Руководство по эксплуатации Сервиса инференса MarQus

(Руководство разработчика)

Сервис инференса нейросетевых моделей MarQus для автоматизации процессов распознавания и сортировки различных типов ТБО.

2023

Содержание

Аннотация

Настоящая документация необходима для быстрой адаптации существующих и новых продуктов в компании с сервисом инференса нейросетевых моделей MarQus для автоматизации процессов распознавания и сортировки различных типов ТБО. Данная документация содержит информацию, необходимую для эксплуатации экземпляра ПО «СИ MarQus».

1. API сервиса

Текущая версия API: v1.

1.1. Протоколы, версия протоколов

Сервис реализует поддержку двух классов протоколов. HTTP/1.1 и HTTP/2 в качестве протокола для REST API и ZMTP 3.0 для передачи видеопотока.

1.2. Описание API

1.2.1. REST API

После запуска контейнера REST API доступен по порту 8001. Например http://127.0.0.1:8001

OpenAPI спецификация REST API СИ MarQus


```
{
   "openapi":"3.0.2",
   "info":{
      "title":"Производит процессинг видеопотока.",
      "description":"Производит процессинг видеопотока. Детекцию, сегментацию,
классификацию и трекинг объектов",
      "version":"2023.0.1"
   },
   "paths":{
      "/":{
         "get":{
            "summary":"Main",
            "operationId":"main__get",
            "responses":{
               "200":{
                  "description":"Successful Response",
                  "content":{
                     "application/json":{
                        "schema":{

                        }
                     }
                  }
               }
            }
         }
      }
```

```
      },
      "/recotracker/recognize":{
         "post":{
            "summary":"Post Recognize",
            "description":"Производит сегментацию, детекцию, классификацию и
трекинг ТБО на изображении",
            "operationId":"post_recognize_recotracker_recognize_post",
            "parameters":[
               {
                  "required":false,
                  "schema":{
                     "title":"Version",
                     "type":"integer",
                     "default":1
                  },
                  "name":"version",
                  "in":"query"
               }
            ],
            "requestBody":{
               "content":{
                  "multipart/form-data":{
                     "schema":{

"$ref":"#/components/schemas/Body_post_recognize_recotracker_recognize_post"
                     }
                  }
               },
               "required":true
            },
            "responses":{
               "200":{
                  "description":"Successful Response",
                  "content":{
                     "application/json":{
                        "schema":{

                        }
                     }
                  }
               },
               "422":{
                  "description":"Validation Error",
                  "content":{
                     "application/json":{
                        "schema":{
                           "$ref":"#/components/schemas/HTTPValidationError"
                        }
                     }
                  }
               }
            }
         }
      },
      "/recotracker/recognize/async":{
         "get":{
            "summary":"Get Recognize Async",
```

```
            "description":"Возвращает результат по квитанции",
            "operationId":"get_recognize_async_recotracker_recognize_async_get",
            "parameters":[
               {
                  "required":true,
                  "schema":{
                     "title":"Ticket",
                     "type":"string"
                  },
                  "name":"ticket",
                  "in":"query"
               },
               {
                  "required":false,
                  "schema":{
                     "title":"Version",
                     "type":"integer",
                     "default":1
                  },
                  "name":"version",
                  "in":"query"
               }
            ],
            "responses":{
               "200":{
                  "description":"Successful Response",
                  "content":{
                     "application/json":{
                        "schema":{

                        }
                     }
                  }
               },
               "422":{
                  "description":"Validation Error",
                  "content":{
                     "application/json":{
                        "schema":{
                           "$ref":"#/components/schemas/HTTPValidationError"
                        }
                     }
                  }
               }
            }
         },
         "post":{
            "summary":"Post Recognize Async",
            "description":"Запускает процесс сегментации, детекции, классификации
и трекина ТБО на изображении. Возвращает квитанцию, по которой позже можно будет
забрать результат",
            "operationId":"post_recognize_async_recotracker_recognize_async_post",
            "parameters":[
               {
                  "required":false,
                  "schema":{
                     "title":"Version",
```

```
                  "type":"integer",
                  "default":1
               },
               "name":"version",
               "in":"query"
            }
         ],
         "requestBody":{
            "content":{
               "multipart/form-data":{
                  "schema":{
"$ref":"#/components/schemas/Body_post_recognize_async_recotracker_recognize_async
_post"
                  }
               }
            },
            "required":true
         },
         "responses":{
            "200":{
               "description":"Successful Response",
               "content":{
                  "application/json":{
                     "schema":{
                        "$ref":"#/components/schemas/Ticket"
                     }
                  }
               }
            },
            "422":{
               "description":"Validation Error",
               "content":{
                  "application/json":{
                     "schema":{
                        "$ref":"#/components/schemas/HTTPValidationError"
                     }
                  }
               }
            }
         }
      }
   },
   "/recotracker/recognize/tickets":{
      "get":{
         "summary":"List Tickets",
         "description":"Возвращает список квитанцийdefault в очереди",
         "operationId":"list_tickets_recotracker_recognize_tickets_get",
         "parameters":[
            {
               "required":false,
               "schema":{
                  "title":"Version",
                  "type":"integer",
                  "default":1
               },
               "name":"version",
```

```
                  "in":"query"
               }
            ],
            "responses":{
               "200":{
                  "description":"Successful Response",
                  "content":{
                     "application/json":{
                        "schema":{
                           "title":"Response List Tickets Recotracker Recognize
Tickets Get",
                           "type":"array",
                           "items":{

                           }
