Министерство науки и высшего образования Российской Федерации
Федеральное государственное автономное образовательное учреждение высшего образования
«Московский государственный технический университет имени Н.Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ **ИНФОРМАТИКА И СИСТЕМЫ УПРАВЛЕНИЯ**

КАФЕДРА **КОМПЬЮТЕРНЫЕ СИСТЕМЫ И СЕТИ (ИУ6)**

НАПРАВЛЕНИЕ ПОДГОТОВКИ **09.04.01 Информатика и вычислительная техника**

МАГИСТЕРСКАЯ ПРОГРАММА **09.04.01/12 Интеллектуальный анализ больших данных в системах поддержки принятия решений.**

РАСЧЕТНО-ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

К КУРСОВОМУ ПРОЕКТУ

НА ТЕМУ:

Разработка сервиса для распознавания атрибутов участников дорожного движения на видео

Студент ____ИУ6-33М____ _____ _Д.А. Шестаков_____
(Группа) (Подпись, дата) (И.О.Фамилия)

Руководитель курсового проекта _____ _М.В. Мурашов_____
(Подпись, дата) (И.О.Фамилия)

2025 г.

Министерство науки и высшего образования Российской Федерации
Федеральное государственное автономное образовательное учреждение
высшего образования
«Московский государственный технический университет имени Н.Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

УТВЕРЖДАЮ
Заведующий кафедрой ИУ-6
(Индекс)
_____ А.В. Пролетарский
(И.О.Фамилия)
« _____ » _____ 2025 г.

З А Д А Н И Е
на выполнение курсового проекта

по дисциплине ___ Технологии организационно-аналитической деятельности_____

Студент группы ___ИУ6-33М_____

_____Шестаков Данил Алексеевич_____
(Фамилия, имя, отчество)

Тема курсового проекта « Разработка сервиса для распознавания атрибутов участников дорожного движения на видео»_____
_____
_____

Направленность КП (учебный, исследовательский, практический, производственный, др.)
_____учебная_____
Источник тематики (кафедра, предприятие, НИР) _кафедра_____

График выполнения проекта: 25% к 4 нед., 50% к 9 нед., 75% к 14 нед., 100% к 17 нед.

Задание проанализировать методы и подходы к распознаванию объектов на видео,_____
разработать схему базы данных, разработать схему алгоритма работы сервиса,_____
продемонстрировать работу сервиса с помощью пользовательского интерфейса_____
_____
_____
_____

Оформление курсового проекта:

Расчетно-пояснительная записка на _25-30_ листах формата А4.
Перечень графического (иллюстративного) материала (чертежи, плакаты, слайды и т.п.)
____нет_____
_____
_____
_____

Дата выдачи задания « 01 » сентября 2025 г.

Руководитель курсового проекта _____ ____М.В. Мурашов__
(Подпись, дата) (И.О.Фамилия)

Студент _____ ____Д.А. Шестаков_
(Подпись, дата) (И.О.Фамилия)

Примечание: Задание оформляется в двух экземплярах: один выдается студенту, второй хранится на кафедре.

РЕФЕРАТ

РПЗ 31 страниц, 14 рисунков, 11 источников, 1 приложение

ОБНАРУЖЕНИЕ, ОТСЛЕЖИВАНИЕ, РАСПОЗНАВАНИЕ, YOLO, BOTSORT, POSTGRESQL, MINIO, СЕГМЕНТАЦИЯ, PGVECTOR, CLIP, RESNET.

Целью данного курсового проекта является разработка сервиса для распознавания атрибутов участников дорожного движения на видео.

В ходе выполнения курсового проекта были выполнены следующие задачи:

- проанализированы методы и подходы к распознаванию объектов на видео;
- разработана схема базы данных;
- разработана схема алгоритма работы сервиса;
- продемонстрирована работа сервиса с помощью пользовательского интерфейса.

СОДЕРЖАНИЕ

ОПРЕДЕЛЕНИЯ, ОБОЗНАЧЕНИЯ И СОКРАЩЕНИЯ

BoT-SORT – алгоритм отслеживания объектов (трекинг), использующий информацию о движении камеры и признаки реидентификации для улучшения точности отслеживания.

CLIP (Contrastive Language-Image Pre-Training) – нейронная сеть, обученная на парах изображение-текст, способная находить семантические связи между визуальными и текстовыми данными.

CNN (Convolutional Neural Network) – сверточная нейронная сеть, архитектура глубокого обучения, специально предназначенная для обработки изображений.

COCO (Common Objects in Context) – популярный набор данных для обучения моделей обнаружения объектов, сегментации и создания описаний изображений.

FPS (Frames Per Second) – кадровая частота, количество кадров, сменяющих друг друга за одну секунду на видео.

JSON (JavaScript Object Notation) – текстовый формат обмена данными, основанный на JavaScript.

MinIO – высокопроизводительное объектное хранилище, совместимое с S3 API.

OCR (Optical Character Recognition) – технология оптического распознавания символов, позволяющая преобразовывать изображения текста в машиночитаемый текстовый формат.

PAR (Pedestrian Attribute Recognition) – задача распознавания атрибутов пешеходов (пол, возраст, одежда и т.д.) по изображению.

PostgreSQL – свободная объектно-реляционная система управления базами данных.

PromptPAR – метод и модель для распознавания атрибутов пешеходов, основанная на настройке подсказок (промптов) для предварительно обученных моделей (в данном случае CLIP).

ViT (Vision Transformer) – архитектура нейронной сети, основанная на механизме внимания (трансформерах), применяемая для задач компьютерного зрения.

YOLO (You Only Look Once) – семейство архитектур сверточных нейронных сетей для обнаружения объектов в реальном времени.

ВВЕДЕНИЕ

Целью данного курсового проекта является разработка сервиса для распознавания атрибутов участников дорожного движения на видео. Данный сервис является частью системы для анализа дорожного движения, который будет позволять выявлять различные аномалии или аварийные ситуации, что является актуальным направлением разработки, поскольку позволит обеспечивать большую безопасность в современных городах.

1 Анализ методов и подходов для решения задачи

Извлечение атрибутов объектов, присутствующих на видео – тяжелая задача, поскольку мы должны обработать не просто один кадр, а некоторую последовательность кадров, объекты на которой могут в какие-то моменты времени перекрываться другими объекта, некорректно распознаваться из-за плохого освещения. Тем не менее, наличие нескольких кадров, на которых находится объект, позволяет более точно распознать объект, поскольку по итогу обработки нескольких кадров у нас может накопиться наиболее полная информация об атрибутах объекта.

Наиболее популярным подходом к распознаванию объектов на видео является подход, в котором обработка подразделяется на следующие этапы:

- обнаружение объектов;
- отслеживание объектов;
- распознавание объектов.

Такое разделение позволяет на каждом этапе использовать модели и методы, наиболее оптимизированные и подходящие под решение этой задачи. Более того, такой подход проще реализуется, поскольку датасеты для обучения моделей под отдельные задачи более распространены, чем датасеты, в которых учтена вся необходимая для трех этапов информация об участниках видео. Также стоит учитывать, что такая архитектура позволяет гибко заменять модели или встраивать между этапами дополнительные модели для других задач (например, можно между этапами отслеживания и распознавания встроить модель для получения эмбеддинга объекта для задачи реиндетификации).

Далее рассмотрим, что из себя представляют упомянутые выше этапы и какие модели на них мы будем использовать.

1.1 Обнаружение объектов

Обнаружение объектов (англ. Object Detection) – задача компьютерного зрения, в рамках которой каждому объекту на изображении мы должны присвоить ограничивающие рамки (координаты) и класс объекта. В

современных реалиях задача решается с помощью моделей, называемых детекторами объектов, в основе которых лежат сверточные нейронные сети или визуальные трансформеры. Одними из наиболее популярных детекторов является семейство моделей YOLO, поскольку они являются одноэтапными (предсказывают объект за один проход по изображению, в отличие от двухэтапных моделей, которые сначала выбирают регионы на изображении, на которых уже будут искать объект), что обеспечивает высокую скорость работы, но тем не менее, данные модели имеют высокую точность предсказания объектов [1]. В нашем сервисе будем использовать последнюю модель YOLO12, предобученную на стандартном датасете для обнаружения объектов COCO, что позволит нам обнаружать следующие классы объектов, которые можно отнести к участникам дорожного движения: person, bicycle, car, motorcycle, bus, truck.

1.2 Отслеживание объектов

Отслеживание объектов (англ. Object Tracking) – задача компьютерного зрения, в рамках которой мы отслеживаем положение одного или нескольких объектов на видео. В отличие от детекции, которая обнаруживает объекты на отдельных кадрах, трекинг показывает, как они перемещаются во времени. Для этого система фиксирует последовательности координат (т.е. трекер получает на вход ограничивающие рамки от детектора объектов) и связывает их между собой, чтобы выстроить непрерывную траекторию движения. Использование модели для трекинга необходимо для того, чтобы мы могли засекать, какие промежутки времени на видео присутствует объект. К тому же, трекер каждому объекту присваивает уникальный в рамках видео идентификатор, по которому мы можем для каждого объекта накапливать несколько распознаваний, что позволит наиболее полно и точно предсказывать атрибуты объектов. В нашем сервисе мы будем использовать трекер BoT-SORT, поскольку он является одним из лучших на данный момент [2]. Это обеспечивается за счет модуля реидентификации, который позволяет эффективней отслеживать объекты при сложных сценариях с перекрытиями.

К тому же, данный трекер встроен в библиотеку ultralytics, из которой мы используем детектор объектов YOLO12 [1]. В реализации библиотеки BoT-SORT в модуле реидентификации используются извлеченные признаки из детектора, что ускоряет работу, поскольку нам нет необходимости использовать отдельную модель для извлечения признаков.

