

## Введение

В современном мире для обнаружения пожаров широко применяются различные автоматические системы, основанные на датчиках. Однако у них имеются существенные проблемы, включая задержку в обнаружении, серьезные трудности с выявлением пожаров на открытых территориях и в просторных помещениях.

В случае возгорания на открытых пространствах или в просторных помещениях стандартные датчики срабатывают уже тогда, когда пожар трудно поддаётся тушению и причиняет значительный ущерб. Например, высокие потолки на складах или пространство улиц не способствуют обнаружению дыма на ранних этапах, когда его можно было бы легко потушить. В связи с этим использование систем видеонаблюдения более предпочтительно для подобных пространств. Но применение таких систем требует постоянного мониторинга камер человеком.

Применение систем видеонаблюдения исключает ошибки автоматических систем, но ограничивает их до мониторинга человеком. Чтобы уменьшить влияние человеческого фактора и автоматизировать обнаружение пожаров, необходимо использовать системы с применением обученных нейронных сетей.

## 2 Подготовка датасета

Сбор датасета является критически важным этапом в процессе обучения и разработки моделей машинного обучения и компьютерного зрения. Качество и объем данных напрямую влияют на способность модели корректно функционировать в различных условиях и сценариях. В рамках поставленной цели необходимо собрать достаточное количество данных, чтобы исключить проблемы с обучением.

Интернет служит основным источником для получения изображений и видео. В нём можно обнаружить как уже готовые датасеты, так и отдельные файлы, которые потребуется собирать вручную или с применением парсеров. Дополнительно, некоторые данные можно получить, организовав самостоятельную ручную съемку нужных объектов.

Для датасета не подойдут все подряд изображения и видео с дымом, поэтому выделен ряд требований к ним:

- 1) изображения и видео должны иметь приемлемое разрешение и быть достаточно четкими, чтобы обеспечить достаточное количество деталей для анализа и распознавания;
- 2) датасет должен включать изображения и видео, снятые в различных условиях освещения, погоды и времени суток, чтобы модель могла корректно функционировать в различных реальных сценариях;
- 3) необходимо собрать данные с различными типами дыма, чтобы улучшить способность модели различать их в разнообразных условиях;
- 4) изображения и видео не должны содержать водяных знаков.

Эти требования также применимы к сбору данных с другими аэрозолями. Однако для облаков существует специальное правило: их вид должен быть с земли, и они должны перекрывать какие-либо объекты, создавая тем самым иллюзию возгорания.

### 2.1 Сбор данных

Сбор необходимых изображений с дымом в сети Интернет имел свои сложности, но существует множество доступных датасетов с изображениями пожаров. Некоторые из них были получены, после чего вручную был проведен отбор подходящих изображений с дымом. Кроме того, были собраны видеозаписи с пожарами из различных источников. Эти видеозаписи были тщательно проанализированы и сегментированы, чтобы выделить фрагменты, содержащие четкие примеры задымленности.

Для улучшения качества данных и обеспечения их соответствия критериям обучения, все отобранные изображения и видеозаписи были предварительно обработаны. Эта обработка включала корректировку яркости и контрастности, удаление шума и артефактов,

а также масштабирование до стандартного разрешения, подходящего для использования в моделях машинного обучения.

Реальные данные оказались недостаточными, так как, несмотря на распространённость пожаров, найти достаточное количество изображений и видео, соответствующих требованиям, было сложно. В качестве дополнительного источника данных был использован метод генерации синтетических изображений с помощью Stable Diffusion XL. Параметры генерации включали значение 7 для CFG и 40 для Steps, с использованием семплера DPM++ 2M Karras. Для генерации изображений использовались запросы следующего содержания: «city, road, car in fire, building in fire, fire, flame, smoke, traffic, realistic photo, photo, hires, 4K, CCTV camera, view from camera angle, view from above». Пример синтетических изображений с дымом представлен на рисунке 5.



Рисунок 5 – Примеры синтетических изображений с дымом

Эксперименты с синтетическими данными принесли интересные результаты в процессе разработки моделей для распознавания дыма. Они показали, что модели машинного обучения не различают реальный дым от дыма, сгенерированного искусственно. Это открытие существенно расширяет возможности использования синтетических данных в обучении. Использование синтетических данных не только облегчает сбор нужного объема данных для обучения, но и повышает гибкость и масштабируемость процесса тренировки моделей. Эти факторы делают синтетические данные неотъемлемой частью процесса обучения моделей для распознавания дыма, позволяя создавать более мощные и точные системы.

Создавать синтетические данные для других аэрозолей не имеет смысла, так как они легко доступны в реальных условиях и не требуют сложных процедур генерации. Например, туман и облака являются естественными явлениями, которые можно легко запечатлеть в различных климатических условиях, особенно в России. Это позволяет использовать реальные изображения и видео для обучения моделей, улучшая их точность

и надежность. Также они не являются главным классом для распознавания, поэтому количество данных не обязательно должно быть большим.

Описание используемых датасетов как изображений так и видео, собранных из различных источников, представлено в таблице 3.

Таблица 3 – Описание используемых датасетов с изображениями и видеозаписями дыма

Датасет	Количество изображений или видеозаписей	Год	Описание
Images search	543	2023	Изображения, собранные с различных интернет-ресурсов
FIRE Dataset [13]	755	2018	Набор данных был создан во время NASA Space Apps Challenge, целью было использование набора данных для разработки модели, которая может распознавать изображения с огнем.
Fire Detection from CCTV [14]	288	2019	Кадры с различных видеозаписей пожаров
Synthetic	559	2024	Синтетические изображения, созданные с помощью Stable Diffusion XL
Videos search	345	2023	Видеозаписи, собранные с различных интернет-ресурсов

Изображения из найденных датасетов демонстрируют дым в различных его проявлениях. Пример изображений с дымом представлен на рисунке 6.

В ходе проведения текущего исследования использовались 2145 изображений и 345 видеозаписей, полученных из двух готовых датасетов, а также изображения и видеозаписи, собранные вручную с различных интернет-ресурсов. Так же использовались синтетические данные. Такого количества изображений и видеозаписей было достаточно для проведения экспериментов с обучением.



Рисунок 6 – Примеры изображений с дымом из датасетов

Для минимизации возможных ложных срабатываний, связанных с схожестью дыма с другими аэрозолями, был создан датасет, также включающий изображения и видеозаписи с различными аэрозолями. Этот датасет будет использоваться для проверки способности нейронной сети распознавать указанные аэрозоли как дым.

Была выдвинута гипотеза о том, что обученная нейронная сеть сможет определять эти аэрозоли как отдельные классы, а не как отсутствие дыма, может снизить количество ложных срабатываний. Для проверки гипотезы был создан не только датасет, но и проведена аннотация изображений и видеозаписей со следующими аэрозолями:

- 1) туман, который в реальных условиях перепутать с дымом довольно сложно, так как туман куда более рассеянный, чем дым;
- 2) облака, которые в реальности сложно идентифицировать как дым, так как характер расположения облаков не схож с дымом.

Найти подходящие изображения с туманом для обучения было непросто. Как таких готовых датасетов с изображением тумана в свободном доступе в сети Интернет нет. Есть некоторые датасеты, которые предлагают синтетические изображения тумана, но они не пригодны, так как имеют неудовлетворительное качество генерации. В итоге изображения были собраны и отобраны вручную из различных интернет-ресурсов, включая два готовых датасета, а также небольшую часть изображений, сделанных с использованием личного фотооборудования. Что касается видеозаписей, их поиск оказался менее трудоемким. В случае с туманом видеозаписи оказались особенно важными, поскольку

туман, в отличие от дыма, обладает меньшей подвижностью, что упрощает их различение. Описание используемых датасетов представлено в таблице 4.

