

Министерство цифрового развития, связи и массовых коммуникаций Российской  
Федерации

Ордена Трудового Красного Знамени

федеральное государственное бюджетное образовательное учреждение высшего  
образования

МОСКОВСКИЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ СВЯЗИ И ИНФОРМАТИКИ

Кафедра «Математическая кибернетика и информационные технологии»

Введение в ИТ.

Введение в информационные технологии

Разработка веб-сервиса на основе нейросетевой модели обнаружения с отслеживанием  
объектов.

Выполнили:

студенты группы БВТ2108

Старостин И. Д.,

Разумовский И. Д.,

Юношева К. Р.,

Комлев А. В.,

Томилов Ю. А.,

Чугунова Т. А.

Москва2022

1. Использование сверточной нейронной сети YOLO для детектирования объектов. Выгружаем Dataset разметки (Рис. 1-2). Обучение нейросети за 199 эпох (Рис. 3 – 5).

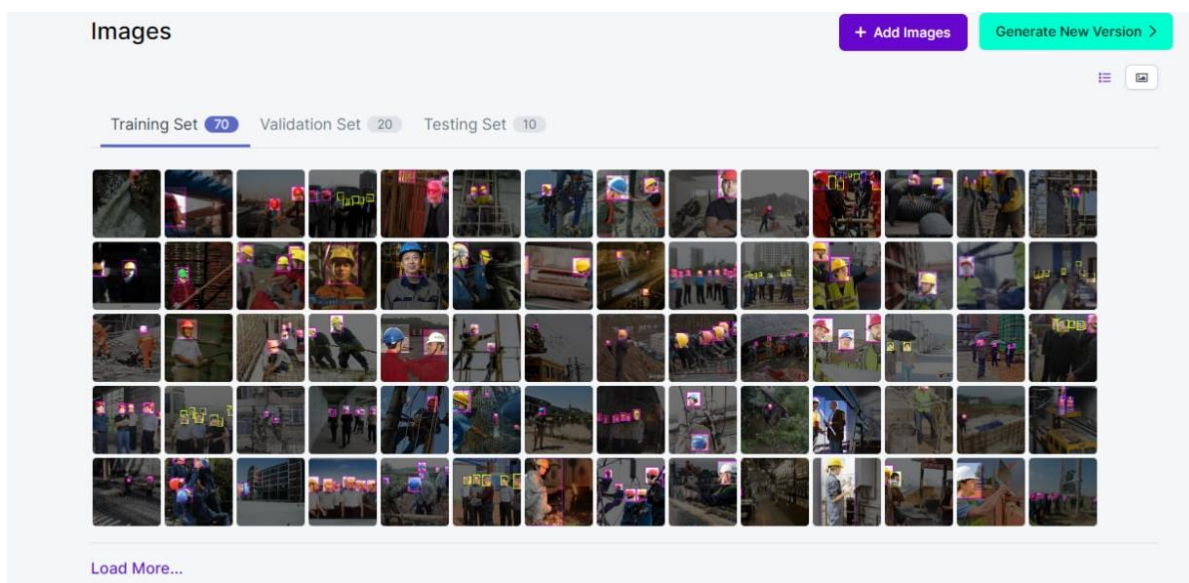


Рис.1

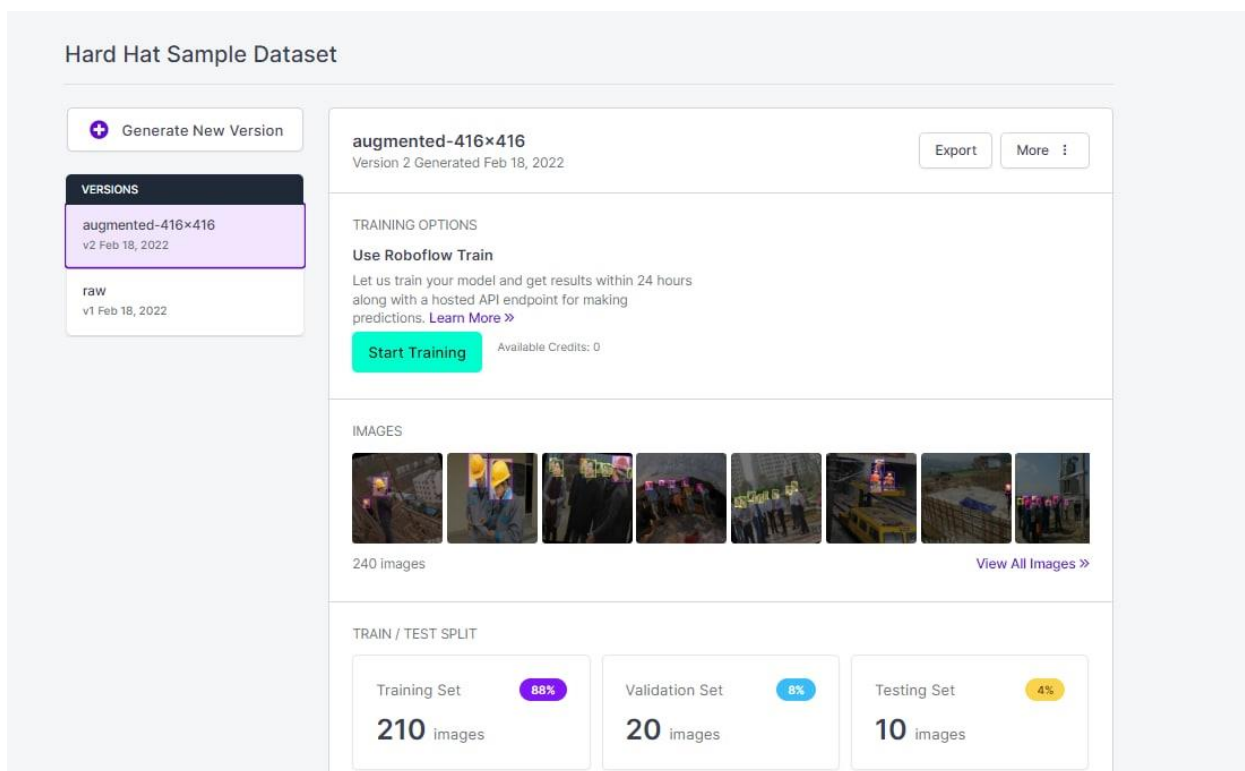


Рис. 2

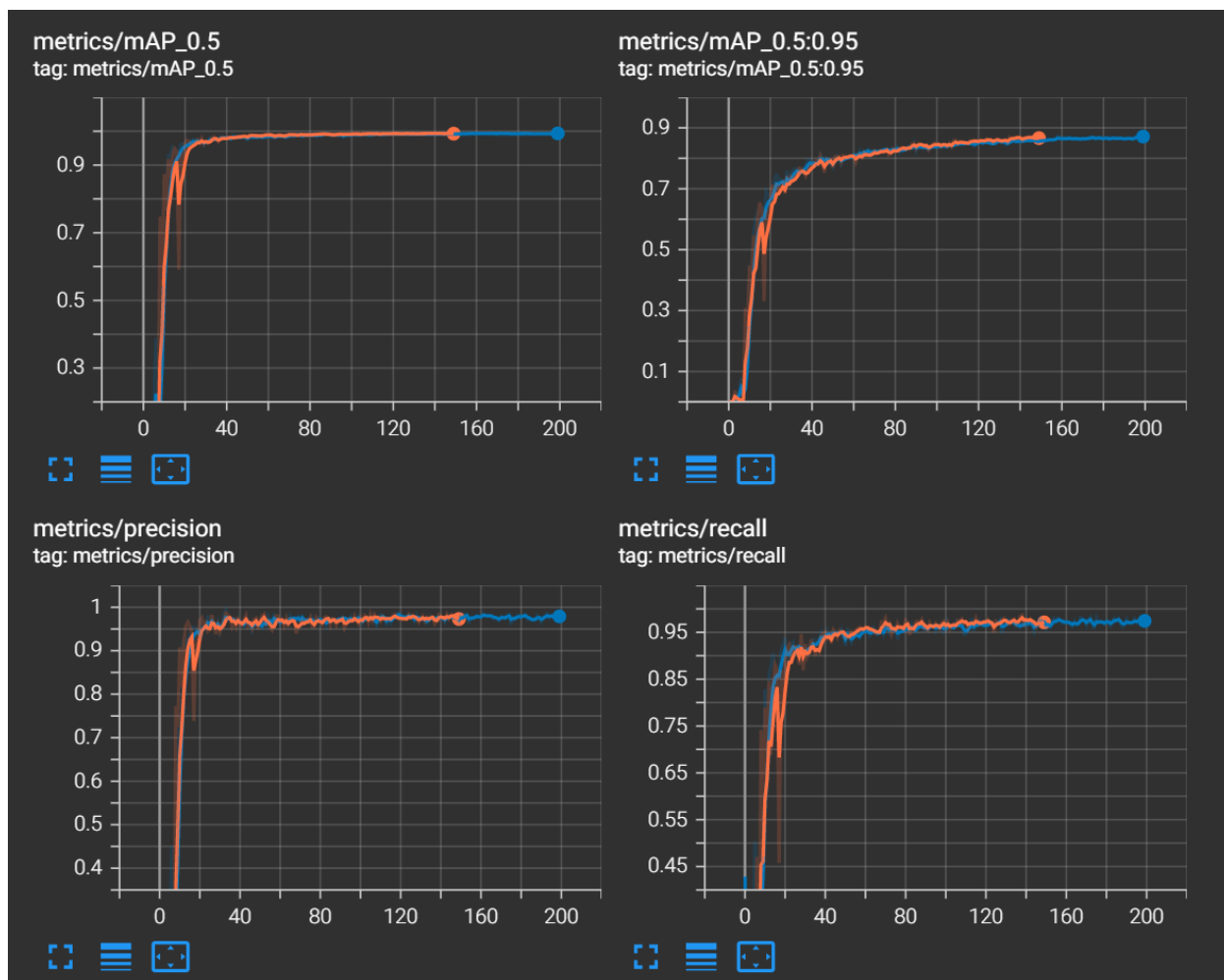


Рис. 3

Epoch	gpu_mem	box	obj	cls	total	targets	img_size
1/199	1.39G	0.07972	0.02273	0	0.1025	40	416: 100% 134/134 [00:18<00:00, 7.25it/s]
	Class	Images	Targets		P	R	mAP@.5 mAP@.5:.95: 100% 19/19 [00:03<00:00, 5.75it/s]
	all	601	612	0.00301	0.105	0.000876	0.00012
Epoch	gpu_mem	box	obj	cls	total	targets	img_size
2/199	1.39G	0.07749	0.02324	0	0.1007	40	416: 100% 134/134 [00:18<00:00, 7.42it/s]
	Class	Images	Targets		P	R	mAP@.5 mAP@.5:.95: 100% 19/19 [00:03<00:00, 5.62it/s]
	all	601	612	0.00865	0.263	0.00305	0.000404
Epoch	gpu_mem	box	obj	cls	total	targets	img_size
3/199	1.39G	0.07402	0.02424	0	0.09826	35	416: 100% 134/134 [00:18<00:00, 7.41it/s]
	Class	Images	Targets		P	R	mAP@.5 mAP@.5:.95: 100% 19/19 [00:03<00:00, 5.64it/s]
	all	601	612	0.0974	0.368	0.0567	0.0113

Рис. 4

Epoch	gpu_mem	box	obj	cls	total	targets	img_size
196/199	1.39G	0.0138	0.006737	0	0.02054	41	416: 100% 134/134 [00:17<00:00, 7.51it/s]
	Class	Images	Targets		P	R	mAP@.5 mAP@.5:.95: 100% 19/19 [00:02<00:00, 6.63it/s]
	all	601	612	0.977	0.977	0.993	0.874
Epoch	gpu_mem	box	obj	cls	total	targets	img_size
197/199	1.39G	0.01436	0.006904	0	0.02127	38	416: 100% 134/134 [00:17<00:00, 7.49it/s]
	Class	Images	Targets		P	R	mAP@.5 mAP@.5:.95: 100% 19/19 [00:02<00:00, 6.64it/s]
	all	601	612	0.977	0.977	0.994	0.875

Рис. 5

2. Результат обработки (Рис. 6-7).

