и `bbox2` отменяют значение этого параметра.

Возвращает:

- Результат верификации: совпадает или не совпадает. Результат возвращается, если задано пороговое значение верификации. При наличии нескольких лиц на фотографиях результат указывается как для каждой пары лиц, так и на уровне фотографий. На уровне фотографий результат верификации будет положительным, если каждая пара лиц на них совпадает.
- Координаты рамок с лицами на фотографиях.
- Вероятность совпадения (степень схожести) лиц от 0 до 1.

Пример

Запрос

```
POST /v0/verify/ HTTP/1.1
Host: 127.0.0.1
Authorization: Token e93437ccdae66d57a45a5c6d9aa7602e
Content-Type: application/json
Content-Length: [length]

{
  "photo1": "http://static.findface.pro/sample.jpg",
  "photo2": "http://static.findface.pro/sample2.jpg"
}
```

Ответ

```
HTTP/1.1 200 OK
Date: Mon, 13 Jun 2016 12:23:56 GMT
Content-Type: application/json
Content-Length: [length]

{
  "results": [
    {
      "bbox1": {
        "x1": 225,
        "x2": 307,
        "y1": 345,
        "y2": 428
      },
      "bbox2": {
        "x1": 78,
        "x2": 185,
        "y1": 114,
        "y2": 222
      },
      "confidence": 0.4206026792526245,
      "verified": true
    }
  ],
  "verified": true
}
```

9.2.3 Метод /identify POST

Данный метод используется для поиска лица в базе биометрических данных и возвращает выборку лиц, если степень их схожести с искомым выше определенного *порогового значения*.

Параметры:

- `photo`: загруженная фотография или публичный URL фотографии.
- `x1, y1, x2, y2` [опционально]: координаты прямоугольника на фотографии, внутри которого находится искомое лицо(а).
- `threshold` [опционально]: пороговая степень схожести лиц для принятия положительного решения о совпадении. Задайте значение от 0 до 1 или используйте предустановленный порог: «strict» (высокий), «medium» (средний), «low» (низкий, установлен по умолчанию), «none» (не задан). По умолчанию сравнение выполняется при 0.75.
- `n` [опционально]: максимальное число лиц в выборке, одно по умолчанию.
- `strict` [опционально]: задает поведение Сервера на случай, если один или несколько шардов `tntapi` недоступны. Данный параметр имеет приоритет над параметром `tntapi_ignore_search_errors` в файле конфигурации `findface-facenapi`.
  - `True`: возвращает ошибку, если некоторые шарды `tntapi` недоступны.
  - `False` [по умолчанию]: Сервер использует доступные шарды `tntapi` для получения результатов идентификации лица и указывает в ответе отношение количества доступных шардов `tntapi` к их общему количеству в заголовке `X-Live-Servers`.

- `mf_selector` [optional]: задает поведение Сервера при наличии нескольких лиц на фотографии или внутри заданного прямоугольника. Возможные значения:
  - `"reject"`: возвращает ошибку, если хотя бы на одной из фотографий присутствует более одного лица.
  - `"biggest"` [по умолчанию]: поиск по базе данных выполняется для самого крупного лица на фотографии.
  - `"all"`: поиск по базе данных выполняется по всем лицам на фотографии.

Возвращает:

- Координаты рамок с лицами, найденными на фотографии. Для каждого набора координат возвращается массив схожих лиц из базы биометрических данных вместе с вероятностью совпадения от 0 до 1.

Пример

Запрос

```
POST /v0/identify/ HTTP/1.1
Host: 127.0.0.1
Authorization: Token e93437ccdae66d57a45a5c6d9aa7602e
Content-Type: application/json
Content-Length: [length]

{
  "n": 10,
  "photo": "http://static.findface.pro/sample.jpg"
}
```

Ответ

```
HTTP/1.1 200 OK
Date: Mon, 13 Jun 2016 12:23:56 GMT
Content-Type: application/json
Content-Length: [length]

{
  "results": {
    "[419, 236, 345, 311]": [
      {
        "confidence": 1,
        "face": {
          "galleries": ["default", "ppl"]
          "id": 316275,
          "meta": "Sam Berry",
          "photo": "http://static.findface.pro/sample.jpg",
          "photo_hash": "dc7ac54590729669ca869a18d92cd05e",
          "timestamp": "2016-07-01T12:18:27.477653",
          "x1": 236,
          "x2": 311,
          "y1": 345,
```

(continues on next page)

(продолжение с предыдущей страницы)


```
          "y2": 419
        }
      },
      {
        "confidence": 0.723975,
        "face": {
          "galleries": ["default", "ppl"]
          "id": 316283,
          "meta": "Sam Berry",
          "photo": "http://test.flexify.io/img/sample2.jpg",
          "photo_hash": "9b1dd93259fe87df122cd678ce95b9f9",
          "timestamp": "2016-07-01T13:19:36.376548",
          "x1": 78,
          "x2": 185,
          "y1": 114,
          "y2": 222
        }
      }
    ]
  }
}
```


9.2.4 Метод /face POST

Данный метод обрабатывает загруженное изображение или изображение в сети Интернет, обнаруживает на нем лица и добавляет их вместе с векторами признаков в базу биометрических данных. Если на изображении несколько лиц, то по умолчанию в базу данных добавляется только самое крупное. Вместе с лицом вы также можете добавить метаданные, которые являются уникальными для его обладателя, например, идентификатор или имя. Не рекомендуется назначать одинаковые метаданные разным людям. При необходимости вы можете указать имя галереи, в которую нужно добавить лицо помимо галереи по умолчанию.

Параметры:

- `photo`: загруженная фотография или публичный URL фотографии.
- `meta` [опционально]: пользовательские метаданные.
- `bbox` [опционально]: массив рамок, содержащих лица на изображении.
- `mf_selector` [опционально]: задает поведение Сервера при наличии нескольких лиц на фотографии или внутри заданного прямоугольника. Возможные значения:
  - `"reject"`: возвращает ошибку, если хотя бы на одной из фотографий присутствует более одного лица.
  - `"biggest"` [по умолчанию]: проверяет самое большое лицо на фотографии.
  - `"all"`: добавляет все лица, обнаруженные на фотографии. Имейте в виду, что в этом случае метаданные будут одинаковыми для всех лиц.
- `galleries` [опционально]: список имен галерей, в которые необходимо добавить лицо.
- `cam_id` [опционально]: UUID видеокамеры, от которой пришло изображение лица.