Таблица 4 – Описание используемых датасетов с изображениями и видеозаписями тумана

Датасет	Количество изображений или видеозаписей	Год	Описание
Camera	2	2023	Изображения сделанные на личном фотоустройстве
Foggy images [15]	7	2023	Датасет изображений взятый с сайта Kaggle. Автором не указано описание
Search	213	2023	Изображения собранные с различных интернет-ресурсов
Weather Image Recognition [16]	24	2021	Датасет изображений взятый с сайта Kaggle. Датасет содержит 6852 изображения для различных типов погоды (11). Авторами собран специально для задачи классификации. Из данного датасета были отобраны только изображения с туманом
Videos search	23	2023	Видеозаписи, собранные с различных интернет-ресурсов

Изображения с туманом условно можно разделить на два вида: туман покрывает все изображение и туман покрывает некоторую часть изображения. В обоих случаях его можно перепутать с дымом, так как дым, при рассеивании, может быть похож на подобную пелену. Пример изображения, где туман покрывает его полностью, представлен на рисунке 7.



Рисунок 7 – Изображение, где туман покрывает его полностью

Пример изображения, где туман покрывает некоторую его часть, представлен на рисунке 8.



Рисунок 8 – Изображение, где туман покрывает некоторую его часть

Дым нечасто принимает сильно рассеянную форму, но все же умение обнаруживать туман как отдельный класс может снизить количество ложных распознаний дыма, когда на кадре на самом деле туман.

Собрать необходимые изображения облаков в сети Интернет стало невозможной задачей. Все найденные готовые датасеты содержат изображения облаков снятые со спутников и совершенно не подходят для поставленной цели. На различных интернет-ресурсах также найти изображения, отвечающие критериям, не удалось, поэтому все

изображения были сделаны вручную на личном фотоустройстве. Видеозаписи также были сделаны на личное фотоустройство. Описание используемого датасета представлено в таблице 5.

Таблица 5 – Описание используемого датасета с изображениями облаков

Датасет	Количество изображений или видеозаписей	Год	Описание
Camera	86	2023	Изображения сделанные на личном фотоустройстве
Videocamera	63	2023	Видеозаписи, сделанные на личном фотоустройстве

Главным критерием отбора изображений с облаками было наличие некоторого пересечения облаков с какими-либо объектами. Пример таких изображений представлен на рисунке 9.

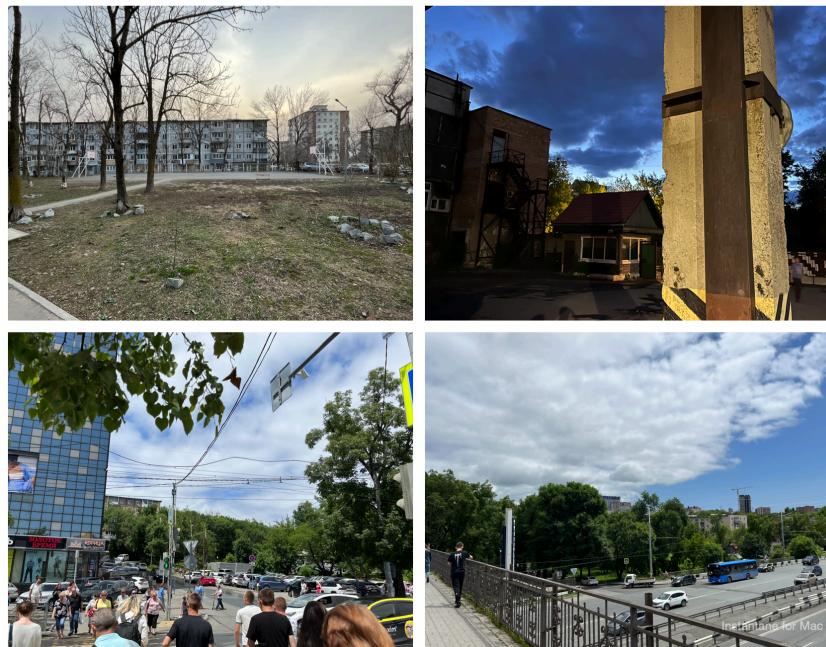


Рисунок 9 – Изображения с облаками

В процессе разработки и обучения моделей для распознавания дыма, большое значение имеет не только наличие изображений и видео, содержащих дым, но и включение в датасет материалов, где дым отсутствует. Это дополнение играет критическую роль в обучении моделей, повышая их способность корректно идентифицировать отсутствие дыма, что является неотъемлемой частью задачи распознавания.

Включение изображений и видео без дыма в тренировочные датасеты требует тщательной подготовки и баланса. Важно обеспечить, чтобы количество данных без дыма не перевешивало данные с дымом, чтобы не снизить чувствительность модели к его присутствию. Соотношение и распределение данных были оптимизированы с учетом конкретной задачи и условий, в которых будет работать модель.

## 2.2 Аннотация данных

После сбора изображений и видеозаписей с аэрозолями, их необходимо было аннотировать. Качественная аннотация является ключевым этапом подготовки данных, позволяющим превратить сырье изображения в информативный датасет, способный эффективно обучать и тестировать алгоритмы машинного обучения. Аннотация данных включает в себя процессы идентификации и маркировки объектов или явлений на изображениях, что позволяет моделям нейронных сетей обучаться на конкретных примерах и вырабатывать точные предсказания на основе визуальной информации.

В качестве инструмента для аннотирования был выбран Label Studio. Это гибкий инструмент для аннотации данных, который поддерживает различные типы данных, включая изображения, тексты, аудио и видео. Он позволяет вручную отсматривать каждое изображение или видео и выставлять бокс с нужным классом на необходимую область. Label Studio позволяет проводить как ручную аннотацию, так и интегрировать механизмы автоматической аннотации.

Для автоматической аннотации использовалась модель Yolo (You Only Look Once), предназначенная для обнаружения объектов. Несмотря на то, что модель Yolo не идеально подходила для достижения главной цели исследования, она эффективно справлялась с идентификацией и начальной разметкой определенных типов объектов. Модель была обучена на первоначально аннотированных данных, после чего интегрирована в проект Label Studio для проведения автоматической аннотации. Порядок работы был следующим:

- 1) ручное аннотирование части изображений и видео;
- 2) обучение модели Yolo на аннотированной части данных;
- 3) подключение обученной модели Yolo к проекту в Label Studio;
- 4) запуск автоматического аннотирования;
- 5) ручное исправление автоматической разметки.

Автоматическое аннотирование, хоть и не идеально, играет ключевую роль в оптимизации процесса разметки данных. Оно позволяет значительно сократить время аннотации и повысить эффективность работы.

Аннотирование дыма, как и других аэрозолей, имеет свои особенности, так как у аэрозолей нет четких границ своей формы. Дым аннотировался согласно нескольким принципам.

Если дым занимает практически всю площадь изображения и не имеет четко выраженных элементов, то такой дым аннотируется одним большим боксом. Пример такого аннотированного изображения представлен на рисунке 10.



Рисунок 10 – Дым не имеет четких форм

Если дым представляет собой четко выраженные формы, то каждая из этих форм выделяется отдельным боксом и дополнительно набор из нескольких (или все) форм выделяются одним общим боксом. Пример такого аннотированного изображения представлен на рисунке 11.



Рисунок 11 – Дым имеет четкие формы

Если дым небольшой и имеет четко различимые фрагменты, то каждый фрагмент выделяется отдельным боксом и весь дым одним общим боксом. Пример такого аннотированного изображения представлен на рисунке 12.



Рисунок 12 – Дым состоит из различных фрагментов

Аннотирование видеозаписей имеет свои особенности. С помощью специального инструмента в Label Studio дым на видеозаписях аннотировался покадрово боксами по тем же принципам, что и изображения. Пример такого аннотирования представлен на рисунке 13.