1.3 Распознавание объектов

Распознавание объектов – достаточно общий термин, у которого нет однозначной интерпретации, часто его употребляют как синоним обнаружения объектов. Но мы под распознаванием будем понимать извлечение признаков об объекте на изображении. Также не стоит это путать с классификацией, поскольку мы присваиваем объекту на изображении не один класс, а извлекаем несколько различных атрибутов. Прежде чем обсуждать, какие атрибуты пешеходов и транспортных средств и с помощью каких моделей, мы будем распознавать, считаю важным сказать, что на вход моделей мы будем подавать вырезанные изображения объектов из исходного кадра, что реализуется засчет обрезания кадра по рамкам, полученным от детектора объектов. Теперь перейдем к нашим участникам дорожного движения, каждого из которых рассмотрим отдельно.

1.3.1 Распознавание пешеходов

Прежде, чем рассматривать модели, выделим характерные атрибуты пешеходов: пол, возраст, вид одежды верхней части тела, вид одежды нижней части тела, обувь, тип волос (длинные, короткие, отсутствуют) или головной убор, наличие каких-то дополнительных аксессуаров (рюкзак, сумка, очки и так далее). Также можно у выделенных частей тела (голова, тело, ноги, обувь) предсказывать цвет. Все данные атрибуты достаточно специфичные, их тяжело распознавать с помощью классических классификаторов на основе сверточных нейронных сетей, поэтому в современных моделях все чаще используются трансформеры, механизмы внимания на отдельные части тела пешехода, и уже даже начинают пробовать распознавать с помощью больших

языковых моделей. В нашем сервисе мы решили использовать модель PromtPAR [3].

PromptPAR — это модель для распознавания атрибутов пешеходов (Pedestrian Attribute Recognition, PAR), которая использует предобученную модель CLIP (Contrastive Language-Image Pre-Training) в качестве основы [4]. Основная идея — не использовать стандартные CNN (как ResNet), а сформулировать задачу как проблему слияния визуальной и текстовой информации (vision-language fusion). На рисунке 1 представлена архитектура модели.

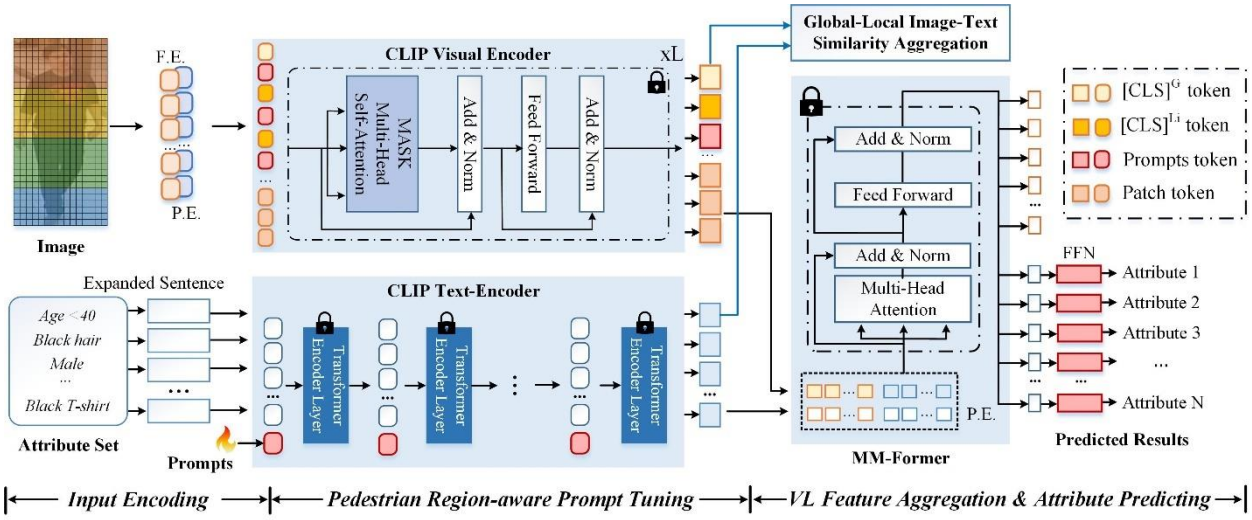


Рисунок 1 – Архитектура PromtPAR

Архитектуру можно условно разделить на три основных этапа:

1. Input Encoding (Входное кодирование):

– изображение (Image): исходное изображение пешехода разбивается на «патчи» (patches) — небольшие квадратные участки. Эти патчи, вместе с информацией об их положении (P.E. - Positional Encoding), подаются в визуальный кодировщик.

– атрибуты (Attribute Set): набор текстовых атрибутов (например, «Black hair») расширяется до полных предложений («Expanded Sentence»). К этим предложениям добавляются специальные обучаемые векторы — «промпты» (Prompts). Они также смешиваются с P.E. и подаются в текстовый кодировщик.

2. Pedestrian Region-aware Prompt Tuning (Настройка промптов с учетом регионов):

– CLIP Visual Encoder**:** это визуальная часть CLIP, основанная на Vision Transformer (ViT). Он принимает патчи изображения. Важно, что этот кодировщик заморожен, то есть его веса не обновляются во время обучения. Блок MASK Multi-Head Self-Attention используется для того, чтобы модель могла фокусироваться на определенных регионах изображения.

– CLIP Text-Encoder**:** это текстовая часть CLIP, основанная на Transformer. Он принимает предложения с промптами. Этот кодировщик также заморожен.

– «Prompt Tuning»: вместо того чтобы обучать всю огромную модель CLIP (миллиарды параметров), мы обучаем только маленькие векторы-промпты, которые «направляют» замороженную модель на правильное выполнение нужной задачи (распознавание атрибутов).

3. VL Feature Aggregation & Attribute Predicting (Агрегация V-L признаков и предсказание):

– Global-Local Image-Text Similarity Aggregation: выходы из обоих кодировщиков (визуальные токены и текстовые токены) поступают в этот блок. Он сопоставляет глобальные (все изображение/весь текст) и локальные (патчи/слова) признаки.

– MM-Former (Multi-Modal Transformer): это еще один, но уже мультимодальный трансформер, который принимает на вход и визуальные, и текстовые признаки и осуществляет их слияние (fusion). Этот блок также заморожен во время обучения.

– FFN (Feed-Forward Network) & Predicted Results: Слитые признаки из MM-Former подаются в простую полносвязную нейронную сеть (FFN), которая и является классификатором. Она выдает финальные предсказания для каждого атрибута (Attribute 1, 2, ... N).

Если подытожить, то ключевая идея, следующая**:** обучаются только промпты (малая часть параметров) и финальный классификатор (FFN). Все

остальное (CLIP Encoders, MM-Former) — заморожено. Это является «parameter-efficient» обучение (всего 0.75% обучаемых параметров).

Для обучения модели использовался датасет PETA, который является классическим в задаче распознавания атрибутов пешеходов (PAR) [5]. Код модели PromtPAR был отредактирован, чтобы обучать модель только на некоторых отобранных из датасета PETA атрибутах. На рисунке 2 приведено изображение и полученное для него предсказание атрибутов в виде JSON.

Рисунок 2 – Предсказание атрибутов пешехода по фото с помощью PromtPAR

1.3.2 Распознавание транспортных средств

Также, как и с пешеходами, выделим атрибуты, которые будем распознавать: номер, цвет (и в виде кода, и в виде метки), марку, модель, класс (автобус, легковой и так далее). В данном случае мы не сможем обойтись одной моделью, как это было с пешеходами, здесь практически под каждый атрибут требуется свой подход. Также стоит помнить, что класс транспортного средства у нас будет определяться на этапе детекции, поскольку YOLO12 обучена обнаруживать такие классы, как car, bus, motorcycle и другие, в следствие чего далее на этом атрибуте останавливаться не будем.

Начнем с распознавания номера, который из себя представляет 2 этапа: обнаружение номера и его оптическое распознавание. Для обнаружения также

используются вышеупомянутые детекторы, но уже дообученные для этой конкретной задачи (обнаружение номеров). Так в детектор номеров мы передаем вырезанное изображение транспортного средства (как и в другие модели для распознавание остальных атрибутов). Далее к вырезанному номеру можно применять библиотеку или собственную модель для оптического распознавания символов. В нашем случае, наилучшее качество показала библиотека PaddleOCR на изображениях номеров без предобработки [6].

Перейдем к распознаванию цвета. Большая часть датасетов и открытых моделей заточены под определение метки цвета (красный, черный и т.д.), что можно понять, поскольку код цвета для одного и того же объекта может сильно отличаться от освещения и устройства, на которое объект был снят, но для нашей системы необходимо получать код цвета, чтобы была возможность проводить более глубокий анализ, из-за чего был предложен следующий подход: сегментировать изображение транспортного средства на различные части, и к частям, которые относятся в кузову, применить кластеризацию k-means с двумя кластерами (два кластера обуславливаются тем, что транспортного средства может состоять из нескольких цветов, или при сегментации можем захватить ненужные пиксели) для получения среднего значения цвета в наибольшем кластере. Далее уже на основе полученного кода цвета можно сопоставить метку цвета по таблице основных цветов. Для сегментации использовалась предобученная YOLO11-seg (у семейства YOLO есть и такие модели), дообученная на датасете «Car Parts Segmentation». Для кластеризации использовался k-means из библиотеки Scikit-learn [7].

Теперь нам остается рассмотреть распознавание марки и модели. Данная задача является тяжелой, поскольку марок и моделей автомобилей огромное количество, которое постоянно пополняется. К тому же, некоторые модели одной и той же марки может быть крайне тяжело различить. Еще стоит помнить, что в каждом регионе своя специфика автомобилей, поэтому датасет, собранный в одной стране, может быть нерелевантным для другой. В

современных подходах все чаще начинают для этой задачи использовать большие языковые модели, но мы ограничились простым решением: дообучение resnet50 [8] на датасетах «Stanford Cars» [9] и «VMMRdb». Данная модель может предсказывать около 900 классов, каждый класс себя представляет сразу марку и модель (например, «Chevrolet Camaro 2010»), чего на начальном этапе разработки системы будет вполне достаточно.