                        }
                     }
                  }
               },
               "422":{
                  "description":"Validation Error",
                  "content":{
                     "application/json":{
                        "schema":{
                           "$ref":"#/components/schemas/HTTPValidationError"
                        }
                     }
                  }
               }
            }
         }
      },
      "/recotracker/queue_receiver/info":{
         "get":{
            "summary":"List Queue Receiver Info",
            "description":"Возвращает информацию об очереди ресивера",

"operationId":"list_queue_receiver_info_recotracker_queue_receiver_info_get",
            "parameters":[
               {
                  "required":false,
                  "schema":{
                     "title":"Version",
                     "type":"integer",
                     "default":1
                  },
                  "name":"version",
                  "in":"query"
               }
            ],
            "responses":{
               "200":{
                  "description":"Successful Response",
                  "content":{
                     "application/json":{
                        "schema":{
                           "title":"Response List Queue Receiver Info Recotracker
```

```
Queue Receiver Info Get",
                                "type":"object"
                             }
                          }
                       }
                    },
                    "422":{
                       "description":"Validation Error",
                       "content":{
                          "application/json":{
                             "schema":{
                                "$ref":"#/components/schemas/HTTPValidationError"
                             }
                          }
                       }
                    }
                 }
              }
           },
           "/recotracker/queue_request/info":{
              "get":{
                 "summary":"List Queue Request Info",
                 "description":"Возвращает информацию об очереди запросов на
распознавание",

"operationId":"list_queue_request_info_recotracker_queue_request_info_get",
                 "parameters":[
                    {
                       "required":false,
                       "schema":{
                          "title":"Version",
                          "type":"integer",
                          "default":1
                       },
                       "name":"version",
                       "in":"query"
                    }
                 ],
                 "responses":{
                    "200":{
                       "description":"Successful Response",
                       "content":{
                          "application/json":{
                             "schema":{
                                "title":"Response List Queue Request Info Recotracker
Queue Request Info Get",
                                "type":"object"
                             }
                          }
                       }
                    },
                    "422":{
                       "description":"Validation Error",
                       "content":{
                          "application/json":{
                             "schema":{
                                "$ref":"#/components/schemas/HTTPValidationError"
```

```
                        }
                     }
                  }
               }
            }
         },
         "/recotracker/info/system":{
            "get":{
               "summary":"Get Info",
               "description":"Информация о системе и окружении",
               "operationId":"get_info_recotracker_info_system_get",
               "parameters":[
                  {
                     "required":false,
                     "schema":{
                        "title":"Version",
                        "type":"integer",
                        "default":1
                     },
                     "name":"version",
                     "in":"query"
                  }
               ],
               "responses":{
                  "200":{
                     "description":"Successful Response",
                     "content":{
                        "application/json":{
                           "schema":{
                              "title":"Response Get Info Recotracker Info System
Get",
                              "type":"object"
                           }
                        }
                     }
                  },
                  "422":{
                     "description":"Validation Error",
                     "content":{
                        "application/json":{
                           "schema":{
                              "$ref":"#/components/schemas/HTTPValidationError"
                           }
                        }
                     }
                  }
               }
            }
         }
      },
      "components":{
         "schemas":{
            "Body_post_recognize_async_recotracker_recognize_async_post":{
               "title":"Body_post_recognize_async_recotracker_recognize_async_post",
               "required":[
                  "upload_file"
```

```
      ],
      "type":"object",
      "properties":{
         "upload_file":{
            "title":"Upload File",
            "type":"string",
            "format":"binary"
         }
      }
   },
   "Body_post_recognize_recotracker_recognize_post":{
      "title":"Body_post_recognize_recotracker_recognize_post",
      "required":[
         "upload_file"
      ],
      "type":"object",
      "properties":{
         "upload_file":{
            "title":"Upload File",
            "type":"string",
            "format":"binary"
         }
      }
   },
   "HTTPValidationError":{
      "title":"HTTPValidationError",
      "type":"object",
      "properties":{
         "detail":{
            "title":"Detail",
            "type":"array",
            "items":{
               "$ref":"#/components/schemas/ValidationError"
            }
         }
      }
   },
   "Ticket":{
      "title":"Ticket",
      "required":[
         "ticket"
      ],
      "type":"object",
      "properties":{
         "ticket":{
            "title":"Ticket",
            "type":"string",
            "format":"uuid"
         }
      }
   },
   "ValidationError":{
      "title":"ValidationError",
      "required":[
         "loc",
         "msg",
         "type"
```

```
            ],
            "type":"object",
            "properties":{
               "loc":{
                  "title":"Location",
                  "type":"array",
                  "items":{
                     "anyOf":[
                        {
                           "type":"string"
                        },
                        {
                           "type":"integer"
                        }
                     ]
                  }
               },
               "msg":{
                  "title":"Message",
                  "type":"string"
               },
               "type":{
                  "title":"Error Type",
                  "type":"string"
               }
            }
         }
      }
   }
}
```