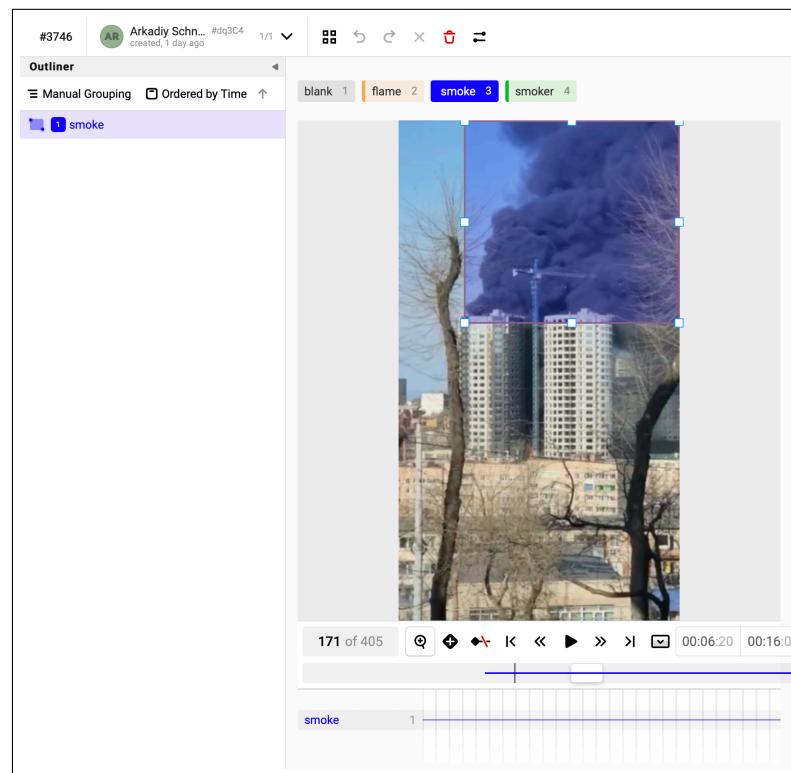


Рисунок 13 – Аннотация видеозаписи

Принцип разметки видео заключается в тщательной идентификации и маркировке объектов на каждом кадре видеозаписи. На изображении видно, как область с дымом была выделена прямоугольной рамкой и помечена меткой «smoke». Далее при отходе дыма из выделенной области, ее необходимо сдвинуть так, чтобы боксы покрывали необходимые участки дыма.

Также для проверки гипотезы существуют следующие классы:

- 1) smoke – дым;
- 2) fog – туман;
- 3) cloud – облако.

Аннотирование изображений и видеозаписей осуществлялось с применением различных принципов с целью улучшения точности распознавания дыма в различных его проявлениях. В процессе аннотирования возникали трудности при работе с изображениями, содержащими сложные формы аэрозоля, что могло повлиять на качество дальнейшего обучения. Тем не менее, полученные результаты были достаточными для последующего корректного обучения.

Аннотирование тумана простое, так как из-за своей рассеянности он имеет равную зону расположения, которую необходимо просто выделить одним боксом. Пример аннотированного изображения, где туман покрывает изображение практически полностью, представлен на рисунке 14.



Рисунок 14 – Аннотированное изображение, где туман покрывает его полностью

Пример аннотированного изображения, где туман покрывает его некоторую часть, представлен на рисунке 15. В данном случае туман можно перепутать с дымом, так как дым вдали при рассеянии создаёт пелену в определённой области, что может повлиять на работу нейронной сети.



Рисунок 15 – Аннотированное изображение, где туман покрывает часть

Аннотирование видеозаписей сделано по аналогии с аннотированием дыма. Видео-записи покадрово размечены с помощью боксов.

Аннотирование облаков схоже с аннотированием дыма, но здесь чаще встречается вариант с хорошо различимыми фрагментами. Пример аннотированных изображений с облаками представлен на рисунке 16.



Рисунок 16 – Аннотированные изображения с облаками

Аннотирование видеозаписей сделано по аналогии с аннотированием дыма. Видео-записи покадрово (с использованием ключевых кадров) размечены с помощью боксов.

Облака меньше всего походят на дым, но в некоторых ракурсах и при определенной освещенности легко можно спутать с дыром.

Теперь, когда данные тщательно подготовлены и проверены, можно с уверенностью приступить к следующему важному этапу исследования – разработке и тестированию моделей машинного обучения, что позволит изучить и оценить эффективность различных алгоритмов в задаче распознавания дыма.

### 2.3 Разделение данных

Разделение данных на обучающий и тестовый наборы является критически важным этапом в процессе подготовки данных для обучения и оценки нейронных сетей. Этот этап не только определяет, как модель будет обучаться и адаптироваться к новым данным, но и как точно она сможет оценить свои реальные возможности перед применением в реальных условиях. Основная цель разделения данных – обеспечить, чтобы обучение модели проходило на разнообразном и репрезентативном наборе примеров, в то время как в тестирование проводились на отдельных, независимых наборах, что позволяет избежать переобучения и обеспечивает объективную оценку производительности.

Для автоматизации разделения данных на обучающий и тестовый наборы был написан специальный скрипт на языке программирования Python. Изначально необходимо подготовить папку с двумя папками «images» и «labels». В первой папке располагаются изображения, во второй – текстовый файл, содержащий данные о разметке. Пример структуры представлен на рисунке 17.

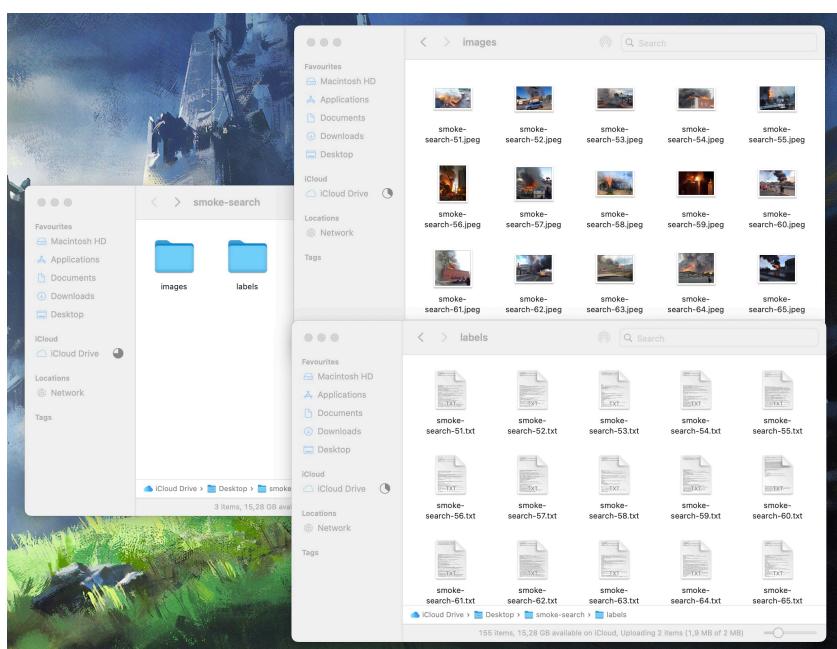


Рисунок 17 – Данные до разделения на два набора

Запуск скрипта осуществляется с помощью командной строки с передачей аргументов командой строки: путь до папки с изображениями, путь до папки с текстовыми файлами с разметкой, путь до выходной папки, размер тестового набора в процентах. Пример использования команды представлен на рисунке 18.

```



```

Рисунок 18 – Пример команды запуска скрипта

После того, как скрипт закончить работу, в выходной папке появятся папка «train» для обучающего набора и папка «val» для тестового набора. Данные разделяются случайным образом. Разделенные данные представлены на рисунке 19.

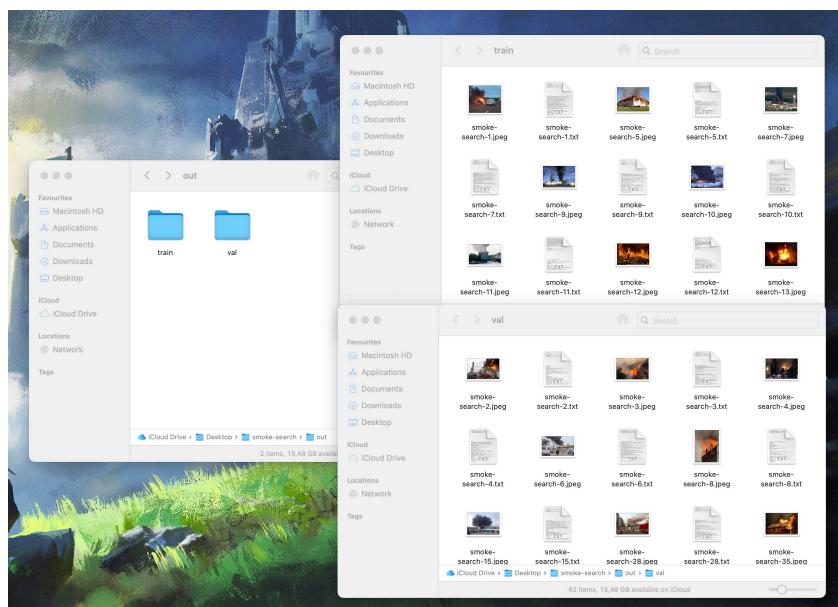


Рисунок 19 – Данные после разделения на два набора

Согласно примеру было 155 файлов. После запуска скрипта для разделения данных с 20% данных на тестовый набор, на обучение пойдет 124 файла и на тестирование – 31 файл.