2 Проектирование и реализация сервиса

В данной главе приступим к проектированию и реализации нашего сервиса.

Сервис должен выполнять обработку файлов, находящихся в базе данных, по полученному запросу. Запросы (задачи) сервис получает через очередь Celery. Организация хранения файлов в системе устроена следующим образом: в реляционной базе данных PostgreSQL [10] хранится справочная информация о файле (тип файла, название файла, статус обработки и т.д.), а сами файлы хранятся в объектном хранилище MinIO [11]. Соответственно, логика работы сервиса следующая:

- получение id файла в PostgreSQL;
- скачивание файла из MinIO для обработки (путь к файлу в MinIO лежит в PostgreSQL);
- выполнение обработки файла и параллельное обновление статуса обработки файла в PostgreSQL (необходимо для отображения пользователю на фронтенде);
- загрузка извлеченных данных об объектах в PostgreSQL, а также загрузка миниатюр этих объектов в MinIO;
- формирование задачи для сервиса кластеризации объектов (передаются id обнаруженным объектов).

Также в наш цикл обработки, рассмотренный в первой главе («Обнаружение – Отслеживание – Распознавание») будут встроены модели для получения эмбеддингов для реидентификации объектов. Это необходимо для сервиса кластеризации объектов, который на основе этих эмбеддингов ищет объекты, присутствующие в нескольких файлах. Для хранения эмбеддингов используется расширение PGvector для PostgreSQL. Далее подробней рассмотрим проектирование схемы базы данных.

2.1 Проектирование схемы базы данных

Как говорилось выше, у нас в системе используется реляционная база данных PostgreSQL и объектное хранилище MinIO. Начнем с рассмотрения таблиц в PostgreSQL.

В PostgreSQL у нас будут следующие таблицы:

– «media_files» – хранение данных об исходных медиафайлах;

– «detected_objects» – хранение данных об обнаруженных объектах;

– «object_embeddings» – хранение эмбеддингов обнаруженных объектов;

– «brand_models» – хранение пар известных марок и моделей (данная таблица необходима для задач фронтенда).

Рассмотрим поля таблицы «media_files»:

– «id» – числовое поле, является первичным ключом, автоматически присваивается при загрузке в базу данных медиафайла;

– «filename» – текстовая колонка, в которой хранится имя файла;

– «storage_path» – путь к исходному объекту в MinIO;

– «media_type» – колонка с перечисляемым типом данных с двумя возможными значениями: «video» и «photo», т.е. здесь указывается тип файла;

– «fps» – вещественная колонка, в котором хранится фреймрейт видео (в случае фото записываем единицу);

– «status» – колонка с перечисляемым типом данных со следующими возможными значениями: «unprocessed», «pending», «processing», «completed» и «failed», т.е. здесь указывается статус обработки файла;

– «status_message» – текстовое поле, в которое формируется сообщение пользователю о прогрессе обработки файла (например, «Обработано 63 кадра из 103»);

– «uploaded_at» – поле, в котором хранится время загрузки файла в базу данных;

– «processed_at» – поле, в котором хранится время завершения обработки файла;

– «thumbnail_path» – текстовое поле, в котором хранится путь к миниатюре файла в MinIO (данная миниатюра формируется в начале обработки файла и используется для отображения файла на фронтенде);

– «vehicle_amount» – числовое поле, в которое записывается количество транспортных средств, обнаруженных при обработке медиафайла;

– «person_amount» – числовое поле, в которое записывается количество пешеходов, обнаруженных при обработке медиафайла;

– «video_length» – числовое поле, в которое записывается количество кадров в видео (для фото записывается единица).

Рассмотрим поля таблицы «detected_objects»:

– «id» – числовое поле, является первичным ключом, автоматически присваивается при загрузке обнаруженного объекта в таблицу;

– «media_file_id» – числовое поле, является внешним ключом, ссылающимся на медиафайл, в котором объект был обнаружен;

– «object_type» – колонка с перечисляемым типом данных с двумя возможными значениями: «person» и «vehicle», т.е. здесь указывается тип объекта;

– «detection_details» – поле формата JSONB, в котором хранится следующая информация: «track_id» объекта (присваивается трекером), список пар «кадр появления объекта – кадр исчезновения объекта» (используется список, поскольку объект может исчезнуть при перекрытиях, но потом появится снова), число кадров присутствия объекта на видео, словарь, в котором в качестве ключа выступает номер кадра присутствия объекта на видео, а в качестве значения – кортеж с координатами объекта на кадре.

– «thumbnail_path» – текстовое поле, в котором хранится путь к миниатюре объекта в MinIO (данная миниатюра формируется в ходе обработки файла и используется для отображения объекта на фронтенде);

– «attributes» – поле формата JSONB, в котором хранится информация об распознанных атрибутах объекта (для пешеходов и транспортных средств JSONB будут разного формата);

– «name» – текстовое поле, в котором формируется краткое названия объекта для отображения на фронтенде (например, «black car with plate «AXO 6973» или «male with short hair, t-shirt, shorts»).

В таблице выше упоминались поля с JSONB, давайте подробней рассмотрим, какие данные будут в них храниться. Начнем с JSONB для транспортных средств:

– «vehicle_type» – тип транспортного средства из следующего списка: «car», «truck», «bus», «motorcycle», «bicycle»«;

– «color_label» – метка цвета транспортного средства из следующего списка: «white», «black», «red», «blue», «green», «yellow», «silver», «grey», «gray», «brown», «unknown», «orange», «beige», «golden», «purple», «pink», «non-existent»;

– «color_code» – список с кодом цвета в RGB;

– «brand» – марка транспортного средства;

– «model» – модель транспортного средства;

– «plate_text» – текст номера транспортного средства;

Теперь рассмотрим JSONB для пешеходов:

– «gender» – пол пешехода из следующего списка: «male» и «female»;

– «age» – возраст пешехода из следующего списка: «less_18», «18-60», «over_60»;

– «hair_type» – тип волос из следующего списка: «long», «short», «bald»;

– «upper_body_clothes_type» – тип одежды верхней части тела пешехода из следующего списка: «tshirt», «jacket», «suit», «sweater», «shirt», «vest»;

– «lower_body_clothes_type» – тип одежды нижней части тела пешехода из следующего списка: «shorts», «jeans», «short_skirt», «trousers», «capri», «long_skirt», «suit»;

– «shoes_type» – тип обуви пешехода из следующего списка: «sport», «leather_shoes», «sandals», «shoes», «sneakers», «boots», «stockings»;

– «hair_color», «upper_body_clothes_color», «lower_body_clothes_color», «shoes_color» – цвет различных частей пешехода из следующего списка: «white», «black», «red», «blue», «green», «yellow», «brown», «orange», «purple», «pink», «grey»;

– «has_glasses» – булево поле наличия очков у пешехода;

– «has_backpack» – булево поле наличия рюкзака у пешехода;

– «has_bag» – булево поле наличия сумки у пешехода.

Возвращаемся к таблицам. Рассмотрим поля таблицы «object_embeddings»:

– «id» – числовое поле, является первичным ключом, автоматически присваивается при загрузке эмбеддинга объекта в таблицу;

– «object_id» – чиловое поле, является внешним ключом, который ссылается на объект, к которому эмбеддинг относится;

– «embedding» – поле для хранения эмбеддинга размерностью 512;

– «created_at» – поле, в котором хранится время загрузки эмбеддинга в таблицу.

Рассмотрим поля таблицы «brand_models»:

– «id» – числовое поле, является первичным ключом, автоматически присваивается при загрузке записи в таблицу;

– «brand» – текстовое поле, в котором хранится марка транспортного средства;

– «model» – текстовое поле, в котором хранится модель транспортного средства.

Теперь перейдем к рассмотрению MinIO. Так как оно является объектным хранилищем, то там нет никаких таблиц, а файлы хранятся в бакетах (англ. bucket), которые можно воспринимать как папки в файловой системе. У нас в MinIO будет 3 бакета:

– «mediafiles» – в нем по папкам «photo» и «video» хранятся исходные медиафайлы;

– «mediafile-thumbnails» – в нем по папкам «photo» и «video» хранятся миниатюры для файлов;

«object-thumbnails» – в нем по папкам «pedestrian» и «vehicle» хранятся миниатюры для обнаруженным объектов.

Теперь попробуем визуализировать схему базы данных. На рисунке 3 представлена полученная схемы базы данных.