Более подробно изучить спецификацию Open API можно по ссылке (3)

1.2.2. ImageZMQ

В сервисе используется библиотека imageZMQ для стриминга потока изображений OpenCV с одного компьютера на другой с помощью обмена сообщениями PyZMQ.

После запуска контейнера сервис слушает TCP порт 5500.

Более подробно изучить ZMQ можно по ссылке (1)

2. Примеры

2.1. Python

Минимальный код на Python для стриминга изображений для Raspberry Pi выглядит так:

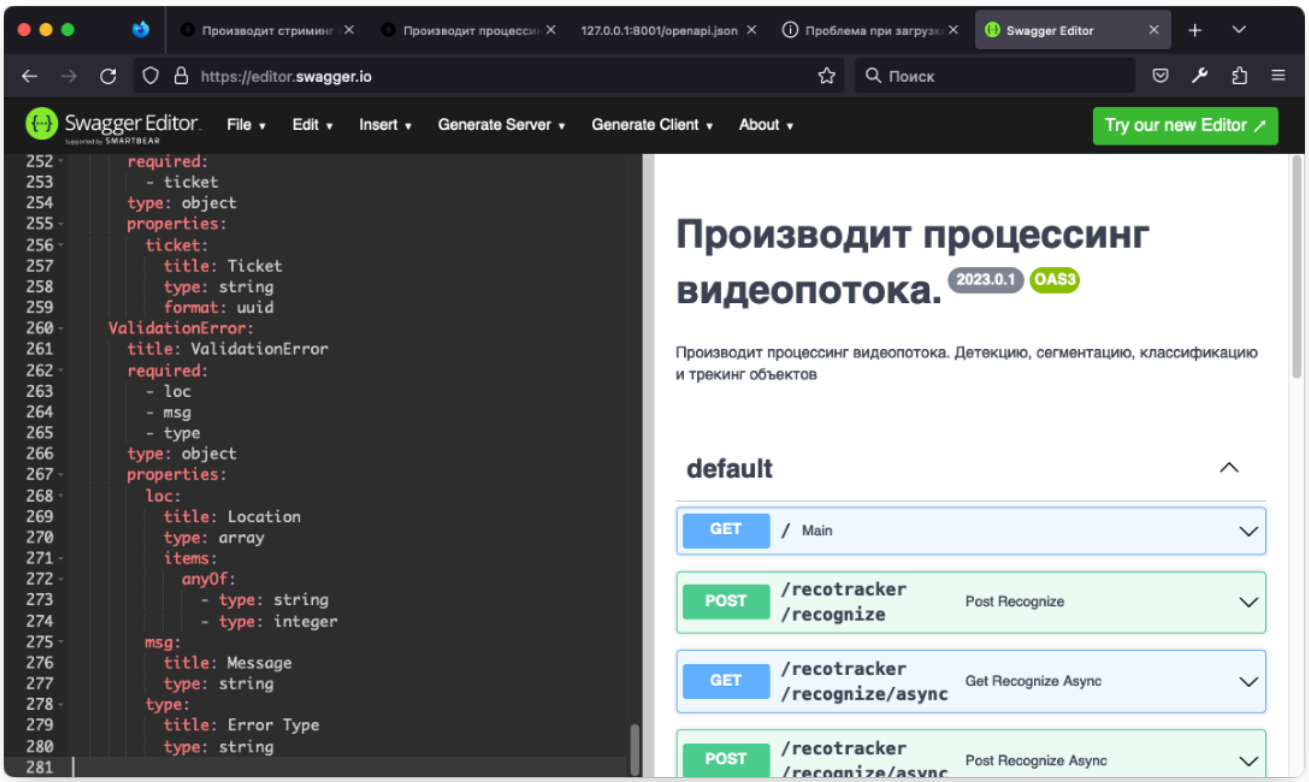
```
import time

from dynaconf import settings
from imutils.video import VideoStream

from vss import VideoStreamSender

# Stream from Raspberry Pi camera
if __name__ == "__main__":
    picam = VideoStream(usePiCamera=True).start()
    time.sleep(2)
    sender = imagezmq.ImageSender(connect_to="tcp://127.0.0.1:5500",
REQ_REP=false)
    while True:
        img = picam.read()
        sender.send_image(socket.gethostname(), img)
        time.sleep(0.1)
```

Более подробно про захват и передачу изображений можно изучить по ссылке (2)