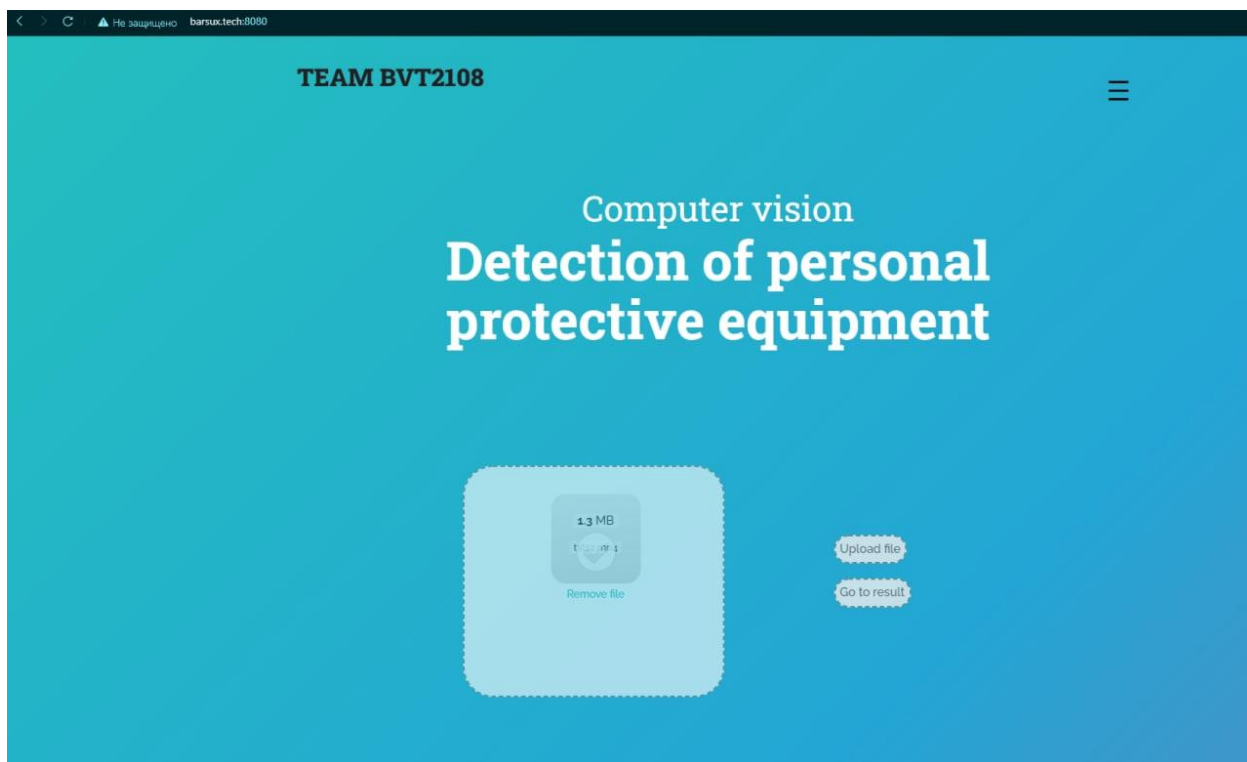


Рис. 6

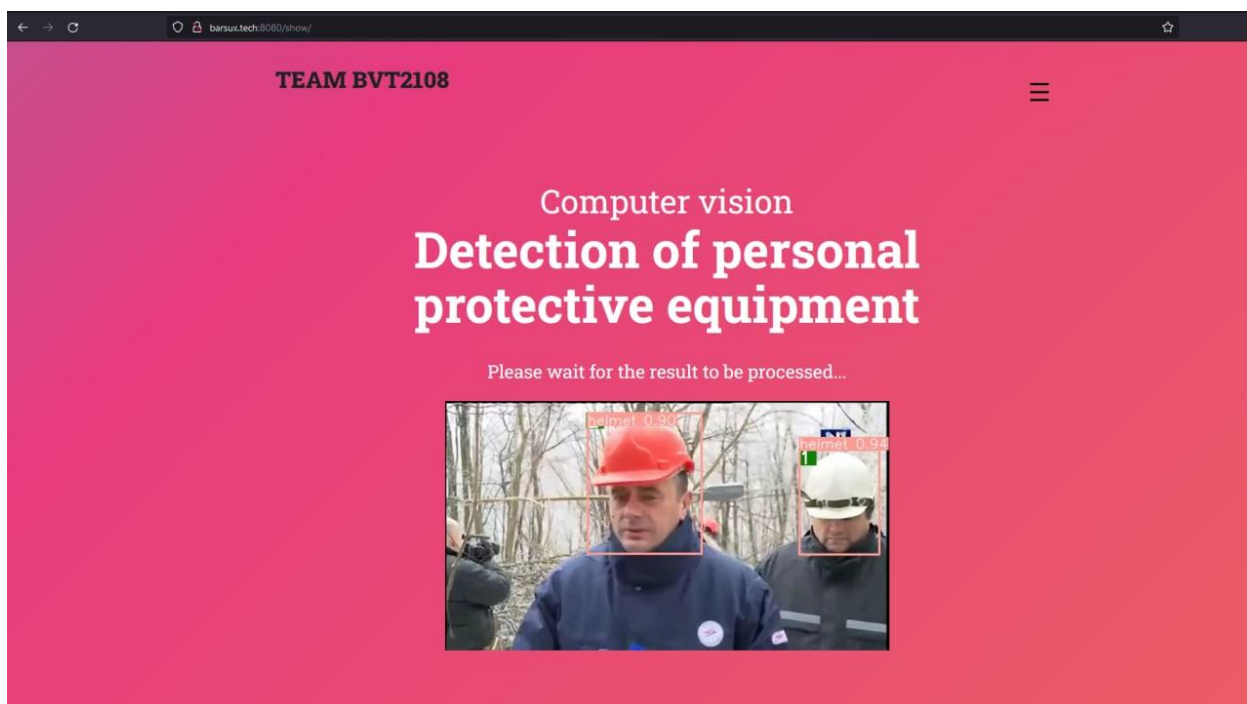
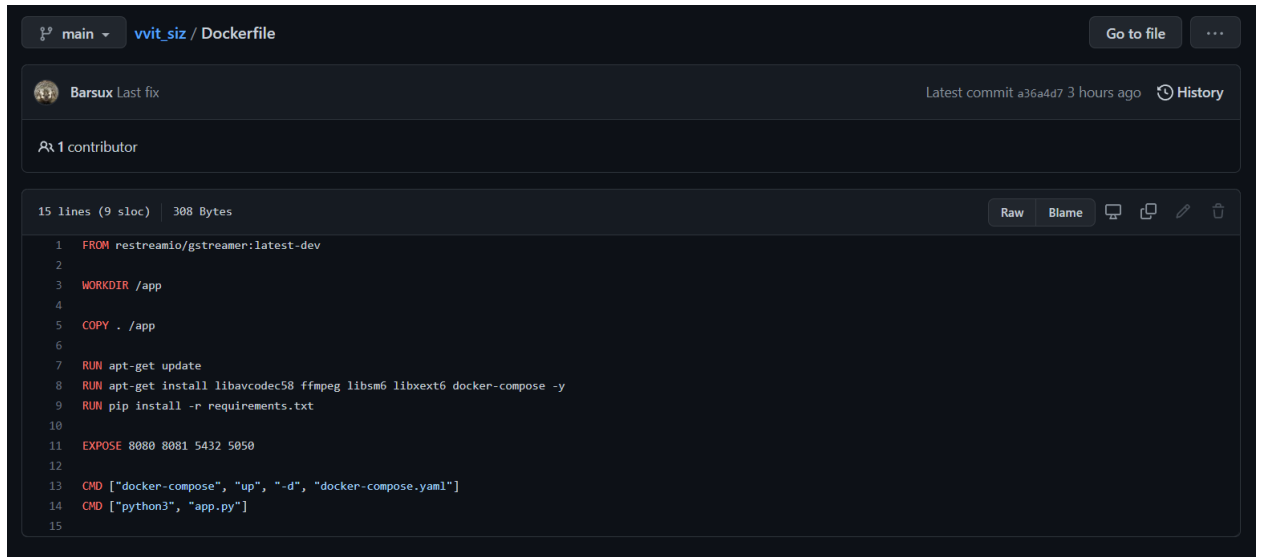


Рис. 7

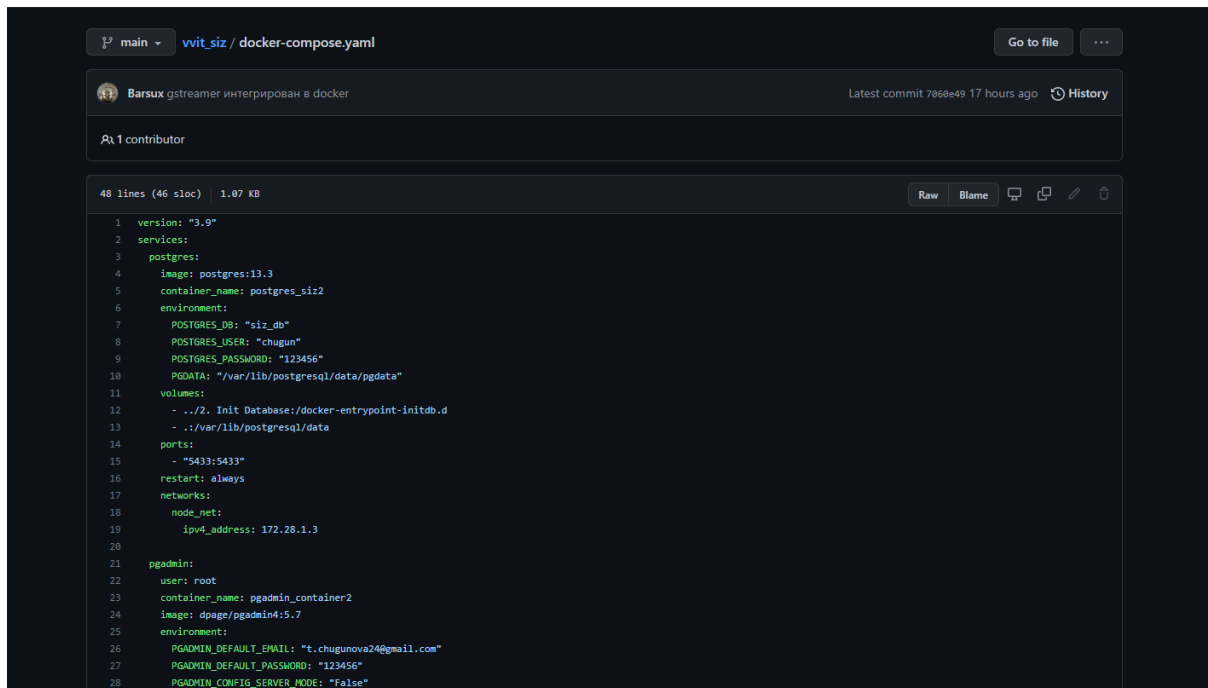
3. Использование контейнера Docker для развертывания на сервер. В Dockerfile мы находим контейнер с gstream, скачиваем его, создаем и копируем директорию и в