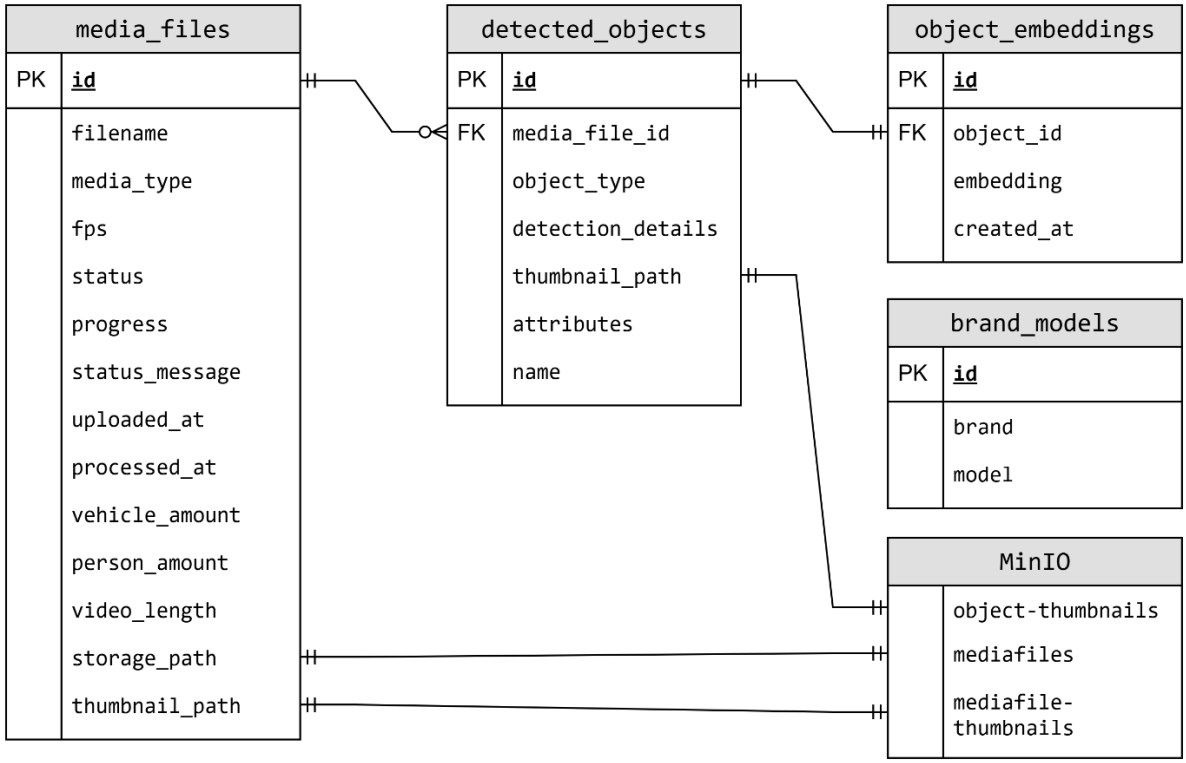


Рисунок 3 – Схема базы данных

2.2 Проектирование алгоритма работы сервиса

Сервис реализуется в виде функции, которая вызывается для обработки задачи. Основная последовательность действий сервиса была описана в начале второй главы, но есть еще несколько моментов которые стоит упомянуть. Сервис в первую очередь проектировался для анализа видео, но также в нем реализована возможность анализа фото. Как говорилось ранее, на видео мы можем по нескольким кадрам определять атрибуты объектов. В нашем случае, мы для каждого объекта копим распознанные атрибуты, а затем берем из них те, которые чаще всего встречались (а эмбеддинги – усредняем). Возможно, стоило придумать более интеллектуальный подход для отбора атрибутов, но на начальном этапе разработки системы этого вполне достаточно. Для формирования миниатюр объектов используется похожая логика с сохранением вырезок объекта на каждом кадре, но по итогу в MinIO мы записываем миниатюру из середины пребывания объекта в видео. Такая логика позволяет получать более качественные миниатюры, поскольку в моменты появления или исчезновения объекта с видео, мы видим только его часть, из-за чего такая миниатюра менее информативная. А вот с анализом фото все проще – мы только один раз распознаем все атрибуты объектов и сразу формируем миниатюры.

Теперь перейдем к схеме алгоритма, которая представлена на рисунке 4.

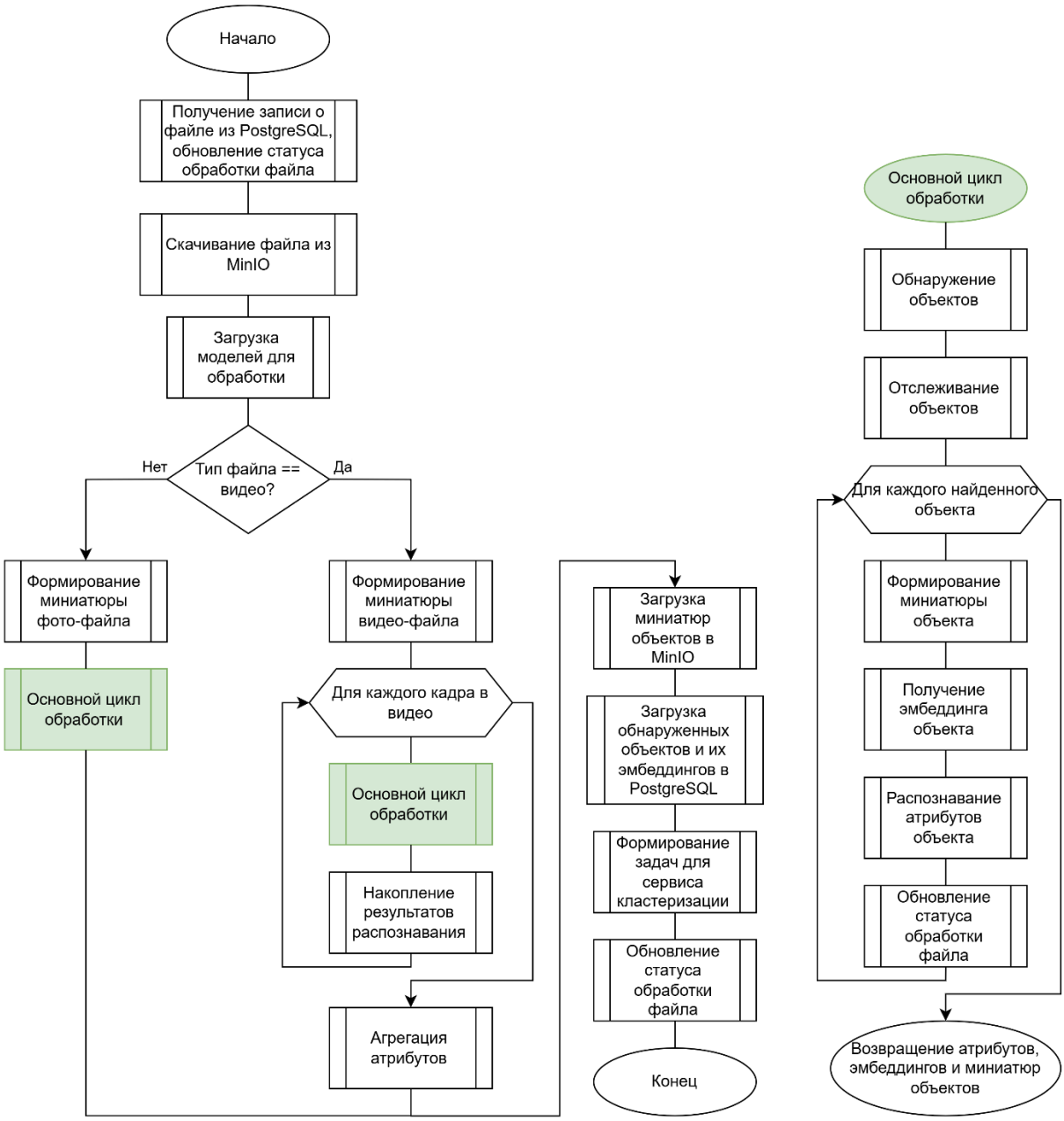


Рисунок 4 – Схема алгоритма работы сервиса

На схеме алгоритма отображены все тонкости работы алгоритма, упомянутые выше. Фрагмент исходного кода программы приведен в приложении А.

3 Демонстрация результатов

В качестве демонстрации работы сервиса приводим скриншоты работы с сервисом через фронтенд и скриншоты таблиц баз данных.

Начнем с демонстрации интерфейса, на котором будет видно, как используются извлеченные атрибуты объектов. На рисунке 5 представлен скриншот, на котором отображается список файлов. Миниатюры для этих файлов формируются в ходе обработки файла. Также у карточек видна информация о количестве обнаруженных объектов того или иного типа, тип файла и другая служебная информация.

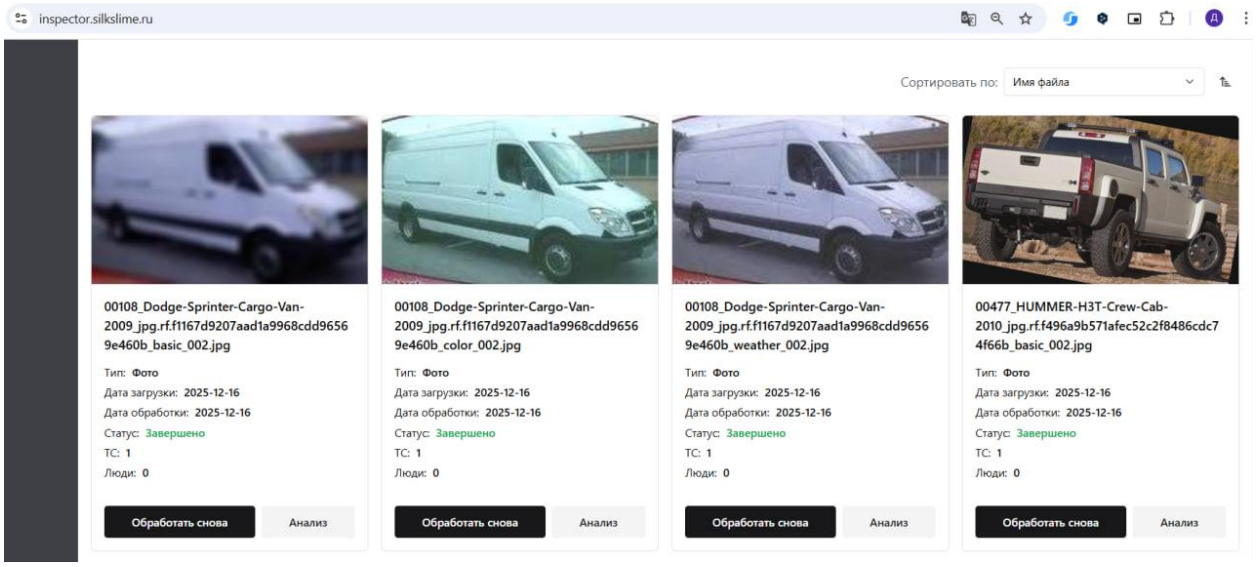


Рисунок 5 – Список файлов в интерфейсе пользователя

Для каждого обработанного файла можно просмотреть более подробный «анализ». Так при просмотре «анализа» видеофайла в верхней части страницы пользователя отображается исходное видео, но при нажатии паузы для объектов рисуются ограничивающие рамки, которые подписываются некоторыми распознанными атрибутами. На рисунке 6 представлен скриншот видеоплеера на паузе.

