СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Промышленная безопасность — состояние защищённости жизненно важных интересов личности и общества от аварий на опасных производственных объектах. Основная цель промышленной безопасности - предотвращение и/или минимизация последствий аварий на опасных производственных объектах.

На производствах достаточно часто нарушается техника безопасности, вследствие чего происходят травмы и летальные исходы. По данным статистики общее число происшествий за год снижается, но очень медленными темпами. Это все также приносит большой ущерб предприятиям. Чтобы решить эту проблему, было решено разработать такую архитектуру на базе глубоких нейронных сетей, которая в реальном времени детектирует наличие и правильность применения средства индивидуальной защиты.

Разработка данного модуля поможет не только снизить процент травм и летальных исходов, но и автоматизировать контроль за соблюдением техники безопасности.

Основная цель выпускной квалификационной работы – разработать модуль проверки наличия и правильности применения средств индивидуальной защиты. для интеллектуальной системы.

Для достижения данной цели были поставлены следующие задачи:

1. Проведение анализа существующих отечественных и зарубежных аналогов.
2. Выбор технологии, языка и среды программирования для разработки.
3. Сбор и разметка выборочных данных для обучения.
4. Проектирование моделей нейронных сетей.
5. Написание пользовательского интерфейса.
6. Написание API (Application Programming Interface).
7. Тестирование.

1 АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ

1.1 Постановка задачи

Компания ООО «Малленом Системс» — ведущая российская компания в области разработки и внедрения систем видеоаналитики и промышленного контроля на основе технологий машинного зрения и искусственного интеллекта (машинное обучение, нейронные сети глубокого обучения) [1].

Основные направления деятельности компании «Малленом Системс»:

1. Визуальный контроль продукции на производстве.
2. Прослеживание продукции в алкогольной, фармацевтической и других отраслях промышленности.
3. Видеоконтроль и учет автомобильного и железнодорожного транспорта.
4. Автоматизация взвешивания автомобильного и железнодорожного транспорта.
5. Транспортное моделирование и управление дорожным движением.
6. Поставка и внедрение продукции Cognex (умные камеры, оптические датчики, сканеры кодов и маркировок) в РФ и СНГ.

Для освоения нового направления деятельности компанией «Малленом Системс» было предложено разработать модуль проверки наличия и правильности применения средств индивидуальной защиты для интеллектуальной системы.

Описание задачи.

Разработать алгоритмы и программное обеспечение для обнаружения людей на видеоизображениях, проверки использования средств индивидуальной защиты и проверки правильности применения средств индивидуальной защиты.

Основные требования к функциональным характеристикам:

1. Анализ видеопотока в режиме реального времени.

2. Реализация событий нарушения и устранения нарушений применения средств индивидуальной защиты.

3. Реализация API модуля.

4. Для демонстрации работы алгоритма и оценки его эффективности разработать приложение на языке C#.

5. Предусмотреть в приложении ручную загрузку видеофайла.

1.2 Анализ существующих аналогов

Первым делом перед началом разработки был проведен анализ существующих отечественных и зарубежных аналогов данной системы. Были выявлены плюсы и минусы каждого и сделан вывод.

Среди отечественных аналогов стоит выделить следующие системы: Детектор касок SecurOS Helmet Detector компании ISS, Детектор отсутствия касок компании Macroscop, Нейросетевой модуль TRASSIR Hardhat Detector компании DSSL и Система промышленной видеоаналитики CenterVision компании Центр2М.

Среди зарубежных стоит выделить: Smart Hard Hat Improves Industrial Safety компании GuardHat и Платформа IoT для промышленной безопасности компании NNTC.

Детектор касок SecurOS Helmet Detector — специализированное решение компании ISS, обеспечивающее детекцию людей, перемещающихся в зоне контроля без каски [2]. Комплексные решения ISS применяются на промышленных предприятиях для обеспечения охраны труда. Алгоритм работы SecurOS Helmet Detector представлен на рисунке 1.1.

Рисунок 1.1 - Алгоритм работы SecurOS Helmet Detector

Основные точки внедрения:

1) логистические терминалы;
2) морские порты;
3) промышленные предприятия;
4) строительные площадки.

Детектор отсутствия касок — система, позволяющая снизить уровень производственного травматизма и смертности работников предприятий из-за нарушения ими требований охраны труда.

Детектор отсутствия касок анализирует получаемое с камеры видеонаблюдения в реальном времени изображение. При этом он учитывает наличие прочих предметов, например, закреплённых на каске очков и шумоподавляющих наушников [3]. Обрабатывая получаемую информацию, детектор определяет отсутствие каски на голове сотрудника, для создания доказательной базы сохраняет отдельный кадр, фиксирующий нарушения, и отправляет автоматическое уведомление уполномоченному лицу удобным для него способом: на мобильный телефон, в мессенджер или на электронную почту [3].

Основные точки внедрения:

1) критическая инфраструктура (Тепловые, атомные электростанции, Гидроэлектростанции);

2) промышленность (Заводы, фабрики, Строительство, Добыча полезных ископаемых);

3) логистические центры, склады (Морские и речные порты, Логистические центры, склады).

Нейросетевой модуль TRASSIR Hardhat Detector помогает контролировать соблюдение техники безопасности. В случае появления в рабочей зоне сотрудника без защитной каски интеллектуальный модуль отправляет уведомление в режиме реального времени и сохраняет кадр для доказательной базы [4].

Модуль Hardhat Detector решает задачи:

1) снижение травматизма на производстве;

2) повышение дисциплины сотрудников;

3) снижение расходов на оплату больничных;

4) снижение репутационных рисков компании.

Для корректной работы модуля необходимы соблюдать правила по настройке детектора и размещению камер:

1. Настройка. При обработке видео исходный кадр сжимается, поэтому качество изображения снижается.

2. Дальность детекции. В зданиях с высокими потолками не рекомендуется размещать камеру прямо под потолком, в комнатах с большой площадью лучше устанавливать несколько камер в разных частях помещения, иначе люди в кадре будут настолько маленькими, что детектор не распознает каску.

3. Освещение. Неравномерное, слишком слабое или чересчур сильное освещение снижает качество распознавания.

4. Ракурс. Модуль хорошо распознает каски практически с любого ракурса, но установка под углом больше 75º относительно потолка не рекомендуется.

5. Требования к каскам. Белый и оранжевый цвета распознаются лучше остальных оттенков, однотонные каски — лучше касок с надписями и наклейками.

Модуль Hardhat Detector работает только на видеорегистраторах TRASSIR серии NeuroStation.

Также для работы необходимо:

1) модуль TRASSIR Hardhat Detector;
2) регистратор TRASSIR серии NeuroStation;
3) VMS TRASSIR;
4) видеокамеры любого производителя и лицензии для подключения.

Система промышленной видеоаналитики CenterVision использует нейронные сети для распознавания объектов. Отслеживает производственный процесс в режиме реального времени, передает автоматическое оповещение в ситуационный центр, готовит отчёты по всем фактам нарушения правил безопасности [5]. Технология работы системы промышленной видеоаналитики CenterVision представлена на рисунке 1.2.

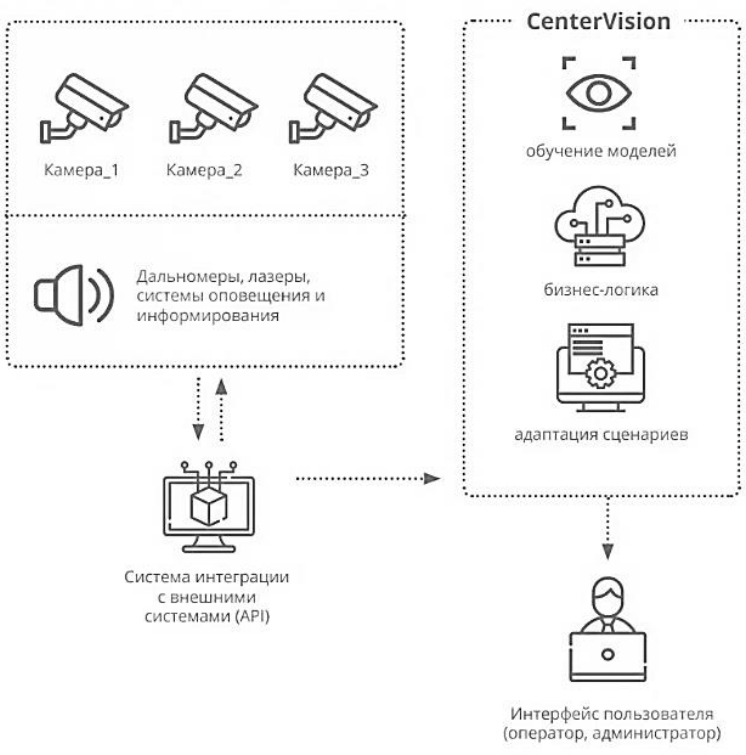


Рисунок 1.2 - Технология работы Системы промышленной видеоаналитики CenterVision

Smart Hard Hat Improves Industrial Safety — интегрированная система безопасности, созданная компанией GuardHat, которая помогает обнаруживать, предупреждать и предотвращать опасные производственные инциденты, связанные с работой [6].

Решения GuardHat предназначены для работы в нескольких областях:

1) мобильный рабочий;
2) динамично строящиеся объекты;
3) производственные цеха.

GuardHat представляет индивидуальные решения, в зависимости от потребности клиента и специфики отрасли, работает с компаниями в таких отраслях как:

1) химическая промышленность;
2) строительство;
3) металлургия;
4) транспорт;
5) бумажная, нефтяная и пластмассовая промышленности и другое.

Платформа IoT для промышленной безопасности, разработанная NNTC, классифицирует данные, поступающие из любой системы видеоаналитики и локального позиционирования, промышленного и носимого IoT, контроля доступа и управления [7]. Технология работы платформы IoT для промышленной безопасности представлена на рисунке 1.3.

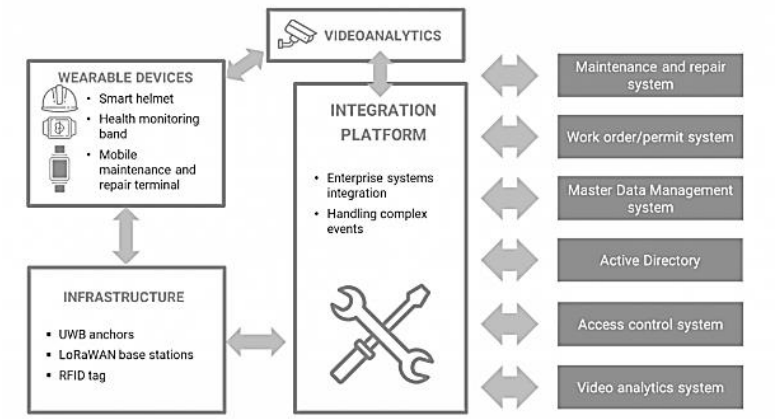


Рисунок 1.3 - Технология работы Платформы IoT для промышленной безопасности

Все проанализированные аналоги можно наглядно представить в виде таблицы и выделить их преимущества и недостатки по нескольким основным критериям (Таблица 1.1).

Таблица 1.1 - Анализ отечественных и зарубежных аналогов

	SecurOS Helmet Detector	Детектор отсутствия касок	TRASSIR Hardhat Detector	Center Vision	Smart Hard Hat	Платформа IoT
Определение положения человека в опасной зоне	+	-	-	-	+	-
Не требует дополнительных настроек/оборудования	+	+	-	+	+	+
Обнаружение больше одного элемента средств индивидуальной защиты	-	+	-	+	-	-
Работа в режиме реального времени	+	+	-	+	+	+
Формирование сообщения об нарушении (с наличием видеофрагмента или кадра)	+	+	+	+	+	+
Возможность подключения больше одного потока (камеры)	+	+	+	+	-	-
Невысокая цена	-	+	+	-	-	-

Вывод: аналогов данной системы не мало, но у каждой есть свои недостатки. В большинстве случаев такие программы пишутся под конкретное место (будь то цех или строительная площадка). Разрабатываемая система нацелена на универсальность и возможность работы при любых условия и освещении.

1.3 Выбор средств разработки

В ходе разработки потребуются:

1) работа с нейронными сетями;
2) разметка данных и ее обучение;
3) написание пользовательского интерфейса;
4) написание API.