Возвращает:

- Представление данных добавленного лица в JSON или ошибку при добавлении.
- В случае если на изображении было обнаружено несколько лиц и значение `mf_selector="reject"`, метод возвращает код ошибки `400` (`Bad Request`), а также список координат рамок для каждого обнаруженного лица.

Пример №1

Запрос

```
POST /v0/face/ HTTP/1.1
Host: 127.0.0.1
Authorization: Token e93437ccdae66d57a45a5c6d9aa7602e
Content-Type: application/json
Content-Length: [length]

{
  "meta": "Sam Berry",
  "photo": "http://static.findface.pro/sample.jpg",
  "galleries": ["gal1", "niceppl"]
}
```

Ответ

```
HTTP/1.1 200 OK
Date: Mon, 13 Jun 2016 06:04:02 GMT
Content-Type: application/json; charset=UTF-8
Content-Length: [length]

{
  "results": [
    {
      "galleries": ["default", "gal1", "niceppl"]
      "id": 2334,
      "meta": "Sam Berry",
      "photo": "http://static.findface.pro/sample.jpg",
      "photo_hash": "dc7ac54590729669ca869a18d92cd05e",
      "timestamp": "2016-06-13T11:11:29.425339",
      "x1": 225,
      "x2": 307,
      "y1": 345,
      "y2": 428
    }
  ]
}
```

Пример №2

Запрос

```
POST /v0/face/ HTTP/1.1
Host: 127.0.0.1
Authorization: Token e93437ccdae66d57a45a5c6d9aa7602e
Content-Type: application/json
Content-Length: [length]

{
  "mf_selector": "reject",
  "photo": "http://static.findface.pro/sample-multiface.jpg"
}
```

Ответ

```
HTTP/1.1 400 Bad Request
Date: Mon, 13 Jun 2016 06:04:02 GMT
Content-Type: application/json; charset=UTF-8
Content-Length: [length]

{
  "code": 400,
  "faces": [
    {
      "x1": 1952,
      "x2": 2137,
      "y1": 838,
      "y2": 1023
    },
    {
      "x1": 1766,
      "x2": 1952,
      "y1": 1312,
      "y2": 1498
    },
    {
      "x1": 1385,
      "x2": 1540,
      "y1": 939,
      "y2": 1094
    },
    {
      "x1": 2452,
      "x2": 2607,
      "y1": 664,
      "y2": 818
    },
    {
      "x1": 1609,
      "x2": 1764,
      "y1": 767,
      "y2": 922
    }
  ],
  "reason": "Too many faces: 5"
}
```

9.2.5 Метод /face/id/<id> GET

Данный метод возвращает подробную информацию о лице с заданным id.

Параметры:

- Отсутствуют.

Возвращает:

- Представление данных лица с определенным id в JSON.

Пример

Запрос

```
GET /v0/face/id/2333/ HTTP/1.1
Host: 127.0.0.1
Authorization: Token e93437ccdae66d57a45a5c6d9aa7602e
```

Ответ

```
HTTP/1.1 200 OK
Date: Mon, 13 Jun 2016 12:23:56 GMT
Content-Type: application/json
Content-Length: [length]

{
  "galleries": ["default", "ppl"]
  "id": 2333,
  "meta": "Sam Berry",
  "photo": "http://static.findface.pro/sample.jpg",
  "photo_hash": "dc7ac54590729669ca869a18d92cd05e",
  "timestamp": "2016-06-13T11:06:42.075754",
  "x1": 225,
  "x2": 307,
  "y1": 345,
  "y2": 428
}
```

9.2.6 Метод /face/id/<id> PUT

Данный метод используется для изменения значений полей объекта `лицо` с заданным id.

Параметры:

- `meta`: новые метаданные.
- `person_id`: уникальный идентификатор персоны в базе данных.

- `galleries`: словарь JSON с одной парой ключ-значение. Вы можете добавить, удалить или полностью изменить список пользовательских галерей, в которых хранится лицо: `{"add":["list","of","galleries"]}`, `{"del":["list","of","galleries"]}`, `{"set":["list","of","galleries"]}`.

Возвращает:

- Представление обновленных данных лица в JSON.

Пример

Запрос

```
PUT /v0/face/id/5/ HTTP/1.1
Host: 127.0.0.1
Authorization: Token e93437ccdae66d57a45a5c6d9aa7602e
Content-Type: application/json
Content-Length: [length]

{
  "meta": "Sam Berry #2"
}
```

Ответ

```
HTTP/1.1 200 OK
Date: Mon, 13 Jun 2016 12:23:56 GMT
Content-Type: application/json
Content-Length: [length]

{
  "id": 2333,
  "meta": "Sam Berry #2",
  "photo": "http://static.findface.pro/sample2.jpg",
  "photo_hash": "dc7ac54590729669ca869a18d92cd05e",
  "timestamp": "2016-06-13T11:06:42.075754",
  "x1": 225,
  "x2": 307,
  "y1": 345,
  "y2": 428
}
```

9.2.7 Метод /face/id/<id> DELETE

Данный метод удаляет лицо с заданным id.

Параметры:

- Отсутствуют.

Возвращает:

- HTTP 204 No Content в случае успеха или причину ошибки.

Пример

Запрос

```
DELETE /v0/face/id/2332/ HTTP/1.1
Host: 127.0.0.1
Authorization: Token ca7916cdac260628c411cb5d895dd566
Content-Length: 0
```

Ответ

```
HTTP/1.1 204 No Content
```

9.2.8 Метод /face/meta/<meta> GET

Возвращает список лиц с заданными метаданными. Имейте в виду, что данный метод чувствителен к регистру. Строка с метаданными должна быть в кодировке URL. Пробелы между словами в метаданных должны быть закодированы как %20.

Параметры:

- Отсутствуют.

Возвращает:

- Возвращает список лиц с заданными метаданными.

