# VISIONGATE

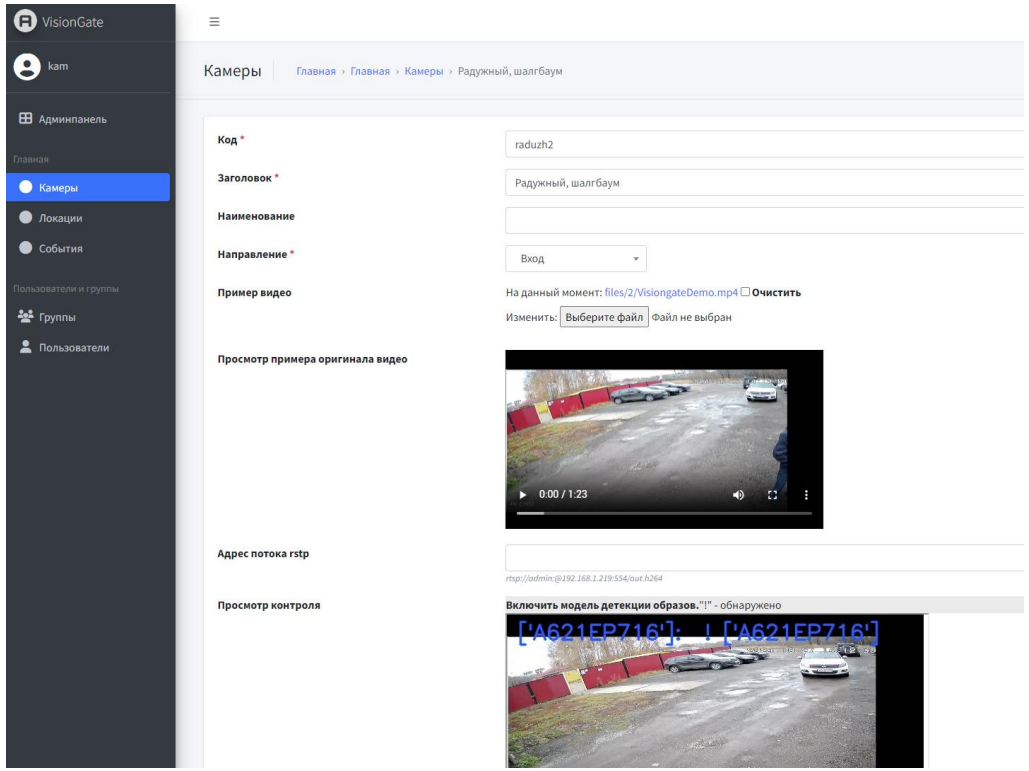
КОНТРОЛЬ ДОСТУПА ПО РАСПОЗНАВАНИЮ НОМЕРОВ

# РЕАЛИЗАЦИЯ СИСТЕМЫ РАСПОЗНАВАНИЯ НОМЕРОВ VISIONGATE

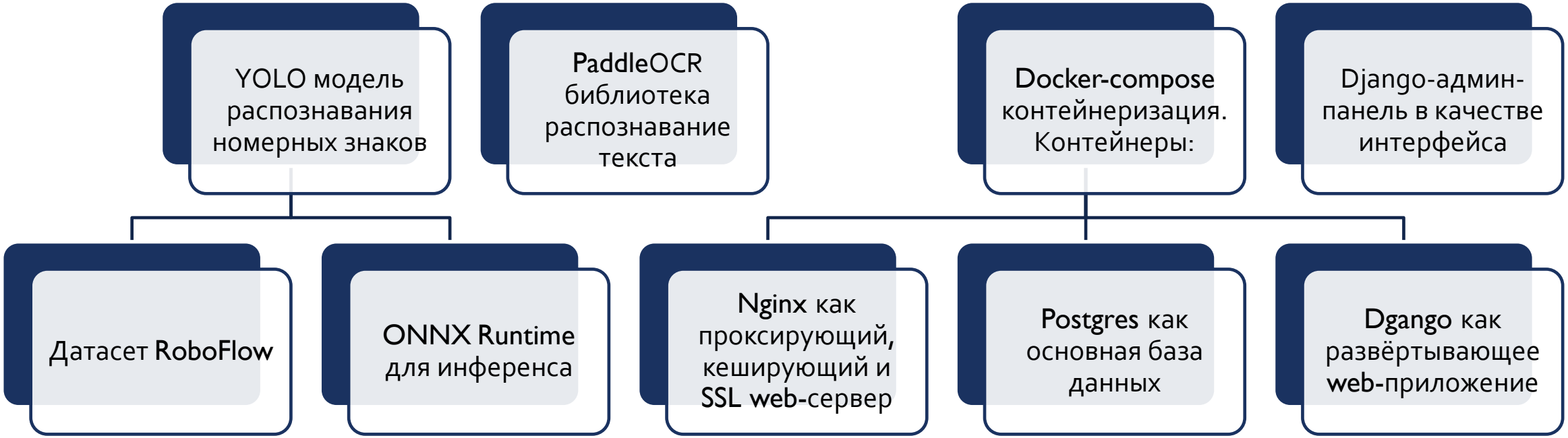
- VisionGate – это распознавание номеров авто для автоматизации проезда через шлагбаумы
- Последовательность реализации проекта
  - Обучить YOLO модель на датасете из roboflow и выгрузить её в ONNX для инференса в web-приложении
  - Опубликовать web-приложение в Django-Admin, и упаковать его в docker-compose
- Использовать следующие технологии Computer Vision:
  - YOLO для распознавания номерной платы, детекция единственного класса licence_plate
  - Передать часто изображения выделенной зоны кадра в PaddleOCR для распознавания текста номерного знака
- Демонстрация - настройка списка номеров и просмотр потока видео в Django-Admin-панели
- Упаковать в Django-приложение, которое использует ONNX-формат пред-обученной YOLO модели
- Развернуть систему на сервере с помощью docker-compose и привязать к домену visiongate.ru

# БИЗНЕС-ЦЕННОСТЬ И РЕШАЕМАЯ ПРОБЛЕМА

- 1. Локальное или облачное решение по контролю доступа через распознавание номеров на базе стандартного ПК
- 2. Возможность подключения любых rstp-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 обнаруженных событий

