УТВЕРЖДЕН
ru.red-soft.00001-01 39 01

ГЕМБАФЕЙС (GEMBAFACE)

Руководство пользователя

ru.red-soft.00001-01 39 01

Листов 19

Инв. № подл.	Подп. и дата	Взам. инв. №	Инв. № дубл.	Подп. и дата

АННОТАЦИЯ

ГЕМБАФЕЙС (GEMBAFACE) это программа для распознавания лиц. Предназначен для формирования баз данных поисковых образов (биометрических данных) по фотографиям людей и последующего поиска по предъявленной фотографии. Программа предполагает информационное взаимодействие со смежными информационными системами по технологии REST API.

Настоящий документ содержит сведения для эксплуатации программы. Установлен общий порядок подключения и эксплуатации программы. Описаны методы API основного сервиса программы и приведены соответствующие примеры.

СОДЕРЖАНИЕ

1. НАЗНАЧЕНИЕ И УСЛОВИЯ ПРИМЕНЕНИЯ ПРОГРАММЫ

1.1. Общая информация

Программа ГЕМБАФЕЙС представляет собой набор специальных сервисов. Работа основного сервиса, с которым производится взаимодействие клиентского ПО, производится по протоколу HTTP. Программа позволяет сформировать на основе фотографий лиц людей базу с биометрическими данными людей изображенными на фотографиях для последующего поиска в данной базе людей по фотографиям, а также определения степени близости биометрических данных.

1.2. Функции программы

К основным функциям программы ГЕМБАФЕЙС относятся:

1) Получение поискового образа из фотографий людей.
2) Формирование базы данных поисковых образов.
3) Поиск по базе данных поисковых образов.
4) Сравнение двух фотографий людей.
5) Сравнение двух лиц на одной фотографии.

1.3. Сценарии

Программа позволяет организовать различные сценарии использования распознавания:

1) Черный список.
2) Белый список.
3) Поиск.
4) Верификация.
5) Идентификация.
6) прочее...

1.4. Системы и задачи

Программа обеспечивает широкие направления использования в различных системах и задачах:

1) Системы безопасности (СКУД, СОВН, ...).
2) Системы биометрической аутентификации.
3) Как часть системы валидации документов (паспорт, права...).

4) В системах для оперативно-розыскной деятельности.
5) В торговле, для улучшения качества обслуживания.
6) ...

1.5. Методы программы

Программа содержит методы для:
1) Ведения (создание, удаление, вывод списка) записей о персонах;
2) Ведения (создание, удаление, вывод списка) записей о фотографиях персоны;
3) Поиска персоны по предъявленной фотографии.

2. ХАРАКТЕРИСТИКИ ПРОГРАММЫ

2.1. ГЕМБАФЕЙС программа, представляет собой набор сервисов, относится к служебным программам, к программам для разработки конечных продуктов. Предполагается одновременная работа сервисов ГЕМБАФЕЙС со смежной системой или клиентским сервисом.

2.2. ГЕМБАФЕЙС позволяет для смежной информационной системы создать до 10 000 000 записей о персонах и до 10 000 записей о фотографии персоны для каждой персоны.

2.3. Фотографии персоны не хранятся в информационных ресурсах программы ГЕМБАФЕЙС, а используются для вычисления поискового образа (вектора) персоны на фотографии.

2.4. Ожидается, что на одной фотографии присутствует одна персона. В случае наличия нескольких персон, выбирается персона, лицо которой имеет наибольшую площадь на изображении.

2.5. В записях о фотографии персоны сохраняется сопроводительная информация о положении лица на фотографии и внешний идентификатор фотографии (для вывода изображения и рамки на нем в клиентской части). Предполагается, что хранилище фотографий находится в информационных ресурсах смежной информационной системы.

2.6. Рекомендуемый размер фотографий 1024x768 в формате JPEG или PNG размером до 2 Мб.

2.7. Обмен между ГЕМБАФЕЙС и смежной информационной системой осуществляется по протоколу HTTP по технологии REST.