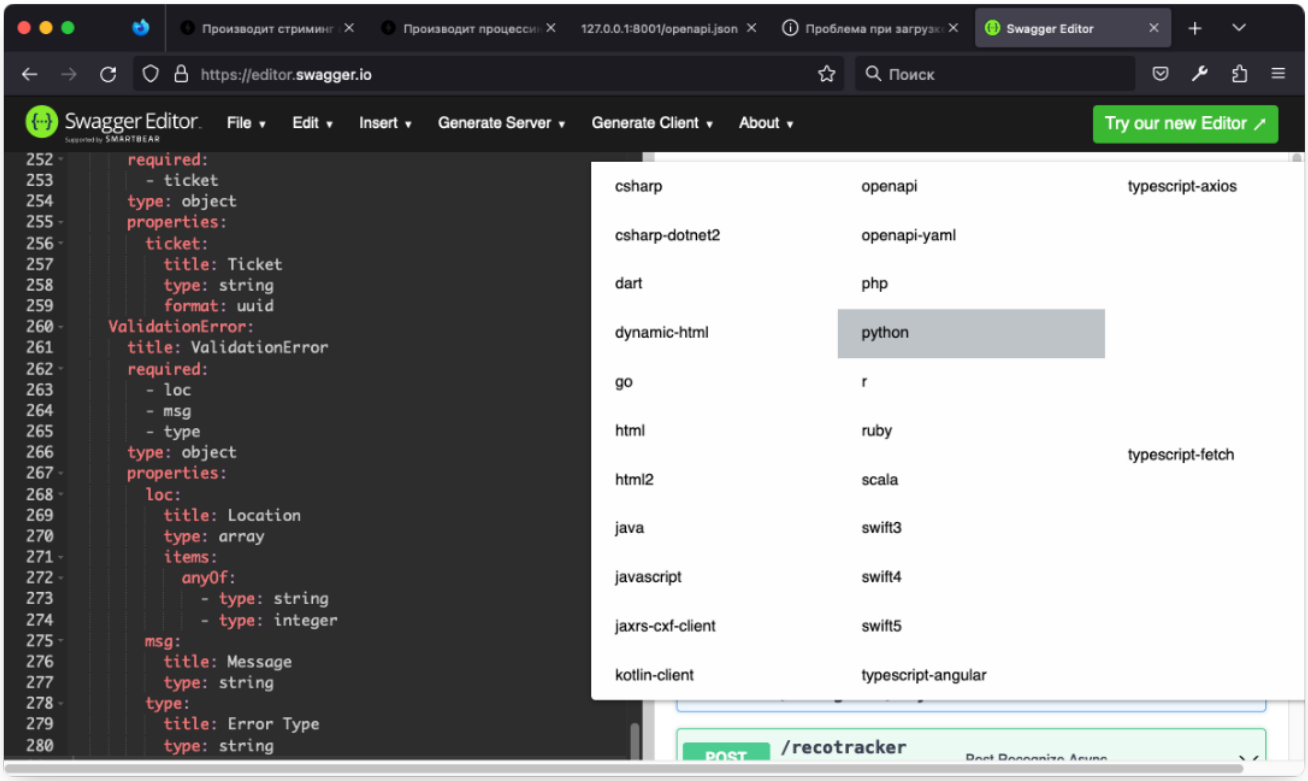
2.2. Генерация кода интерфейса

Используя OpenAPI спецификацию REST API СИ MarQus можно сгенерировать код клиента. Для этого необходимо:

1. перейти по ссылке (4)
2. вставить в редактор json из пункта 1.2.1 REST API



3. В меню Generate Client выбрать необходимый язык программирования. Например, Python.



4. Скачать получившийся python-client-generated.zip и использовать в приложении использующем СИ MarQus через REST API

Альтернативный путь - использовать устоявшиеся практики разработки в компании, что может дать в результате более качественный код.

3. Обзор сервиса

3.1. Установка и запуск

Для установки и запуска сервиса необходимо:

1. Скачать образ, используя команду

```
docker pull harbor.zebrains.team/mlearning/marqus-video-recotracker:latest
```

