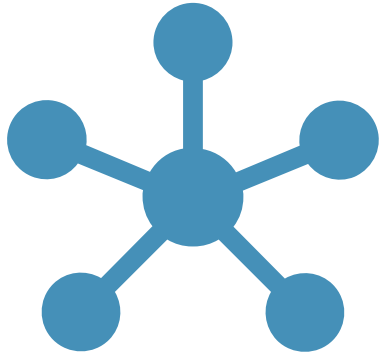


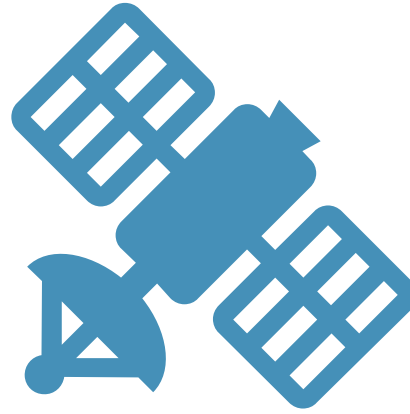


VISIONGATE

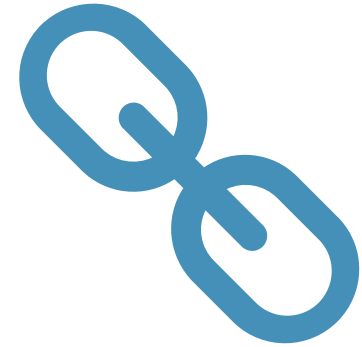
ИНФРАСТРУКТУРА СИСТЕМЫ АВТОМАТИЗАЦИИ КОНТРОЛЯ ДОСТУПА НА ОСНОВЕ РАСПОЗНАВАНИЯ ИЗОБРАЖЕНИЙ



Распознавание
номеров



Управление списком
доступа



Аналитика и просмотр
событий

ВОЗМОЖНОСТИ

СХЕМА РАБОТЫ СИСТЕМЫ



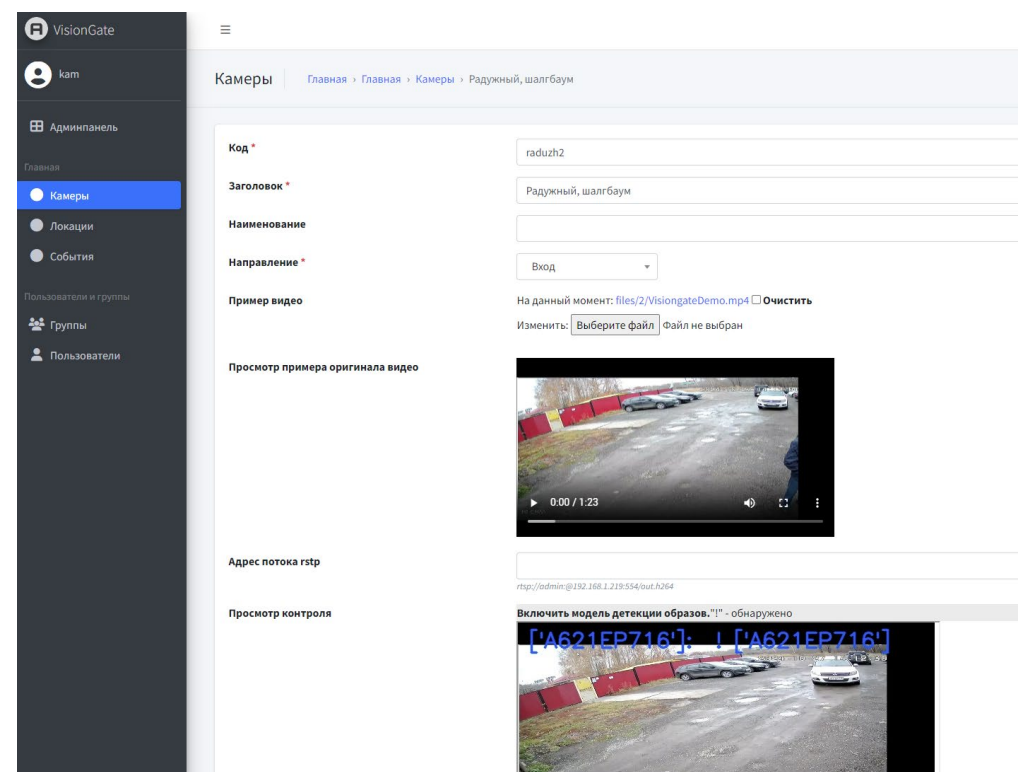
YOLO распознавание блоков номерных знаков