3. АУТЕНТИФИКАЦИЯ

3.1. Все http запросы на сервис ГЕМБАФЕЙС требуют авторизации. Авторизация запросов происходит проверкой токена в заголовке `X-Auth-Token` http-запросов. Конечным пользователям продукта, для получения действующего токена, необходимо связываться с администраторами сервиса.

3.2. Для администраторов системы: токен авторизации настраивается в конфигурационном файле сервиса gemba-face *face.sdl* в поле *token*.

4. ЗАПРОСЫ К ПРОГРАММЕ

4.1. Основной сценарий использования программы (основного сервиса) следующий:

1) первичное создание записей о персонах и записей о фотографиях;

2) добавление/удаление записей о персонах и записей о фотографиях по мере изменения соответствующих информационных массивов в смежной информационной системе;

3) поиск персон по предъявленной фотографии.

4.2. Первичное создание записей о персонах и записей о фотографиях связано со следующими действиями:

1) формирование на стороне смежной информационной системы информационного массива записей о персонах, включающих уникальный для данной системы идентификатор персоны, который будет использоваться для идентификации записи о персоне в программе;

2) формирование на стороне смежной информационной системы информационного массива записей о фотографиях каждой персоны, включающих уникальный для данной персоны идентификатор фотографии, который будет использоваться для идентификации записи о фотографии в программе;

3) вызов метода создания записи о персоне (на стороне ГЕМБАФЕЙС);

4) вызов метода создания записи о фотографии (на стороне ГЕМБАФЕЙС).

4.3. Добавление/удаление записей о персонах и записей о фотографиях по мере изменения соответствующих информационных массивов в смежной информационной системе связано со следующими действиями:

1) вызов метода создания записи о персоне (на стороне ГЕМБАФЕЙС) в момент добавления соответствующей записи в смежную информационную систему;

2) вызов метода удаления записи о персоне (на стороне ГЕМБАФЕЙС) в момент удаления соответствующей записи в смежной информационной системе;

3) вызов метода создания записи о фотографии (на стороне ГЕМБАФЕЙС) в момент добавления соответствующей записи в смежную информационную систему;

4) вызов метода удаления записи о фотографии (на стороне ГЕМБАФЕЙС) в момент удаления соответствующей записи в смежной информационной системе;

4.4. Для целей проверки соответствия информационных массивов ГЕМБАФЕЙС и смежной информационной системы доступны следующие методы:

1) метод вывода списка записей о персонах;

2) метод вывода списка записей о фотографиях персоны.

4.5. Поиск персоны по предъявленной фотографии осуществляется вызовом соответствующего метода и получением в ответ списка похожих персон в порядке степени похожести. Длина списка и порог похожести передается в параметрах вызова.

5. ВХОДНЫЕ И ВЫХОДНЫЕ ДАННЫЕ

5.1. Получить список персон (GET /persons)

5.1.1. Запрос

```
Content-Type - application/json

Параметры QUERY:

    <limit> - Количество запрашиваемых записей
    <offset> - Отступ от начала списка
```

5.1.2. Ответ

```
Content-Type - application/json

Пример ответа (JSON):
```

```
{
    "persons":[{
        "id": "sdfsdf",
        "tags": ["one", "two"]
    },{
        "id": "sdfsdd",
        "tags": ["one", "two"]
    }],
    "meta":{
        "offset":0,
        "total":5,
        "limit":5
    }
}
```

5.2. Получение сведений о персоне (GET *persons*<pid>)

5.2.1. Запрос

```
Content-Type - application/json

Параметры URL:

    <pid> - Идентификатор персоны
```

5.2.2. Ответ

```
Content-Type - application/json

Пример ответа (JSON):
```

```
{
    "id": "test",
    "tags": ["one", "two"]
}
```

5.3. Добавить в базу новую персону (POST /persons)

5.3.1. Запрос

```
Content-Type - application/json

Содержание запроса (BODY):

{
    "id": "test",
    "tags": ["one", "two"]
}
```