## 4 Создание бэкенд части системы распознавания пожароопасных ситуаций

Разработанная модель для обнаружения дыма включена в более широкую систему, предназначенную для выявления ситуаций, которые могут привести к пожарам. Эта система также включает в себя модели для обнаружения пламени и курильщиков, однако они не рассматриваются в данной работе. Чтобы убедиться в эффективности этих моделей и возможности их интеграции, необходимо создать инфраструктуру, которая может обрабатывать видеопотоки в реальном времени и по результатам анализа моделей предоставлять предупреждения о возможных пожарах.

Система hiFireCam, предназначенная для автоматического обнаружения пламени, дыма и курильщиков, разработана для анализа видеоданных с целью идентификации потенциально пожароопасных ситуаций. Эта система особенно подходит для использования в больших помещениях, таких как аэропортские залы, склады или ангары, а также на открытых пространствах, где обычные датчики дыма могут быть неэффективны. Основным пользователем системы является оператор, который работает в тесном контакте с пожарными службами.

### 4.1 Проектирование бэкенд части общей системы

#### 4.1.1 Требования к реализуемой системе

Система обнаружения дыма и пожаров должна отвечать ряду специфических требований, чтобы быть максимально эффективной и надежной в различных условиях эксплуатации. Эти требования включают в себя как технические характеристики, так и функциональные возможности, которые обеспечивают высокую точность распознавания и минимизацию ложных срабатываний. Ниже перечислены ключевые требования, которые должны быть учтены при разработке и внедрении такой системы:

- 1) анализировать видеоданные (с подключенными к ней видеокамер) на предмет возникновения пожаров и уведомлять об этом оператора;
- 2) помимо определения факта возникновения пожара, должна быть записана вся история и процесс возникновения пожара;
- 3) система может обеспечить обнаружение курящих людей, тем самым предоставляя возможность автоматизированного контроля за пожароопасными зонами;
- 4) система предоставляет удобный интерфейс пользователя-оператора для доступа к стримам с камер, архиву записей, текущему статусу анализа, истории выдачи предупреждений.

Система не должна отвечать следующим требованиям:

- 1) заниматься вызовом пожароохраных служб, для этого будут предусмотрены специальные интерфейсы и интеграции;
- 2) заниматься пожаротушением или управлением пожарными системами;
- 3) заменять сертифицированную систему пожарной охраны (лишь дополнять).

На основании данных требований спроектирована архитектура системы, состоящая из множества связанных систем.

На рисунке 37 представлена схема с описанием архитектуры всей системы.

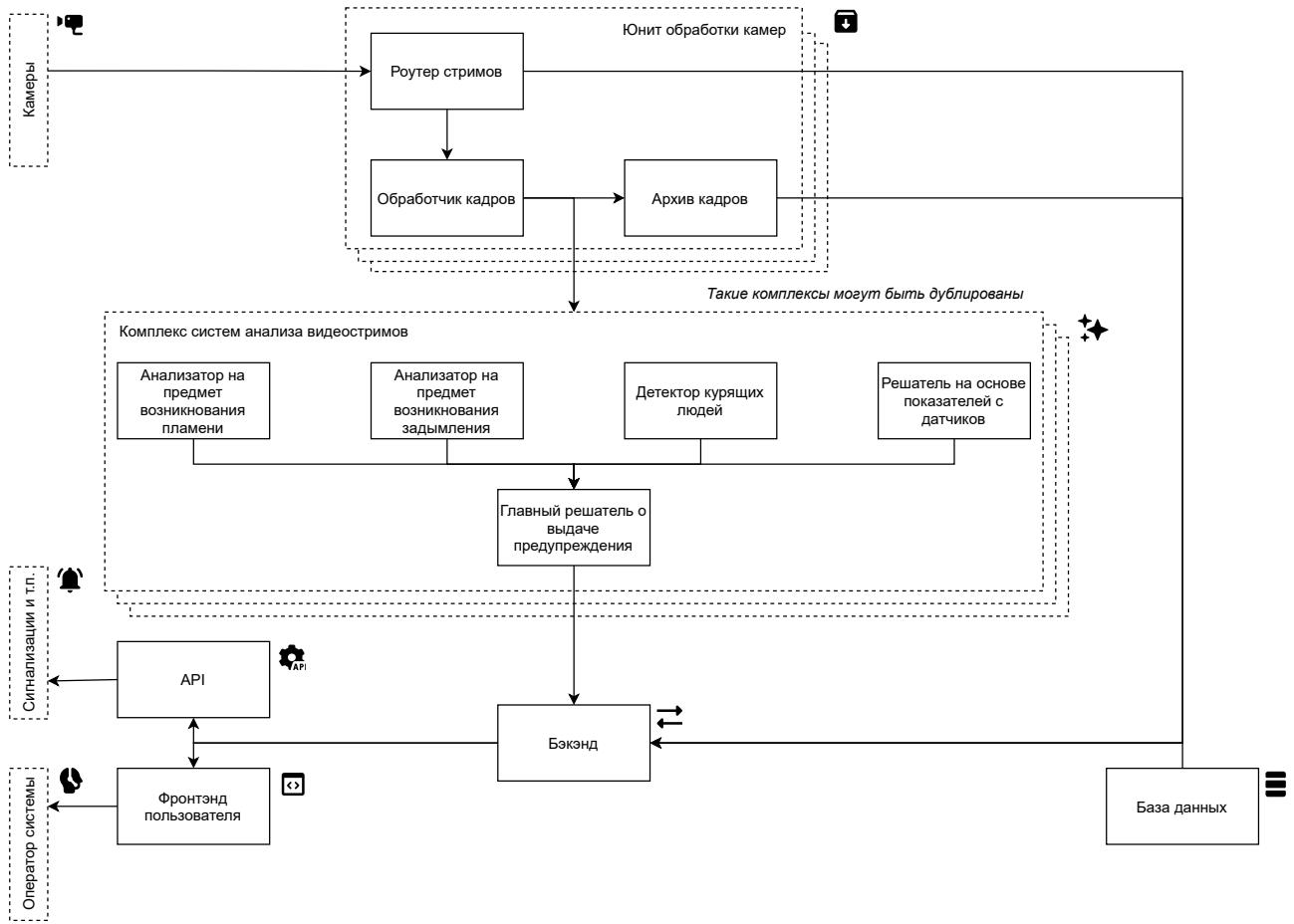


Рисунок 37 – Архитектура системы и ее компоненты

Система поделена на независимые компоненты:

- 1) юнит обработки камер – комплекс систем, занимающихся обработкой стримов с камер, сохранением и архивацией кадров, а так же передачей группы кадров для анализа;
- 2) обработчик кадров – вытаскивает из стрима кадры, делает первичную обработку (например, ресайз под заданный стандарт обработки нашими алгоритмами), передает кадры для сохранения в архив, а также передает группу кадров в комплекс анализа;
- 3) архив кадров и стримов – блоковое хранилище на базе S3 (S3 – Simple Storage Service, сервис хранения объектов), сохраняет всю историю кадров и стримов;

- 4) комплекс систем анализа видео стримов – это множество алгоритмов, на основе которых комплекс выдает разного рода предупреждения;
- 5) база данных – обычная реляционная БД;
- 6) бэкэнд – занимается обработкой и сохранением предупреждений, обработкой запросов пользователя, выдачей уведомлений;
- 7) API;
- 8) фронтэнд пользователя.

Большинство компонентов предполагают возможность репликации (запуска нескольких копий). Связь между компонентами осуществляется через брокер сообщений (на схеме отсутствует).

В данной работе рассмотрена разработка бэкенда и базы данных.

#### **4.1.2 Анализ и выбор инструментальных средств для создания бэкенда**

Для проектирования бэкенда необходимо воспользоваться программным обеспечением, с помощью которого можно создавать различные диаграммы и схемы. Такие инструменты позволяют визуализировать архитектуру системы, продемонстрировать взаимодействие между компонентами и обеспечить ясное представление о структуре и логике работы бэкенда.