OCR распознавание текста внутри фреймов

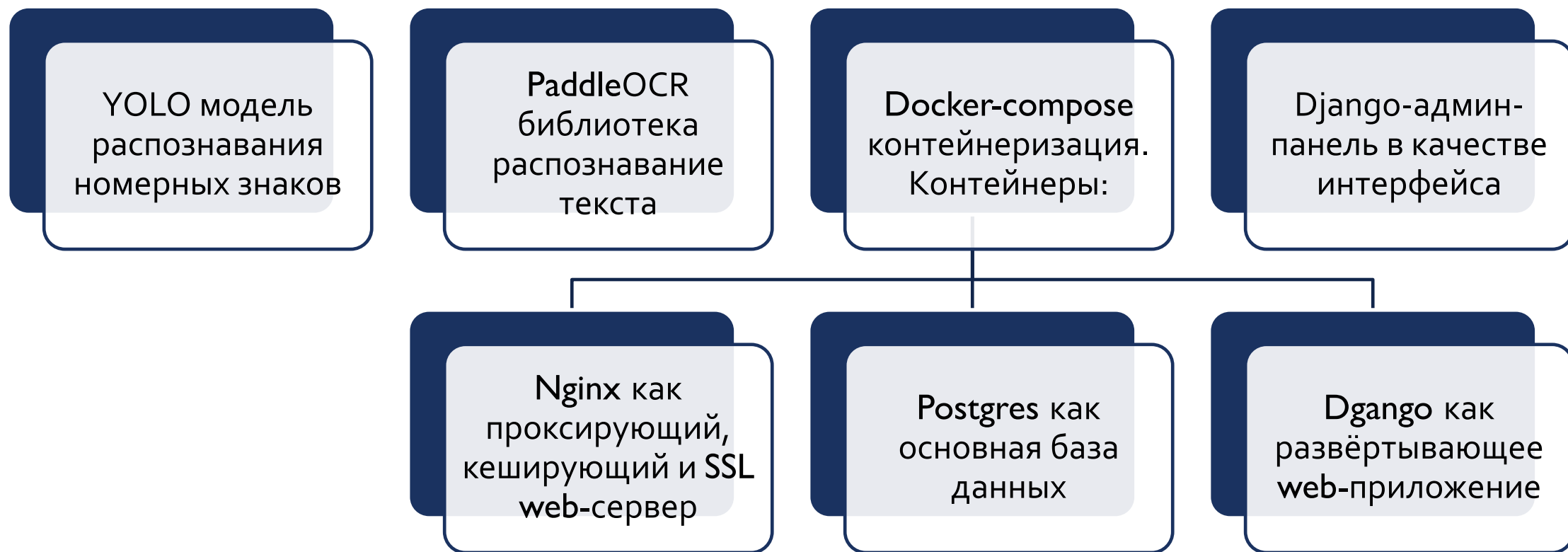
Django-админ-панель как простая система
конфигурирования

БИЗНЕС-ЦЕННОСТЬ И РЕШАЕМАЯ ПРОБЛЕМА

- 1. Локальное или облачное решение по контролю доступа на базе стандартного ПК
- 2. Возможность подключения любых rtsp-источников, например rtsp-ip-камер
- 3. Аналитика событий: и в web-интерфейсе, и в БД Postgres
- 4. Возможность удобной настройки разрешённых номеров в web-интерфейсе
- 5. Возможность расширения YOLO модели распознавания образов посредством дообучения
- 6. Возможность расширить функционал, например реализовать распознавание лиц при наличии GPU
- 7. Выгрузка CSV-файла событий