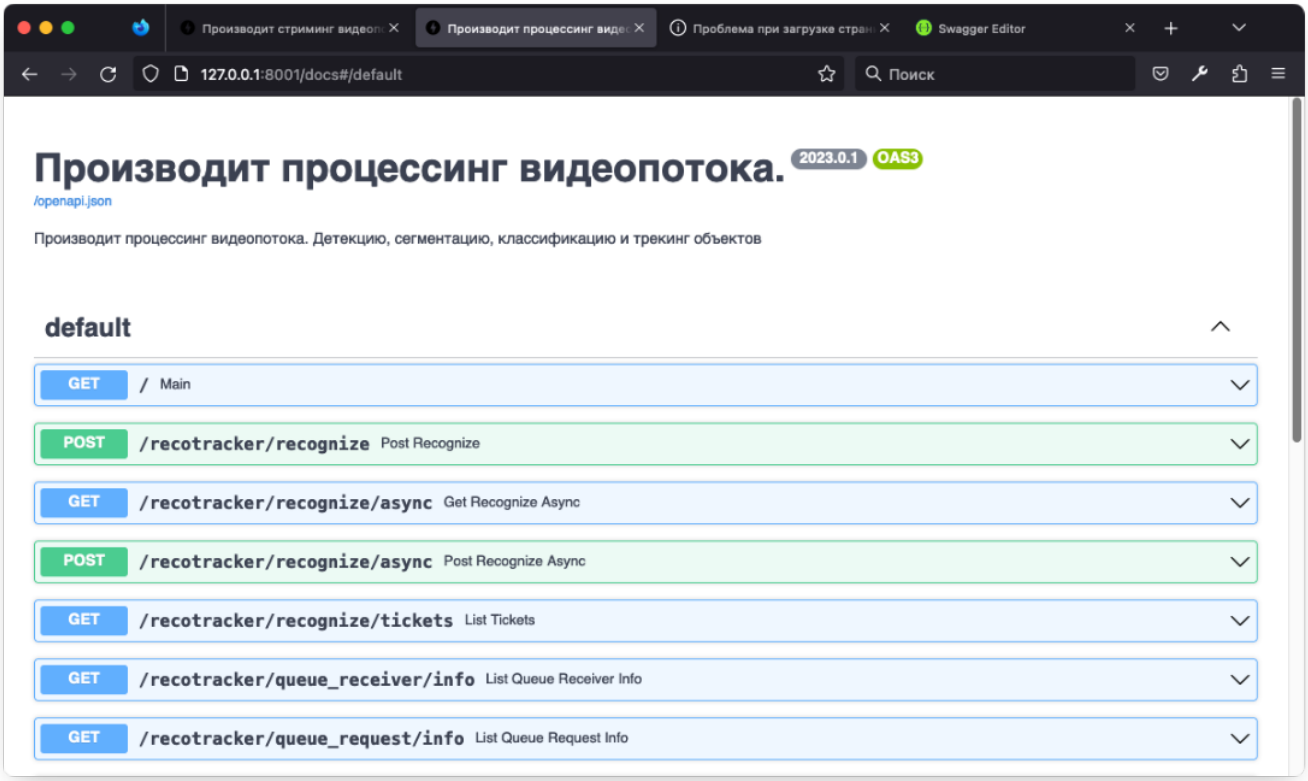
2. Запустить новый контейнер из образа

```
docker run --name marqus-video-recotracker -p 8001:8001/tcp -p 8002:8002/tcp
harbor.zebrains.team/mlearning/marqus-video-recotracker:latest
```

3. Проверить в браузере

Открыть ссылку http://127.0.0.1:8001

Должны увидеть API сервиса marqus-video-recotracker:



4. Проверить эндпоинт /recotracker/info

В Response body должны увидеть информацию о доступности cuda.

```
"torch_info": {
  "cuda.is_available": true
},
```

Если cuda недоступна, то сервис всё равно будет работать, но производительность сервиса упадёт на порядок.

3.2. Совместимость версий

Инструкции настоящего документа применимы к следующим версиям сервиса: 2023.0.X

Проверить версию сервиса можно с момощью команды при запущенном контейнере.

X - минорная версия сервиса, не влияющая на функционал сервиса.

```
curl http://127.0.0.1:8001/openapi.json|jq -r .info.version
```

4. Поддерживаемая функциональность

4.1. Методы REST API

Методы REST API СИ MarQus не предназначены для эксплуатации под нагрузкой и предназначены для разработки прототипов и тестирования интеграций.

1. Метод REST POST /recotracker/recognize

Производит все стадии пайплайна (детекцию, сегментацию, классификацию и трекинг) и возвращает результат. Трекинг в случае со статичной картинкой особого смысла не имеет и представляет собой просто ID объекта и его координаты.

```
curl -X 'POST' \
  'http://127.0.0.1:8001/recotracker/recognize?version=1' \
  -H 'accept: application/json' \
  -H 'Content-Type: multipart/form-data' \
  -F 'upload_file=@example.png;type=image/png'
```

Результат вызова метода - изображение с визуализацией распознавания.

2. Метод REST POST /recotracker/recognize/async

Запускает процесс сегментации, детекции, классификации и трекина ТБО на изображении. Возвращает квитанцию, по которой позже можно будет забрать результат.

```
curl -X 'POST' \
  'http://127.0.0.1:8001/recotracker/recognize/async?version=1' \
  -H 'accept: application/json' \
  -H 'Content-Type: multipart/form-data' \
  -F 'upload_file=@example.png;type=image/png'
```

вернёт

```
{
  "ticket": "dbca1b67-5395-445a-9308-e48b8a7fcba4"
}
```

3. Метод REST GET /recotracker/recognize/tickets

Возвращает список обработанных квитанций в очереди.

```
curl -X 'GET' \
  'http://127.0.0.1:8001/recotracker/recognize/tickets?version=1' \
  -H 'accept: application/json'
```

вернёт

```
[
  "dbca1b67-5395-445a-9308-e48b8a7fcba4"
]
```

4. Метод REST GET /recotracker/recognize/async

Возвращает результат по квитанции.

```
curl -X 'GET' \
  'http://127.0.0.1:8001/recotracker/recognize/async?ticket=dbca1b67-5395-445a-
9308-e48b8a7fcba4&version=1' \
  -H 'accept: application/json'
```

Результат вызова метода - изображение с визуализацией распознавания.