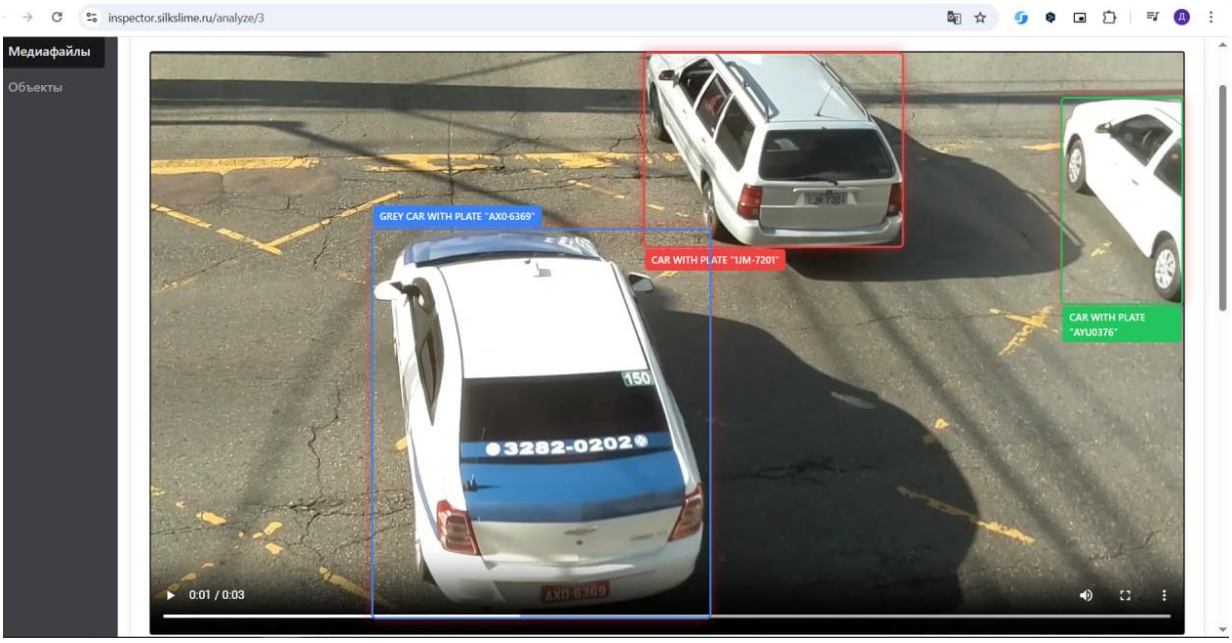


Рисунок 6 – Отрисовка ограничивающих рамок объектов на видео

Ниже видео на странице пользователя отображается временная шкала, на которой видно, в какие промежутки времени объект присутствовал. При нажатии на шкалу отображается миниатюра объекта и его атрибуты. Скриншот временной шкалы приведен на рисунке 7.

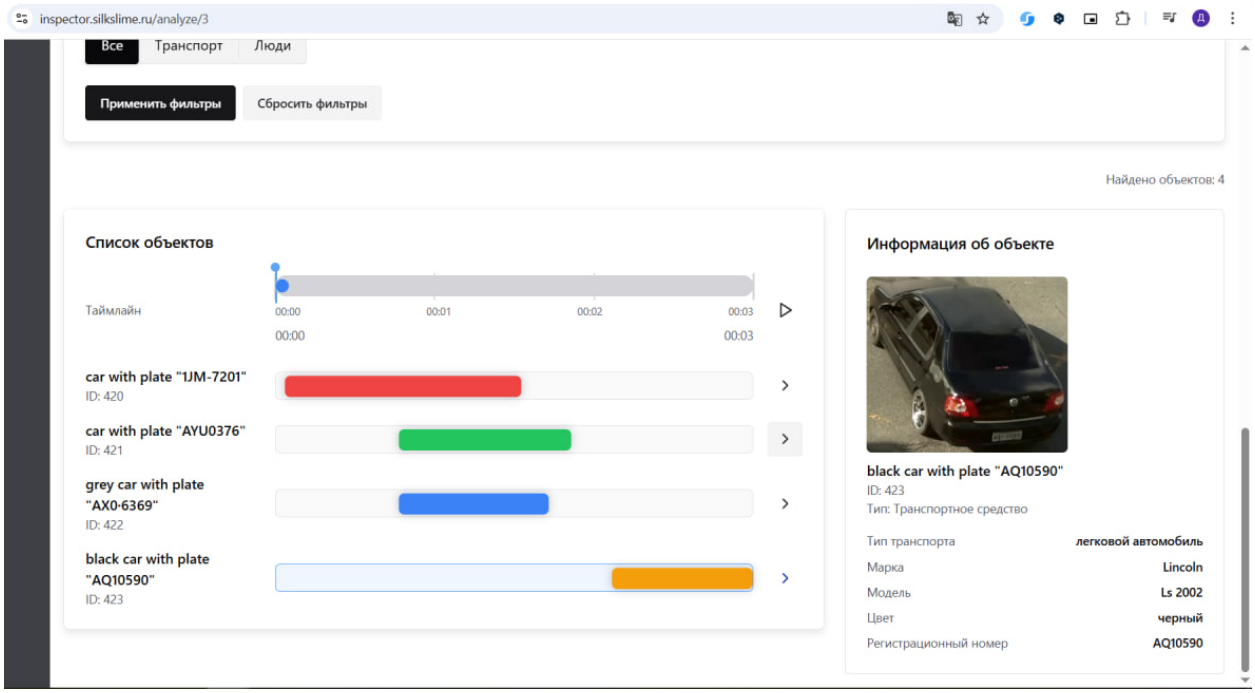


Рисунок 7 – Временная шкала обработанного видео

Для фото в «анализе» будут также отрисованы ограничивающие рамки для объектов, но вместо временной шкалы будет просто список объектов. На рисунке 8 представлено обработанное фото с дорисованными ограничивающими рамками. На рисунке 9 представлен список объектов, обнаруженных на фото.

Рисунок 8 – Отображение обработанного фото

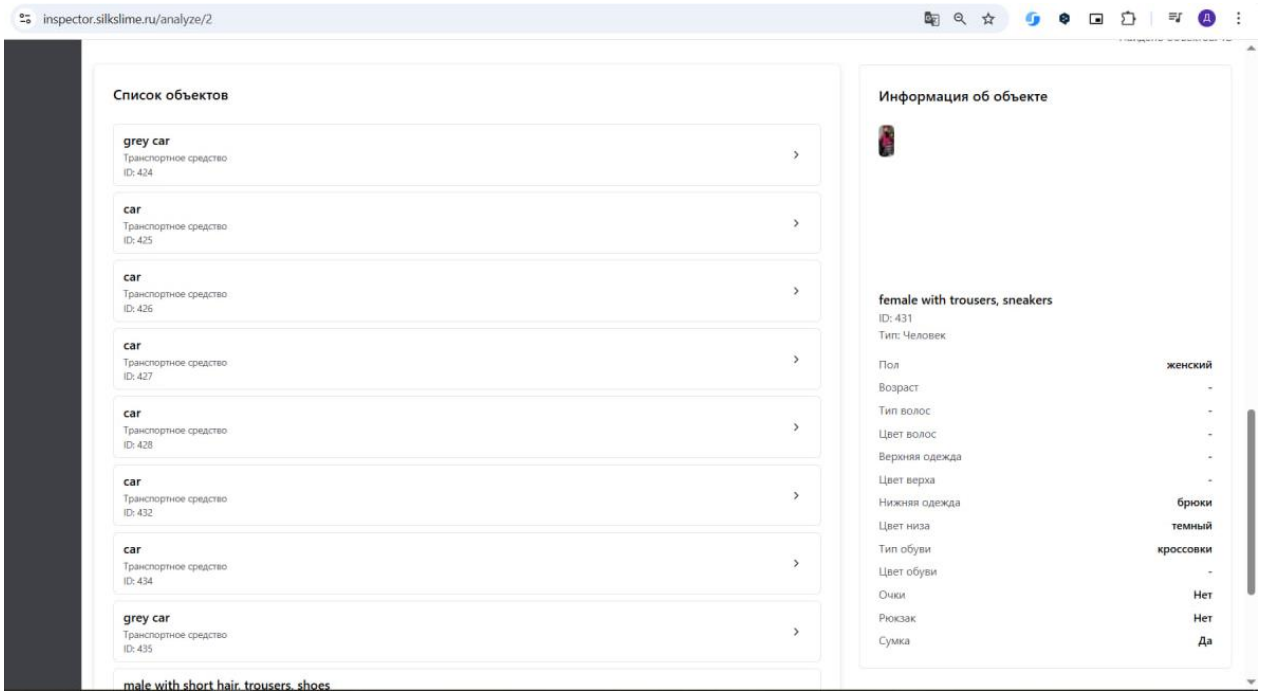
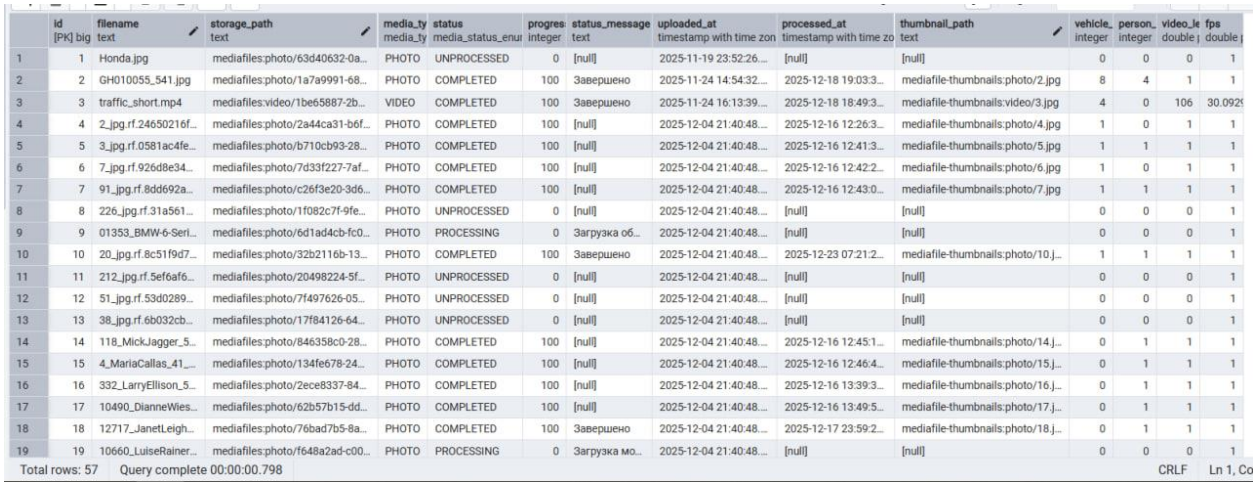