5.3.2. Ответ

```
Content-Type - application/json

Пример ответа (JSON):

{
    "id": "test",
    "tags": ["one", "two"]
}
```

5.4. Удалить персону из базы (DELETE *persons*<pid>)

5.4.1. Запрос

```
Content-Type - application/json

Параметры URL:

    <pid> - Идентификатор персоны
```

5.4.2. Ответ

```
Content-Type - application/json

Пример ответа (JSON):

{
  "result": true
}
```

5.5. Получить список фотографий персоны (GET *persons*<pid>/faces)

5.5.1. Запрос

```
Content-Type - application/json

Параметры URL:

    <pid> - Идентификатор персоны

Параметры QUERY:

    <limit> - Количество запрашиваемых записей
    <offset> - Отступ от начала списка
```

5.5.2. Ответ

```
Content-Type - application/json

Пример ответа (JSON):
```


```
{
    "faces": [{
        "eyeLeft": {
            "x": 118,
            "y": 47
        },
        "rect": {
            "x": 90,
            "y": 125,
            "w": 167,
            "h": 168
        },
        "id": "sdf",
        "person": {
            "id": "111",
            "tags": ["one", "two"]
        },
        "eyeRight": {
            "x": 48,
            "y": 49
        }
    }],
    "meta": {
        "offset": 0,
        "total": 1,
        "limit": 5
    }
}
```


5.6. Добавить фотографию в базу (POST *persons*<pid>/faces)

5.6.1. Запрос

```
Content-Type - multipart/form-data

Параметры URL:

    <pid> - Идентификатор персоны

Параметры FORM:

    <id> - Идентификатор фотографии
    <face> - Файл фотографии
```

5.6.2. Ответ

```
Content-Type - application/json

Пример ответа (JSON):
```

```
{
    "id": "45",
    "person": {
        "id": "111",
        "tags": ["one", "two"]
    },
    "eyeLeft": {
        "x": 118,
        "y": 47
    },
    "rect": {
        "x": 90,
        "y": 125,
        "w": 167,
        "h": 168
    },
    "eyeRight": {
        "x": 48,
        "y": 49
    }
}
```

5.7. Получение информации о фотографии (GET *persons*<pid>/faces/<fid>)

5.7.1. Запрос

```
Content-Type - application/json

Параметры URL:

    <pid> - Идентификатор персоны
    <fid> - Идентификатор фотографии
```

5.7.2. Ответ

```
Content-Type - application/json

Пример ответа (JSON):
```

```
{
    "id": "45",
    "person": {
        "id": "111",
        "tags": ["one", "two"]
    },
    "eyeLeft": {
        "x": 118,
        "y": 47
    },
    "rect": {
        "x": 90,
        "y": 125,
        "w": 167,
        "h": 168
    },
    "eyeRight": {
        "x": 48,
        "y": 49
    }
}
```


5.8. Удаление информации о фотографии (DELETE *persons*<pid>/faces/<fid>)

5.8.1. Запрос

```
Content-Type - application/json

Параметры URL:

    <pid> - Идентификатор персоны
    <fid> - Идентификатор фотографии
```

5.8.2. Ответ

```
Content-Type - application/json

Пример ответа (JSON):
```


```
{
  "result": true
}
```


5.9. Получить список фотографий (GET /faces)

5.9.1. Запрос

```
Content-Type - application/json

Параметры QUERY:

    <limit> - Количество запрашиваемых записей
    <offset> - Отступ от начала списка
```

5.9.2. Ответ

```
Content-Type - application/json

Пример ответа (JSON):
```


```
{
    "faces": [{
        "eyeLeft": {
            "x": 118,
            "y": 47
        },
        "rect": {
            "x": 90,
            "y": 125,
            "w": 167,
            "h": 168
        },
        "id": "sdf",
        "person": {
            "id": "111",
            "tags": ["one", "two"]
        },
        "eyeRight": {
            "x": 48,
            "y": 49
        }
    }],
    "meta": {
        "offset": 0,
        "total": 1,
        "limit": 5
    }
}
```