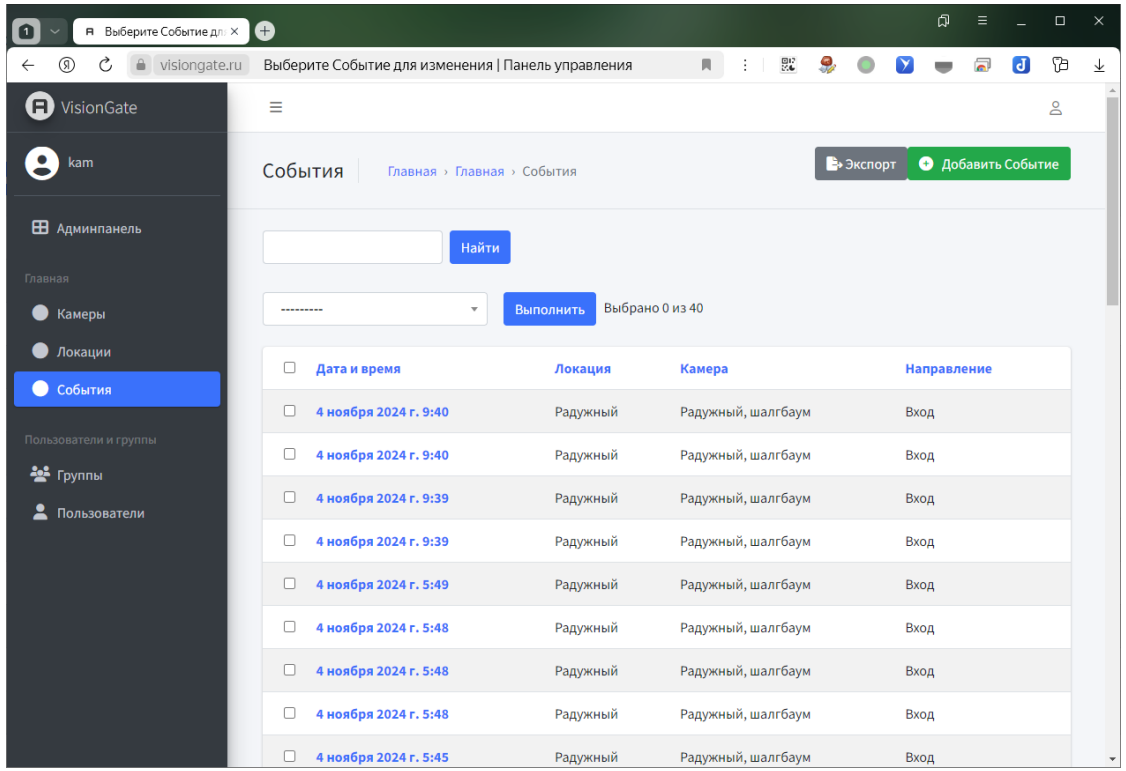


# ПРИМЕНЯЕМЫЕ ТЕХНОЛОГИИ



# ПРИНЦИПИАЛЬНАЯ АРХИТЕКТУРА СИСТЕМЫ

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



# YOLO МОДЕЛЬ РАСПОЗНАВАНИЯ НОМЕРНЫХ ЗНАКОВ

1. Использовать roboflow.com ресурс и забрать YOLO-совместимый датасет разметки номерных знаков https://universe.roboflow.com/roboflow-universe-projects/license-plate-recognition-rxg4e/dataset/4
2. Переобучить ultralytics YOLO модель по-умолчанию (yolo11n.pt) на этом датасете на распознавание единственного класса «**licence_plate**»
3. Достигнуты функции потерь, например dfl_loss=0.952
4. Пример распознанного класса:

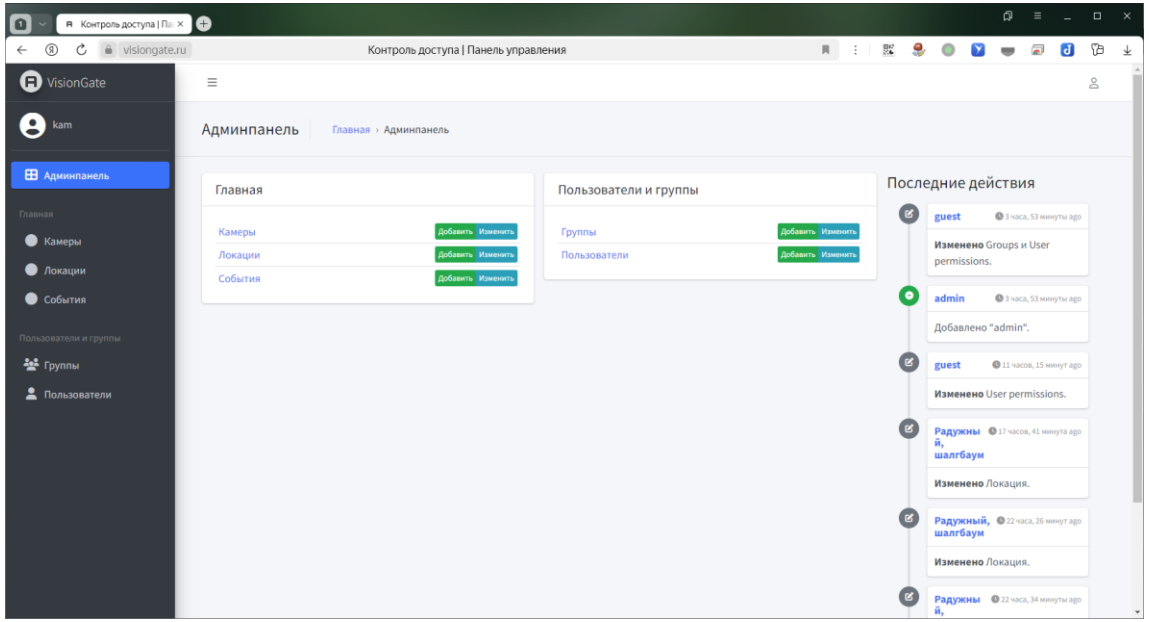
# РАБОТА С ONNX МОДЕЛЬЮ НА ИНФЕРЕНСЕ В DJANGO

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+

```python
# 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]
    # отвеивание 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 возвращатся
    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
```