Нейронные сети используются для решения сложных задач, которые требуют аналитических вычислений подобных тем, что делает человеческий мозг. Самыми распространенными применениями нейронных сетей является:

1. Классификация — распределение данных по параметрам. Например, на вход дается набор людей и нужно решить, кому из них давать кредит, а кому нет. Эту работу может сделать нейронная сеть, анализируя такую информацию как: возраст, платежеспособность, кредитная история и т. д.

2. Предсказание — возможность предсказывать следующий шаг. Например, рост или падение акций, основываясь на ситуации на фондовом рынке.

3. Распознавание — в настоящее время самое широкое применение нейронных сетей. Используется в Google, когда вы ищете фото или в камерах телефонов, когда оно определяет положение вашего лица и выделяет его и многое другое.

Основная задача нейронной сети в данной работе — это распознавание и классификация объектов. В связи с чем была выбрана архитектура сверточной нейронной сети. На данный момент сверточная нейронная сеть и ее модификации считаются лучшими по точности и скорости алгоритмами нахождения объектов.

В настоящее время для написания нейронных сетей больше всего подходят такие языки программирования, как Python и R.

R — интерпретируемый язык программирования, основным способом работы с которым является командный интерпретатор. Главным недостатком

данного языка является то, что он сложен в обучении и достаточно медленный [8].

Python — высокоуровневый язык программирования общего назначения, ориентированный на повышение производительности разработчика и читаемости кода. Синтаксис ядра Python минималистичен, однако стандартная библиотека включает большой объём полезных функций. Основное преимущество языка Python в том, что он является универсальным и многоцелевым языком программирования [9]. Вследствие чего для работы с нейронными сетями был выбран язык программирования Python.

Для написания интерфейса была выбрана Microsoft Visual Studio — линейка продуктов компании Microsoft, включающих интегрированную среду разработки программного обеспечения и ряд других инструментальных средств [10]. Данные продукты позволяют разрабатывать как консольные приложения, так и приложения с графическим интерфейсом, в том числе с поддержкой технологии Windows Forms, а также веб-сайты, веб-приложения и веб-службы [10]. А точнее Visual Studio Community — полнофункциональная, расширяемая и бесплатная интегрированная среда разработки для создания современных приложений Android, iOS и Windows, а также веб-приложений и облачных служб. Любой индивидуальный разработчик может создавать бесплатные или платные приложения с помощью Visual Studio Community [10].

В качестве языка был выбран соответственно объектно-ориентированный язык программирования C#.

Для разработки API были выбраны технология ASP.NET Core Web API: ASP.NET Core представляет собой переписанную и расширенную версию фреймворка ASP.NET, которая работает на платформе .NET Core. ASP.NET Core Web API предоставляет средства для создания легковесных и высокопроизводительных API. Он также поддерживает различные форматы данных и предоставляет гибкую конфигурацию и маршрутизацию.

2 ОБЗОР МОДЕЛЕЙ НЕЙРОННЫХ СЕТЕЙ

Нейронная сеть - математическая модель, а также ее программные и аппаратные реализации, предназначенные для поиска неких полезных представлений данных в пространстве представлений.

Нейронные сети состоят из следующих компонентов:

- данные – входные и выходные (иногда только входные);
- модель нейронной сети, состоящая из слоев;
- весовые параметры - те параметры, которые непосредственно изменяются в процессе обучения;
- функция потерь - определяет, каким образом будут обновляться весовые параметры сети для достижения желаемых результатов;
- оптимизатор - непосредственно отвечает за обновление весовых коэффициентов сети.

Процесс обновления весовых параметров нейронной сети, при котором ошибка сети, характеризующаяся функцией потерь, стремится к минимуму, называется обучением сети

Полносвязные нейронные сети могут выступать в качестве средства для решения задачи классификации изображений, однако на данный момент доминирующей архитектурой для решения этой задачи являются сверточные нейронные сети (CNN), которые и будут использоваться в данной работе.

Классическая нейронная сеть для классификации изображении состоит и следующих компонентов:

- сверточный слой;
- пулинговый слой (или слой субдискретизации);
- слой дропаута (или прореживания);
- блок линейных слоев.

Сверточный слой реализует операцию свертки. Она состоит в следующем: на вход слою подается изображение (исходное или уже обработанное), каждый фрагмент каждого канала изображения умножается на ядро сверткии

поэлементно, результаты по фрагменту суммируются и определяются в позицию, аналогичную таковой у исходного фрагмента. Таким образом формируется карта признаков.

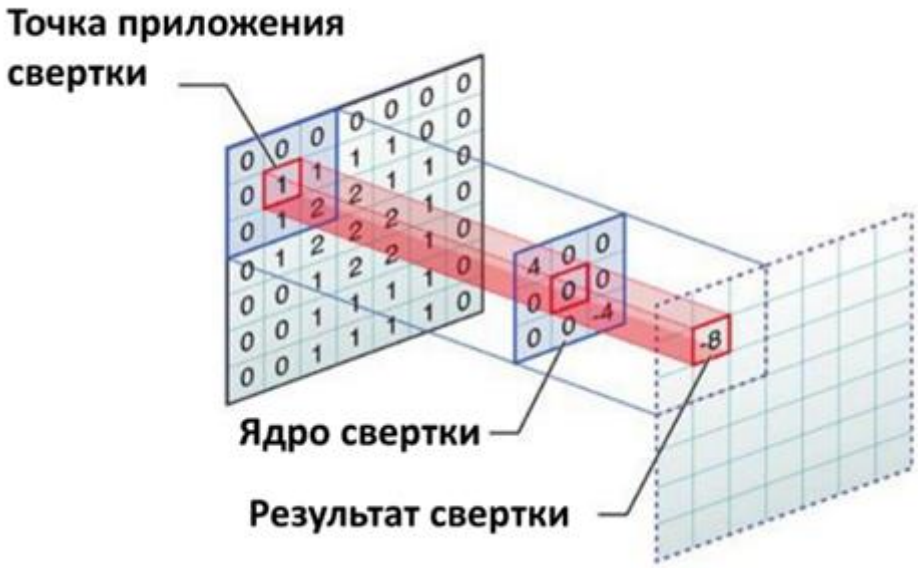


Рисунок 2.1 – Формирование карты признаков сверточным слоем

Пулинговый слой (или субдискретизации) уплотняет поступающие на него карты признаков, при этом группы пикселей сжимаются путем нелинейного преобразования, обычно таким преобразованием является максимизация.

Рисунок 2.2 – Работа пулингового слоя

Слой дропаута обнуляет в процессе обучения некоторый случайный набор параметров сети. Применение этого слоя положительно влияет на склонность сверточной сети к обобщению данных.

Рассмотрим сверточные нейронные сети и выберем наиболее подходящие для задач данной работы.

2.1 Выбор нейросетевой модели для обнаружения человека

Faster R-CNN (Region-based Convolutional Neural Network) - архитектур нейронных сетей для обнаружения объектов. Она использует двухэтапный подход, включающий обнаружение предлагаемых областей и классификацию объектов внутри этих областей. Faster R-CNN обеспечивает высокую точность обнаружения.

SSD (Single Shot MultiBox Detector).

SSD предлагает более быстрый одноэтапный подход, обнаруживая объекты на разных масштабах и различных уровнях свертки. SSD хорошо подходит для задач реального времени.

Основные преимущества нейронной сети SSD:

1. Архитектура SSD позволяет детектировать объекты в реальном времени.
2. Качество работы близко к Faster R-CNN.
3. Используется большое количество default box-ов, покрывающих входное изображение на разных масштабах.
4. На этапе Inference архитектура SSD осуществляет детектирование 7308 объектов, большая часть из которых впоследствии фильтруется.

YOLOv4 (You Only Look Once) — очень популярная архитектура сверточных нейронных сетей, которая используется для распознавания множественных объектов на изображении. Главная особенность этой архитектуры по сравнению с другими состоит в том, что большинство систем применяют сверточные нейронные сети несколько раз к разным регионам изображения, в YOLO CNN применяется один раз ко всему изображению сразу. Сеть делит изображение на своеобразную сетку и предсказывает

ограничивающие рамки и вероятности того, что там есть искомый объект для каждого участка.

Преимуществом данного подхода является то, что сеть смотрит на все изображение сразу и учитывает контекст при детектировании и распознавании объекта. Используя эту систему, нужно всего один раз пропустить изображение через нейронную сеть, чтобы предсказать объекты и их расположение. Это обеспечивает YOLO высокую скорость по сравнению с другими методами обнаружения объектов.

После проведенного анализа была выбрана архитектура модели нейронной сети YOLOv4 по следующим критериям:

1. Данная модель лучше детектирует небольшие объекты по сравнению с Faster R-CNN и SDD.
2. Имеет больше FPS (кадров в секунду) по сравнению с Faster R-CNN и проще в использовании по сравнению с SSD.
3. Более эффективная в режиме реального времени по сравнению с Faster R-CNN.

2.2 Выбор нейросетевых моделей для обнаружения средств индивидуальной защиты

Для задач классификации рассмотрим такие нейронные сети, как MixNets и EfficientNets.

MixNets и EfficientNets - это два разных подхода к созданию нейросетей, которые могут быть эффективны в различных задачах машинного обучения.

MixNets имеет следующие преимущества по сравнению с EfficientNets:

- меньшее число параметров (MixNets использует меньше параметров, чем EfficientNets, что делает ее более легкой и быстрой в работе);
- повышенная точность при обучении на маленьких наборах данных (MixNets показала лучшие результаты в классификации изображений на меньших наборах данных, т.к. использует оптимизированные функции свертки);

- более легкая архитектура (MixNets использует свертки с меньшим числом фильтров, что позволяет ей работать с менее ресурсной аппаратурой).

Исходя из данных преимуществ будем использовать MixNets:

1. MixNet_S – для классификации правильности применения и обнаружения красной каски;

2. MixNet_L – для классификации СИЗ и положения человека (лицом или спиной к камере).

MixConvs - концепция объединения ядер нескольких размеров в один слой, которая является основным строительным блоком для MixNets. На рисунке 2.3 приведена архитектура MixNet-M, показывающая различные группы ядер.

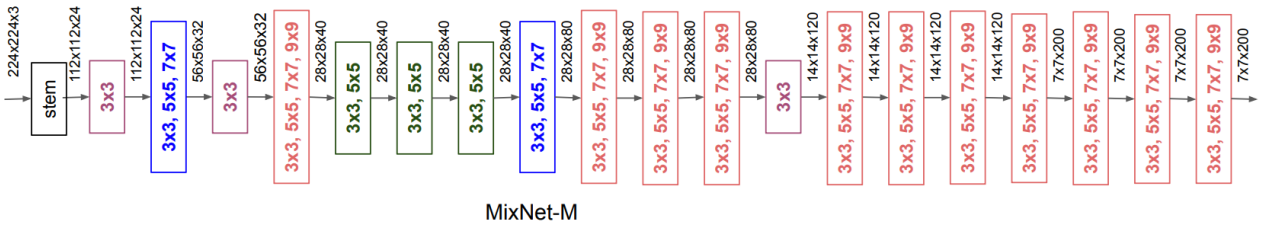


Рисунок 2.3 – Архитектура сети MixNet-M

Вначале используются меньшие ядра, затем более крупные ядра постепенно интегрируются по мере прохождения данных через слои. MixNet-L - это просто MixNet-M с коэффициентом глубины 1,3.

Коэффициент глубины является параметром, который используется для сокращения числа параметров и операций в сверточных слоях. Это позволяет ускорить вычисления и снизить требования к памяти.

Установка коэффициента глубины в 1 соответствует применению стандартных сверточных операций. При увеличении коэффициента глубины увеличивается количество параметров и вычислений, что может улучшить точность, но также сделать нейронную сеть более трудоемкой.

3 РЕАЛИЗАЦИЯ

Модуль, рассматриваемый в данной работе, разработан для системы видеоаналитики - EYECONT компании Mallenom Systems.

EYECONT - гибкая система видеоаналитики на основе машинного зрения в режиме реального времени анализирует видео с IP-камер, детектирует на нем людей, отслеживает их поведение и перемещение, а также контролирует попадание человека в определенные (в том числе, опасные) зоны. Дополнительным функционалом является выявление опасных событий, таких как задымление, возгорание, оставленные предметы и др [11].

EYECONT - унифицированное и стандартизированное API, предоставляемое для интеграторов. Основано на микросервисном клиент-серверном подходе, который позволяет распределить функциональность системы на отдельные сервисы. Решение является кроссплатформенным. Также использует событийную асинхронную передачу уведомлений от модулей.