Рисунок 9 – Список объектов обработанного фото

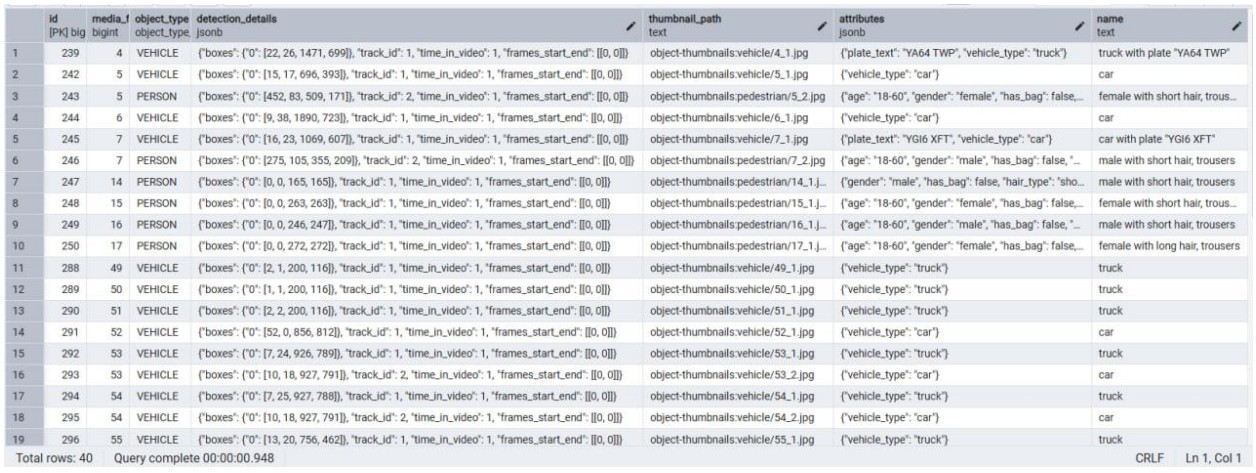
Теперь приведем скриншоты таблиц баз данных, которые заполняются согласно логике, описанной выше. На рисунке 10 представлен скриншот таблицы «media_files». На рисунке 11 представлен скриншот таблицы «detected_objects». На рисунке 12 представлен скриншот таблицы «object_embeddings». На рисунке 13 представлен скриншот таблицы «brand_models».



	id [PK] big	filename text	storage_path text	media_ty media_ty	status media_status_enu	progres integer	status_message text	uploaded_at timestamp with time zon	processed_at timestamp with time zo	thumbnail_path text	vehicle_ integer	person_ integer	video_le double	fps double
1	1	Honda.jpg	mediafiles:photo/63d40632-0a...	PHOTO	UNPROCESSED	0	[null]	2025-11-19 23:52:26....	[null]	[null]	0	0	0	1
2	2	GH010055_541.jpg	mediafiles:photo/1a7a9991-68...	PHOTO	COMPLETED	100	Завершено	2025-11-24 14:54:32....	2025-12-18 19:03:3...	mediafile-thumbnails:photo/2.jpg	8	4	1	1
3	3	traffic_short.mp4	mediafiles:video/1be65887-2b...	VIDEO	COMPLETED	100	Завершено	2025-11-24 16:13:39....	2025-12-18 18:49:3...	mediafile-thumbnails:video/3.jpg	4	0	106	30.092
4	4	2_jpg.rf.24650216f...	mediafiles:photo/2a44ca31-b6f...	PHOTO	COMPLETED	100	[null]	2025-12-04 21:40:48....	2025-12-16 12:26:3...	mediafile-thumbnails:photo/4.jpg	1	0	1	1
5	5	3_jpg.rf.0581ac4fe...	mediafiles:photo/b710cb93-28...	PHOTO	COMPLETED	100	[null]	2025-12-04 21:40:48....	2025-12-16 12:41:3...	mediafile-thumbnails:photo/5.jpg	1	1	1	1
6	6	7_jpg.rf.926d8e34...	mediafiles:photo/7d33f227-7af...	PHOTO	COMPLETED	100	[null]	2025-12-04 21:40:48....	2025-12-16 12:42:2...	mediafile-thumbnails:photo/6.jpg	1	0	1	1
7	7	91_jpg.rf.8dd692a...	mediafiles:photo/c26f3e20-3d6...	PHOTO	COMPLETED	100	[null]	2025-12-04 21:40:48....	2025-12-16 12:43:0...	mediafile-thumbnails:photo/7.jpg	1	1	1	1
8	8	226_jpg.rf.31a561...	mediafiles:photo/1f082c7f-9fe...	PHOTO	UNPROCESSED	0	[null]	2025-12-04 21:40:48....	[null]	[null]	0	0	0	1
9	9	01353_BMW-6-Seri...	mediafiles:photo/6d1ad4cb-fc0...	PHOTO	PROCESSING	0	Загрузка об...	2025-12-04 21:40:48....	[null]	[null]	0	0	0	1
10	10	20_jpg.rf.8c51f9d7...	mediafiles:photo/32b2116b-13...	PHOTO	COMPLETED	100	Завершено	2025-12-04 21:40:48....	2025-12-23 07:21:2...	mediafile-thumbnails:photo/10.j...	1	1	1	1
11	11	212_jpg.rf.5ef6af6...	mediafiles:photo/20498224-5f...	PHOTO	UNPROCESSED	0	[null]	2025-12-04 21:40:48....	[null]	[null]	0	0	0	1
12	12	51_jpg.rf.53d0289...	mediafiles:photo/7f497626-05...	PHOTO	UNPROCESSED	0	[null]	2025-12-04 21:40:48....	[null]	[null]	0	0	0	1
13	13	38_jpg.rf.6b032cb...	mediafiles:photo/17f84126-64...	PHOTO	UNPROCESSED	0	[null]	2025-12-04 21:40:48....	[null]	[null]	0	0	0	1
14	14	118_MickJagger_5...	mediafiles:photo/846358c0-28...	PHOTO	COMPLETED	100	[null]	2025-12-04 21:40:48....	2025-12-16 12:45:1...	mediafile-thumbnails:photo/14.j...	0	1	1	1
15	15	4_MariaCallas_41_...	mediafiles:photo/134fe678-24...	PHOTO	COMPLETED	100	[null]	2025-12-04 21:40:48....	2025-12-16 12:46:4...	mediafile-thumbnails:photo/15.j...	0	1	1	1
16	16	332_LarryEllison_5...	mediafiles:photo/2ece8337-84...	PHOTO	COMPLETED	100	[null]	2025-12-04 21:40:48....	2025-12-16 13:39:3...	mediafile-thumbnails:photo/16.j...	0	1	1	1
17	17	10490_DianneWies...	mediafiles:photo/62b57b15-dd...	PHOTO	COMPLETED	100	[null]	2025-12-04 21:40:48....	2025-12-16 13:49:5...	mediafile-thumbnails:photo/17.j...	0	1	1	1
18	18	12717_JanetLeigh...	mediafiles:photo/76bad7b5-8a...	PHOTO	COMPLETED	100	Завершено	2025-12-04 21:40:48....	2025-12-17 23:59:2...	mediafile-thumbnails:photo/18.j...	0	1	1	1
19	19	10660_LuiseRainer...	mediafiles:photo/f648a2ad-c00...	PHOTO	PROCESSING	0	Загрузка мо...	2025-12-04 21:40:48....	[null]	[null]	0	0	0	1