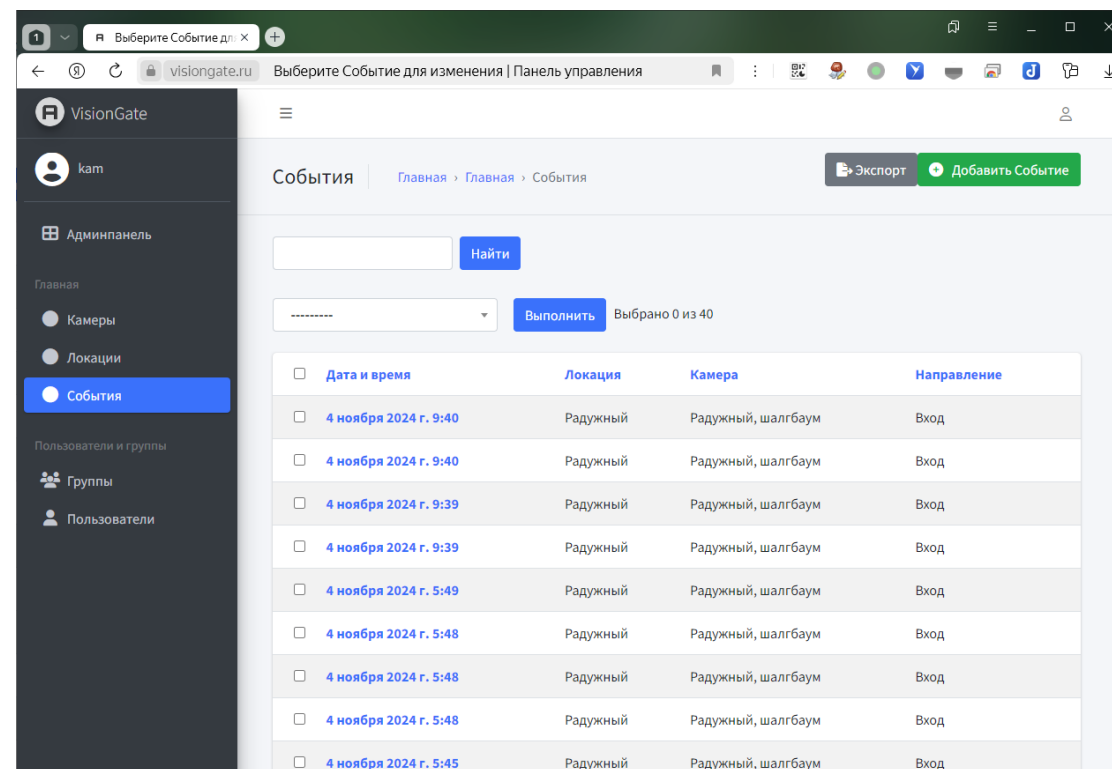


ПРИМЕНЯЕМЫЕ ТЕХНОЛОГИИ



ПРИНЦИПИАЛЬНАЯ АРХИТЕКТУРА СИСТЕМЫ

1. Распознавание bounding-box образов номерных знаков на основе YOLO модели, обученной на распознавание номерных знаков
2. OCR-распознавание текста внутри bounding-box зон кадра
3. Django-админ-панель в качестве развёртывания самой модели и настройки управления системой
4. Локации - сущность, где устанавливаются источники видео-потоков
5. Локации содержат списки Разрешённых номерных знаков (соответствующее поле-список)
6. Камеры - привязываются к Локации, имеют Направление, а также и включают rstp ссылки и примеры файлов видеопотока
7. События - журнал распознанных в Локации объектов-номеров, при совпадении из списка из Разрешённых (на скриншоте)



YOLO МОДЕЛЬ РАСПОЗНАВАНИЯ НОМЕРНЫХ ЗНАКОВ

- YOLO-совместимый датасет разметки номерных знаков
<https://universe.roboflow.com/roboflow-universe-projects/license-plate-recognition-rxg4e/dataset/4>
- Обучение модели на распознавание одного класса «**licence_plate**»
- Выбрана модель по умолчанию библиотеки YOLO
- Достигнуты метрики $dfl_loss=0.952$
- Дообучение на альтернативных датасетах позволит улучшить распознавание при необходимости
- Пример распознанного класса:



РАБОТА С ONNX МОДЕЛЬЮ НА ИНФЕРЕНСЕ В DJANGO

1. Технология ONNX: библиотека onnxruntime, экспорт модели: `model.export(format='onnx')`
2. Подключение обученной модели для её использования: `onnxruntime.InferenceSession`
3. Нормировка и преобразование openCV-кадра к нужной размерности 640, и транспонирование для формата ONNX-модели
4. Получение результата работы модели: 8400 боксов классов с их вероятностями
5. Отбор искомых номерных плат через вычисление IOU по пересечению площади 0.7+ и вероятности класса 0.5+

```
# преобразование выходов модели в нужный формат, удаление лишних боксов по вероятности и по IOU
def model_output_to_boxes(model_output, img_width, img_height, prob_threshold, iou_threshold) -> List[List[Union[int, float]]]:
    detections = model_output[0]
    # перестановка размерностей выходов модели для удобства, чтобы кол-во боксов стояло нулевой размерностью
    detections = detections.transpose()

    # конвертация детекций в формат [4, координаты, индекс_класса, вероятность_класса]
    detections = [convert_detections(detection, img_width, img_height) for detection in detections]
    # отсевание только боксов вероятность которых более prob_threshold (0.5)
    boxes = [detection for detection in detections if detection[5] > prob_threshold]
    # сортировка по убыванию боксов по вероятности
    boxes.sort(key=lambda x: x[5], reverse=True)

    # итоговый список для боксов который будет возвращаться
    boxes_list = []
    # удаление лишних боксов (которые сильно пересекаются), пока не останется ни одного
    while len(boxes) > 0:
        # добавляем первый самый вероятный бокс в итоговый список
        boxes_list.append(boxes[0])
        # из остальных оставляем только те которые пересекаются с текущим не сильно (меньше iou_threshold)
        boxes = [box for box in boxes if iou(box, boxes[0]) < iou_threshold]
    return boxes_list

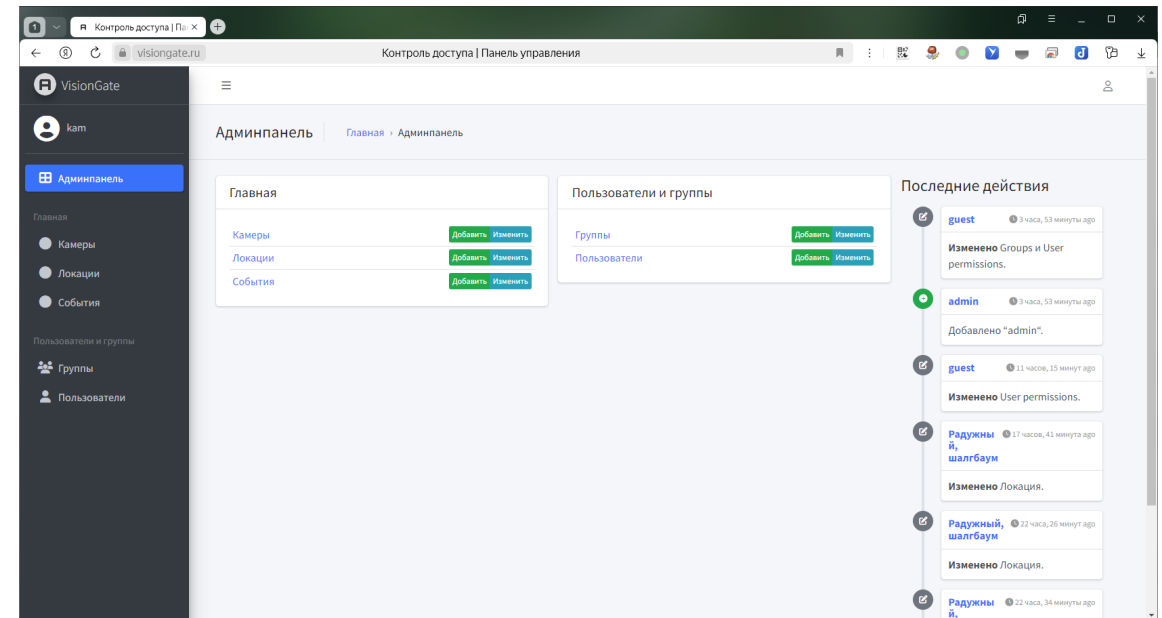
# порог вероятности для отсеивания боксов
prob_threshold = 0.5
# порог IOU для отсеивания боксов
iou_threshold = 0.7

# получение высоты и ширины изображения
img_height = cv2_image.shape[0]
img_width = cv2_image.shape[1]

# проверка работы функции преобразования выходов модели в нужный формат
boxes_list = model_output_to_boxes(outputs, img_width, img_height, prob_threshold, iou_threshold)
boxes_list
```