EYECONT API позволяет:

1. работать с контекстами распознавания и обработки кадров: создавать, удалять и настраивать параметры;
3. добавлять и настраивать зоны контроля, линии подсчета;
4. получать информацию обо всех отслеживаемых объектах.

Для описания, документации и визуализации в EYECONT API интегрирован Swagger.

Swagger — это набор инструментов, который позволяет автоматически описывать API на основе его кода.

В разрабатываемом модуле необходимо предусмотреть обнаружения человека и набора средств индивидуально защиты на кадре. При нарушении (отсутствии одного или более СИЗ) формировать событие с данным нарушением.

Набор детектируемых средств индивидуальной защиты:

- каска;

- комбинезон;
- диэлектрические перчатки;
- термостойки комбинезон;
- защитный лицевой щиток.

Нарушения правильности применения детектируемых средств индивидуальной защиты:

- закатанные рукава;
- не застегнутый комбинезон;
- сапоги поверх брюк;
- перчатки под комбинезоном;
- края перчаток загнуты.

Алгоритм определения наличия СИЗ и правильности их применения:

- если обнаружена красная каска проверяются антистатические перчатки, термостойки комбинезон, защитный лицевой щиток;
- если обнаружена обычная каска, проверяется обычный комбинезон;
- правильность применения СИЗ, добавляется в событие только если есть соответствующий СИЗ: простая или термостойкая спецодежда – комбинезон не застегнут, рукава закатаны, сапоги поверх брюк; диэлектрические перчатки - перчатки под спецодеждой, края перчаток загнуты.

3.1 Формирование выборочных данных

Выборочные данные для обучения нейронных сетей, были предоставлены компанией ООО «Малленом Системс».

Разметка выборочных данных осуществлялась с помощью программного обеспечения Computer Vision Annotation Tool (CVAT) – это инструмент с открытым исходным кодом для разметки цифровых изображений и видео. Основной его задачей является предоставление пользователю удобных и эффективных средств разметки наборов данных. CVAT – универсальный сервис, поддерживающий разные типы и форматы разметки. Для конечных

пользователей CVAT – это web-приложение, работающее в браузере. Он поддерживает разные сценарии работы и может быть использован как для персональной, так и для командной работы. Окно разметки представлено на рисунке 3.1.

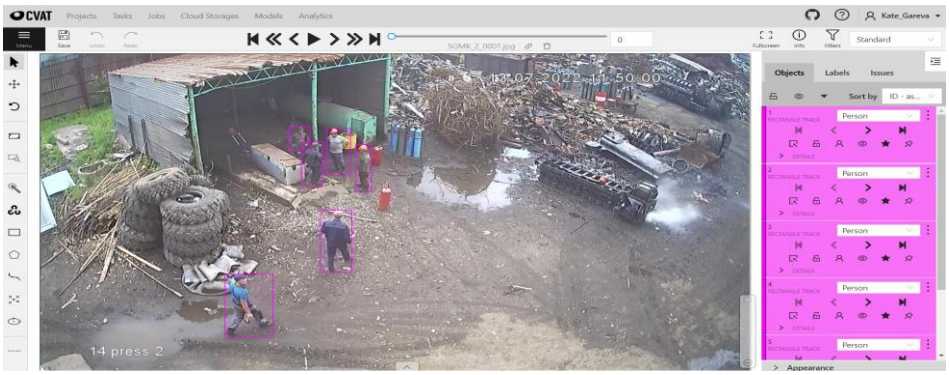


Рисунок 3.1 – Окно разметки

Разметка средств индивидуальной защиты.

False – когда СИЗ отсутствует на конкретном кадре (рисунок 3.2).

Рисунок 3.2 - Отсутствие СИЗ

True – когда СИЗ надет правильно и виден на конкретном кадре (рисунок 3.3).

Рисунок 3.3 - Наличие (спецодежда, обувь, перчатки)

Doubt – когда СИЗ не видно/перекрыт на конкретном кадре (рисунок 3).

Рисунок 3.4 - Не видно СИЗ (каска, маска, ботинки)

После экспорта каждого набора данных в формат CVAT for images 1.1 получен архив со структурой, представленной на рисунке 3.5.

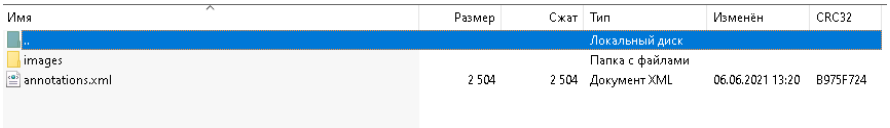
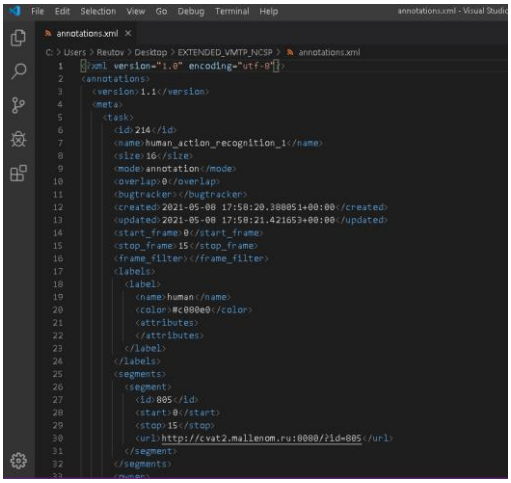


Рисунок 3.5 - Структура архива

В архиве имеются папка images, в которой находятся размеченные изображения и файл annotations.xml в котором хранятся координаты левой верхней и правой нижней точек ограничивающих прямоугольников и классы

объектов, к которому они принадлежат. Структура xml-файла представлена на рисунке 3.6.


File Edit Selection View Go Debug Terminal Help
annotations.xml - Visual Studio
annotations.xml
C: > Users > Reutov > Desktop > EXTENDED_VMTP_NICSP > annotations.xml
<?xml version="1.0" encoding="utf-8"?>
<annotations>
<version>1.1</version>
<meta>
<task>
<id>214</id>
<name>human_action_recognition_1</name>
<size>16</size>
<mode>annotation</mode>
<overlap>0</overlap>
<bugtracker></bugtracker>
<created>2021-05-08 17:58:20.380051+00:00</created>
<updated>2021-05-08 17:58:21.421653+00:00</updated>
<start_frame>0</start_frame>
<stop_frame>15</stop_frame>
<frame_filter></frame_filter>
<labels>
<label>
<name>human</name>
<color>#c080e0</color>
<attributes>
</attributes>
</label>
</labels>
<segments>

<id>805</id>
<start>0</start>
<stop>15</stop>
<url>http://cvat2.mallenom.ru:8080/?id=805</url>

</segments>

Рисунок 3.6 - Структура xml-файла разметки

Для обучения нейронной сети для обнаружения человека было собрано 10320 кадров, для классификатора красной каски - 9187 кадров, классификатор СИЗ - 41100 кадров, классификатор правильности применения СИЗ - 16239. 70% выборки были взяты в качестве обучающих данных, а 30% – в качестве тестовых.

3.2 Разработка модуля средств индивидуальной защиты в нейросетевом сервисе

Нейросетвеой сервис отвечает за загрузку нейронных сетей, подачу изображения на обработку и получение результатов обработки. Данный сервис предусматривает загрузку установленных в конфигурации сетей модуля, возможность выбора загружаемого набора сетей, их смены.

В результате разработки модуля средств индивидуальной защиты в нейросетевом сервисе был реализован пайплайн.

Пайплайн (pipeline) - это концепция организации последовательности операций или этапов обработки данных, которые выполняются в определенном порядке.

Состав классов для работы с нейросетевой обработкой представлен на рисунке 3.7.

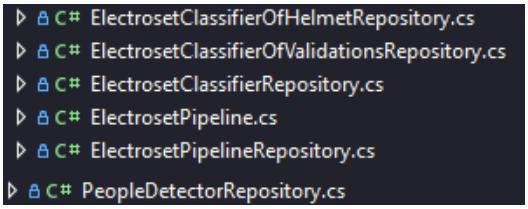


Рисунок 3.7 – Классы нейросетевого сервиса

ElectrosetPipelineRepository – отвечает за загрузку пайплайна с установленным набором сетей, также хранит все загруженные пайплайны, возвращает и обновляет их по требованию.

ElectrosetPipeline – сам пайплайн разработанного модуля. Содержит метод, отправляющий изображение на обработку, который возвращает результаты обработки детектором и классификаторами.

ElectrosetClassifierRepository, ElectrosetClassifierOfValidationsRepository, ElectrosetClassifierOfHelmetRepository – отвечают за загрузку классификаторов, установленных в загружаемом пайплайне.

PeopleDetectorRepository – отвечает за загрузку детектора людей, который установлен в загружаемом пайплайне.

На рисунке 3.8 представлены контракты данных модуля в нейросетевом сервисе.

Контракты данных (data contracts) - это соглашения или спецификации, определяющие формат и структуру данных, которые передаются между компонентами системы или между различными системами.

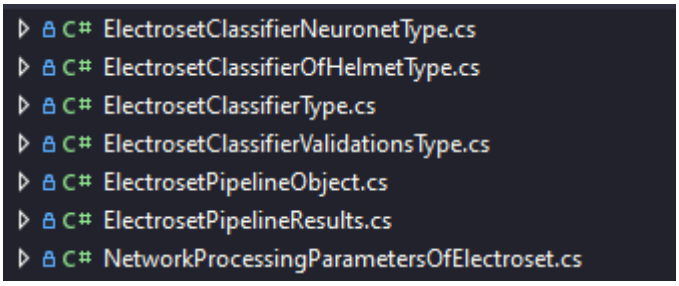


Рисунок 3.8 – Контракты данных нейросетевого сервиса

ElectrosetClassifierNeuronetType, ElectrosetClassifierOfHelmetType, ElectrosetClassifierType, ElectrosetClassifierValidationsType – классы перечислений с типами средств индивидуальной защиты.

NetworkProcessingParametersOfElectroset – параметры нейросетевой обработки.

ElectrosetPipelineResults, ElectrosetPipelineObject – содержат результаты распознования объектов на кадре.

Контроллеры для работы с данным модулем представлены на рисунке 3.9.

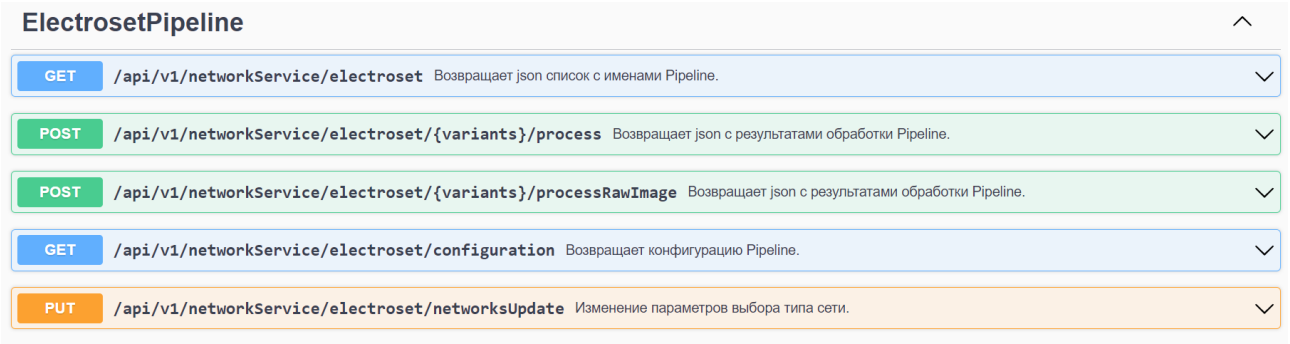


Рисунок 3.9 – Контроллеры нейросетевого сервиса

3.3 Разработка модуля средств индивидуальной защиты в сервисе ядра

Сервис ядра позволяет создавать и конфигурировать контексты обработки.

Контекст представляет собой сущность работы с видеопотоком. В контексте есть возможность настраивать параметры подключения к видеопотоку, выбирать пайплайны обработки, создавать, удалять, конфигурировать зоны контроля, выбирать детектируемые средства индивидуальной защиты, устанавливать параметры для нейросетевой обработки, такие как порог распознавания, история распознавания и др.

Состав классов для работы с ядром представлен на рисунке 3.10.