5. Метод REST GET /recotracker/queue_request/info

Возвращает информацию об очереди запросов на распознавание.

```
curl -X 'GET' \
  'http://127.0.0.1:8001/recotracker/queue_request/info?version=1' \
  -H 'accept: application/json'
```

вернёт

```
{
  "qsize": 0,
  "maxsize": 100
}
```

6. Метод REST GET /recotracker/queue_receiver/info

Возвращает информацию об очереди ресивера.

```
curl -X 'GET' \
  'http://127.0.0.1:8001/recotracker/queue_receiver/info?version=1' \
  -H 'accept: application/json'
```

вернёт

```
{
  "qsize": 2,
  "maxsize": 100
}
```

7. Метод REST GET /recotracker/info/system

Возвращает информацию о системе и окружении. Полезен для понимания того, как расходуется ресурс сервера при работающем сервисе. Также полезен для понимания наличия ресурсов, например доступности GPU.

```
curl -X 'GET' \
  'http://127.0.0.1:8001/recotracker/info/system?version=1' \
  -H 'accept: application/json'
```

вернёт

```
{
  "torch_info": {
    "cuda.is_available": true
  },
  "platform_info": {
```

```
  "system": "Linux",
  "node_name": "ai-reference-sys-info-staging-deployment-dcd5c4cd-8rfsk",
  "release": "5.4.0-144-generic",
  "version": "#161-Ubuntu SMP Fri Feb 3 14:49:04 UTC 2023",
  "machine": "x86_64",
  "[processor": "x86_64"
},
"ip_info": {
  "ip_address_by_socket": "10.233.80.94"
},
"mem_info": {
  "total": "251.54 GB",
  "available": "230.09 GB",
  "used": "19.49 GB",
  "percentage": 8.5
},
"swap_info": {
  "total": "0 B",
  "free": "0 B",
  "used": "0 B",
  "percentage": 0
},
"partitions_info": {
  "total": "0 B",
  "free": "0 B",
  "used": "0 B",
  "percentage": 0
},
"gpu_info": {
  "available": [
    0,
    1
  ],
  "all": [
    {
      "id": 0,
      "uuid": "GPU-4ce7ff43-236d-2484-3784-b131af5cc9f1",
      "load": 0,
      "memoryUtil": 0.1441947565543071,
      "memoryTotal": 24564,
      "memoryUsed": 3542,
      "memoryFree": 20705,
      "driver": "525.85.05",
      "name": "NVIDIA RTX A5000",
      "serial": "1322721041028",
      "display_mode": "Disabled",
      "display_active": "Disabled",
      "temperature": 27
    },
    {
      "id": 1,
      "uuid": "GPU-e6d944c4-b209-0f2f-eddc-d8221aa56d8d",
      "load": 0,
      "memoryUtil": 0.21095912717798404,
      "memoryTotal": 24564,
      "memoryUsed": 5182,
      "memoryFree": 19065,
```

```
        "driver": "525.85.05",
        "name": "NVIDIA RTX A5000",
        "serial": "1324221006645",
        "display_mode": "Disabled",
        "display_active": "Disabled",
        "temperature": 25
      }
    ]
  },
  "env_info": {
    "PATH": "/usr/local/cuda-
11.1/bin:/usr/local/nvidia/bin:/usr/local/cuda/bin:/usr/local/sbin:/usr/local/bin:
/usr/sbin:/usr/bin:/sbin:/bin",
    "HOSTNAME": "ai-reference-sys-info-staging-deployment-dcd5c4cd-8rfsk",
    "AI_REFERENCE_SYS_INFO_STAGING_SVC_SERVICE_HOST": "10.233.38.164",
    "AI_REFERENCE_SYS_INFO_STAGING_SVC_PORT_80_TCP_PROTO": "tcp",
    "KUBERNETES_SERVICE_PORT": "443",
    "KUBERNETES_SERVICE_PORT_HTTPS": "443",
    "KUBERNETES_PORT_443_TCP": "tcp://10.233.0.1:443",
    "KUBERNETES_PORT_443_TCP_PROTO": "tcp",
    "AI_REFERENCE_SYS_INFO_STAGING_SVC_SERVICE_PORT": "80",
    "AI_REFERENCE_SYS_INFO_STAGING_SVC_PORT_80_TCP": "tcp://10.233.38.164:80",
    "KUBERNETES_PORT_443_TCP_ADDR": "10.233.0.1",

"AI_REFERENCE_SYS_INFO_STAGING_SVC_SERVICE_PORT_AI_REFERENCE_SYS_INFO_STAGING":
"80",
    "AI_REFERENCE_SYS_INFO_STAGING_SVC_PORT_80_TCP_ADDR": "10.233.38.164",
    "AI_REFERENCE_SYS_INFO_STAGING_SVC_PORT_80_TCP_PORT": "80",
    "KUBERNETES_SERVICE_HOST": "10.233.0.1",
    "KUBERNETES_PORT": "tcp://10.233.0.1:443",
    "KUBERNETES_PORT_443_TCP_PORT": "443",
    "AI_REFERENCE_SYS_INFO_STAGING_SVC_PORT": "tcp://10.233.38.164:80",
    "NVARCH": "x86_64",
    "NVIDIA_REQUIRE_CUDA": "cuda>=11.8 brand=tesla,driver>=450,driver<451
brand=tesla,driver>=470,driver<471 brand=unknown,driver>=470,driver<471
brand=nvidia,driver>=470,driver<471 brand=nvidiartx,driver>=470,driver<471
brand=geforce,driver>=470,driver<471 brand=geforcertx,driver>=470,driver<471
brand=quadro,driver>=470,driver<471 brand=quadrortx,driver>=470,driver<471
brand=titan,driver>=470,driver<471 brand=titanrtx,driver>=470,driver<471
brand=tesla,driver>=510,driver<511 brand=unknown,driver>=510,driver<511
brand=nvidia,driver>=510,driver<511 brand=nvidiartx,driver>=510,driver<511
brand=geforce,driver>=510,driver<511 brand=geforcertx,driver>=510,driver<511
brand=quadro,driver>=510,driver<511 brand=quadrortx,driver>=510,driver<511
brand=titan,driver>=510,driver<511 brand=titanrtx,driver>=510,driver<511
brand=tesla,driver>=515,driver<516 brand=unknown,driver>=515,driver<516
brand=nvidia,driver>=515,driver<516 brand=nvidiartx,driver>=515,driver<516
brand=geforce,driver>=515,driver<516 brand=geforcertx,driver>=515,driver<516
brand=quadro,driver>=515,driver<516 brand=quadrortx,driver>=515,driver<516
brand=titan,driver>=515,driver<516 brand=titanrtx,driver>=515,driver<516",
    "NV_CUDA_CUDART_VERSION": "11.8.89-1",
    "NV_CUDA_COMPAT_PACKAGE": "cuda-compat-11-8",
    "CUDA_VERSION": "11.8.0",
    "LD_LIBRARY_PATH": "/usr/local/cuda-
11.1/lib64:/usr/local/nvidia/lib:/usr/local/nvidia/lib64",
    "NVIDIA_VISIBLE_DEVICES": "all",
    "NVIDIA_DRIVER_CAPABILITIES": "compute,utility",
    "CUDA_HOME": "/usr/local/cuda-11.1",
```

```
    "TZ": "Europe/Moscow",
    "HOME": "/root",
    "LC_CTYPE": "C.UTF-8"
  }
}
```