Total rows: 57 Query complete 00:00:00.798 CRLF Ln 1, Co

Рисунок 10 – Скриншот таблицы «media_files»



	id [PK] big	media_f bigint	object_type object_type	detection_details jsonb	thumbnail_path text	attributes jsonb	name text
1	239	4	VEHICLE	{"boxes": {"0": [22, 26, 1471, 699]}, "track_id": 1, "time_in_video": 1, "frames_start_end": [[0, 0]]}	object-thumbnails:vehicle/4_1.jpg	{"plate_text": "YA64 TWP", "vehicle_type": "truck"}	truck with plate "YA64 TWP"
2	242	5	VEHICLE	{"boxes": {"0": [15, 17, 696, 393]}, "track_id": 1, "time_in_video": 1, "frames_start_end": [[0, 0]]}	object-thumbnails:vehicle/5_1.jpg	{"vehicle_type": "car"}	car
3	243	5	PERSON	{"boxes": {"0": [452, 83, 509, 171]}, "track_id": 2, "time_in_video": 1, "frames_start_end": [[0, 0]]}	object-thumbnails:pedestrian/5_2.jpg	{"age": "18-60", "gender": "female", "has_bag": false,...	female with short hair, trous...
4	244	6	VEHICLE	{"boxes": {"0": [9, 38, 1890, 723]}, "track_id": 1, "time_in_video": 1, "frames_start_end": [[0, 0]]}	object-thumbnails:vehicle/6_1.jpg	{"vehicle_type": "car"}	car
5	245	7	VEHICLE	{"boxes": {"0": [16, 23, 1069, 607]}, "track_id": 1, "time_in_video": 1, "frames_start_end": [[0, 0]]}	object-thumbnails:vehicle/7_1.jpg	{"plate_text": "YGI6 XFT", "vehicle_type": "car"}	car with plate "YGI6 XFT"
6	246	7	PERSON	{"boxes": {"0": [275, 105, 355, 209]}, "track_id": 2, "time_in_video": 1, "frames_start_end": [[0, 0]]}	object-thumbnails:pedestrian/7_2.jpg	{"age": "18-60", "gender": "male", "has_bag": false, "...	male with short hair, trousers
7	247	14	PERSON	{"boxes": {"0": [0, 0, 165, 165]}, "track_id": 1, "time_in_video": 1, "frames_start_end": [[0, 0]]}	object-thumbnails:pedestrian/14_1.j...	{"gender": "male", "has_bag": false, "hair_type": "sho...	male with short hair, trousers
8	248	15	PERSON	{"boxes": {"0": [0, 0, 263, 263]}, "track_id": 1, "time_in_video": 1, "frames_start_end": [[0, 0]]}	object-thumbnails:pedestrian/15_1.j...	{"age": "18-60", "gender": "female", "has_bag": false,...	female with short hair, trous...
9	249	16	PERSON	{"boxes": {"0": [0, 0, 246, 247]}, "track_id": 1, "time_in_video": 1, "frames_start_end": [[0, 0]]}	object-thumbnails:pedestrian/16_1.j...	{"age": "18-60", "gender": "male", "has_bag": false, "...	male with short hair, trousers
10	250	17	PERSON	{"boxes": {"0": [0, 0, 272, 272]}, "track_id": 1, "time_in_video": 1, "frames_start_end": [[0, 0]]}	object-thumbnails:pedestrian/17_1.j...	{"age": "18-60", "gender": "female", "has_bag": false,...	female with long hair, trousers
11	288	49	VEHICLE	{"boxes": {"0": [2, 1, 200, 116]}, "track_id": 1, "time_in_video": 1, "frames_start_end": [[0, 0]]}	object-thumbnails:vehicle/49_1.jpg	{"vehicle_type": "truck"}	truck
12	289	50	VEHICLE	{"boxes": {"0": [1, 1, 200, 116]}, "track_id": 1, "time_in_video": 1, "frames_start_end": [[0, 0]]}	object-thumbnails:vehicle/50_1.jpg	{"vehicle_type": "truck"}	truck
13	290	51	VEHICLE	{"boxes": {"0": [2, 2, 200, 116]}, "track_id": 1, "time_in_video": 1, "frames_start_end": [[0, 0]]}	object-thumbnails:vehicle/51_1.jpg	{"vehicle_type": "truck"}	truck
14	291	52	VEHICLE	{"boxes": {"0": [52, 0, 856, 812]}, "track_id": 1, "time_in_video": 1, "frames_start_end": [[0, 0]]}	object-thumbnails:vehicle/52_1.jpg	{"vehicle_type": "car"}	car
15	292	53	VEHICLE	{"boxes": {"0": [7, 24, 926, 789]}, "track_id": 1, "time_in_video": 1, "frames_start_end": [[0, 0]]}	object-thumbnails:vehicle/53_1.jpg	{"vehicle_type": "truck"}	truck
16	293	53	VEHICLE	{"boxes": {"0": [10, 18, 927, 791]}, "track_id": 2, "time_in_video": 1, "frames_start_end": [[0, 0]]}	object-thumbnails:vehicle/53_2.jpg	{"vehicle_type": "car"}	car
17	294	54	VEHICLE	{"boxes": {"0": [7, 25, 927, 788]}, "track_id": 1, "time_in_video": 1, "frames_start_end": [[0, 0]]}	object-thumbnails:vehicle/54_1.jpg	{"vehicle_type": "truck"}	truck
18	295	54	VEHICLE	{"boxes": {"0": [10, 18, 927, 791]}, "track_id": 2, "time_in_video": 1, "frames_start_end": [[0, 0]]}	object-thumbnails:vehicle/54_2.jpg	{"vehicle_type": "car"}	car
19	296	55	VEHICLE	{"boxes": {"0": [13, 20, 756, 462]}, "track_id": 1, "time_in_video": 1, "frames_start_end": [[0, 0]]}	object-thumbnails:vehicle/55_1.jpg	{"vehicle_type": "truck"}	truck

Total rows: 40 Query complete 00:00:00.948 CRLF Ln 1, Col 1

Рисунок 11 – Скриншот таблицы «detected_objects»

	id [PK] bigint	object_id bigint	embedding vector	created_at timestamp with time zone
1	235	239	[-0.04995783,0.019800678,0.012002662,-0.015012731,-0.023226094,-0.017585...	2025-12-16 12:26:39.019874+...
2	238	242	[-0.053175975,0.024162795,-0.095939845,-0.018003022,0.047052823,0.068247...	2025-12-16 12:41:34.361656+...
3	239	243	[0,0.02878877,0.020779535,0.00026127294,0,0,0.011081463,0.063293345,0,0,0,...	2025-12-16 12:41:34.365168+...
4	240	244	[0.043536022,0.019004747,-0.038984,0.07708602,-0.025008637,-0.045825634,...	2025-12-16 12:42:23.861309+...
5	241	245	[0.014923437,0.053922646,0.008317961,0.013159405,-0.043743677,-0.036110...	2025-12-16 12:43:09.301794+...
6	242	246	[0,0.09119325,0.11093493,0.019563446,0,0.048686028,0,0.08401084,0.003268...	2025-12-16 12:43:09.306503+...
7	243	247	[0.052009284,0.061562985,0.0051452625,0,0,0.10630274,0,0,0.02110997,0.057...	2025-12-16 12:45:16.914593+...
8	244	248	[0,0.045723844,0,0,0,0.083624795,0,0.027462155,0.047567002,0.08731998,0,0,...	2025-12-16 12:46:43.485884+...
9	245	249	[0.036269918,0.08508503,0.034224823,0,0,0.16072264,0,0.09535895,0.039735...	2025-12-16 13:39:36.884061+...
10	246	250	[0.05265354,0.037318505,0.009722395,0,0,0.10745582,0,0.016247747,0.07494...	2025-12-16 13:49:54.008714+...
11	283	288	[0.033610005,-0.04164961,-0.07921009,0.011575373,0.038928255,0.00857837...	2025-12-17 00:22:07.537197+...
12	284	289	[0.018420814,-0.029780021,-0.054231968,-0.005353793,0.097668946,0.014259...	2025-12-17 00:22:52.713962+...
13	285	290	[0.03694152,-0.05798891,-0.08596611,0.013171609,0.035745192,0.009017664,...	2025-12-17 00:23:38.242474+...
14	286	291	[0.023454791,0.060279947,0.10367703,0.03437544,-0.014595866,-0.02887645...	2025-12-17 00:24:16.929561+...
15	287	292	[0.023312852,0.030214971,0.09755307,0.012350961,-0.028598856,-0.0225169...	2025-12-17 00:24:57.454942+...
16	288	293	[0.007713399,0.038366456,0.11146646,0.020831514,-0.031067535,-0.0232766...	2025-12-17 00:24:57.459591+...
17	289	294	[0.021905689,0.034256168,0.10445873,0.013707735,-0.033443864,-0.0323686...	2025-12-17 00:25:37.49477+03
18	290	295	[0.011111551,0.042576734,0.11308443,0.023077438,-0.0280711,-0.02802839,0...	2025-12-17 00:25:37.499672+...
19	291	296	[-0.01740349,0.011256501,-0.058464818,0.04002981,-0.043390393,-0.0431922...	2025-12-17 00:26:17.441808+...