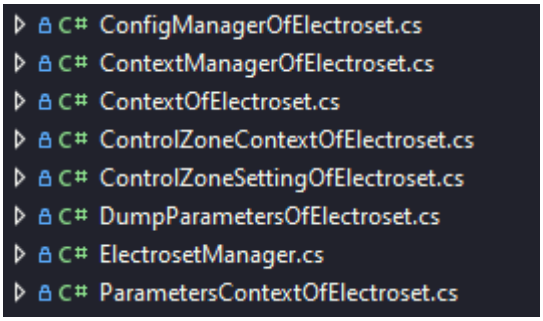


Рисунок 3.10 – Классы сервиса ядра

Для данного модуля были разработаны следующие классы:

1. управление контекстами, сохраненными в конфигурационный файл;

2. управление загруженными контекстами;
3. создание, удаление, настройка зон контроля;
4. управление сохранением дампов работы системы;
5. настройка загруженных контекстов.

Контроллеры для работы с данным модулем представлены на рисунке 3.11.

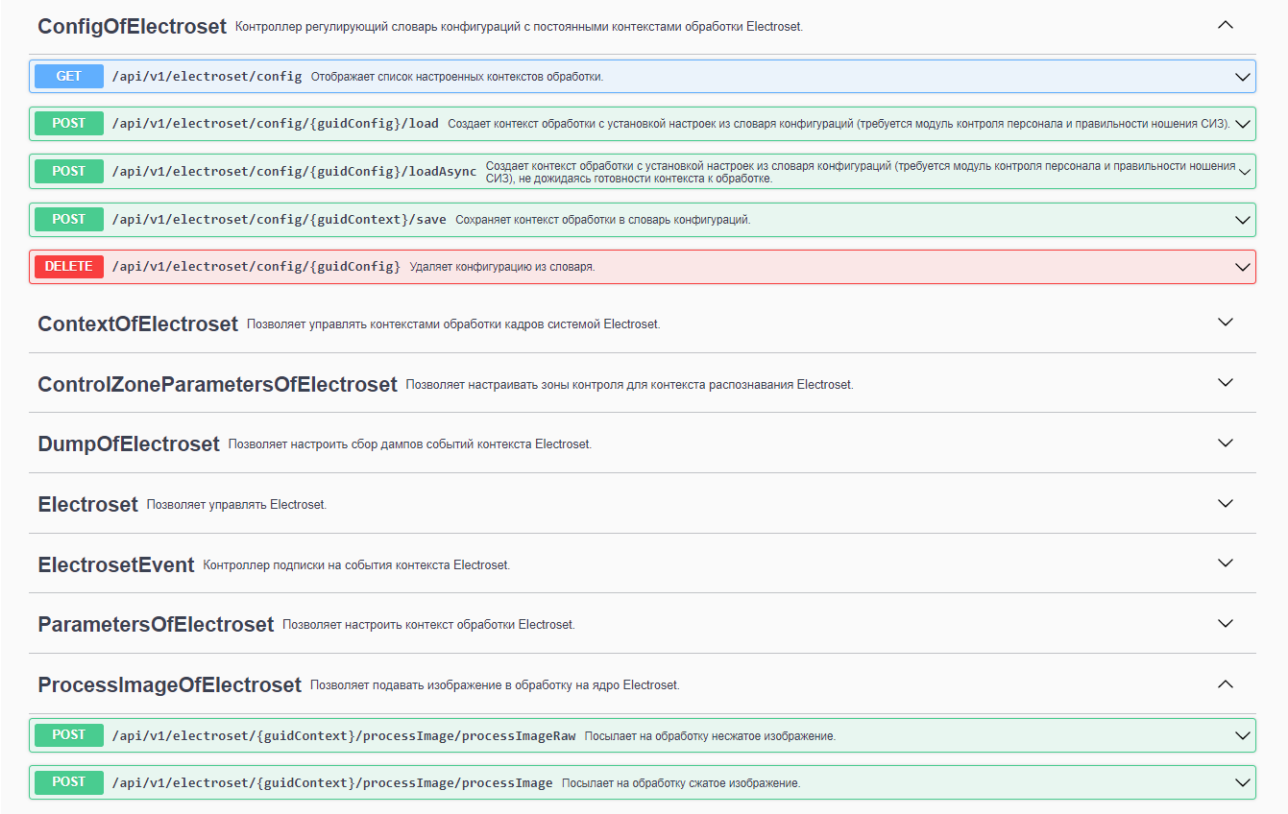


Рисунок 3.11 – Контроллеры сервиса ядра

3.4 Разработка клиента для взаимодействия с API модуля средств индивидуальной защиты

Для взаимодействия созданного API и отладочного интерфейса необходимо было написать клиент обращения к данному API.

Функции, которые выполняет клиента:

1. Формирование запросов: формирует запросы, указывая необходимые методы (GET, POST, PUT, DELETE) и URL-адреса конечных точек API. Добавляет параметры запроса, заголовки и тело запроса, в зависимости от требуемых данных.

2. Отправка запросов: отправляет сформированные запросы на сервер API. Устанавливает соединение с сервером и передает запрос с необходимыми данными.

3. Обработка ответов: обрабатывает полученные данные. Анализирует статусный код ответа для определения успешности запроса, чтения содержимого ответа.

3.5 Разработка отладочного интерфейса

Отладочный интерфейс необходим для тестирования и демонстрации работы модуля.

На рисунке 3.12 представлено главное окно отладочного интерфейса, здесь отображается запущенный видеопоток, на левой панели в таблицах отображаются результаты работы классификаторов обнаружения средств индивидуальной защиты. Также фиксируются события нарушений.

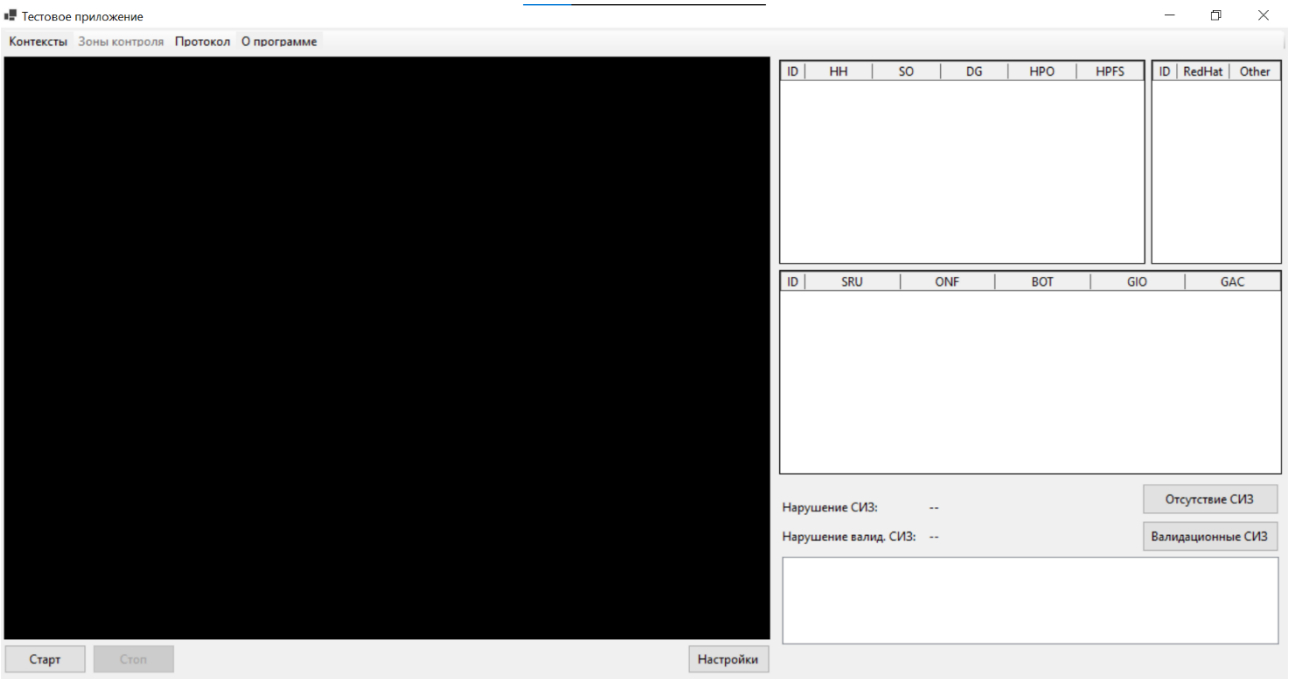


Рисунок 3.12 – Главное окно отладочного интерфейса

Окна отображения событий представлены на рисунках 3.13 и 3.14. При срабатывании какого-либо события, информация о нем и кадр отображаются в данных окнах.

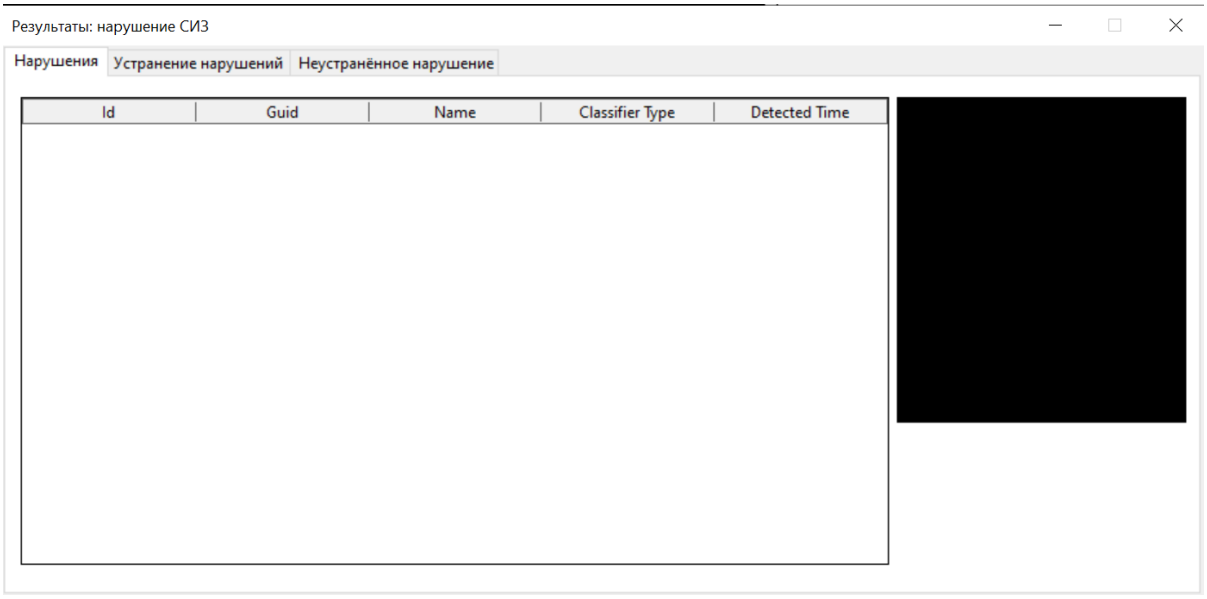


Рисунок 3.13 – Окно отображения событий

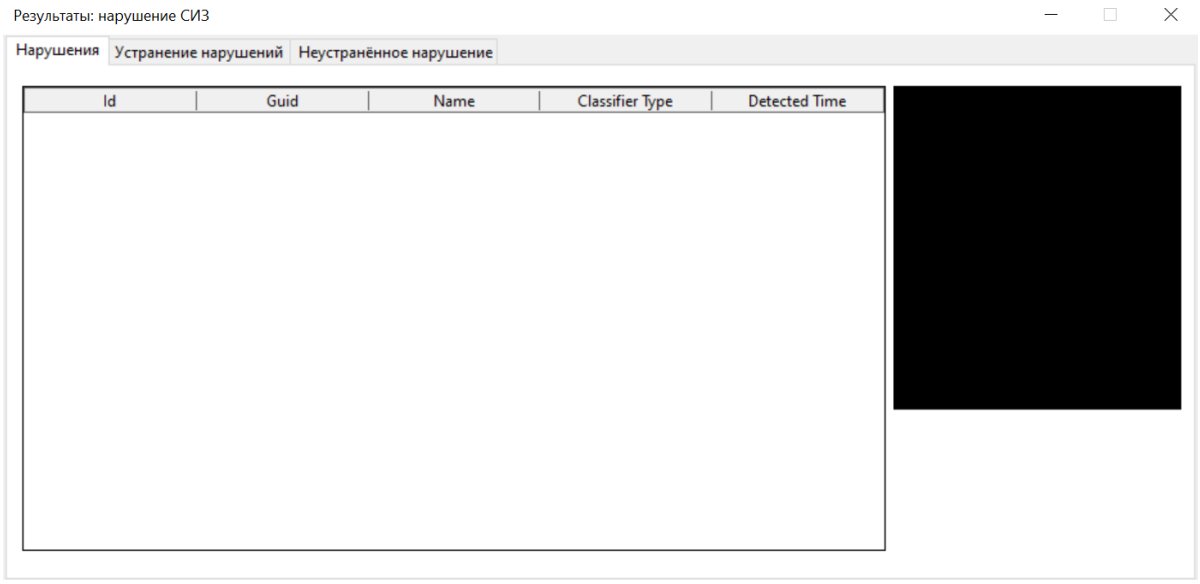


Рисунок 3.14 – Окно отображения событий

На рисунке 3.15 представлена таблица, в которой отображаются обнаруженные средства индивидуальной защиты:

- HH – каска;
- SO – комбинезон;
- DG – диэлектрические перчатки;
- HPO – термостойки комбинезон;
- HPFS – защитный лицевой щиток.