Разработка высокоэффективного и надежного бэкенда для современных приложений требует использования специализированного программного обеспечения. Правильный выбор инструментов и технологий позволяет обеспечить стабильную работу системы, эффективное управление данными и легкость в развёртывании и поддержке приложения. Для разработки бэкенда необходимо следующее программное обеспечение:

- 1) операционная система;
- 2) программное обеспечение для автоматизации развёртывания и управления приложениями;
- 3) СУБД (система управления базами данных);
- 4) фреймворк для создания бэкенда.

Каждый из этих компонентов имеет свои особенности и требования, которые необходимо учитывать для создания эффективного и стабильного бэкенда.

Для анализа инструментальных средств для построения диаграмм и схем были выбраны такие популярные решения как Draw.io [20], Visio [21], Lucidchart [22]. В таблице 7 представлено сравнение этого программного обеспечения.

Таблица 7 – Сравнение программного обеспечения для построения схем и диаграмм

Критерий	Draw.io	Visio	Lucidchart
Бесплатное использование	Полностью бесплатный	Требует покупки лицензии	Бесплатная версия с ограниченными возможностями
Оффлайн режим	Поддерживается через десктопное приложение	Полностью поддерживается	Не поддерживается
Поддержка шаблонов и символов	Обширная библиотека шаблонов	Очень широкий выбор профессиональных шаблонов	Хорошая поддержка с современными шаблонами
Совместная работа	Поддержка реального времени через веб	Ограниченнaя, через SharePoint или OneDrive	Отличная поддержка совместной работы в реальном времени

В результате сравнения программного обеспечения, был выбран Draw.io, так как он удобен в использовании, полностью бесплатный и имеет оффлайн режим.

В качестве программного обеспечения для автоматизации развёртывания и управления приложениями был выбран Docker, как самое популярное и удобное средство для контейнеризации, аналогов которому нет. Каждый компонент системы запускается в собственном контейнере.

Для запуска любого программного обеспечения необходима операционная система. Docker хорошо работает с linux дистрибутивами, поэтому выбор осуществлялся среди них. Для анализа выбраны три самых популярных дистрибутива: Debian [23], Ubuntu [24], CentOS [25]. В таблице 8 представлено сравнение этих операционных систем.

Таблица 8 – Сравнение операционных систем

Критерий	Debian	Ubuntu	CentOS
Стабильность	Высокая	Средняя	Высокая
Поддержка Docker	Хорошая, требует настройки	Хорошая, более интуитивная установка	Отличная, оптимизирована под серверы
Сообщество и поддержка	Обширное, множество форумов и документации	Огромное, с активной поддержкой пользователей	Большое, с акцентом на корпоративное использование
Частота обновлений	Регулярные, стабильные релизы	Частые, с быстрым внедрением новшеств	Реже, с фокусом на стабильность и безопасность
Совместимость с оборудованием	Широкая поддержка различного оборудования	Отличная поддержка современного оборудования	Хорошая поддержка, особенно в серверных конфигурациях
Легкость использования	Требует больше настроек и понимания Linux	Дружелюбнее для новичков	Настроен на использование в бизнес-среде
Подходит для разработки	Отлично подходит для опытных разработчиков	Отлично для начинающих и профессионалов	Идеально для корпоративных разработчиков и продакшн сред

При сравнении операционных систем были выявлены их достоинства и недостатки. В качестве операционной системы был выбран CentOS как система с наилучшей поддержкой Docker.

Для хранения данных системы необходима база данных, которая будет реализоваться с помощью СУБД. В контексте системы подойдет классическая реляционная модель,

поэтому для выбора были представлены две самые популярные и бесплатные СУБД – MySQL [26] и PostgreSQL [27]. Сравнение этих систем представлено в таблице 9.

Таблица 9 – Сравнение СУБД

Критерий	MySQL	PostgreSQL
Поддержка SQL стандартов	Частичная	Полная
Расширяемость	Ограниченнaя, не поддерживает создание пользовательских функций	Полная поддержка создания новых типов данных, операторов, функций
Поддержка сложных запросов	Базовая, сложные запросы могут требовать дополнительной оптимизации	Превосходная, включая оптимизированную поддержку сложных операций и CTE
Производительность при работе с большими объемами данных	Хорошая, но может страдать в некоторых кейсах	Отличная, эффективно обрабатывает большие объемы и сложные транзакции
Поддержка конкурентных записей	Может испытывать замедление при высокой конкуренции	Использует многоуровневую модель MVCC для высокой производительности при конкурентных операциях
Открытость и сообщество	Активное, но под управлением Oracle	Очень активное, полностью открытое и независимое сообщество

На основе проведенного анализа была выбрана СУБД PostgreSQL, так как полная поддержка SQL стандартов упрощает интеграцию и масштабирование приложений. PostgreSQL также выделяется своей расширяемостью, позволяя разработчикам создавать собственные функции и типы данных, что идеально подходит для сложных или специализированных бизнес-задач. Эффективность обработки больших объемов данных и высокая производительность при конкурентных операциях делают её предпочтительным выбором для систем, требующих высокой надежности и производительности.

Для написания кода бэкенда необходим специализированный фреймворк. Для выбора были отобраны три популярных фреймворка, используемые различные языки программирования – FastAPI [28], Express [29], Laravel [30]. В таблице 10 представлено сравнение этих фреймворков.

Таблица 10 – Сравнение фреймворков для разработки бэкенда

Критерий	FastAPI	Express	Laravel
Производительность	Высокая	Средняя	Средняя
Поддержка асинхронности	Полная	Частичная (через сторонние модули)	Ограниченнная
Встроенная поддержка API	Отличная (с автоматической генерацией документации)	Базовая	Хорошая (требует настройки)
Легкость разработки	Высокая (минимальный бойле рплейт, быстрая разработка)	Средняя (требуется много настройки)	Средняя (требует глубокого понимания фреймворка)
Масштабируемость	Отличная	Средняя	Хорошая
Безопасность	Сильные встроенные механизмы	Модульная безопасность	Сильные встроенные механизмы

В итоге был выбран FastAPI, так как этот фреймворк демонстрирует отличную производительность и полную поддержку асинхронности, что важно для разработки высокопроизводительных веб-приложений. FastAPI упрощает создание чистых API с автоматической генерацией документации, значительно сокращая время, необходимое для разработки и документирования.

#### 4.1.3 Описание основного прецедента бэкенда

Проектирование с использованием Унифицированного языка моделирования (UML) стало стандартом де-факто в сфере разработки программного обеспечения. Этот мощный инструмент моделирования позволяет архитекторам и разработчикам визуализировать структуру и поведение системы на разных этапах разработки. UML не только облегчает

процесс проектирования сложных систем, но и способствует лучшему пониманию между разработчиками и стейкхолдерами за счет стандартизованных диаграмм и символов.

Использование UML в проектировании включает создание ряда диаграмм, каждая из которых охватывает различные аспекты системы, такие как диаграммы классов, диаграммы последовательности, диаграммы состояний и диаграммы компонентов. Эти диаграммы предоставляют ценную информацию о классах и объектах, их взаимодействиях и жизненном цикле, а также об архитектуре системы в целом. Подход на основе UML способствует повышению качества проекта за счет тщательного анализа и планирования перед фактической разработкой кода, что минимизирует риски и способствует эффективной коммуникации внутри команды.

В работе бэкенда можно выделить следующие основные прецеденты:

- 1) получение видеопотока – пользователь запрашивает видео через фронтенд, система обрабатывает и передает видеопоток со стороны бэкенда;
- 2) генерация оповещений – бэкенд получает данные на предмет различных событий (например, обнаружение курильщиков, дыма, пламени) и отправляет оповещения о детектируемых событиях на фронтенд;
- 3) работа с базой данных – бэкенд, получая различные данные из других модулей, сохраняет их в базу данных.

Главным прецедентом в работе бэкенда является генерация оповещений. Алгоритм работы данного прецедента продемонстрирован в виде диаграммы последовательности на рисунке 38.