ОСНОВНЫЕ ФАЙЛЫ, РЕАЛИЗУЮЩИЕ РАБОТУ МОДЕЛИ И ПРИЛОЖЕНИЯ

- visiongate/main/models.py - схема данных
- visiongate/main/views.py - работа с onnx моделью в эндпойнте Django
- visiongate/main/numberplate.py - iou-вычисление bounding-box и запуск PaddleOCR visiongate/main/admin.py - интерфейсная часть админ-панели
- Итоговая_Аттестация.ipynb - обучение модели, выгрузка ONNX, принципиальная проверка работоспособности https://colab.research.google.com/drive/1IGudyk6Hvj-adFBQs86s-ol0WK_IQwB4?usp=drive_link
- Colab-демонстрация работы модели и распознавания



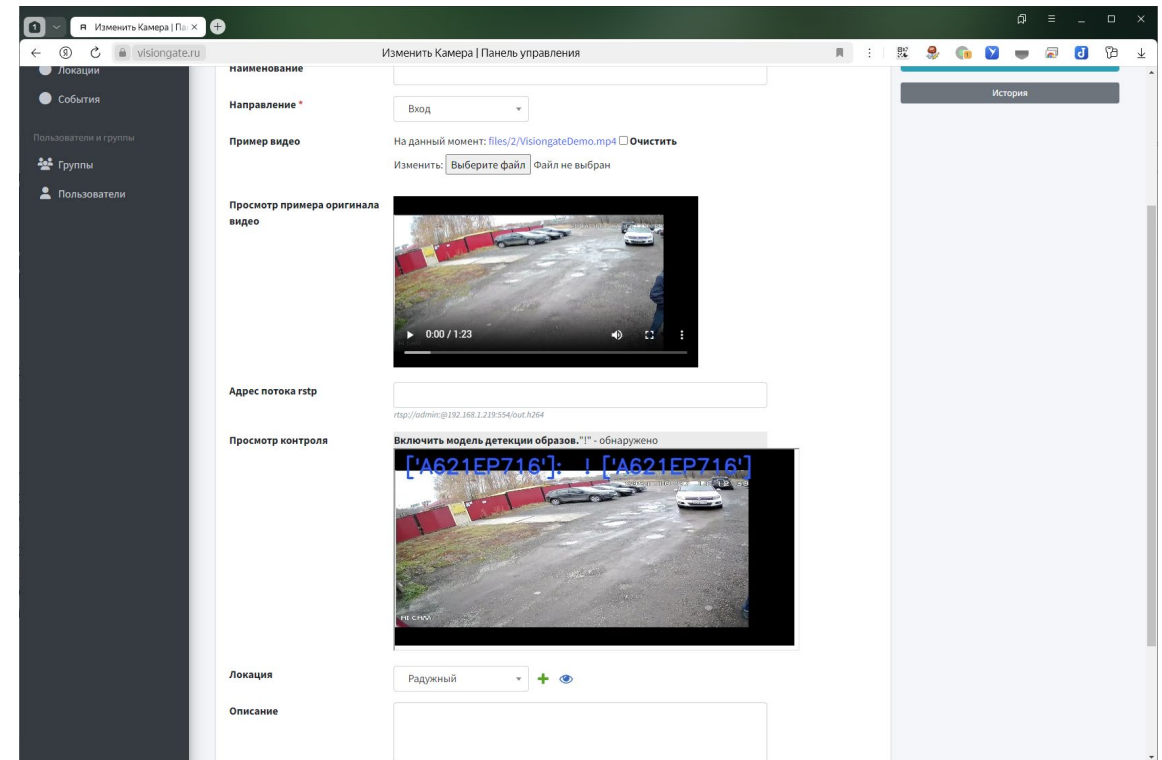
НАСТРОЙКА ЛОКАЦИЙ И КАМЕР

1. Авторизоваться на сайте <https://visiongate.ru/admin>
2. Перейти в раздел Локации
<https://visiongate.ru/admin/main/location/> ,
нажать Создать
3. Указать в поле "Разрешённые" построчный список разрешённых номерных знаков
4. Перейти в раздел Камеры
<https://visiongate.ru/admin/main/camera/> ,
нажать Создать
5. Указать Локацию, Направление (вход-выход),
загрузить демо-ролик и указать rtsp ссылку на камеру

The screenshot shows the 'Добавить Камера' (Add Camera) form in the VisionGate admin panel. The interface is in Russian. On the left is a sidebar with navigation links: 'Админпанель' (Admin Panel), 'Главная' (Home), 'Камеры' (Cameras - selected), 'Локации' (Locations), 'События' (Events), 'Пользователи и группы' (Users and Groups), 'Группы' (Groups), and 'Пользователи' (Users). The main content area is titled 'Добавить Камера | Панель управления' (Add Camera | Control Panel). It contains several input fields: 'Код *' (Code), 'Заголовок *' (Title), 'Наименование' (Name), 'Направление *' (Direction) with a dropdown menu showing 'Вход' (Entrance), 'Пример видео' (Video Example) with a file selection button, 'Просмотр примера оригинала видео' (View original video example), 'Адрес потока rtsp' (RTSP stream address) with a pre-filled URL, 'Просмотр контроля' (View control) with a checkbox for 'Включить модель детекции образов' (Enable image detection model), 'Локация' (Location) with a dropdown