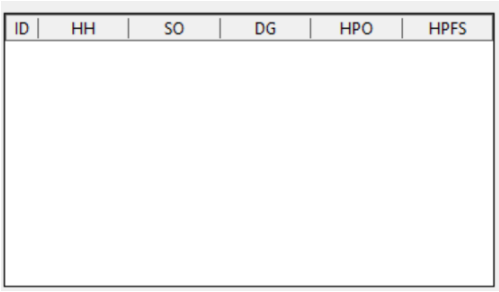


Рисунок 3.15 – Таблица обнаружения СИЗ

На рисунке 3.16 таблица обнаружения красной каски: RedHat – красная каска, Other – каска другого цвета.

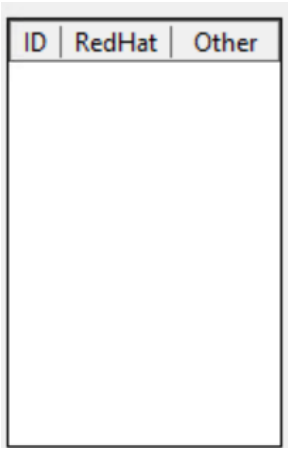


Рисунок 3.16 – Таблица обнаружения красной каски

На рисунке 3.17 таблица, в которой обнаружения правильности применения средств индивидуальной защиты:

- SRU – закатанные рукава;
- ONF – не застегнутый комбинезон;
- BOT – сапоги поверх брюк;
- GIO – перчатки под комбинезоном;
- GAC – края перчаток загнуты.

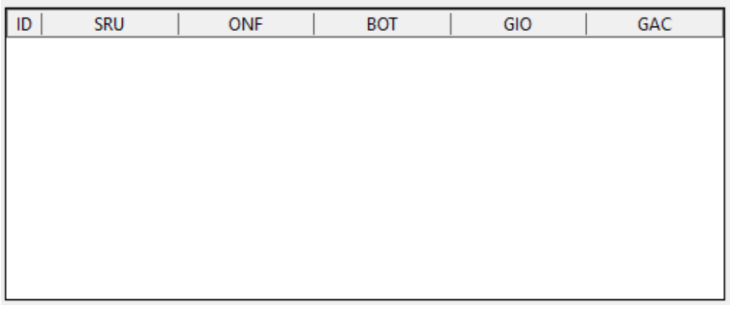


Рисунок 3.17 – Таблица обнаружения правильности применения СИЗ

4 ТЕСТИРОВАНИЕ

После обучения модели нейронной сети YOLOv4 были получены следующие результаты:

1. mAP (мера IoU, средний показатель точности) – 90,4%.
2. Функция потерь минимизирована до 0,5251.

На рисунке 4.1 представлен результат обучения нейронных сетей.

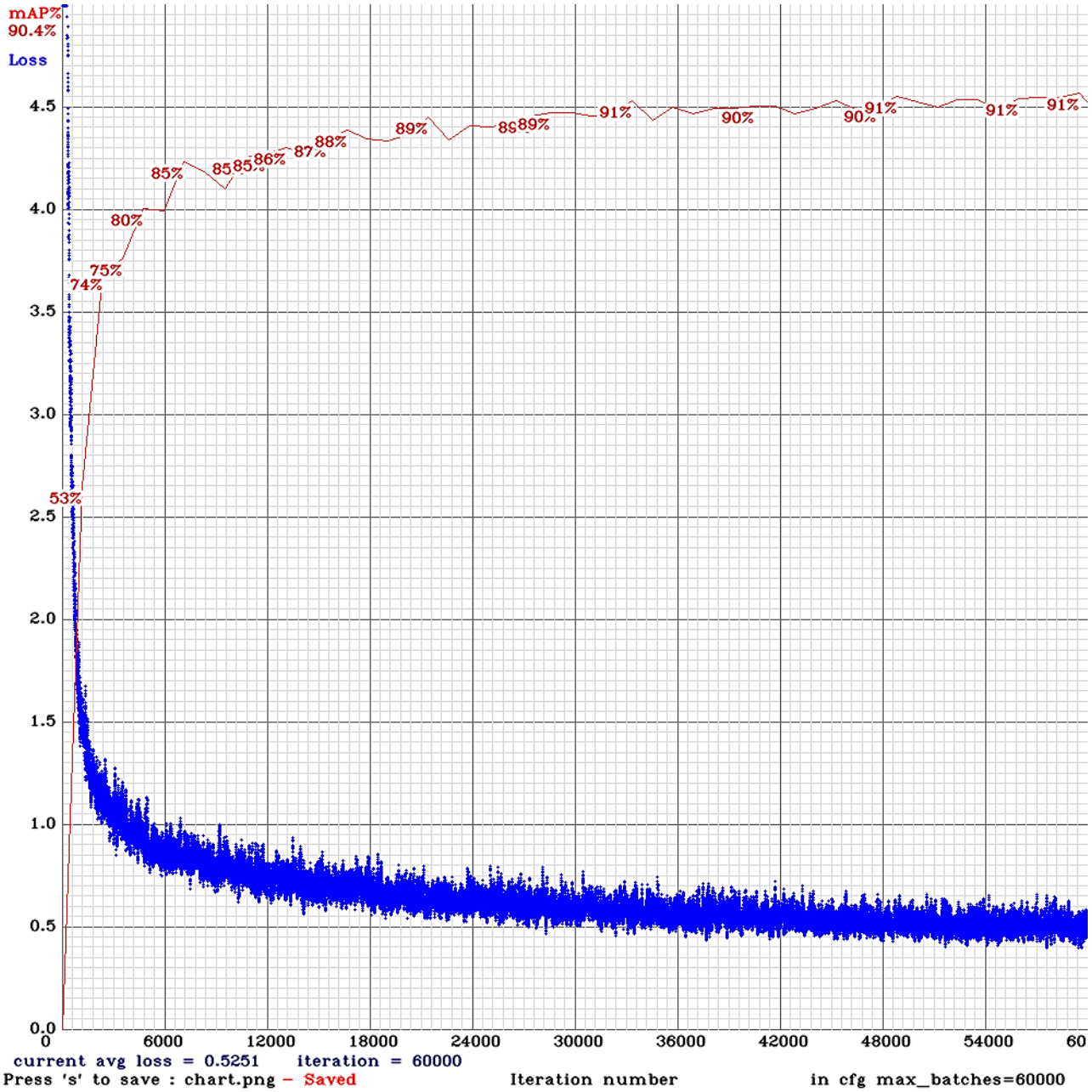


Рисунок 4.1 - Результат обучения нейронной сети

Человек считается правильно обнаруженным на кадре, если среди предложенных детектором для изображения зон есть зона, для которой мера IoU ≥ 0.5.

Для оценки эффективности детектора используется метрика mAP, зависящая от показателей Recall (R) и Precision (P):

$$R = \frac{n_{rd}}{n_{tnp}}, \quad (1)$$

$$P = \frac{n_{rd}}{n_{czp}}, \quad (2)$$

где n_rd – количество правильно обнаруженных людей;

n_tnp – общее количество людей в выборке;

n_czp – общее количество предложенных алгоритмом вариантов зон с людьми.

$$R = \frac{66\ 537}{69\ 943} = 0{,}95$$

$$P = \frac{66\ 537}{69\ 816} = 0{,}95$$

Для оценки правильности определения наличия средств индивидуальной защиты используется показатель S:

$$S = \frac{n_{rd}}{n_{tnp}}, \quad (3)$$

где n_rd – количество людей, для которых верно определен факт использования СИЗ;

n_tnp – количество правильно обнаруженных людей.

$$S = \frac{60\ 449}{66\ 537} = 0{,}9$$

Средний FPS на GPU NVIDIA 1660 составил 43 FPS (кадров в секунду).

Работа обученной нейронной сети, а именно детектирование и классификация обнаруженных объектов в разработанном интерфейсе пользователя и API представлена на рисунках 4.2 и 4.3.

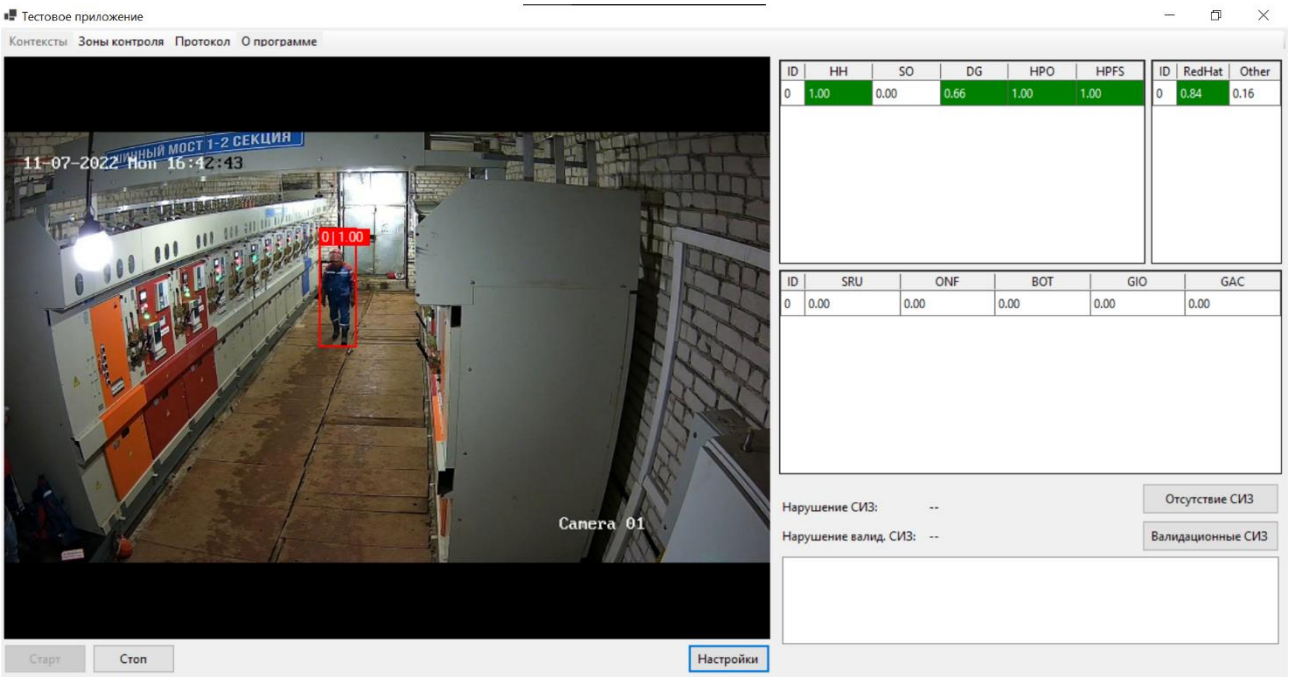


Рисунок 4.1 – Работа программы в интерфейсе пользователя


```
{
  "is_processing_success": true,
  "error_description": null,
  "result": [
    {
      "IsOk": true,
      "ObjectTracking": {
        "id": 0,
        "position": {
          "x": 0.58556676,
          "y": 0.2676165,
          "width": 0.08744413,
          "height": 0.38932705
        },
        "object": {
          "detection_result": {
            "probability": 0.9999504,
            "ClassProbabilty": null,
            "type": "PersonFront",
            "position": {
              "x": 0.58556676,
              "y": 0.2676165,
              "width": 0.08744413,
              "height": 0.38932705
            }
          },
```


Рисунок 4.2 – Работа программы в Swagger

5 ПРИМЕР РАБОТЫ СИСТЕМЫ

Для работы системы необходимо запустить нейросетевой сервис (рисунок 5.1) и сервис ядра, создать контекст обработки (рисунок 5.2), запустить видеопоток.