Пример

Запрос

```
GET /v0/face/meta/Sam%20Berry/ HTTP/1.1
Host: 127.0.0.1
Authorization: Token e93437ccdae66d57a45a5c6d9aa7602e
```

Ответ

```
HTTP/1.1 200 OK
Date: Mon, 13 Jun 2016 12:23:56 GMT
Content-Type: application/json
Content-Length: [length]
```

(continues on next page)

(продолжение с предыдущей страницы)


```
{
  "results": [
    {
      "galleries": ["default", "ppl"],
      "id": 2333,
      "meta": "Sam Berry",
      "photo": "http://static.findface.pro/sample.jpg",
      "photo_hash": "dc7ac54590729669ca869a18d92cd05e",
      "timestamp": "2016-06-13T11:06:42.075754",
      "x1": 225,
      "x2": 307,
      "y1": 345,
      "y2": 428
    },
    {
      "galleries": ["default", "ppl"],
      "id": 2378,
      "meta": "Sam Berry",
      "photo": "http://static.findface.pro/sample2.jpg",
      "photo_hash": "dc7ac54590729669ca869a18d92cd05e",
      "timestamp": "2016-06-13T11:06:42.075754",
      "x1": 46,
      "x2": 502,
      "y1": 472,
      "y2": 789
    }
  ]
}
```


9.2.9 Метод /faces GET

Параметры:

- Отсутствуют.

Возвращает:

- Данный метод возвращает список всех лиц в базе биометрических данных.

Пример

Запрос

```
GET /v0/faces/ HTTP/1.1
Host: 127.0.0.1
Authorization: Token e93437ccdae66d57a45a5c6d9aa7602e
```

Ответ

```
HTTP/1.1 200 OK
Date: Mon, 13 Jun 2016 12:23:56 GMT
Content-Type: application/json
Content-Length: [length]

{
  "results": [
    {
      "galleries": ["default", "ppl"]
      "id": 2333,
      "meta": "Sam Berry",
      "photo": "http://static.findface.pro/sample.jpg",
      "photo_hash": "dc7ac54590729669ca869a18d92cd05e",
      "timestamp": "2016-06-13T11:06:42.075754",
      "x1": 225,
      "x2": 307,
      "y1": 345,
      "y2": 428
    },
    {
      "galleries": ["default", "ppl"]
      "id": 2335,
      "meta": "",
      "photo": "http://static.findface.pro/sample2.jpg",
      "photo_hash": "9879efb38d2dae550460c9edb6f36982",
      "timestamp": "2016-06-13T11:34:57.275394",
      "x1": 8,
      "x2": 152,
      "y1": 406,
      "y2": 550
    }
  ]
}
```

9.2.10 Метод /faces/gallery/<gallery> GET

Возвращает список всех лиц в базе данных.

9.2.11 Метод /meta GET

Данный метод возвращает все уникальные строки с метаданными, хранящимися в базе данных. Для каждой строки возвращается одно из связанных с ней лиц. Для того чтобы получить все лица, связанные с той или иной строкой, используйте метод `GET /v0/face/meta/<meta>`.

Параметры:

- Отсутствуют.

Возвращает:

- Список объектов, содержащих уникальные строки с метаданными, количество лиц, связанных с той или иной строкой, JSON-представление данных первого лица, связанного со строкой.

Пример

Запрос

```
GET /v0/meta/ HTTP/1.1
Host: 127.0.0.1
Authorization: Token e93437ccdae66d57a45a5c6d9aa7602e
```

Ответ

```
HTTP/1.1 200 OK
Date: Mon, 13 Jun 2016 12:23:56 GMT
Content-Type: application/json
Content-Length: [length]

{
  "results": [
    {
      "count": 1,
      "face": {
        "galleries": ["default", "ppl"]
        "id": 2333,
        "meta": "Sam Berry",
        "photo": "http://static.findface.pro/sample.jpg",
        "photo_hash": "dc7ac54590729669ca869a18d92cd05e",
        "timestamp": "2016-06-13T11:06:42.075754",
        "x1": 225,
        "x2": 307,
        "y1": 345,
        "y2": 428
      },
      "meta": "Sam Berry"
    },
    {
      "galleries": ["default", "ppl"]
      "count": 15,
      "face": {
        "id": 2563,
        "meta": "Angelina Jolie",
        "photo": "http://static.findface.pro/sample2.jpg",
        "photo_hash": "dc7ac54590729669ca869a18d92cd05e",
        "timestamp": "2016-06-13T11:06:42.075754",
        "x1": 225,
        "x2": 307,
        "y1": 345,
        "y2": 428
      },
      "meta": "Angelina Jolie"
    }
```

(continues on next page)

(продолжение с предыдущей страницы)

```
  ]
}
```

9.2.12 Метод /galleries GET

Данный метод возвращает список всех существующих галерей в базе данных.

Возвращает:

- Словарь JSON со списком имен (id) галерей.

Пример

Запрос

```
GET /v0/galleries/ HTTP/1.1
Host: 127.0.0.1
Authorization: Token e93437ccdae66d57a45a5c6d9aa7602e
```

Ответ

```
HTTP/1.1 200 OK
Date: Mon, 13 Jun 2016 12:23:56 GMT
Content-Type: application/json
Content-Length: [length]

{
  "results": [
    "default",
    "test"
    "57bd75f941741d36ab4614a0",
    "57bd76a241741d377bf881ac",
  ]
}
```

9.2.13 Метод /galleries/<gallery> POST

Данный метод создает новую галерею с заданным именем. Имя галереи может содержать латинские буквы, числа, знак подчеркивания и минус ([a-zA-Z0-9_-]+) и не может быть длиннее 48 символов.

Параметры:

Отсутствуют.

Пример

Запрос

```
POST /v0/galleries/testgal HTTP/1.1
Host: 127.0.0.1
Authorization: Token e93437ccdae66d57a45a5c6d9aa7602e
Content-Type: application/json
```

Ответ

```
HTTP/1.1 201 Created
Date: Mon, 13 Jun 2016 06:04:02 GMT
```

9.2.14 Метод /galleries/<gallery> DELETE

Данный метод удаляет галерею и все лица в ней.

Возвращает:

- HTTP 204 No content.

Пример

Запрос

```
DELETE /v0/galleries/niceppl HTTP/1.1
Host: 127.0.0.1
Authorization: Token e93437ccdae66d57a45a5c6d9aa7602e
Content-Length: 0
```

Ответ

```
HTTP/1.1 204 No Content
```

9.2.15 Метод /docs GET

Данный метод возвращает список всех задокументированных версий API. Доступен без авторизации.

9.2.16 Метод /docs/<version> GET

Данный метод возвращает документацию к заданной версии API. Доступен без авторизации.