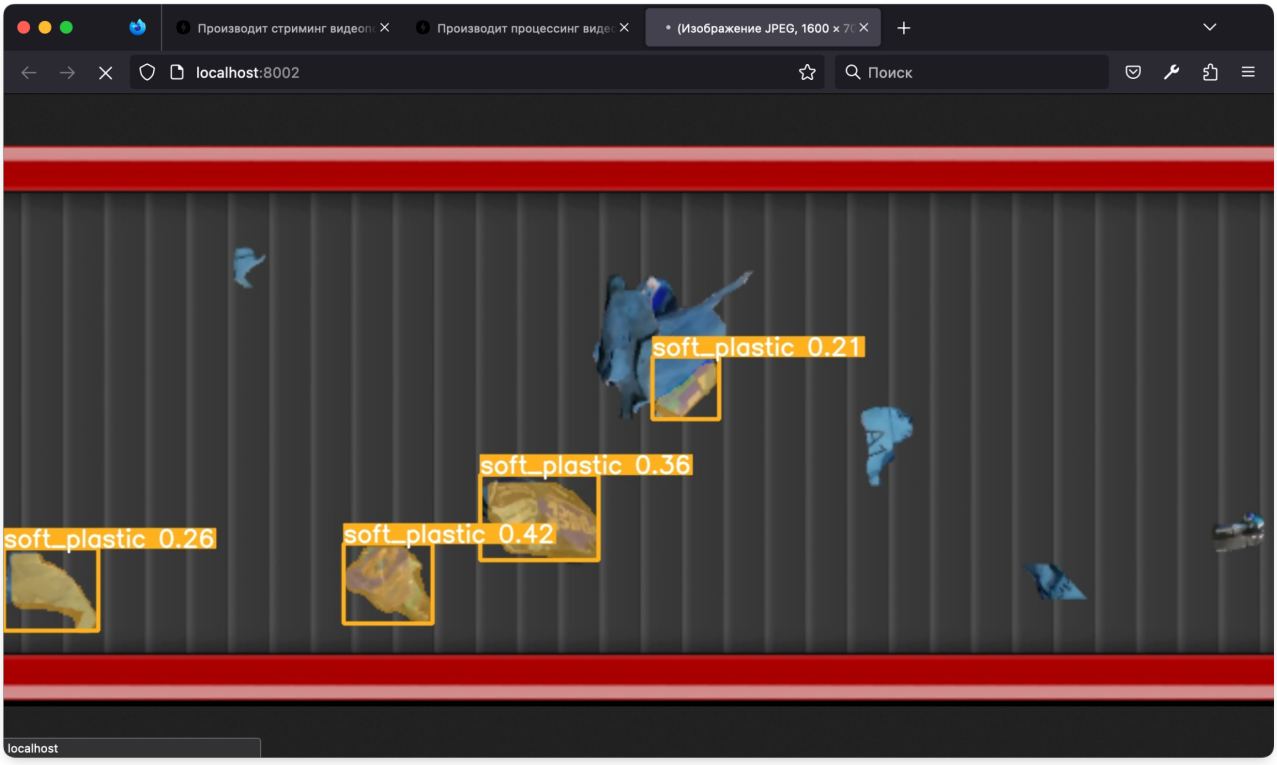
В этом примере показано, что на сервере присутствует 2 GPU NVIDIA RTX A5000, свободна вторая GPU и может использоваться сервисом. `"cuda.is_available": true`.