Рисунок 12 – Скриншот таблицы «object_embeddings»

	id [PK] bigint	object_id bigint	embedding vector	created_at timestamp with time zone
1	235	239	[-0.04995783,0.019800678,0.012002662,-0.015012731,-0.023226094,-0.017585...	2025-12-16 12:26:39.019874+...
2	238	242	[-0.053175975,0.024162795,-0.095939845,-0.018003022,0.047052823,0.068247...	2025-12-16 12:41:34.361656+...
3	239	243	[0,0.02878877,0.020779535,0.00026127294,0,0,0.011081463,0.063293345,0,0,0,...	2025-12-16 12:41:34.365168+...
4	240	244	[0.043536022,0.019004747,-0.038984,0.07708602,-0.025008637,-0.045825634,...	2025-12-16 12:42:23.861309+...
5	241	245	[0.014923437,0.053922646,0.008317961,0.013159405,-0.043743677,-0.036110...	2025-12-16 12:43:09.301794+...
6	242	246	[0,0.09119325,0.11093493,0.019563446,0,0.048686028,0,0.08401084,0.003268...	2025-12-16 12:43:09.306503+...
7	243	247	[0.052009284,0.061562985,0.0051452625,0,0,0.10630274,0,0,0.02110997,0.057...	2025-12-16 12:45:16.914593+...
8	244	248	[0,0.045723844,0,0,0,0.083624795,0,0.027462155,0.047567002,0.08731998,0,0,...	2025-12-16 12:46:43.485884+...
9	245	249	[0.036269918,0.08508503,0.034224823,0,0,0.16072264,0,0.09535895,0.039735...	2025-12-16 13:39:36.884061+...
10	246	250	[0.05265354,0.037318505,0.009722395,0,0,0.10745582,0,0.016247747,0.07494...	2025-12-16 13:49:54.008714+...
11	283	288	[0.033610005,-0.04164961,-0.07921009,0.011575373,0.038928255,0.00857837...	2025-12-17 00:22:07.537197+...
12	284	289	[0.018420814,-0.029780021,-0.054231968,-0.005353793,0.097668946,0.014259...	2025-12-17 00:22:52.713962+...
13	285	290	[0.03694152,-0.05798891,-0.08596611,0.013171609,0.035745192,0.009017664,...	2025-12-17 00:23:38.242474+...
14	286	291	[0.023454791,0.060279947,0.10367703,0.03437544,-0.014595866,-0.02887645...	2025-12-17 00:24:16.929561+...
15	287	292	[0.023312852,0.030214971,0.09755307,0.012350961,-0.028598856,-0.0225169...	2025-12-17 00:24:57.454942+...
16	288	293	[0.007713399,0.038366456,0.11146646,0.020831514,-0.031067535,-0.0232766...	2025-12-17 00:24:57.459591+...
17	289	294	[0.021905689,0.034256168,0.10445873,0.013707735,-0.033443864,-0.0323686...	2025-12-17 00:25:37.49477+03
18	290	295	[0.011111551,0.042576734,0.11308443,0.023077438,-0.0280711,-0.02802839,0...	2025-12-17 00:25:37.499672+...
19	291	296	[-0.01740349,0.011256501,-0.058464818,0.04002981,-0.043390393,-0.0431922...	2025-12-17 00:26:17.441808+...

Рисунок 13 – Скриншот таблицы «brand_models»

Отображение наполнения MinIO менее интересное, поэтому приведем только один скриншот наполнения бакета «mediafiles» в папке «video», что видно на рисунке 14.

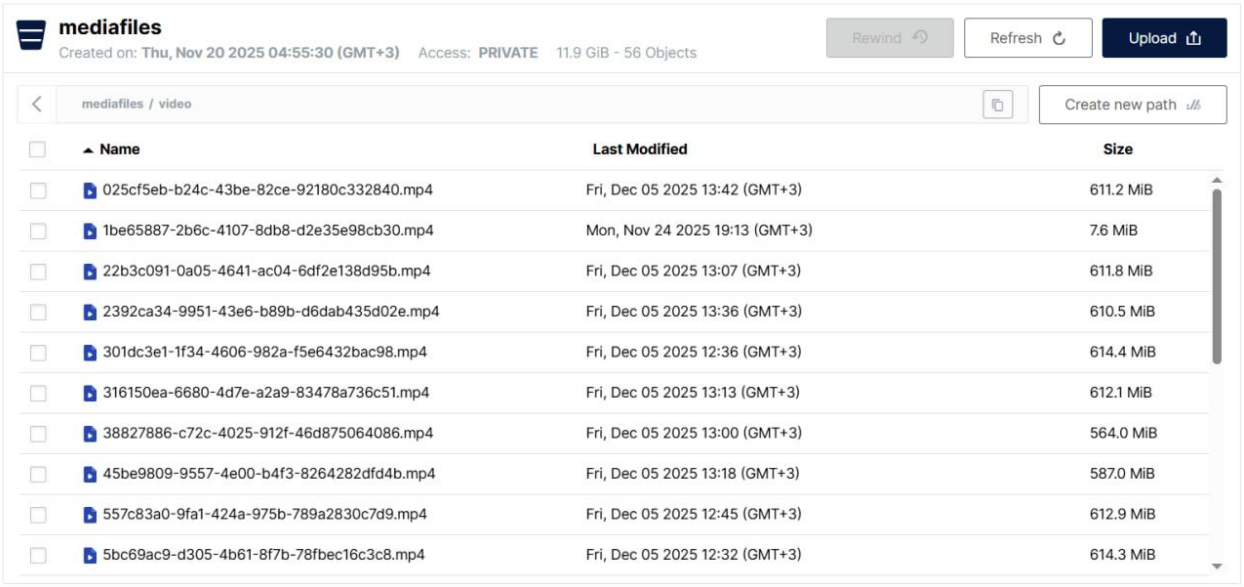


Рисунок 14 – Скриншот бакета «mediafiles» папки «video» в MinIO

ЗАКЛЮЧЕНИЕ

В ходе выполнения курсового проекта был разработан сервис для распознавания атрибутов участников дорожного движения на видео.

Для выполнения проекта были выполнены следующие задачи:

- проанализированы методы и подходы к распознаванию объектов на видео;
- разработана схема базы данных;
- разработана схема алгоритма работы сервиса;
- продемонстрирована работа сервиса с помощью пользовательского интерфейса.

Для дальнейшей работы выделены следующие направления:

- улучшение работы моделей;
- продумывания технологии тестирования сервиса;
- оптимизация кода работы сервиса;
- разработка более глубокого анализа на основе извлеченных данных.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1 Jocher G., Chaurasia A., Qiu J. Ultralytics YOLO [Электронный ресурс]. URL: https://github.com/ultralytics/ultralytics (дата обращения: 22.12.2025).

2 Aharon N., Orfaig R., Bobrovsky B. BoT-SORT: Robust Associations Multi-Pedestrian Tracking // arXiv preprint arXiv:2206.14651. 2022.

// International conference on machine learning. PMLR, 2021. P. 8748–8763.

3 Xiao Wang, Jiandong Jin, Chenglong Li, Jin Tang, Cheng Zhang, Wei Wang. Pedestrian Attribute Recognition via CLIP based Prompt Vision-Language Fusion// arXiv preprint. 2023.

4 Radford A. et al. Learning transferable visual models from natural language supervision // International conference on machine learning. PMLR, 2021. P. 8748–8763.

5 Deng Y. et al. Pedestrian attribute recognition at far distance // Proceedings of the 22nd ACM international conference on Multimedia. 2014. P. 789–792.

6 Du Y. et al. PP-OCR: A practical ultra lightweight OCR system // arXiv preprint arXiv:2009.09941. 2020.

7 Pedregosa F. et al. Scikit-learn: Machine Learning in Python // Journal of Machine Learning Research. 2011. Vol. 12. P. 2825–2830.

8 He K. et al. Deep residual learning for image recognition // Proceedings of the IEEE conference on computer vision and pattern recognition. 2016. P. 770–778.

9 Krause J. et al. 3d object representations for fine-grained categorization // Proceedings of the IEEE international conference on computer vision workshops. 2013. P. 554–561.

10 PostgreSQL: The World's Most Advanced Open Source Relational Database [Электронный ресурс]. URL: https://www.postgresql.org/docs/ (дата обращения: 22.12.2025).

11 MinIO Object Storage Documentation [Электронный ресурс]. URL: https://min.io/docs/minio/linux/index.html (дата обращения: 22.12.2025).

ПРИЛОЖЕНИЕ А

Фрагменты исходного кода программы

Листов 2

```
if mediafile.media_type == MediaType.PHOTO:
    total_frames = 1
    frame = cv2.imread(temp_path)
    if frame is None:
        raise ValueError(f"Could not read image file from
{temp_path}")
    file_thumbnail_path = minio_put_thumbnail(frame, file_id,
minio, False, False)
    mediafile.thumbnail_path = file_thumbnail_path
    result = detector(frame, classes=classes)[0]
    object_count = 0
    for coordinates, cls in zip(result.boxes.xyxy,
result.boxes.cls):
        object_count += 1
        mediafile.progress = int(object_count * 90 /
len(result.boxes.cls))
        mediafile.status_message = f'Распознавание объекта
{object_count} из {len(result.boxes.cls)}'
        db.commit()
        coordinates = list(map(int, coordinates))
        x1, y1, x2, y2 = coordinates
        object_crop = frame[y1:y2, x1:x2]
        tracked_objects[object_count] = {
            'class': names[int(cls)],
            'frame_start_end': [[0, 0]],
            'time_in_video': 1,
            'boxes': {str(0): coordinates},
            'reid_embedding': get_reid_embedding(object_crop,
int(cls) == 0)
        }
        if int(cls) == 0: # person
            tracked_objects[object_count]['attributes'] =
ped_atr_rec(object_crop)
            tracked_objects[object_count]['thumbnail_path'] =
minio_put_thumbnail(object_crop, file_id, minio, True, False,
object_count)
            pedestrian_count += 1
        else: # vehicle
            tracked_objects[object_count]['attributes'] =
{'plate_text': None,
                                                        'col
or_label': None,
                                                        'col
or_code': None,
                                                        'bra
nd': None,
                                                        'mod
el': None,
                                                        }
            tracked_objects[object_count]['thumbnail_path'] =
minio_put_thumbnail(object_crop, file_id, minio, True, True,
object_count)
            vehicle_count += 1
```

```
            ocr_result = plate_detect_and_ocr(plate_detector,
anpr_model, object_crop)
            if ocr_result:
                tracked_objects[object_count]['attributes']['pla
te_text'] = ocr_result[0]
            color_result = get_car_color(car_segmentator,
object_crop, body_part_ids)
            tracked_objects[object_count]['attributes']['color_l
abel'] = color_result['label']
            tracked_objects[object_count]['attributes']['color_c
ode'] = color_result['rgb']
            brand, car_model =
split_make_and_model_str(get_vehicle_make_and_model_class(object
_crop, vehicle_recognitor, device))
            tracked_objects[object_count]['attributes']['brand']
= brand
            tracked_objects[object_count]['attributes']['model']
= car_model
```