9.2.17 Метод /person/id/<id> GET

Параметры:

- Отсутствуют.

Возвращает:

- Представление данных человека с заданным `person_id` в JSON.

Пример

Запрос

```
GET    /person/history/id/2001 HTTP/1.1
Host:   127.0.0.1
Authorization:  Token   e93437ccdae66d57a45a5c6d9aa7602e
Content-Type:   application/json
Content-Length: [length]
{
    "cam_ids":    [1, 25, 26, 27],
    "start":  "2016-06-13T11:00:00.000000",
    "end":    "2016-06-14T11:00:00.000000"
}
```

Ответ

```
HTTP/1.1   200 OK
Date:   Mon,    13  Jun 2016    12:23:56    GMT
Content-Type:   application/json
Content-Length: [length]
{
    "results":
    [
        {
            "person_id":  2001,
            "face_id":    240344,
            "cam_id": 25,
            "meta":   "Sam   Berry",
    "screenshot":"https://static.findface.pro/57726179d6946f02f3763824/
↪dc7ac54590729669ca869a18d92cd05e_thumb.j
pg",
            "timestamp":  "2016-06-13T11:06:42.075754",
        },
        {
            "person_id":  2001,
            "face_id":    240422,
            "cam_id": 25,
            "meta":   "Sam   Berry",
            "screenshot": "https://static.findface.pro/57726179
d6946f02f3763824/dc7ac54590729669ca869a18d92cd05e_thumb.j
pg",
            "timestamp":  "2016-06-13T11:08:44.073452",
```

(continues on next page)

(продолжение с предыдущей страницы)

```
        }
    ]
}
```

9.2.18 Метод /history/search POST

Данный метод возвращает все события по видеокамере(ам), удовлетворяющие заданным условиям.

Параметры:

- `"person_id"` [опционально]: значение параметра `person_id` интересующего человека.
- `"cam_id"` [опционально]: массив id видеокамер.
- `"start"` [опционально]: время начала событий в формате ISO8601.
- `"end"` [опционально]: время конца событий в формате ISO8601.
- `"friend"` [опционально]: является ли человек «своим».
- `"limit"` [опционально]: количество записей на странице, по умолчанию 0 — не ограничено.

Возвращает:

- Список всех событий истории.
- `next_page`: URL к следующей странице результатов поиска (содержит путь и указатель на следующую порцию результатов). Если такого поля нет в ответе, значит, была выведена последняя страница.

Пример

Запрос

```
POST /v0/history/search    HTTP/1.1
Host:   127.0.0.1
Authorization:  Token   e93437ccdae66d57a45a5c6d9aa7602e
Content-Type:   application/json
Content-Length: [length]
{
    "limit": 2,
}
```

Ответ

```
HTTP/1.1   200 OK
Date:   Mon,    12  Oct 2016    12:23:56    GMT
Content-Type:   application/json
Content-Length: [length]
{
   "next_page": "/v0/history/search?max_id=4",
```

(continues on next page)

(продолжение с предыдущей страницы)


```
   "results":[
      {
         "friend":false,
         "meta":"",
         "photo_hash":"9fda49f2444f93c33ad8aa914e20e53b",
         "cam_id":"12345678123456781234567812345678",
         "person_id":8,
         "timestamp":"2016-10-11T14:36:27.450000",
         "photo":"",
         "id":20146,
         "y1":77,
         "x1":285,
         "x2":552,
         "y2":345
      },
      {
         "friend":false,
         "meta":"",
         "photo_hash":"dc7ac54590729669ca869a18d92cd05e",
         "cam_id":"12345678123456781234567812345678",
         "person_id":8,
         "timesamp":"2016-10-12T12:57:07.509000",
         "photo":"",
         "id":20147,
         "x1":236,
         "y1":345,
         "x2":311,
         "y2":419
      }
   ]
}
```


9.3 Запросы к пользовательским галереям

При установке FindFace Enterprise Server SDK в базе биометрических данных автоматически создается галерея с именем `default`. Галерею `default` нельзя удалить. Распознаваемые лица автоматически добавляются в данную галерею и не могут быть оттуда удалены.

В дополнение к галерее по умолчанию, вы можете создать неограниченное количество пользовательских галерей и добавлять лица также в них. Пользовательские галереи позволяют формировать списки лиц по нужным признакам и вести поиск только в них.

Для создания пользовательской галереи, используйте метод *POST /galleries/gallery_name*.

По умолчанию все методы API применяются к галерее по умолчанию. Однако для некоторых методов можно указать определенную пользовательскую галерею в URI запроса, к которой запрос и будет применен (см. таблицу ниже). Например, чтобы найти человека в галерее `ppl`, используйте метод `POST /faces/gallery/ppl/identify/` вместо `POST /identify/`.

Метод для галереи по умолчанию	Метод для пользовательской галереи
`POST /identify/`	`POST /faces/gallery/<Gallery>/identify/`
`GET /faces/`	`GET /faces/gallery/<Gallery>/`
`GET /face/meta/<Meta>`	`GET /face/gallery/<Gallery>/meta/<Meta>`
`GET /meta/`	`GET /meta/gallery/<Gallery>/`

9.4 Методы для работы с видеокамерами

Методы, приведенные в данном разделе, являются расширением *общих методов API* FindFace Enterprise Server SDK и предназначены для работы с видеокамерами.

В этом разделе:

9.4.1 Метод /camera POST

Описание

Данный метод создает новую видеокамеру на Сервере.

Параметры:

- `meta` [опционально]: пользовательская строка с метаданными видеокамеры
- `url` [опционально]: URL потока видеокамеры
- `detector` [опционально]: виртуальный детектор, соответствующий списку видеокамер.
- `rot` [W,H,X,Y] [опционально]: включает обнаружение и отслеживание лиц только внутри заданного прямоугольника (ROT, region of tracking).
- `roi` [W,H,X,Y] [опционально]: включает отправку на Сервер FindFace лиц, обнаруженных только внутри интересующей области (ROI, region of interest).

Возвращает:

Представление данных добавленной камеры в JSON или причину ошибки.