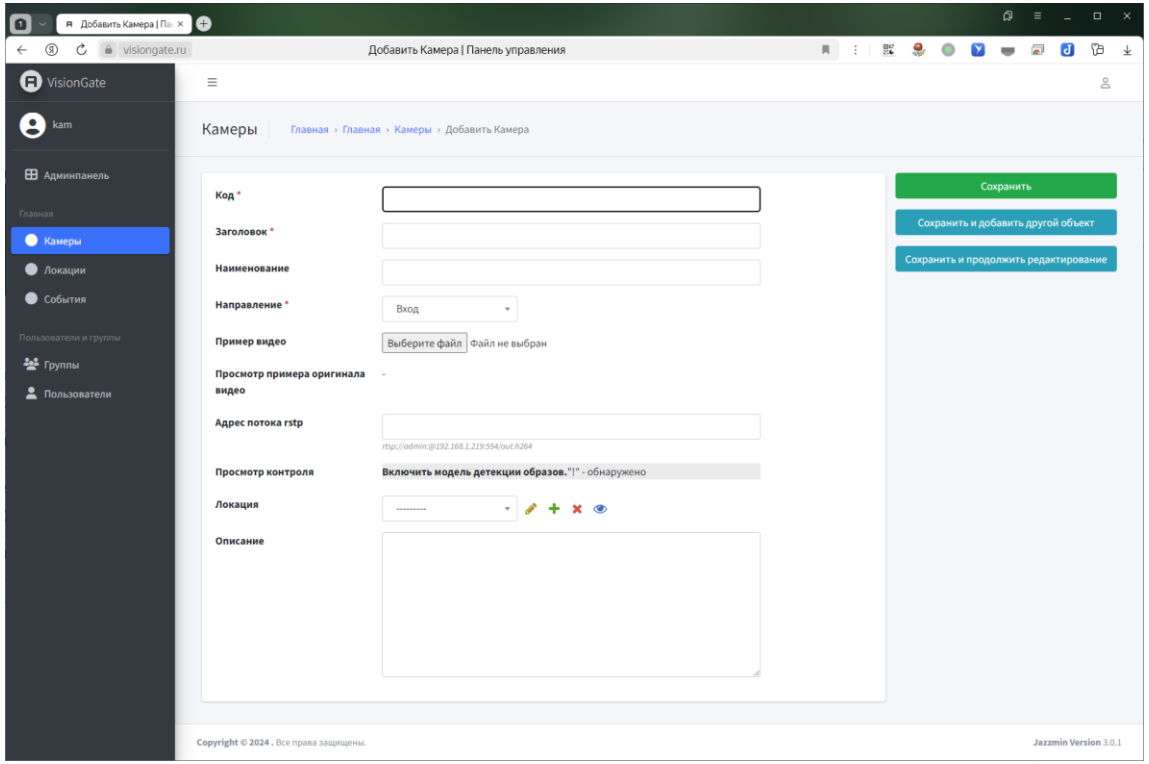
# ОСНОВНЫЕ ФАЙЛЫ, РЕАЛИЗУЮЩИЕ РАБОТУ МОДЕЛИ И ПРИЛОЖЕНИЯ

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



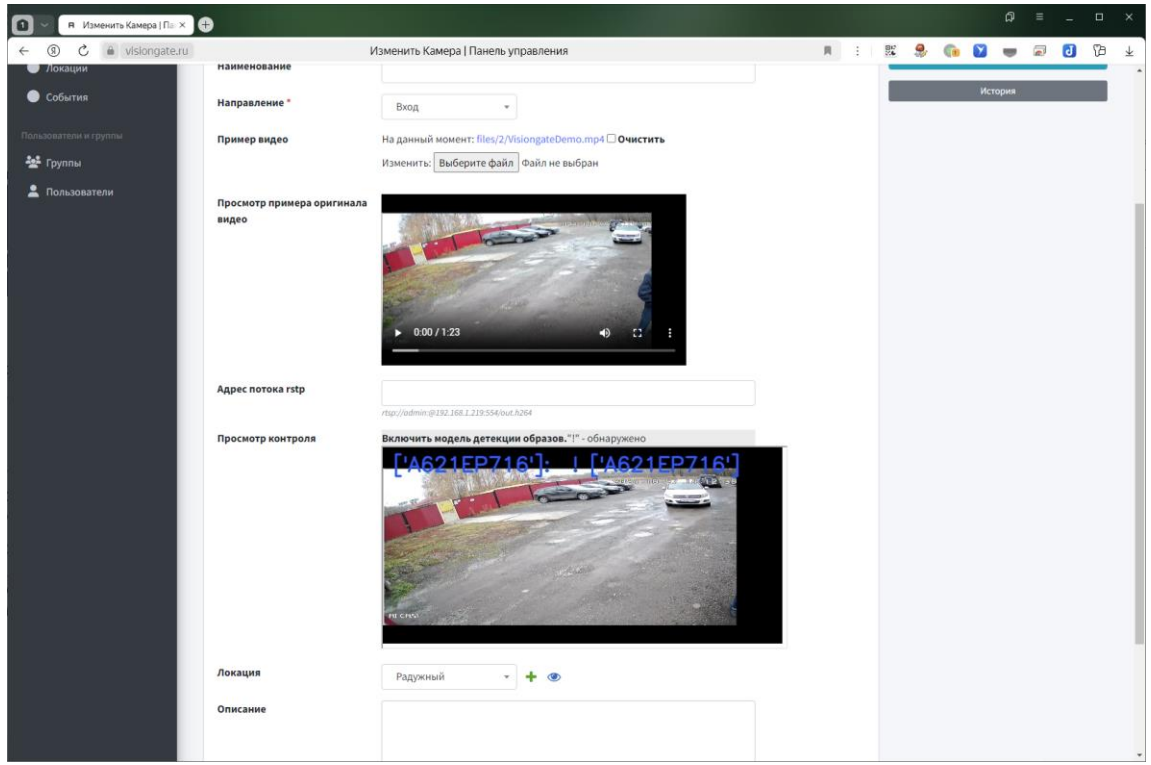
# НАСТРОЙКА ЛОКАЦИЙ И КАМЕР

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



# 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 просмотреть в блоке **Просмотр примера оригинала видео**
4. Демонстрацию работы ONNX-модели распознавания bounding-box образов номерного знака, и последующего распознанного текста - можно посмотреть в блоке **Просмотр контроля**



# УСТАНОВКА НА СЕРВЕРЕ

- Репозитарий github
  https://github.com/ikamil/visiongate
- Для запуска требуется
  сам сервер, доступ к root-консоли или права на docker
- Пример развёрнутого приложения
  https://visiongate.ru

**Скачивание и запуск docker compose сборки**

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

**Создание БД**

```
root@lkwuthwrux:~# 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@lkwuthwrux:~# 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@lkwuthwrux:~# 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