Рисунок 38 – Диаграмма последовательности

На представленной диаграмме последовательности UML описаны взаимодействия между различными модулями системы мониторинга и оповещения, включая взаимодействие с пользовательским интерфейсом и базой данных.

Диаграмма начинается с модуля обработки камер, который отправляет предупреждение о потенциальном событии, таком как возможное возникновение пожара, в модуль по работе с каналом предупреждений. Этот модуль, в свою очередь, передает информацию о возникновении пожара в модуль генерации оповещений, который обрабатывает данную информацию для создания соответствующих уведомлений.

После генерации уведомления, модуль генерации оповещений отправляет данные об оповещении в базу данных, где эта информация хранится для дальнейшего использования. В то же время, информация об оповещении передается в пользовательский интерфейс (фронтенд), позволяя пользователям получать актуальную информацию о состоянии безопасности в режиме реального времени.

Ключевым элементом диаграммы является обратная связь от модуля генерации оповещений к модулю по работе с каналом предупреждений, что позволяет корректировать и оптимизировать процесс уведомления в зависимости от полученных результатов и реакции системы. Эта обратная связь помогает поддерживать актуальность данных и эффективность системы оповещений.

#### **4.1.4 Проектирование базы данных**

Для эффективного управления и анализа данных необходимо разработать специализированную базу данных, которая будет централизованно хранить всю критически важную информацию. Эта база данных должна быть структурирована таким образом, чтобы обеспечивать легкий доступ, высокую производительность и возможности масштабирования для будущего роста и развития. В базе данных необходимо хранить информацию о следующих сущностях:

- 1) анонсы – содержится информация о работающих узлах с камерами;
- 2) узлы – содержится информация о наборах камер;
- 3) камеры – содержится информация о камерах;
- 4) пользователи – содержится информация о пользователях;
- 5) оповещения – содержится информация об отправленных оповещениях пожаров.

На основе анализа выделенных сущностей была создана схема базы данных в виде ER-диаграммы, представленной на рисунке 39.

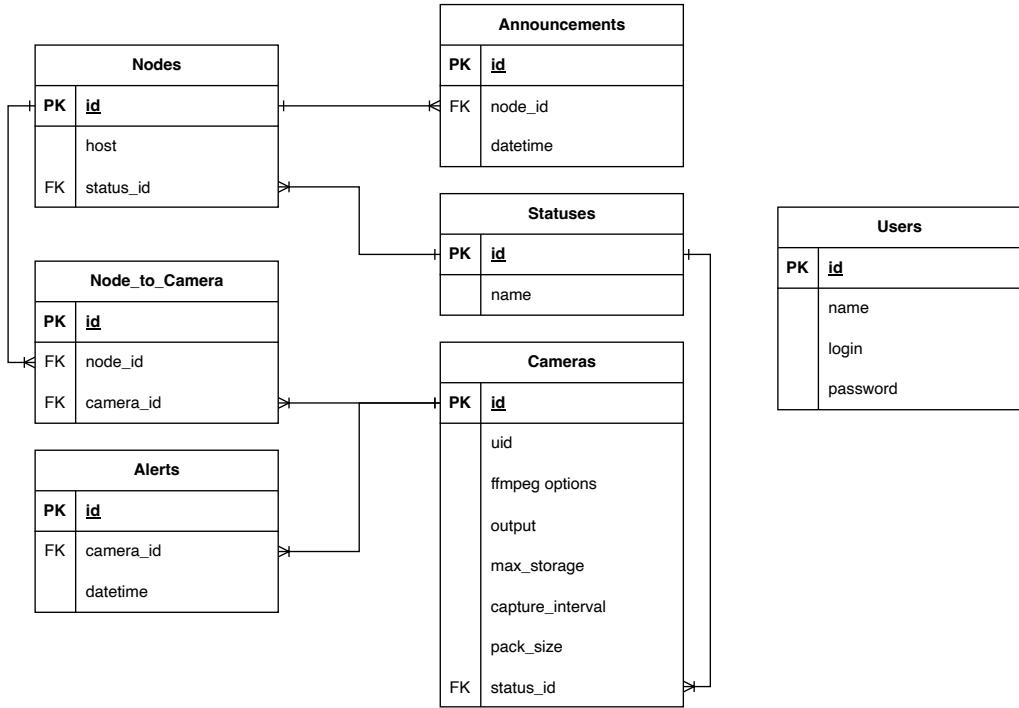


Рисунок 39 – ER-диаграмма базы данных

На схеме имеются следующие таблицы:

- 1) Узлы (Nodes) – содержится информация о наборах камер, каждый узел имеет уникальный идентификатор и хост, каждый узел может быть связан с определённым статусом из таблицы Статусы;
- 2) Статусы (Statuses) – хранит возможные состояния узлов, такие как активен, неактивен;
- 3) Узел к Камере (Node\_to\_Camera) – связывает узлы с камерами, позволяя узлам иметь несколько камер и наоборот;
- 4) Камеры (Cameras) – содержит информацию о камерах, включая уникальный идентификатор, настройки ffmpeg, параметры вывода видео, максимальное хранилище, интервал захвата и размер пакета данных;
- 5) Оповещения (Alerts) – регистрирует оповещения, сгенерированные системой, каждое оповещение связано с камерой и содержит временную метку его создания;
- 6) Пользователи (Users) – содержит информацию о пользователях системы, включая идентификатор, имя, логин и пароль;
- 7) Анонсы (Announcements) – содержит информацию о сообщениях, адресованных пользователям или узлам, каждое объявление связано с узлом и содержит временную метку.

Спроектированная база данных в дальнейшем может масштабироваться при необходимости, что обеспечивает удобную адаптацию к растущим требованиям пользователей и увеличению объемов обрабатываемых данных. Такая гибкость позволяет легко интегрировать новые функциональные возможности и обеспечивать высокую производительность системы даже при значительном увеличении количества обрабатываемых данных и пользователей.

## 4.2 Разработка бэкенда

В современной разработке программного обеспечения, бэкенд системы занимает центральное место, обеспечивая стабильность, безопасность и масштабируемость веб-приложений и сервисов. Бэкенд служит основой для обработки данных, интеграции различных системных модулей и обеспечения взаимодействия между серверной частью и пользовательским интерфейсом. В контексте разработки комплексных систем, таких как системы видеонаблюдения, управления данными или интеллектуального анализа данных, структура бэкенда играет ключевую роль в успешной реализации и функционировании приложений.

Бэкенд системы делится на несколько основных компонентов, каждый из которых выполняет свои специфические задачи и взаимодействует с другими частями системы для обеспечения целостности и эффективности обработки данных:

- 1) часть по работе с базой данных – этот компонент отвечает за все аспекты взаимодействия с базой данных, включая хранение, извлечение, обновление и удаление данных;
- 2) часть по работе с модулем обработки камер – модуль обрабатывает входные данные с камер, выполняет необходимые алгоритмы обнаружения и анализа объектов в реальном времени и передает результаты для дальнейшей обработки или немедленного реагирования в бэкенд;
- 3) часть по работе с фронтеном – эта часть системы отвечает за интеграцию серверных процессов с пользовательским интерфейсом.

Такое разделение облегчает разработку и тестирование каждого из компонентов отдельно, и позволяет более гибко масштабировать систему в зависимости от изменяющихся требований и условий эксплуатации.

### 4.2.1 Разработка части по работе с базой данных

Для успешной интеграции PostgreSQL с FastAPI было выполнено несколько шагов, начиная с установки необходимых библиотек, настройки соединения с базой данных и

заканчивая созданием асинхронных запросов. В качестве основных инструментов были выбраны библиотеки «asynccpg» и «SQLAlchemy», которые предназначены для эффективной работы с базой данных PostgreSQL в асинхронном режиме.