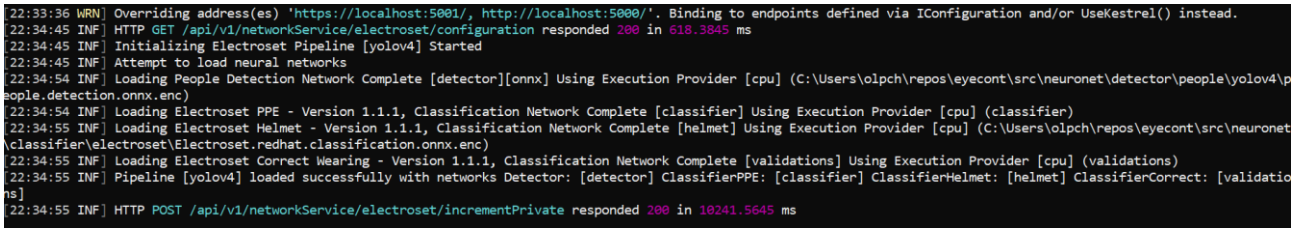


Рисунок 5.1 – Запуск нейросетевого сервиса и загрузка сетей

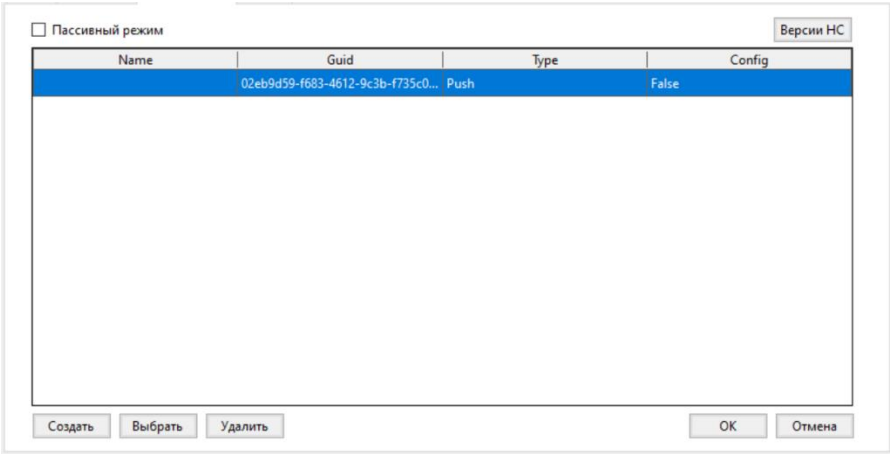


Рисунок 5.2 – Создание контекста обработки

На рисунке 5.3 продемонстрирована работа системы на видео с отсутствием нарушений.

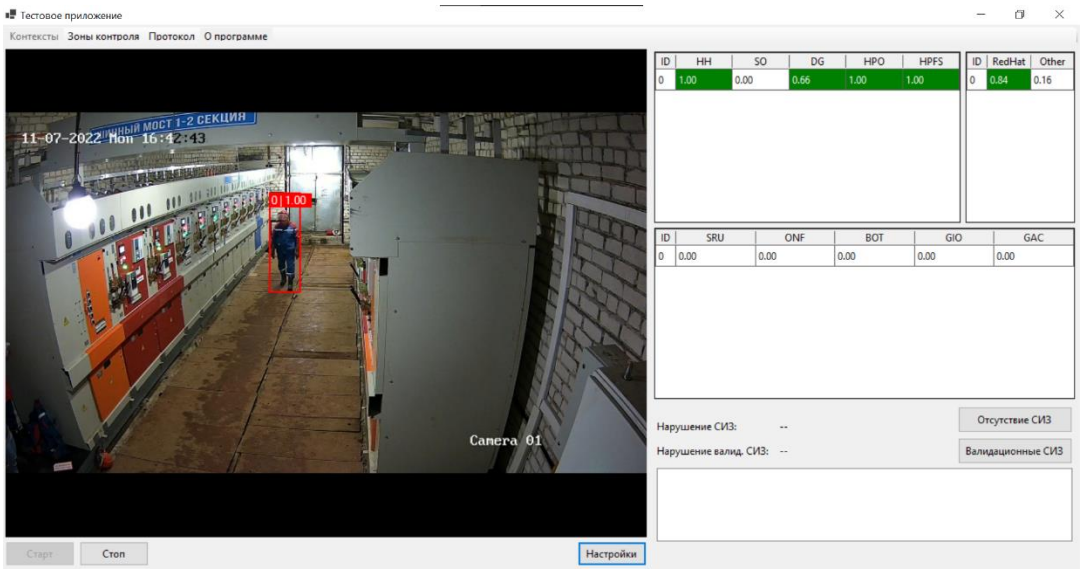


Рисунок 5.3 – Отсутствие нарушений

Работа системы с видео, где присутствует нарушение правильности применения средств индивидуальной защиты, а именно, у рабочего в красной каске отсутствуют диэлектрические перчатки, представлена на рисунках 5.4 и 5.5.

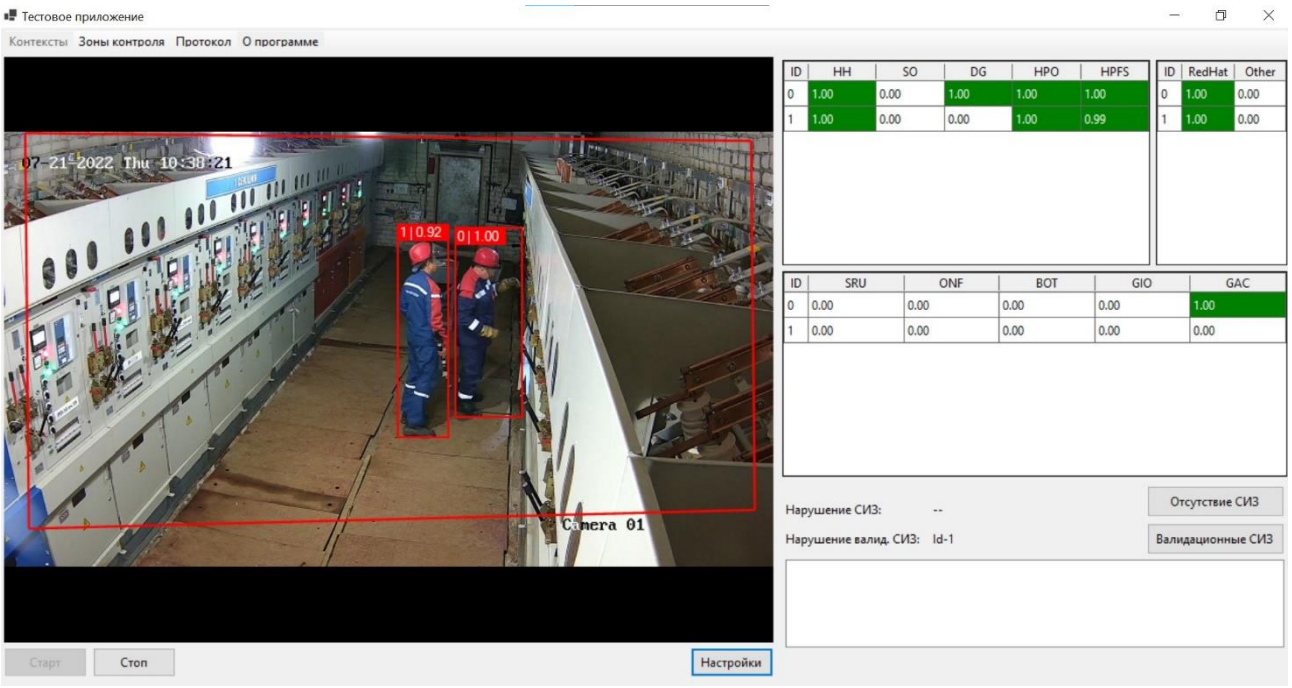


Рисунок 5.4 – Нарушение

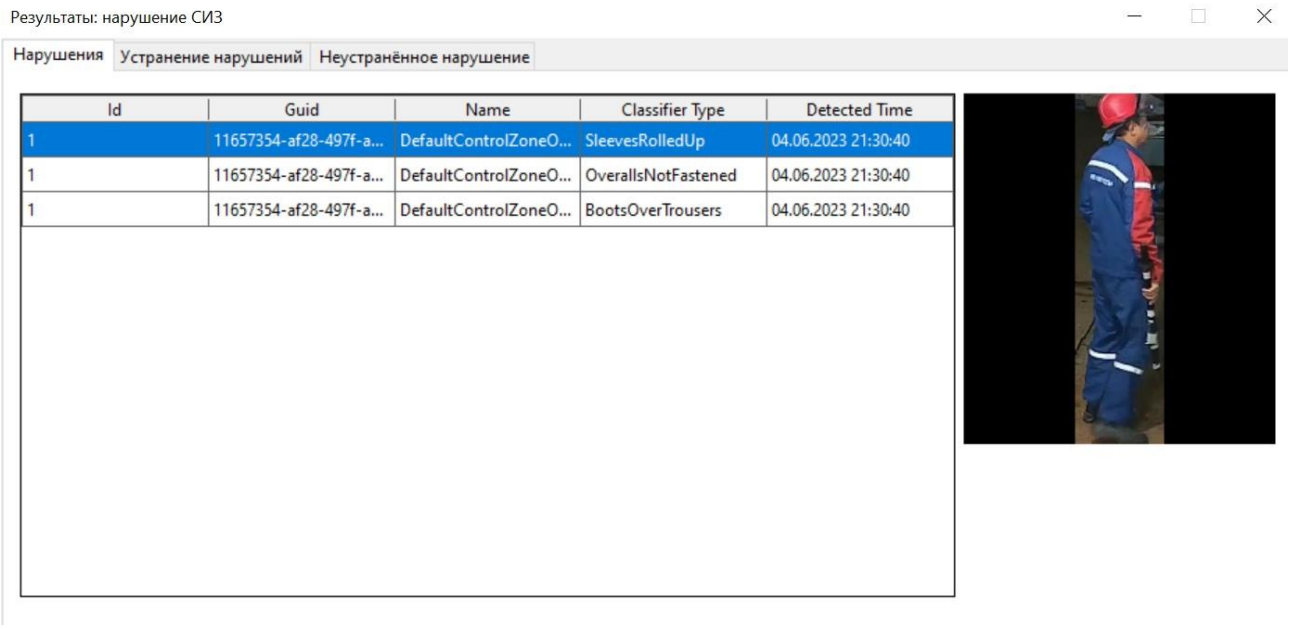


Рисунок 5.5 – Окно с событиями

ЗАКЛЮЧЕНИЕ

Промышленная безопасность – это комплекс мероприятий, цель которых – предотвратить аварийную ситуацию. Система промышленной безопасности начинает действовать с момента создания организации, во время производственной деятельности и заканчивается при полной ликвидации. Соблюдение ее норм и правил контролируется ответственными лицами, несущими ответственность за эксплуатацию оборудования и проведение работ согласно должностным обязанностям. Однако всегда найдутся люди, пренебрегающие указанными правилами по разным причинам, что может повлечь за собой различные травмы и даже летальные исходы.

Разработанная модуль поможет не только снизить процент травм и летальных исходов, но и автоматизировать контроль за соблюдением установленной техники безопасности.

В результате выполнения выпускной квалификационной работы была проанализирована литература по данной теме; выявлены аналоги разрабатываемой системы; выбраны технологии, языки программирования и среды разработки; разработан прототип системы; определены и выполнены основные этапы разработки системы; проведено тестирование системы.

В итоге получены следующие результаты: точность обнаружения людей по видеоданным по метрике mAP составила 90,4%. Все ключевые показатели, прописанные в техническим задании, выполнены.

Данный программный модуль был интегрирован в систему промышленной безопасности и охраны труда «EYECONT» компании ООО «Малленом Системс».