4.2. Входящий поток ImageZMQ

Сервис принимает входящий поток изображений OpenCV на порт 5500. Минимальный код на Python, реализующий поток передачи изображений приведён в [пункте 2.1. настоящего руководства]. (#21)

4.3. Исходящий поток фреймов с визуализацией изображения

Сервис транслирует фреймы видеопотока с визуализацией распознавания на порт 8002. Если при запущенном сервисе открыть ссылку в браузере http://127.0.0.1:8002, то можно увидеть исходящий поток.



5. Ссылки

1. Протокол передачи сообщений ZeroMQ

2. Использование imageZMQ в проектах распределенного компьютерного зрения
3. Спецификация открытого API
4. Редактор открытого API

Перечень терминов и сокращений

Термин/сокращение	
Сервис	Сервис инференса MarQus нейросетевых моделей для автоматизации процессов распознавания и сортировки различных типов ТБО.
СИ MarQus	Сервис инференса MarQus нейросетевых моделей для автоматизации процессов распознавания и сортировки различных типов ТБО.
API	Программный интерфейс приложения, интерфейс прикладного программирования (англ. application programming interface) – описание способов (набор классов, процедур, функций, структур или констант), которыми одна компьютерная программа может взаимодействовать с другой программой. Используется программистами при написании всевозможных приложений.
REST	(англ. Representational State Transfer — «передача репрезентативного состояния» или «передача „самоописываемого“ состояния») — архитектурный стиль взаимодействия компонентов распределённого приложения в сети.
Docker	ПО с открытым исходным кодом для автоматизации развёртывания и управления Программы с использованием технологии контейнеризации. В состав «Docker» включен пакетный менеджер «Docker Compose», обеспечивающий запуск многоконтейнерных приложений.
JSON	(англ. JavaScript Object Notation) – текстовый формат обмена данными, основанный на JavaScript.
OpenSource	Открытое программное обеспечение (англ. open-source software) – программное обеспечение с открытым исходным кодом. Исходный код таких программ доступен для просмотра, изучения и изменения, что позволяет убедиться в отсутствии уязвимостей и неприемлемых для пользователя функций, принять участие в доработке самой открытой программы, использовать код для создания новых программ и исправления в них ошибок – через заимствование исходного кода, если это позволяет совместимость лицензий, или через изучение использованных алгоритмов, структур данных, технологий, методик и интерфейсов (поскольку исходный код может существенно дополнять документацию, а при отсутствии таковой – сам служит документацией).
TCP	Сетевая модель передачи данных, представленных в цифровом виде. Модель описывает способ передачи данных от источника информации к получателю. В модели предполагается прохождение информации через четыре уровня, каждый из которых описывается правилом (протоколом передачи). Наборы правил, решающих задачу по передаче данных, составляют стек протоколов передачи данных, на которых базируется Интернет. Название TCP/IP происходит из двух основных протоколов семейства – Transmission Control Protocol (TCP) и Internet Protocol (IP), которые были первыми разработаны и описаны в данном стандарте.
ZeroMQ , ZMQ	Высокопроизводительная асинхронная библиотека обмена сообщениями, ориентированная на использование в распределённых и параллельных вычислениях. Библиотека реализует очередь сообщений, которая может функционировать без выделенного брокера сообщений.

Термин/сокращение	
ТБО	Твёрдые бытовые отходы.