командной строке обновляем пакеты, скачиваем библиотеки, устанавливаем зависимости, далее открываем 4 порта и после выполняем инструкции прописанные в `docker-compose.yml` и запускаем скрипт `app.py`, запускающий сайт (Рис. 8). В `docker-compose.yml` скачиваем два образа и настраиваем их (задаем бд, создаем пользователя с паролем, открываем порты и задаем ip адрес бд, к которому мы будем обращаться). В `pgadmin` мы указываем имя контейнера и версию приложения, а также ip-адреса, через которые подключаемся к этим контейнерам (Рис. 9-10).



```
1 FROM restreamio/gstreamer:latest-dev
2
3 WORKDIR /app
4
5 COPY . /app
6
7 RUN apt-get update
8 RUN apt-get install libavcodec58 ffmpeg libsm6 libxext6 docker-compose -y
9 RUN pip install -r requirements.txt
10
11 EXPOSE 8080 8081 5432 5050
12
13 CMD ["docker-compose", "up", "-d", "docker-compose.yml"]
14 CMD ["python3", "app.py"]
15
```

Рис. 8



```
1 version: "3.9"
2 services:
3   postgres:
4     image: postgres:13.3
5     container_name: postgres_siz2
6     environment:
7       POSTGRES_DB: "siz_db"
8       POSTGRES_USER: "chugun"
9       POSTGRES_PASSWORD: "123456"
10    PGDATA: "/var/lib/postgresql/data/pgdata"
11    volumes:
12      - ../2. Init Database:/docker-entrypoint-initdb.d
13      - ./var/lib/postgresql/data
14    ports:
15      - "5433:5433"
16    restart: always
17    networks:
18      node_net:
19        ipv4_address: 172.28.1.3
20
21   pgadmin:
22     user: root
23     container_name: pgadmin_container2
24     image: dpage/pgadmin4:5.7
25     environment:
26       PGADMIN_DEFAULT_EMAIL: "t.chugunova24@gmail.com"
27       PGADMIN_DEFAULT_PASSWORD: "123456"
28       PGADMIN_CONFIG_SERVER_MODE: "False"
```

Рис. 9

```
29 volumes:
30   - ${DATA_PATH_HOST}/pgadmin:/var/lib/pgadmin
31 ports:
32   - "5050:80"
33 restart: unless-stopped
34 deploy:
35   resources:
36     limits:
37       cpus: '0.5'
38       memory: 1G
39 networks:
40   node_net:
41     ipv4_address: 172.28.1.4
42
43 networks:
44   node_net:
45     ipam:
46       driver: default
47       config:
48         - subnet: 172.28.0.0/16
```