Сформулированные задачи выпускной квалификационной работы полностью решены. Поставленная цель – разработка модуля проверки наличия и правильности применения средств индивидуальной защиты для интеллектуальной системы – достигнута.

СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ

1. Официальный сайт компании ООО «Малленом Системс» [Электронный ресурс]: офиц. сайт. – Режим доступа: https://www.mallenom.ru
2. ISS – Интеллектуальные Системы Безопасности [Электронный ресурс]: офиц. сайт. – Режим доступа: https://iss.ru/products/securos-helmet-detector
3. Macroscop - профессиональное ПО для IP-камер [Электронный ресурс]: офиц. сайт. – Режим доступа: https://macroscop.com/produkty/programma-dlya-ip-kamer/detektor-otsutstviya-kaski
4. Системы видеонаблюдения: проектирование, производство, продажа оборудования для охранных CCTV [Электронный ресурс]: офиц. сайт. – Режим доступа: https://www.dssl.ru/products/hardhat-detector-detektor-nalichiya-kasok/
5. Центр2М – российский информационный оператор [Электронный ресурс]: офиц. сайт. – Режим доступа: https://center2m.ru/videoanalitika
6. HealthTech Insider [Электронный ресурс]: офиц. сайт. – Режим доступа: https://healthtechinsider.com/2017/10/31/smart-hard-hat-improves-industrial-safety-video/
7. NNTC [Электронный ресурс]: офиц. сайт. – Режим доступа: https://www.nntc.digital/solutions/iot-platform-for-industrial-safety-/
8. R [Электронный ресурс]: офиц. сайт. – Режим доступа: https://ru.wikipedia.org/wiki/R_(язык_программирования)
9. Python [Электронный ресурс]: офиц. сайт. – Режим доступа: https://ru.wikipedia.org/wiki/Python
10. Microsoft Visual Studio [Электронный ресурс]: офиц. сайт. – Режим доступа: https://ru.wikipedia.org/wiki/Microsoft_Visual_Studio
11. EYECONT [Электронный ресурс]: офиц. сайт. – Режим доступа: https://eyecont.ru

ПРИЛОЖЕНИЕ 1

(обязательное)