Пример

Запрос

```
POST /v0/camera/ HTTP/1.1
Host: 127.0.0.1
Authorization: Token 1234567890qwertyuiop
Content-Type:   application/json
Content-Length: [length]
```

(continues on next page)

(продолжение с предыдущей страницы)

```
{
    "meta": "test",
    "url": "http://test.com:1234/stream.flv",
    "detector": "detec1"
}
```

Ответ

```
HTTP/1.1 201 Created
Content-Length: [length]
Content-Type: application/json; charset=UTF-8
{
    "meta": "meta",
    "url": "http://test.com:1234/stream.flv",
    "detector": "detec1",
    "id": "7bb35e9d-9f4f-4e5b-8811-e1dded6de811"
}
```

9.4.2 Метод /camera GET

Описание

Данный метод возвращает список всех камер на Сервере.

Параметры:

Отсутствуют.

Возвращает:

Список всех камер.

Пример

Запрос

```
GET /v0/camera HTTP/1.1
Host: 127.0.0.1
Authorization: Token 1234567890qwertyuiop
```

Ответ

```
HTTP/1.1 200 OK
Content-Length: [length]
Date: Thu, 13 Oct 2016 12:14:22 GMT
Content-Type: application/json; charset=UTF-8
```

(continues on next page)

(продолжение с предыдущей страницы)

```
[
    {
        "meta": "firstcam",
        "url": "http://192.168.133.37:1234/stream.flv"
        "id": "34ba07c4-0677-4d5c-9946-62c625cd7127"
    },
    {
        "meta": "newinfo",
        "url": "http://5.6.7.8:1234/stream.flv",
        "id": "b28a898b-6334-4d37-8888-c9dd858ddc47"
    },
    ...
]
```

9.4.3 Метод /camera/<camera_id> GET

Описание

Данный метод возвращает данные заданной видеокамеры с `id = camera_id`.

Параметры:

Отсутствуют.

Возвращает:

Информацию о камере или причину ошибки.

Пример

Запрос

```
GET /v0/camera/b28a898b-6334-4d37-8888-c9dd858ddc47 HTTP/1.1
Host: 127.0.0.1
Authorization: Token 1234567890qwertyuiop
```

Ответ

```
HTTP/1.1 200 OK
Content-Length: [length]
Content-Type: application/json; charset=UTF-8
{
    "meta": "test info",
    "url": "http://5.6.7.8:1234/stream.flv",
    "id": "b28a898b-6334-4d37-8888-c9dd858ddc47"
}
```

9.4.4 Метод /camera/<camera_id> PUT

Описание

Данный метод используется для изменения метаданных заданной камеры с `id = camera_id`.

Параметры:

- `meta` [опционально]: новая строка с метаданными
- `url` [опционально]: URL потока видеокамеры
- `rot` [W,H,X,Y] [опционально]: включает обнаружение и отслеживание лиц только внутри заданного прямоугольника (ROT, region of tracking). Если вы используете ROT, обязательно передавайте этот параметр на камеру каждый раз, когда вы отправляете запрос PUT, поскольку если данный параметр в запросе PUT отсутствует или его значение не задано, ROT на камере будет удален.
- `roi` [W,H,X,Y] [опционально]: включает отправку на Сервер FindFace лиц, обнаруженных только внутри интересующей области (ROI, region of interest). Если вы используете ROI, обязательно передавайте этот параметр на камеру каждый раз, когда вы отправляете запрос PUT, поскольку если данный параметр в запросе PUT отсутствует или его значение не задано, ROI на камере будет удален.

Возвращает:

Представление данных камеры в JSON.

Пример №1

Запрос

```
PUT /v0/camera/b28a898b-6334-4d37-8888-c9dd858ddc47 HTTP/1.1
Host: 127.0.0.1
Authorization: Token 1234567890qwertyuiop
Content-Type: application/json
Content-Length: [length]
{
    "meta": "newinfo",
    "url": "http://zzzz.com:1234/stream.flv"
}
```

Ответ

```
HTTP/1.1 200 OK
Content-Length: [length]
Content-Type: application/json; charset=UTF-8
{
    "url": "http://zzzz.com:1234/stream.flv",
    "id": "b28a898b-6334-4d37-8888-c9dd858ddc47",
    "meta": "newinfo"
}
```

Пример №2

Запрос

```
PUT /v0/camera/b28a898b-6334-4d37-8888-c9dd858ddc47 HTTP/1.1
Host: 127.0.0.1
Authorization: Token 1234567890qwertyuiop
Content-Type: application/json
Content-Length: [length]
{
    "rot": [
      120,
      120,
      35,
      50
    ],
    "roi": [
      100,
      100,
      40,
      50
    ]
}
```

Ответ

```
HTTP/1.1 200 OK
Content-Length: [length]
Content-Type: application/json; charset=UTF-8
{
    "id": "b28a898b-6334-4d37-8888-c9dd858ddc47",
    "rot": [
      120,
      120,
      35,
      50
    ],
    "roi": [
      100,
      100,
      40,
      50
    ]
}
```

9.4.5 Метод /camera/<camera_id> DELETE

Описание

Данный метод удаляет заданную камеру с `id = camera_id`.

Параметры:

Отсутствуют.

Возвращает:

HTTP 204 No Content в случае успеха или причину ошибки.

Пример

Запрос

```
DELETE /v0/camera/b28a898b-6334-4d37-8888-c9dd858ddc47 HTTP/1.1
Host: 127.0.0.1
Authorization: Token 1234567890qwertyuiop
Content-Length: 0
```

Ответ

```
HTTP 204 No Content
```

Совет: Вы также можете ознакомиться с документаций REST API на нашем сайте и во встроенном фреймворке `http://<facenapi_ip>:8000/v1/docs`.

Глава 10

Обслуживание и устранение неисправностей

10.1 Устранение неполадок с лицензией и NTLS

При устранении неполадок с лицензией и сервером NTLS первым шагом является получение информации о лицензии и статусе сервера. Это можно сделать, отправив API-запрос в 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.