Рис. 10

#### 4. Использование базы данных PostgreSQL для сбора статистики (Рис. 11-14).

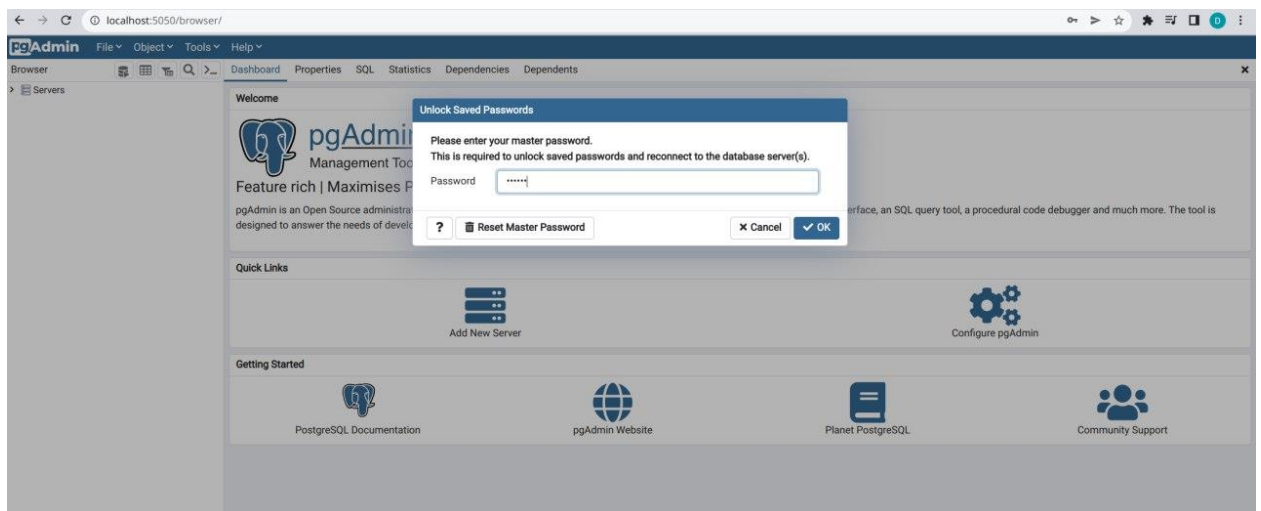


Рис. 11

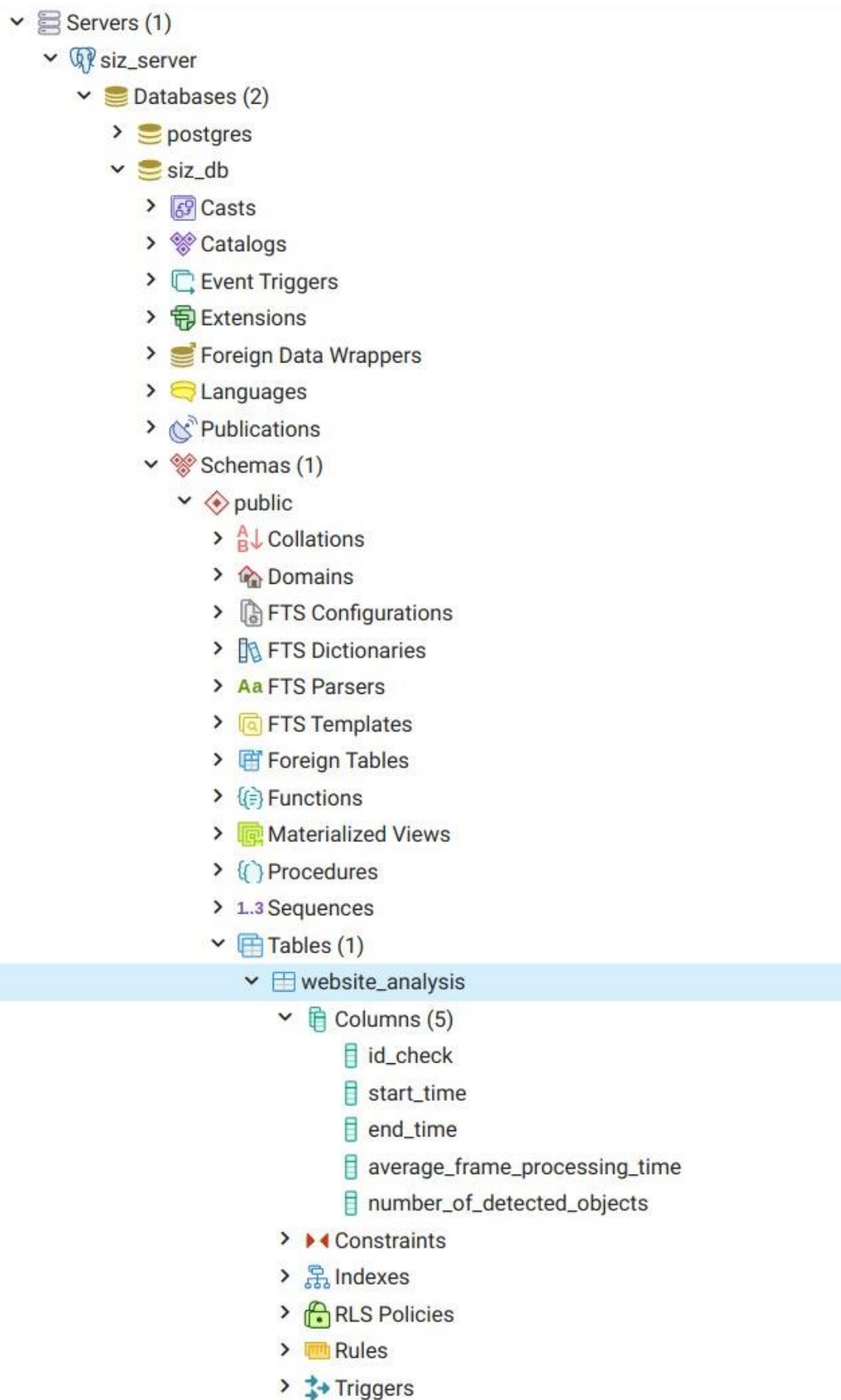


Рис. 12

siz\_db/chugun@siz\_server

website\_analysis

General Columns Advanced Constraints Parameters Security SQL

Name website\_analysis

Owner chugun

Schema public

Tablespace pg\_default

Partitioned table? No

Comment

Cancel Reset Save

Рис. 13

siz\_db/chugun@siz\_server

website\_analysis

General Columns Advanced Constraints Parameters Security SQL

Inherited from table(s) Select to inherit from...

Columns

	Name	Data type	Length/Precision	Scale	Not NULL?	Primary key?
<input checked="" type="checkbox"/>	id_check	integer			Yes	No
<input checked="" type="checkbox"/>	start_time	numeric			No	No
<input checked="" type="checkbox"/>	end_time	numeric			No	No
<input checked="" type="checkbox"/>	average_frame_processing_time	text			No	No
<input checked="" type="checkbox"/>	number_of_detected_objects	integer			No	No

Cancel Reset Save

Рис. 14



5. Мы используем библиотеку `psycopg2` (инструмент для взаимодействия с `postgres` на `python`) и в функции `init` подключаемся к бд, находящейся в контейнере, в функции `table_exists` узнаем существует ли таблица и при необходимости создаем ее, в функции `insert_data` мы добавляем данные через `sql`-запрос а бд и в конце делаем коммит и завершаем соединение (Рис. 15-17).

```
41 lines (37 sloc) | 1.46 KB
1  import os
2  import uuid
3  import time
4
5  from app import app, WINDOWS
6  from flask import render_template, flash, request, redirect, url_for
7  from threading import Thread
8  from yolov5.detect_track import run
9  Last_video = None
10
11 def run_gst_pipeline(filename):