5.10. Поиск персоны по фотографии (POST /find)

5.10.1. Запрос

```
Content-Type - multipart/form-data

Параметры QUERY:

    <count> - Количество найденных персон в ответе
    <tags> - Теги указанные через запятую

Параметры FORM:

    <face> - Файл фотографии для поиска
```

5.10.2. Ответ

```
Content-Type - application/json

Пример ответа (JSON):
```

```
{
  "face": {
    "rect": {
      "x": 186,
      "y": 209,
      "w": 181,
      "h": 181
    },
    "result": [
      {
        "similarity": 100,
        "face_id": "5bf7ecbd-a1d9-4115-9fa4-bf6d65b9ffb6",
        "person_id": "Lena"
      },
      {
        "similarity": 100,
        "face_id": "157374f6-3e4d-11e8-b467-0ed5f89f718b",
        "person_id": "Lena2"
      }
    ]
  }
}
```


5.11. Поиск нескольких персон по фотографии (POST /find/all)

5.11.1. Запрос

Content-Type - multipart/form-data

Параметры QUERY:

<count> - Количество найденных персон в ответе
<tags> - Теги указанные через запятую

Параметры FORM:

<face> - Файл фотографии для поиска

5.11.2. Ответ

Content-Type - application/json

Пример ответа (JSON):


```
{
  "faces": [
    {
      "rect": {
        "x": 186,
        "y": 209,
        "w": 181,
        "h": 181
      },
      "result": [
```

```
        {
          "similarity": 100,
          "face_id": "5bf7ecbd-a1d9-4115-9fa4-bf6d65b9ffb6",
          "person_id": "Lena"
        }
      ]
    },
    {
      "rect": {
        "x": 372,
        "y": 415,
        "w": 145,
        "h": 161
      },
      "result": [
        {
          "similarity": 100,
          "face_id": "07cc0c5a-3e4d-11e8-b467-0ed5f89f718b",
          "person_id": "Olga"
        }
      ]
    }
  ]
}
```


5.12. Сравнение лиц на двух фотографиях (POST /compare)

5.12.1. Запрос

```
Content-Type - multipart/form-data

Параметры FORM:

    <face1> - Файл первой фотографии для сравнения
    <face2> - Файл второй фотографии для сравнения
```

5.12.2. Ответ

```
Content-Type - application/json

Пример ответа (JSON):
```


```
{
    "similarity": 100
}
```


5.13. Сравнение двух лиц на одной фотографии (POST /compare/self)

5.13.1. Запрос

Content-Type - multipart/form-data

Параметры FORM:

<face> - Файл фотографии для сравнения

5.13.2. **Ответ**

Content-Type - application/json

Пример ответа (JSON):


```
{
    "similarity": 100
}
```

6. ПРАКТИЧЕСКИЕ РЕКОМЕНДАЦИИ

6.1. Выбор пороговых значений similarity

В практических задачах распознавания необходимо выбирать пороговое/вые значение/я для величины `similarity` (степень похожести), возвращаемой сервисом. Это задача не так однозначна и сильно зависит от входных данных (например, от качества и размера картинок, от степени резкости фото), от размера БД в которой осуществляется поиск, от сценария использования сервиса.

Наиболее часто встречающийся диапазон значений для порогового значения similarity (рабочий диапазон значений в реальных условиях работы сервиса, на реальных данных) это диапазон от 0.55 до 0.7-0.8.

В некоторых сценариях допускается использовать пониженное значение порога от 0.5.

В некоторых специальных сценариях работы допустимо использовать крайне низкие значения порога от 0.45, см подразд. 6.3.

Факторы влияющие на выбор пороговых значений:

1) Качество съемки/фотографий
2) Размер БД
3) Присутсвие оператора (проверяет ли работу алгоритма человек)
4) Сценарий использования

Данные факторы выстроены без приоритета, однако сценарий использования играет максимальный приоритет.

Выбор пограничного значения similarity является неотъемлемой частью любого сценария и целиком определяет качество работы алгоритма сравнения людей.