Инициализация базы данных осуществлялась на стороне бэкенда с использованием Object-Relational Mapping (ORM), что позволяет абстрагировать и упростить взаимодействие с базой данных в рамках объектно-ориентированной парадигмы. После установления подключения к базе данных были разработаны модели данных для каждой из таблиц. Эти модели отражают структуру таблиц и определяют взаимосвязи между данными. Пример модели данных для таблицы Cameras представлен на рисунке 40.

```
class Camera(Base):
    __tablename__ = 'cameras'

    id = Column(Integer, primary_key=True)
    uid = Column(String, unique=True, nullable=False)
    ffmpeg_options = Column(String)
    output = Column(String)
    max_storage = Column(Integer)
    capture_interval = Column(Integer)
    pack_size = Column(Integer)
    status_id = Column(Integer)
```

Рисунок 40 – Пример модели для таблицы камер

Для работы с данными был создан API, содержащий роуты или эндпоинты для каждой из моделей. Эти эндпоинты обеспечивают интерфейс для выполнения операций CRUD (создание, чтение, обновление, удаление), что является стандартной практикой для современных веб-приложений. Например, для модели Cameras были разработаны эндпоинты для получения списка всех камер и добавления новой камеры. Пример этих эндпоинтов представлен на рисунке 41.

Для обеспечения безопасности и контроля доступа к API, были внедрены механизмы аутентификации и авторизации. Это позволяет ограничить доступ к критически важным операциям и защитить данные от несанкционированного доступа. Использование токенов и ролей пользователей обеспечивает гибкое управление правами доступа.

Все созданные эндпоинты были документированы с использованием OpenAPI (Swagger), что позволяет разработчикам и пользователям легко понимать и использовать API. Генерация автоматической документации существенно упрощает интеграцию

API с другими системами и приложениями, обеспечивая прозрачность и удобство работы с интерфейсом.

```
# Эндпоинт для получения списка всех камер
@app.get("/cameras/")
async def read_cameras():
    async with AsyncSession() as session:
        async with session.begin():
            result = await session.execute(select(Camera))
            cameras = result.scalars().all()
    return cameras

# Эндпоинт для добавления новой камеры
@app.post("/cameras/")
async def create_camera(camera_data: Camera):
    async with AsyncSession() as session:
        session.add(camera_data)
        await session.commit()
    return camera_data
```

Рисунок 41 – Пример эндпоинтов

Интеграция FastAPI с базой данных PostgreSQL через асинхронные библиотеки обеспечивает высокую производительность и масштабируемость веб-приложения. Это позволяет эффективно обрабатывать большое количество запросов, обеспечивая быстродействие и надежность системы.

Тестирование разработанного API проводилось с помощью «pytest» в сочетании с «httpx» для создания тестов. Также использовалось специализированное программное обеспечение «Postman», который позволяет формировать запросы и отправлять их на сервер приложения.

FastAPI автоматически генерирует документацию с использованием «Swagger UI», доступную по URL «/docs». Данная документация может упростить работу разработчиков в будущем.

#### **4.2.2 Разработка части по работе с модулем обработки камер**

Главной задачей части по работе с модулем обработки камер является прием анонсов, благодаря которым можно узнать какие камеры в данный момент активны в системе, а также получение предупреждений о том, что где-то в кадре было замечено появление дыма, пламени или курильщика. Вся полученная информация должна записываться в базу данных. Также эта часть должна запустить часть работы с фронтендом, которая будет генерировать и отправлять оповещения пользователям.

Модуль обработки камер присыпает на бэкенд JSON вида, пример которого представлен на рисунке 42.

```
{
  "id": "camunit1",
  "host": "10.0.0.65",
  "cameras": [
    {
      "id": "camera1",
      "stream_url": "rtsp://user:pass@10.0.0.30:554/ISAPI/Streaming/channels/201",
      "ffmpeg_options": "-c copy -an",
      "output": "/stream1",
      "max_storage": 604800,
      "capture_interval": 300,
      "pack_size": 30
    },
    {
      "id": "camera2",
      "stream_url": "rtsp://user:pass@10.0.0.30:554/ISAPI/Streaming/channels/301",
      "ffmpeg_options": "-c copy -an",
      "output": "/stream2",
      "max_storage": 604800,
      "capture_interval": 300,
      "pack_size": 30
    }
  ]
}
```

Рисунок 42 – Пример входящего JSON с анонсом

Для получения этого JSON на стороне бэкенда необходим специальный эндпоинт для приема данных. Код этого эндпоинта представлен на рисунке 43.

```
# Эндпоинт для получения данных
@app.post("/announce/")
async def announce(config: ConfigSchema, db: Session = Depends(get_db)):
    for camera_data in config.cameras:
        camera = Camera(
            uid=camera_data.id,
            ffmpeg_options=camera_data.ffmpeg_options,
            output=camera_data.output,
            max_storage=camera_data.max_storage,
            capture_interval=camera_data.capture_interval,
            pack_size=camera_data.pack_size
        )
        db.add(camera)
    db.commit()
    return {"status": "Configuration updated successfully"}
```

Рисунок 43 – Эндпоинт для приема данных

Сразу после получения данных из JSON они записываются в базу данных через созданную ранее модель. Далее эти данные будут использоваться на стороне фронтенда. В случае, если анонсы перестанут поступать, бэкенд отметит узел, как упавший и отправит оповещение на фронтенд.

Помимо получения анонсов, также существует вариант получения предупреждения о появлении в кадре пламени, дыма или курильщика. На бэкенд придет JSON вида, представленного на рисунке 44.

```
{
  "unit_id": "camunit1",
  "camera_id": "camunit1",
  "type": "flame",
  "timestamp": 1719400050,
  "gridcell": []
}
```

Рисунок 44 – Пример входящего JSON с предупреждением

Принцип получения и работы с данным JSON схож с предыдущим. Далее запуститься специальная функция, которая начнет генерацию и отправку оповещения на фронтенд.

#### 4.3 Тестирование системы

После того, как бэкенд и все остальные части системы были разработаны и интегрированы между собой, необходимо было протестировать эту систему и проверить как будет работать распознавание дыма.

На главной страницы интерфейса изначально представлен список текущих камер с превью в виде последнего кадра, который обновляется раз в 30 секунд. Пример страницы представлен на рисунке 45.

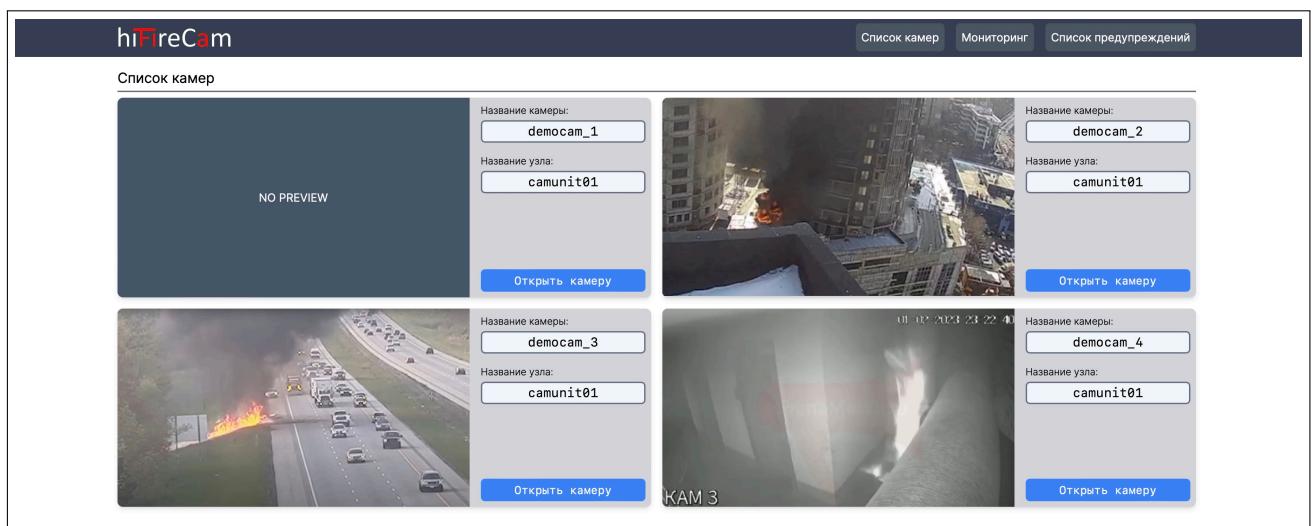


Рисунок 45 – Главная страница системы

Далее откроем камеру «democam\_3». Так как это тестовый эксперимент, здесь будет использоваться заранее записанное видео из датасета. При открытии камеры можно увидеть ее страницу, пример представлен на рисунке 46.