10.1.1 Получение информации о лицензии

Для получения информации о лицензии FindFace Enterprise Server SDK и статусе NTLS, выполните в консоли сервера 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.

```
{
  "name": "NTLS",
  "time": 1504794255,
  "type": "online",
  "license_id": "2e46fed81cc843539f0cf8bd4c1df254",
  "generated": 1503571034,
  "last_updated": 3,
  "valid": {
    "value": true,
    "description": ""
  },
  "source": "/ntech/license/import_
↪803e10f14948d5e8a7583de99b0411635743a01cd7afd8589c475f5b60e202cb.lic",
  "limits": [
    {
      "type": "time",
      "name": "end",
      "value": 4753938994
    },
    {
      "type": "number",
      "name": "faces",
      "value": 1000000000000,
      "current": 80037
    },
    {
      "type": "number",
      "name": "cameras",
      "value": 4294967295,
      "current": 2
    },
    {
      "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": "FindFace-tarantool",
      "ip": "127.0.0.1:37058"
    },
```


(continues on next page)

(продолжение с предыдущей страницы)

```
    {
      "name": "findface-nnapi",
      "ip": "127.0.0.1:37057"
    },
    {
      "name": "findface-extraction-api",
      "ip": "127.0.0.1:37056"
    },
    {
      "name": "fkvideo-detector",
      "ip": "127.0.0.1:37059"
    }
  ]
}
```

10.2 Анализ логов

Активность каждого компонента FindFace Enterprise Server SDK записывается в лог-файл.

findface-facenapi

```
sudo tail -f /var/log/syslog | grep facenapi
Jun 28 17:20:08 ubuntu findface-facenapi[17104]: [I 170628 17:20:08 base:234] 200 POST /v0/face␣
↪(127.0.0.1) 1114 487 241 12
```

Лог `findface-facenapi` содержит следующие значения времени:
1114 — общее время ответа (компоненты FindFace Enterprise Server SDK + MongoDB + Python),
487 — время обнаружения лица,
241 — время ответа `findface-nnapi`,
12 — время ответа `tntapi`.

findface-nnapi

```
sudo tail -f /var/log/syslog | grep nnapi
Jul  7 03:53:05 ubuntu findface-nnapi[49606]: (2017-07-07 10:53:05) [INFO    ] Request: 127.0.0.
↪1:34494 0x7fb100000960 HTTP/1.0 POST /facen
Jul  7 03:53:06 ubuntu findface-nnapi[49606]: (2017-07-07 10:53:06) [INFO    ] Response:␣
↪0x7fb100000960 /facen?x2=0&y1=0&x1=0&y2=0 200 0
```

fkvideo_detector

```
sudo tail -f /var/log/syslog | grep fkvideo_detector
```

extraction-api

```
sudo tail -f /var/log/syslog | grep extraction-api
```

Сервис, сбалансированный по нагрузке

```
sudo tail -f  /var/log/nginx/service_name.access_log
sudo tail -f  /var/log/nginx/nnapi.access_log
```

Tarantool

```
sudo cat /var/log/tarantool/FindFace.log
```

10.3 Миграция на другой детектор или модель

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

Иногда вам может потребоваться миграция экземпляра FindFace Enterprise Server SDK на другой детектор лиц или модель нейронной сети. Как правило, данная процедура необходима при обновлении до последней версии продукта.

Совет: Сводные сведения по моделям см. *здесь*.

При смене детектора лиц вам следует повторно сгенерировать нормализованные изображения и миниатюры лиц, а также векторы признаков.

Предупреждение: Различные детекторы лиц обладают разной чувствительностью к чертам лица. Имейте в виду, что новый детектор может пропустить ранее найденные лица.

В этом разделе:

10.3.1 Утилиты

Для миграции экземпляра FindFace Enterprise Server SDK вам потребуются следующие утилиты:

Утилита	Описание
`findface-regenerate`	Скрипт, который повторно генерирует и перезаписывает данные лиц в MongoDB, применяя новые настройки детектора или модель нейронной сети к изображениям в папке `Uploads`.
`mongo2searchapi`	Скрипт, который копирует перегенерированные векторы признаков из MongoDB в Tarantool.

Обе утилиты автоматически устанавливаются вместе с компонентом *findface-facenapi*.

10.3.2 Требования

Папка `/var/lib/ffupload/uploads/` (`Uploads`) должна быть заполнена как минимум исходными изображениями. Ее содержимое можно просмотреть в браузере по адресу `http://<findface_upload_IP:3333/uploads/`.

В общем случае утилита `findface-regenerate` работает с папкой `Uploads` следующим образом:

Случай использования	Принцип работы
Смена настроек детектора	Утилита `findface-regenerate` прогоняет исходные изображения через компоненты `facenapi` и `nnapi` с новыми настройками детектора [и модели] и возвращает перегенерированные нормализованные изображения лиц, миниатюры лиц и векторы признаков.
Смена модели	Утилита `findface-regenerate` прогоняет нормализованные изображения лиц через компонент `nnapi` с новыми настройками модели и возвращает перегенерированные векторы признаков.

10.3.3 Повторная генерация артефактов

Важно: Перед проведением любых операций с базой данных MongoDB обязательно создайте резервную копию.

Примените утилиту `findface-regenerate` следующим образом:

1. Перейдите в папку `/usr/bin/`. Отобразите и внимательно изучите справку по утилите `findface-regenerate`:

```
cd /usr/bin/
findface-regenerate --help
```

```
## findface-regenerate --help

Usage: /usr/bin/findface-regenerate [OPTIONS]

Options:
```

(continues on next page)

(продолжение с предыдущей страницы)

```
 --help                          show this help information

/usr/lib/python3/dist-packages/facenapi/core/decoders/decoder_threaded.py options:

 --max-size                      Maximum allowed photo width/height (default
                                 4000)

/usr/lib/python3/dist-packages/facenapi/core/detectors/detector_dlib.py options:

 --dlib-adjust-threshold         Adjust face detector threshold (default 0.0)
 --dlib-max-size                 images with width or height larger than
                                 dlib_max_size will be scaled down before
                                 being fed into detector (default 1200)
 --dlib-normalizer               path to normalizer data (default
                                 /usr/share/findface-data/normalizer.dat)

/usr/lib/python3/dist-packages/facenapi/core/detectors/detector_nnd.py options:

 --nnd-max-face-size             Maximum face size in pixels (no limit if 0)
                                 (default 0)
 --nnd-min-face-size             Minimum face size in pixels (default 30.0)
 --nnd-o-net-thresh               (default 0.9)
 --nnd-p-net-thresh               (default 0.5)
 --nnd-r-net-thresh               (default 0.5)
 --nnd-scale-factor               (default 0.79)
 --nnd-workers                   Number of detector workers threads. (0 - as
                                 much as there are cpus) (default 0)

/usr/lib/python3/dist-packages/facenapi/core/main_utils.py options:

 --decoder                       Image decoder (threaded) (default threaded)
 --detector                      Face detector (dlib,nnd) (default nnd)
 --extractor                     Feature extractor (nnapi,extraction-api)
                                 (default nnapi)
 --facen-storage                 Feature vector storage
                                 (searchapi_replicated,tntapi,searchapi)
                                 (default tntapi)
 --id-generator                  Face id generator (tntime,mongo) (default
                                 tntime)

/usr/lib/python3/dist-packages/facenapi/server/context.py options:

 --fetch-proxy                   Fetch images from urls via proxy, ex:
                                 http://1.2.3.4:3128
 --ffupload-url                  url (without path) to PUT images uploaded to
                                 /face, ex: http://127.0.0.1:1234
 --friend-count                   (default 5)
 --friend-interval                (default 604800)
 --gae                           enable Gender, Age and Emotions support
                                 (default False)
 --mongo-host                    mongo database host (default localhost)
 --mongo-port                    mongo database port (default 27017)
 --person-identify               identify persons (default False)
 --person-identify-global        identify persons across all cameras (default
                                 False)
 --person-identify-threshold     threshold for persons identify (default
```