6.2. Сценарий

Во всех случаях выбора порогового значения сценарий играет ключевую роль.

Есть сценарии исключающие возможность ложного срабатывания, то есть когда алгоритм путает людей и считает разных объективно людей одинаковыми или похожими.

Это, к примеру, сценарии безопасности, создания ограждения для входа только зарегистрированного узкого круга людей. Считается допустимым не пропустить человека допущенного к периметру или многократные попытки входа.

Для реализации подобных сценариев, пороговое значение similarity выбирается максимально большим, на предельных значениях (это показатели от 0.7 и вплоть до 0.9 и выше, и выбираются в эксперименте). Для подобных сценариев исключен вариант работы на показателях ниже 0.65 во всех случаях. Не играют значения ни качество съемки ни маленький размер БД. Диапазон значений от 0.65 до 0.7 не рекомендуется, однако допустим на усмотрение разработчиков

системы/сценария.

Есть большое число практических задач, где алгоритму допускается ошибиться. Для таких задач существует нормальный, обычный режим работы с пограничными значениями от 0.55-0.65, в некоторых случаях от 0.5, данный вариант работы не считается рекомендованным. Естественно, пограничные значения близкие к 0.55 увеличивают вероятность ложных срабатываний, допустимость которых определяется сценарием использования сервиса.

Существуют сценарии использования на ультра-низких значениях порога, от 0.45 и выше.

6.3. Примеры специальных сценариев с низким порогом

Можно привести пример специальных сценариев для которых возможен вариант наименьшего порога (от 0.45), в некоторых случаях возможен еще более низкий порог, однако этот вариант не рекомендуется к работе. Во всех случаях, работы на низких порогах (от 0.45 (или ниже) до 0.55) обязательно участие охранника, оператора и тп... или данный режим работы предполагает автоматизированный разбор событий с последующей обработкой опять же оператором.

Гипотетический сценарий работы на низком пороге. У нас комната сотрудников из 10-20 человек и мы точно всегда знаем фиксированный набор человек присутствующих в помещении, исключается вариант появления незнакомца. Все случаи появления на камере нужно классифицировать. Почти наверняка будут ложные срабатывания. Однако нам важнее хоть как-то узнать человека.

Также возможен вариант использования низких пороговых значений на проходных, когда камера стоит неудачно, а нужно помочь охранникам не пропустить преступников. Режим, когда «лучше перебдеть».

Другой сценарий работы, который испытан в практической работе. Есть огромное множество низкокачественных снимков (с сотнями тысяч событий и выше, например проходная) и нам надо найти человека похожего на искомого, необходимо найти в любом случае, камера стоит не удачно. Тогда выставляется минимальный порог и начинается поиск по событиям с участием оператора. Низкий порог помогает в таком случае нам «разгрести помойку».

Также возможен вариант работы, как помощник для оператора, например на кассе. Когда нужно найти человека заходившего в магазин «на днях» в низкопроходных магазинах.

То есть существуют сценарии работы с низкими и ультра-низкими пороговыми значениями. Возможен вариант использования разных пороговых значений для различных реакций системы.

ПЕРЕЧЕНЬ ТЕРМИНОВ

Термин	Определение
1. **API**	программный интерфейс приложения, интерфейс прикладного программирования.
2. **GEMBAFACE**	программа (набор сервисов) для распознавания лица, разработки Ред Софт.
3. **GET, POST, DELETE**	типы запросов HTTP.
4. **HTTP**	протокол прикладного уровня передачи данных.
5. **JSON**	текстовый формат обмена данными, основанный на JavaScript.
6. **REST**	архитектурный стиль взаимодействия компонентов распределённого приложения в сети. Большинство REST-реализаций сервисов используют такие стандарты, как HTTP, URL, JSON и, реже, XML.
7. **ГЕМБАФЕЙС**	рускоязычное название программы GEMBAFACE.
8. **Токен авторизации**	ключ доступа к службам, в нашем контексте указывается в заголовках HTTP.