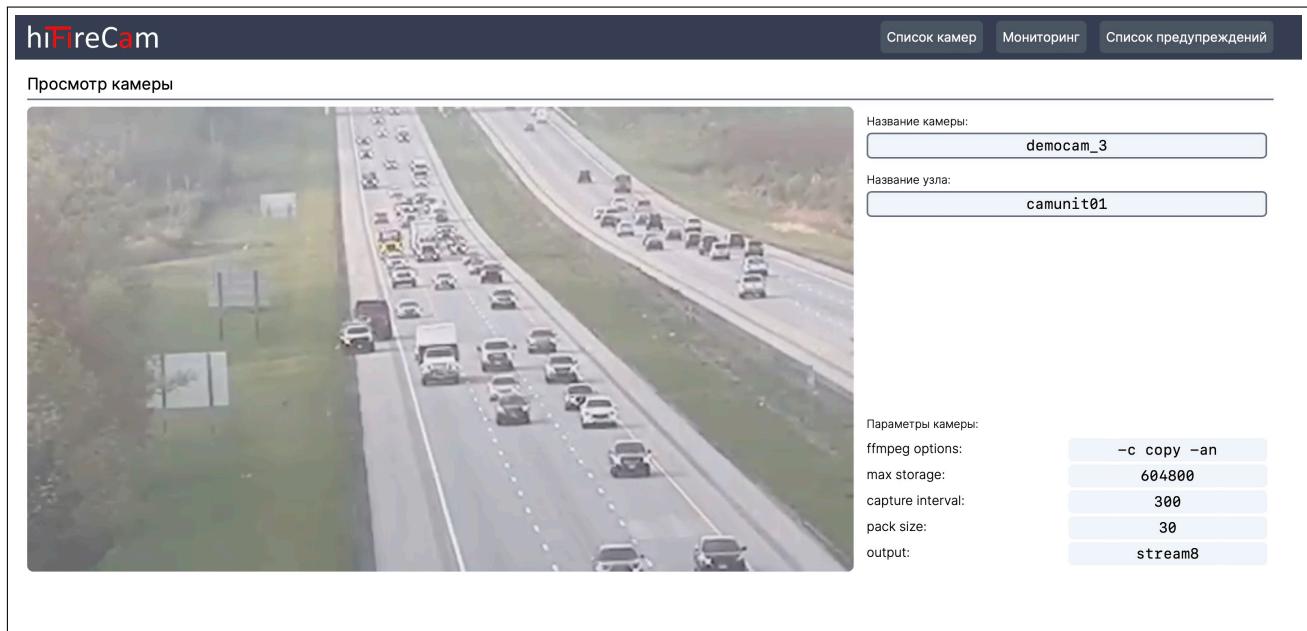


Рисунок 46 – Страница камеры

На странице можно увидеть видео с камеры и ее различные параметры. Данная страница служит для просмотра камеры, проверки ее параметров для корректности настройки. Мониторинг происходит на отдельной специальной странице, представленной на рисунке 47.

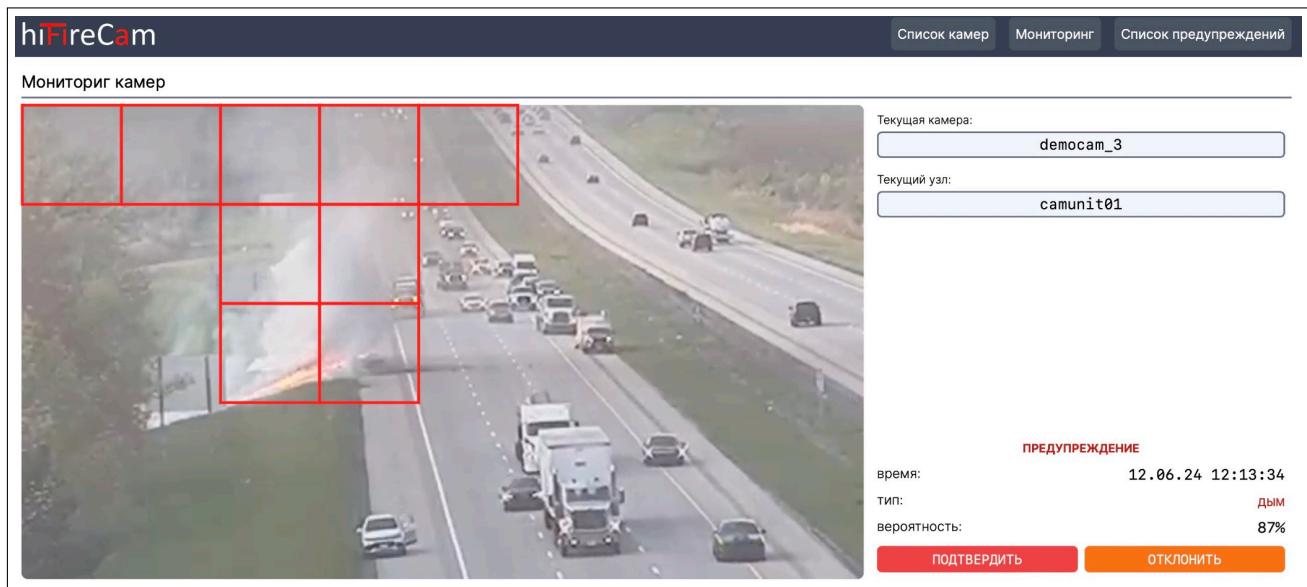


Рисунок 47 – Страница мониторинга камеры

На данной странице можно наблюдать видеокамеры и несколько их параметров. В случае, если в кадре детектируется дым, выводится предупреждение. Также на кадре можно

заметить ячейки с зонами, в которых предположительно имеется дым. В этом примере система предполагает наличие дыма с вероятностью 87 %, и эта оценка оказывается правильной. Тестирование показало, что система корректно определяет наличие дыма.

Проведенные тесты подтвердили высокую точность и надёжность системы в обнаружении дыма. Разработанный бэкенд и алгоритмы распознавания продемонстрировали способность эффективно идентифицировать дым, минимизируя ложные срабатывания. Это делает систему пригодной для использования в реальных условиях, обеспечивая своевременное обнаружение и предупреждение пожароопасных ситуаций.

На данный момент система представлена в виде прототипа и уже демонстрирует значительные успехи в распознавании дыма. Прототип успешно прошёл тестирование и подтвердил свою работоспособность и эффективность. В дальнейшем планируется продолжить работу над улучшением системы, включающим повышение точности детектирования, расширение функциональности и интеграцию дополнительных модулей для обеспечения более комплексного мониторинга и реагирования на пожароопасные ситуации.

Успешное тестирование системы открывает перспективы для её дальнейшего совершенствования и масштабирования. Внедрение более сложных алгоритмов и расширение функциональности могут ещё больше повысить её эффективность. В результате система будет способна обеспечить более высокий уровень безопасности и оперативного реагирования на угрозы пожара, что особенно важно для крупных объектов и территорий с повышенным риском возгораний.

## Заключение

Для достижения поставленной цели, в ходе исследования был проведен анализ предметной области, включающий изучение проблем обнаружения пожаров и методов их регистрации. Для решения задачи обнаружения дыма был выбран подход с использованием видеонаблюдения и машинного обучения.

Для обучения моделей был подготовлен и аннотирован датасет, включающий изображения и видеозаписи с различными типами аэрозолей, что позволило минимизировать количество ложных срабатываний. Были обучены классификатор и детектор на базе YOLO, а также собственная архитектура нейронной сети, результаты которой сопоставлены с результатами YOLO. В работу была выбрана нейронная сеть с собственной архитектурой, а YOLO использовалась как автоматизированный инструмент, помогающий в аннотации изображений.

Была поставлена и опровергнута гипотеза о том, что, если обучить модель различать другие классы аэрозолей как собственные классы, то можно будет сократить количество ложных срабатываний на них. Эксперименты показали, что это не всегда так.

Итоговая система распознавания пожароопасных ситуаций показала свою эффективность и надежность в эксплуатации, успешно прошла тестирование и подтвердила возможность использования в реальных условиях. Разработанный бэкенд обеспечивает интеграцию и взаимодействие всех компонентов системы, что позволяет автоматизировать процесс обнаружения пожаров и уменьшить влияние человеческого фактора.