(continues on next page)

(продолжение с предыдущей страницы)

```
                                0.75)
 --upload-path                  path of $ffupload_url (default uploads)

/usr/lib/python3/dist-packages/facenapi/server/regenerate_facens.py options:

 --config                       path to config file
 --coroutines                   Number of parallel coroutines (default 30)
 --every-other                   (default 1)
 --every-other-offset            (default 0)
 --facen-size                   Facen size in number of floats. (facens of
                                this sizes are not regenerated when smart
                                regeneration is enabled) (default -1)
 --max-id                       Maximum id (inclusive)
 --min-id                       Minimum id (inclusive)
 --regenerate                   What to regenerate: facens, thumbs,
                                normalized (comma-separated). (default
                                facens)

/usr/lib/python3/dist-packages/tornado/log.py options:

 --log-file-max-size            max size of log files before rollover
                                (default 100000000)
 --log-file-num-backups         number of log files to keep (default 10)
 --log-file-prefix=PATH         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          The interval value of timed rotating
                                (default 1)
 --log-rotate-mode              The mode of rotating files(time or size)
                                (default size)
 --log-rotate-when              specify the type of TimedRotatingFileHandler
                                interval other options:('S', 'M', 'H', 'D',
                                'W0'-'W6') (default midnight)
 --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=debug|info|warning|error|none
                                Set the Python log level. If 'none', tornado
                                won't touch the logging configuration.
                                (default info)
```

2. Для смены настроек детектора раскомментируйте и отредактируйте относящиеся к детектору параметры в файле конфигурации `findface-facenapi`.

```
sudo vi /etc/findface-facenapi.ini

detector                        = 'nnd'
...
```

3. Для смены *биометрической модели* лица, отредактируйте параметр `model_facen` в файле конфигурации `findface-nnapi`:

```
sudo vi /etc/findface-nnapi.ini
```

(continues on next page)

(продолжение с предыдущей страницы)

```
model_facen = apricot_320
```

4. Настройте `findface-regenerate`, используя аргументы командной строки, как описано в справке по утилите. Например, для того чтобы сменить детектор лиц, выполните из папки `/usr/bin` следующую команду:

```
sudo findface-regenerate --regenerate=normalized,thumbs,facens --config=/etc/findface-
↪facenapi.ini
```

Для смены модели выполните команду:

```
sudo findface-regenerate --regenerate=facens --config=/etc/findface-facenapi.ini
```

10.3.4 Копирование векторов признаков из MongoDB в Tarantool

Примените утилиту `mongo2searchapi` следующим образом:

1. Создайте резервную копию БД Tarantool.
2. Остановите Tarantool.

```
sudo systemctl stop tarantool@FindFace*
```

3. Удалите файлы снапшотов `.snap`, логов `.xlog` и *быстрого индеса* `.idx` для всех шардов `tntapi`.

Совет: По умолчанию, данные файлы хранятся в следующих папках:

- Одиночная инсталляция:
  - `/opt/ntech/var/lib/tarantool/default/snapshots`
  - `/opt/ntech/var/lib/tarantool/default/xlogs`
  - `/opt/ntech/var/lib/tarantool/default/index`
- Кластерная инсталляция:
  - `/opt/ntech/var/lib/tarantool/shard_`*`N`*`/snapshots`
  - `/opt/ntech/var/lib/tarantool/shard_`*`N`*`/xlogs`
  - `/opt/ntech/var/lib/tarantool/shard_`*`N`*`/index`

---

4. Если векторы признаков в прежней и новой модели *различаются по размеру*, отредактируйте размер вектора признаков в разделе `FindFace.start` файла конфигурации Tarantool `/etc/tarantool/instances.enabled/FindFace_`*`shard_N`*`.lua`. Повторите для каждого шарда.

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

FindFace.start("127.0.0.1", 8001, {license_ntls_server="127.0.0.1:3133", facen_size = 320})
```

5. Запустите `mongo2searchapi` на сервере с `findface-facenapi`:

```
sudo python3 -m facenapi.server.tools.mongo2searchapi --config=/etc/findface-facenapi.ini
```

6. Запустите Tarantool.

```
sudo systemctl start tarantool@FindFace*
```

10.4 Обновление до последней версии

В этом разделе:

10.4.1 Обновление с сохранением данных

Для обновления FindFace Enterprise Server SDK до последней версии с сохранением данных, выполните следующие действия:

1. Остановите все сервисы FindFace Enterprise Server SDK:

```
sudo service 'findface*' stop
sudo service 'fkvideo*' stop
sudo service 'ntls' stop
sudo service 'nginx*' stop
sudo service 'tarantool*' stop
sudo service 'mongod*' stop
```

2. Подготовьте новые пакеты на каждом из серверов развертывания.
3. Выполните апгрейд сервисов командой:

```
sudo apt-get update
sudo apt-get upgrade
```

4. Запустите сервисы FindFace Enterprise Server SDK.