12     print("Running pipeline")
13     pipeline = f"gst-launch-1.0 filesrc location=tracked/{filename} ! qtdemux ! decodebin ! videoconvert ! videoscale ! theoraenc ! oggmux ! tcpserver sink host=127.0.0.1 port=
14     print(pipeline)
15     time.sleep(2)
16     os.system(pipeline)
17
18 @app.route("/", methods = ["GET", "POST"])
19 def upload():
20     global Last_video
21     if request.method == 'POST':
22         f = request.files.get('file')
23         file_ext = f.filename[f.filename.rfind('.'):].lower()
24         new_filename = uuid.uuid1().hex + file_ext
25         Last_video = new_filename
26         if WINDOWS:
27             new_filename_path = f"{app.config['UPLOADED_PATH']}\\{new_filename}"
28         else:
29             new_filename_path = f"{app.config['UPLOADED_PATH']}/{new_filename}"
30         f.save(os.path.join(app.config['UPLOADED_PATH'], new_filename))
31         detect = Thread(target=run, args=(app.config["WEIGHTS_PATH"], new_filename_path, new_filename, False, False))
32         detect.start()
33         detect.join()
34     return render_template('index.html')
35
36 @app.route("/show/", methods = ["GET", "POST"])
37 def show():
38     global Last_video
39     if Last_video:
40         Thread(target=run_gst_pipeline, args=(Last_video,)).start()
41     return render_template("show.html")
```

Рис. 15

```
<div id="video_stream">
<h4 style="color: white; font-family: 'Roboto Slab', serif;">Please wait for the result to be processed...</h4>
<video controls autoplay muted preload="none" >
    <source src="http://127.0.0.1:8081">
</video>
</div>
```

Рис. 16

Computer vision  
**Detection of personal  
protective equipment**

Please wait for the result to be processed...



Рис. 17. Отображение выполненной работы