and location selection icons, and 'Описание' (Description) with a text area. On the right side of the form are three buttons: 'Сохранить' (Save), 'Сохранить и добавить другой объект' (Save and add another object), and 'Сохранить и продолжить редактирование' (Save and continue editing). The footer shows 'Copyright © 2024. Все права защищены.' (All rights reserved.) and 'Jazzmin Version 3.0.1'.

ПРОСМОТР АНАЛИЗА ВИДЕОПОТОКА

1. Выбрать любую из камер
<https://visiongate.ru/admin/main/camera/>
2. Открыть детальную информацию по камере, например
<https://visiongate.ru/admin/main/camera/2/change/>
3. Демонстрационное видео можно посмотреть в блоке Просмотр пример видео
4. Демонстрацию работы ML-модели распознавания bounding-box образов номерного знака, и последующего распознанного текста - можно посмотреть в блоке Просмотр контроля



УСТАНОВКА НА СЕРВЕРЕ

- Репозиторий github
<https://github.com/ikamil/visiongate>
- Сервер, консоль с root или docker доступом

Скачивание и запуск docker compose сборки

```
1. git clone https://github.com/ikamil/visiongate.git
2. cd visiongate
3. docker compose up -d
```

Создание БД

```
root@lkwuthwruх:~# docker exec -it visiongate-pg-1 bash
root@17d007ef0c92:/# su - postgres
postgres@17d007ef0c92:~$ psql
psql (17.0 (Debian 17.0-1.pgdg120+1))
Type "help" for help.

postgres=# create user visiongate password 'visiongate' login;
postgres=# create database visiongate owner visiongate;
```

Опционально: или раскат текущего дампа демо-данных БД

```
root@lkwuthwruх:~# docker exec -it visiongate-pg-1 bash
root@17d007ef0c92:/# su - postgres
postgres@17d007ef0c92:~$ cat /tmp/data/visiongate.sql > psql visiongate
```

Опционально: или создание админ-юзера и раскатка миграций

```
root@lkwuthwruх:~# docker exec -it visiongate-python-1 bash
root@8225c6af5bbb:/# python /code/visiongate/manage.py createsuperuser
root@8225c6af5bbb:/# python /code/visiongate/manage.py makemigrations
root@8225c6af5bbb:/# python /code/visiongate/manage.py migrate
```



СПАСИБО ЗА
ВНИМАНИЕ!

VISIONGATE.RU