```
sudo service 'findface*' start
sudo service 'fkvideo*' start
sudo service 'ntls' start
sudo service 'nginx*' start
sudo service 'tarantool*' start
sudo service 'mongod*' start
```

5. При необходимости *переведите* FindFace Enterprise Server SDK на новый детектор или модель нейронной сети.

10.4.2 Полная переустановка

Для того чтобы полностью переустановить FindFace Enterprise Server SDK, выполните следующие действия:

1. *Удалите* прежний экземпляр со всеми добавленными лицами.
2. Разверните последнюю версию продукта, следуя стандартной *процедуре*.

10.5 Удаление экземпляра продукта

Вы можете автоматически удалить FindFace Enterprise Server SDK и при необходимости базы данных MongoDB и Tarantool с помощью скрипта `ffserver_uninstall.sh`. Выполните следующие действия:

1. Загрузите скрипт `ffserver_uninstall.sh` в любую папку на сервере установки (например, в `/home/username/`).
2. Из данной папки сделайте скрипт `ffserver_uninstall.sh` исполняемым.

```
chmod +x ffserver_uninstall.sh
```

3. Запустите скрипт `ffserver_uninstall.sh`.

```
sudo ./ffserver_uninstall.sh
```

4. Отвечая на вопросы интерактивного мастера удаления, выберите между полным удалением FindFace Enterprise Server SDK со всеми добавленными лицами и удалением с сохранением данных.

10.6 Устранение неполадок при работе с папкой Uploads

Неполадки при работе компонента `findface-upload` приводят к недоступности содержимого папки `Uploads` на странице `http://<findface_upload_IP:3333/uploads/` и в *веб-интерфейсе FindFace*.

Примечание: Папка `Uploads` содержит обработанные Сервером исходные изображения и артефакты Сервера, такие как миниатюры и нормализованные изображения лиц.

В этом разделе:

10.6.1 Отображение содержимого папки Uploads в веб-интерфейсе FindFace

Проблема: Исходные изображения, миниатюры лиц и нормализованные изображения лиц не отображаются в веб-интерфейсе FindFace после изменения IP-адреса сервера `findface-upload`.

Каждый объект типа `лицо` в базе данных *MongoDB* содержит следующие ссылки на папку `Uploads`:

- Ссылка на соответствующее исходное изображение
- Ссылки на соответствующие артефакты Сервера FindFace: миниатюру лица и нормализованное изображение лица

При изменении IP-адреса сервера `findface-upload` ссылки на папку `Uploads` теряют актуальность и исходные изображения и артефакты больше не отображаются в *веб-интерфейсе*.

Для решения проблемы отредактируйте ссылки в полях `photo`, `thumbnail` и `normalized` всех объектов типа `лицо` в MongoDB следующим образом:

1. В консоли перейдите в MongoDB и затем в базу данных `facenapi`.

```
mongo
use facenapi
```

2. Вызовите случайный объект типа `лицо`, чтобы убедиться, что прежний IP-адрес все еще используется в полях `photo`, `normalized` и `thumbnail` (`127.0.0.1` в примере).

```
db.faces.findOne()

{
        "_id" : NumberLong("3871027550645276"),
        "y2" : 383,
        "x2" : 397,
        "x1" : 84,
        "y1" : 71,
        "facen" : BinData(0,"CKftuU5t6j+...+tdKDOE1M29="),
        "gender" : "female",
        "age" : 38.75063705444336,
        "emotions" : [
                "neutral",
                "sad"
        ],
        "meta" : "",
        "photo_hash" : "6209c1a017972f8b18fada3f9e4d2768",
        "timestamp" : ISODate("2017-12-01T09:22:16.950Z"),
        "gallery" : [
                "default"
        ],
        "person_id" : 13,
        "friend" : false,
        "owner" : ObjectId("5a0e96928acdc01dab404bdd"),
        "photo" : "http://127.0.0.1:3333/uploads/5a0e96928acdc01dab404bdd/20171201/
↪3871027550645276_92fc8aa39973_photo.jpeg",
        "normalized" : "http://127.0.0.1:3333/uploads/5a0e96928acdc01dab404bdd/20171201/
↪3871027550645276_41ec18ba44cd_norm.png",
        "thumbnail" : "http://127.0.0.1:3333/uploads/5a0e96928acdc01dab404bdd/20171201/
↪3871027550645276_3bc9e34b60aa_thumb.jpeg"
}
```

3. Примените скрипт замены IP-адреса к полям `photo`, `normalized` и `thumbnail` объектов типа `лицо`. В примере IP-адрес `127.0.0.1` заменяется на `192.168.2.158`.

```
db.faces.find().forEach(function(e,i) {     e.photo=e.photo.replace("//127.0.0.1","//192.168.
↪2.158"); e.normalized=e.normalized.replace("//127.0.0.1","//192.168.2.158"); e.thumbnail=e.
↪thumbnail.replace("//127.0.0.1","//192.168.2.158");     db.faces.save(e); });
```

4. Вызовите случайный объект типа `лицо` еще раз, чтобы убедиться, что IP-адрес был успешно изменен.

```
db.faces.findOne()

...
"photo" : "http://192.168.2.158:3333/uploads/5a0e96928acdc01dab404bdd/20171201/
↪3871027550645276_92fc8aa39973_photo.jpeg",
"normalized" : "http://192.168.2.158:3333/uploads/5a0e96928acdc01dab404bdd/20171201/
↪3871027550645276_41ec18ba44cd_norm.png",
"thumbnail" : "http://192.168.2.158:3333/uploads/5a0e96928acdc01dab404bdd/20171201/
↪3871027550645276_3bc9e34b60aa_thumb.jpeg"
```

(continues on next page)

(продолжение с предыдущей страницы)

```
...
```

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 SDK.

Тип нейронной сети	Имя	В использовании	Размер вектора признаков
Биометрия лица	`model_36`	2016	160
	`model_39c`	2016	160
	`fr_1`	2016-05/04/2017	160
	`en_1`	2016-03/03/2017	320
	`en2_face0`	с 14/03/2017	320
	`apricot_160f`	с 31/07/2017	160
	`apricot_320`	с 31/07/2017	320
	`banana_800f`	с 15/09/2017	800
Распознавание пола	`fr_1_gender0`	с 05/04/2017	N/A
Распознавание возраста	`fr_1_age0`	с 05/04/2017	N/A
Распознавание эмоций	`model_39c_em`	05/04/2017- 11/08/2017	N/A
	`emotion_1`	since 11/08/2017	N/A