Код контроллеров сервисов

```
using System;
using System.Threading;
using System.Threading.Tasks;
using System.Windows.Forms;

using Mallenom;
using Mallenom.Diagnostics.Logs;
using Mallenom.FFmpeg;
using Mallenom.Framework;
using Mallenom.Imaging;
using Mallenom.Imaging.Processing;
using Mallenom.Video;
using Mallenom.Video.FFmpeg;

using EyeCont.Framework;
using EyeCont.Service.Client;
using EyeCont.Service.Contracts;
using EyeCont.TestApplication.Configuration;
using EyeCont.Service.Client.General;
using EyeCont.TestApplication.UI.Forms.General;

namespace EyeCont.TestApplication.Processing;

/// <summary>Процесс обработки видео.</summary>
public class BaseVideoProcessor<TSubscribtionClient, TProcessImageClient, TResults> :
IVideoProcessor<TSubscribtionClient, TProcessImageClient>
      where TSubscribtionClient : BaseSubscriptionClient
      where TProcessImageClient : BaseProcessImage<TResults>
{

      #region Static

      private static readonly ILog Log = LogManager.GetCurrentClassLog();

      #endregion

      #region Fields

      private IDisposable _imageSubscription;
      private FrameQueue<Reference<ImageData>> _frameQueue;
      private TProcessImageClient _imageClient;

      #endregion

      #region Properties

      public IImageSource VideoSource => Parameters.VideoSourceParameters.ImageSource;

      public TSubscribtionClient SubscribtionClient { get; set; }

      public CameraParameters Parameters { get; private set; }

      public VideoParameters ParametersVideoSource { get; private set; } = new();

      public int CameraId { get; set; }

      #endregion
```

```
        #region .ctor

        /// <summary>.ctor.</summary>
        /// <param name="parameters">Настройки камеры.</param>
        public BaseVideoProcessor(CameraParameters parameters)
        {
                Verify.Argument.IsNotNull(parameters, nameof(parameters));

                Parameters = parameters;
        }

        #endregion

        #region Methods

        /// <summary>Запуск обработки изображений.</summary>
        /// <returns></returns>
        public async Task Start()
        {
                try
                {
                        if(VideoSource == null || VideoSource.State.Type ==
ImageSourceStateType.Running) return;

                        if(_imageClient == null)
                        {
                                MessageBox.Show("Не выбран контекст", "Ошибка");
                                return;
                        }

                        if(_imageSubscription == null)
                        {
                                _imageSubscription = VideoSource.Images.Subscribe(
                                        (imageDataRef, metadata) =>
                                        {
                                                var newRef = new Reference<ImageData>();
                                                imageDataRef.CopyTo(newRef);
                                                _frameQueue.Enqueue(newRef);
                                        });
                        }
                        await VideoSource.ChangeStateAsync(ImageSourceTargetState.Running);
                }
                catch(FFmpegException)
                {
                        MessageBox.Show("Проверьте строку подключения", "Ошибка
подключения");
                }
                catch(Exception ex)
                {
                        Log.Warn(ex.Message);
                }

                _frameQueue = new FrameQueue<Reference<ImageData>>(100);

                var thread = new Thread(async () =>
                {
                        while(_frameQueue.TryDequeue(out var frame))
                        {
                                if(frame.HasValue)
                                {
                                        if(_imageClient != null)
                                        {
                                                try
                                                {
```

```
                                                await _imageClient.ProcessImageRaw(frame,
new ImageDescription
                                                {
                                                      Width = frame.Value.Width,
                                                      Height = frame.Value.Height,
                                                      Stride = frame.Value.Slice0.Stride,
                                                      ImagePixelFormat =
frame.Value.Format.ToImagePixelFormat()
                                                });
                                          }
                                          catch(System.IO.IOException ex)
                                          {
                                                Stop();

      Log.Error($"{nameof(BaseVideoProcessor<TSubscribtionClient, TProcessImageClient,
TResults>)}.Start(): a task was canceled. {ex.Message}");
                                          }
                                          catch(System.Net.Http.HttpRequestException ex)
                                          {
                                                Stop();

      Log.Error($"{nameof(BaseVideoProcessor<TSubscribtionClient, TProcessImageClient,
TResults>)}.Start(): neural network service not running. {ex.Message}");
                                          }
                                          catch(TaskCanceledException ex)
                                          {
                                                Stop();

      Log.Error($"{nameof(BaseVideoProcessor<TSubscribtionClient, TProcessImageClient,
TResults>)}.Start(): timeout exception. {ex.Message}");
                                          }
                                    }
                                    if(frame is Reference<ImageData> reference)
                                    {
                                          reference.UnreferenceValue();
                                    }
                              }
                              else
                              {
                                    Log.Error("Wrong image");
                              }
                        }
                  })
                  {
                        IsBackground = false,
                        Name = "Processing",
                        Priority = ThreadPriority.AboveNormal
                  };


                  thread.Start();
            }

            /// <summary>Остановка обработки изображений.</summary>
            /// <returns></returns>
            public async void Stop()
            {
                  if(_imageSubscription != null)
                  {
                        _imageSubscription.Dispose();
                        _imageSubscription = null;
                  }
                  _frameQueue?.Close();
                  await VideoSource.ChangeStateAsync(ImageSourceTargetState.Closed);
            }
```

```
        /// <summary>Настройка видеоисточника.</summary>
        /// <returns></returns>
        public async Task SetupVideoSource()
        {
                using(var setupForm = new SetupForm(ParametersVideoSource))
                {
                        if(setupForm.ShowDialog() == DialogResult.Yes)
                        {
                                ((FFmpegImageSource)VideoSource).Parameters = new
FFmpegImageSourceParameters(
                                        ParametersVideoSource.FileName,
                                        TimeSpan.FromSeconds(10),
                                        ParametersVideoSource.AutoRepeat,
                                        ParametersVideoSource.TimeStamps);
                        }
                }
        }

        /// <summary>Настройка клиента изображений.</summary>
        /// <param name="guid">Канал для видеообработки.</param>
        /// <param name="imageClient">Клиент предоставляющий доступ к контроллеру
изображений</param>
        public void SetupImageClient(
                Guid guid,
                TProcessImageClient imageClient)
        {
                if(VideoSource.State == ImageSourceState.Running) Stop();

                if(imageClient != null)
                {
                        //_imageClient.Dispose();
                        _imageClient = imageClient;
                }
        }

        /// <summary>Настройка подписки.</summary>
        /// <param name="guid">Канал для видеообработки.</param>
        public virtual void SetupSubscriptionClient(Guid guid)
        {

        }

        #endregion
}

using System;
using System.Collections.Generic;
using System.IO;
using System.Threading;
using System.Threading.Tasks;

using Microsoft.AspNetCore.Http;
using Microsoft.AspNetCore.Mvc;

using Mallenom.Framework;
using Mallenom.Swashbuckle.AspNetCore.MultiPart;
using Mallenom.Imaging.Processing;
using Mallenom.Configuration.Json.Polymorphic;
using Mallenom.OnnxRuntime.Classifiers;

using EyeCont.Infrastructure.Convector;
using EyeCont.Infrastructure.Prometheus.Network;
using EyeCont.NetworkService.Contracts;
using EyeCont.NetworkService.Contracts.Data;
```

```
using EyeCont.NetworkService.Contracts.Electroset;
using EyeCont.NetworkService.Contracts.Image;
using EyeCont.NetworkService.Electroset;
using EyeCont.Protection;
using EyeCont.Protection.Data;

namespace EyeCont.NetworkService.Controllers;

[ApiController]
[Route("/api/v1/networkService/electroset")]
public class ElectrosetPipelineController : Controller
{
	private static Serilog.ILogger Log = Serilog.Log.Logger;
	private readonly IPipelineRepository<ElectrosetPipeline,
ElectrosetPipelinesConfiguration, ElectrosetSettingsNetwork> _repository;
	private readonly IStreamConvert _converter;
	private INetworkPublisher _publisher;
	private readonly IImageDecoder _decoder;
	private readonly ILicenseChecker _checker;

	public ElectrosetPipelineController(
		IPipelineRepository<ElectrosetPipeline, ElectrosetPipelinesConfiguration,
ElectrosetSettingsNetwork> repository,
		INetworkPublisher publisher,
		ILicenseChecker checker,
		IImageDecoder decoder,
		IStreamConvert converter)
	{
		_repository = repository;
		_checker    = checker;
		_publisher  = publisher;
		_decoder    = decoder;
		_converter = converter;
	}

	/// <summary>Возвращает json список с именами ElectrosetPipeline.</summary>
	/// <returns>Json список с именами ElectrosetPipeline.</returns>
	/// <response code="200">Json список с именами ElectrosetPipeline
получен.</response>
	[HttpGet]
	[Produces(ContentTypes.Application.Json)]
	[ProducesResponseType(StatusCodes.Status200OK, Type =
typeof(IReadOnlyList<string>))]
	public ActionResult GetJsonPipelineControllers()
	{
		var jsRes = _repository.GetNames();
		return new JsonResult(jsRes);
	}


	/// <summary>Возвращает json с результатами обработки
ElectrosetPipeline.</summary>
	/// <param name="uploadedFile">Сжатое изображение.</param>
	/// <param name="variants">Имя экземпляра.</param>
	/// <param name="networkProcessingParameters">Параметры.</param>
	/// <param name="token">Токен.</param>
	/// <returns>Json с результатами обработки ElectrosetPipeline.</returns>
	/// <response code="200">Json с результатами обработки ElectrosetPipeline
получен</response>
	/// <response code="400">Получено изображение неверного формата</response>
	/// <response code="403">Ключ не найден.</response>
	/// <response code="404">Экземпляр ElectrosetPipeline <paramref
name="variants"/> не настроен в системе</response>
	/// <response code="500">Json с результатами обработки ElectrosetPipeline не
удалось получить</response>
```

```
	[HttpPost("{variants}/process")]
	[Consumes("multipart/form-data")]
	[Produces(ContentTypes.Application.Json)]
	[ProducesResponseType(StatusCodes.Status200OK, Type =
typeof(ProcessingResult<ElectrosetPipelineResults>))]
	[ProducesResponseType(StatusCodes.Status403Forbidden, Type =
typeof(ProcessingResult<ElectrosetPipelineResults>))]
	[ProducesResponseType(StatusCodes.Status404NotFound, Type =
typeof(ProcessingResult<ElectrosetPipelineResults>))]
	[ProducesResponseType(StatusCodes.Status400BadRequest, Type =
typeof(ProcessingResult<ElectrosetPipelineResults>))]
	[ProducesResponseType(StatusCodes.Status500InternalServerError, Type =
typeof(ProcessingResult<ElectrosetPipelineResults>))]
	public async Task<ActionResult> WorkWithImage(
		IFormFile uploadedFile,
		string variants,
		[FromJson] NetworkProcessingParametersOfElectroset
networkProcessingParameters,
		CancellationToken token = default)
	{
		token.ThrowIfCancellationRequested();

		var processing = new PipelineImageProcessing(_decoder, _publisher,
_converter);
		return await
processing.Compressed<NetworkProcessingParametersOfElectroset, ElectrosetPipeline,
ElectrosetPipelineResults, ElectrosetPipelinesConfiguration,
ElectrosetSettingsNetwork>(
			_repository,
			_checker,
			uploadedFile,
			variants,
			networkProcessingParameters,
			token);
	}


	/// <summary>Возвращает json с результатами обработки
ElectrosetPipeline.</summary>
	/// <param name="variants">Имя экземпляра.</param>
	/// <param name="uploadedFile">Несжатое изображение.</param>
	/// <param name="rawImageMeta">Описание изображения.</param>
	/// <param name="networkProcessingParameters">Параметры.</param>
	/// <param name="token">Токен.</param>
	/// <returns>Json с результатами обработки ElectrosetPipeline.</returns>
	/// <response code="200">Json с результатами обработки ElectrosetPipeline
получен</response>
	/// <response code="400">Получено ElectrosetPipeline неверного
формата</response>
	/// <response code="403">Ключ не найден.</response>
	/// <response code="404">Экземпляр ElectrosetPipeline <paramref
name="variants"/> не настроен в системе</response>
	/// <response code="500">Json с результатами обработки ElectrosetPipeline не
удалось получить</response>
	[HttpPost("{variants}/processRawImage")]
	[Consumes("multipart/form-data")]
	[Produces(ContentTypes.Application.Json)]
	[ProducesResponseType(StatusCodes.Status200OK, Type =
typeof(ProcessingResult<ElectrosetPipelineResults>))]
	[ProducesResponseType(StatusCodes.Status400BadRequest, Type =
typeof(ProcessingResult<ElectrosetPipelineResults>))]
	[ProducesResponseType(StatusCodes.Status403Forbidden, Type =
typeof(ProcessingResult<ElectrosetPipelineResults>))]
	[ProducesResponseType(StatusCodes.Status404NotFound, Type =
typeof(ProcessingResult<ElectrosetPipelineResults>))]
```

```
      [ProducesResponseType(StatusCodes.Status500InternalServerError, Type =
typeof(ProcessingResult<ElectrosetPipelineResults>))]
      public async Task<ActionResult> ReadImageRaw(
            IFormFile uploadedFile,
            string variants,
            [FromJson] RawImageMeta rawImageMeta,
            [FromJson] NetworkProcessingParametersOfElectroset
networkProcessingParameters,
            CancellationToken token = default)
      {
            token.ThrowIfCancellationRequested();

            var processing = new PipelineImageProcessing(_decoder, _publisher,
_converter);
            return await processing.Raw<NetworkProcessingParametersOfElectroset,
ElectrosetPipeline, ElectrosetPipelineResults>(
                  _repository,
                  _checker,
                  uploadedFile,
                  variants,
                  rawImageMeta,
                  networkProcessingParameters,
                  token);
      }

      /// <summary>Увеличивает счетчик используемых нейронных сетей.</summary>
      /// <response code="200">Счетчик сетей уменьшен.</response>
      /// <response code="403">Pipeline/сеть выгружен(а).</response>
      /// <response code="404">Pipeline/сеть не настроен(а) в системе.</response>
      [HttpPost("incrementPrivate")]
      [ApiExplorerSettings(IgnoreApi = true)]
      [ProducesResponseType(StatusCodes.Status200OK)]
      [ProducesResponseType(StatusCodes.Status404NotFound)]
      [ProducesResponseType(StatusCodes.Status500InternalServerError)]
      public ActionResult IncrementNetwork(string pipelineName)
      {
            try
            {
                  _repository.IncrementPipeline(pipelineName);

                  return new JsonResult(new ProcessingResult
                  {
                        ProcessingSuccess = true
                  })
                  {
                        StatusCode = StatusCodes.Status200OK
                  };
            }
            catch(NetworkNotFoundException ex)
            {
                  return new JsonResult(new ProcessingResult
                  {
                        ProcessingSuccess = false,
                        ErrorDescription = ex.Message
                  })
                  {
                        StatusCode = StatusCodes.Status404NotFound
                  };
            }
            catch(PipelineNotFoundException ex)
            {
                  return new JsonResult(new ProcessingResult
                  {
                        ProcessingSuccess = false,
                        ErrorDescription = ex.Message
```

```
                })
                {
                        StatusCode = StatusCodes.Status404NotFound
                };
        }
        catch(Exception ex)
        {
                return new JsonResult(new ProcessingResult
                {
                        ProcessingSuccess = false,
                        ErrorDescription = ex.Message
                })
                {
                        StatusCode = StatusCodes.Status500InternalServerError
                };
        }
}

/// <summary>Уменьшает счетчик используемых нейронных сетей.</summary>
/// <response code="200">Счетчик сетей уменьшен.</response>
/// <response code="403">Pipeline/сеть выгружен(а).</response>
/// <response code="404">Pipeline/сеть не настроен(а) в системе.</response>
[HttpPost("decrementPrivate")]
[ApiExplorerSettings(IgnoreApi = true)]
[Produces(ContentTypes.Application.Json)]
[ProducesResponseType(StatusCodes.Status200OK)]
[ProducesResponseType(StatusCodes.Status403Forbidden)]
[ProducesResponseType(StatusCodes.Status404NotFound)]
[ProducesResponseType(StatusCodes.Status500InternalServerError)]
public ActionResult DecrementNetwork(string pipelineName)
{
        try
        {
                _repository.DecrementPipeline(pipelineName);

                return new JsonResult(new ProcessingResult
                {

                        ProcessingSuccess = true
                })
                {
                        StatusCode = StatusCodes.Status200OK
                };
        }
        catch(NetworkNotFoundException ex)
        {
                return new JsonResult(new ProcessingResult
                {
                        ProcessingSuccess = false,
                        ErrorDescription = ex.Message
                })
                {
                        StatusCode = StatusCodes.Status404NotFound
                };
        }
        catch(NetworkUnloadedException ex)
        {
                return new JsonResult(new ProcessingResult
                {
                        ProcessingSuccess = false,
                        ErrorDescription = ex.Message
                })
                {
                        StatusCode = StatusCodes.Status403Forbidden
                };
```

```
            }
            catch(PipelineNotFoundException ex)
            {
                  return new JsonResult(new ProcessingResult
                  {
                        ProcessingSuccess = false,
                        ErrorDescription = ex.Message
                  })
                  {
                        StatusCode = StatusCodes.Status404NotFound
                  };
            }
            catch(Exception ex)
            {
                  return new JsonResult(new ProcessingResult
                  {
                        ProcessingSuccess = false,
                        ErrorDescription = ex.Message
                  })
                  {
                        StatusCode = StatusCodes.Status500InternalServerError
                  };
            }
      }


      /// <summary>Возвращает конфигурацию Pipeline.</summary>
      /// <param name="pipelineName">Имя Pipeline.</param>
      /// <response code="200">Информация получена.</response>
      /// <response code="404">Pipeline/сеть не настроен(а) в системе.</response>
      [HttpGet("configuration")]
      [Produces(ContentTypes.Application.Json)]
      [ProducesResponseType(StatusCodes.Status200OK, Type =
typeof(ProcessingResult<KeyValuePair<string, ElectrosetSettingsNetwork>>))]
      [ProducesResponseType(StatusCodes.Status404NotFound)]
      public ActionResult GetPipelineConfig(string pipelineName)
      {
            try
            {
                  return new
JsonResult(_repository.GetPipelineNetworkConfig(pipelineName));
            }
            catch(PipelineNotFoundException ex)
            {
                  return new JsonResult(new ProcessingResult
                  {
                        ProcessingSuccess = false,
                        ErrorDescription = ex.Message
                  })
                  {
                        StatusCode = StatusCodes.Status404NotFound
                  };
            }
      }

      #region PUT

      #region PUT networks

      /// <summary>Изменение параметров выбора типа сети.</summary>
      /// <param name="pipelineName">Имя pipeline.</param>
      /// <param name="pipelinesConfiguration">Параметры выбора типа сети.</param>
      /// <response code="200">Текущие настройки выбора типа сети изменены.</response>
      /// <response code="403">Pipeline/сеть выгружен(а).</response>
      /// <response code="404">Pipeline/сеть не настроен(а) в системе.</response>
```

```
	/// <response code="500">Текущие настройки выбора типа сети не удалось изменить
из-за внутренней ошибки сервиса.</response>
	[HttpPut("networksUpdate")]
	[ProducesResponseType(StatusCodes.Status200OK)]
	[ProducesResponseType(StatusCodes.Status403Forbidden)]
	[ProducesResponseType(StatusCodes.Status404NotFound)]
	[ProducesResponseType(StatusCodes.Status500InternalServerError)]
	public ActionResult NetworksUpdatePut(
	[FromHeader] string pipelineName,
	[FromBody] ElectrosetPipelinesConfiguration pipelinesConfiguration)
	{
		try
		{
			_repository.UpdateNetwork(pipelineName, pipelinesConfiguration);
			return new JsonResult(new ProcessingResult
			{
				ProcessingSuccess = true
			})
			{
				StatusCode = StatusCodes.Status200OK
			};
		}
		catch(NetworkNotFoundException ex)
		{
			return new JsonResult(new ProcessingResult
			{
				ProcessingSuccess = false,
				ErrorDescription = ex.Message
			})
			{
				StatusCode = StatusCodes.Status404NotFound
			};
		}
		catch(PipelineNotFoundException ex)
		{
			return new JsonResult(new ProcessingResult
			{
				ProcessingSuccess = false,
				ErrorDescription = ex.Message
			})
			{
				StatusCode = StatusCodes.Status404NotFound
			};
		}
		catch(InvalidOperationException ex)
		{
			return new JsonResult(new ProcessingResult
			{
				ProcessingSuccess = false,
				ErrorDescription = ex.Message
			})
			{
				StatusCode = StatusCodes.Status403Forbidden
			};
		}
		catch(Exception ex)
		{
			return new JsonResult(new ProcessingResult
			{
				ProcessingSuccess = false,
				ErrorDescription = ex.Message,
			})
			{
				StatusCode = StatusCodes.Status500InternalServerError
			};
```

```
            }
        }

        #endregion

        #endregion
}

using System;
using System.Collections.Generic;

using Microsoft.AspNetCore.Http;
using Microsoft.AspNetCore.Mvc;

using Mallenom.Framework;

using EyeCont.NetworkService.Contracts;
using EyeCont.Infrastructure.Swagger;
using EyeCont.Service.Contracts.Parameters;
using EyeCont.Service.Data.Electroset;
using Swashbuckle.AspNetCore.Filters;
using EyeCont.Service.Examples;

namespace EyeCont.Service.Controllers.Electroset;

/// <summary>Позволяет настроить сбор дампов событий контекста Electroset.</summary>
[ApiController]
[ApiGroup(ApiGroupNames.Electroset)]
[Route("api/v1/electroset/{guidContext:guid}/dumps")]
public class DumpOfElectrosetController : Controller
{
        private readonly IContextManagerOfElectroset _manager;

        /// <summary>.tor</summary>
        /// <param name="manager">Менеджер сбора дампов.</param>
        public DumpOfElectrosetController(IContextManagerOfElectroset manager)
        {
                _manager = manager;
        }
```

```
	#region ProcessedFrame


	#region GET ProcessedFrame


	/// <summary>Возвращает текущие настройки сбора данных о полных результатах обработки кадра.</summary>

	/// <param name="guidContext">GUID-контекста.</param>

	/// <response code="200">Текущие настройки сбора данных о полных результатах обработки кадра получены.</response>

	/// <response code="404">Контекст с данным GUID <paramref name="guidContext"/> не найден в системе.</response>

	/// <response code="500">Текущие настройки сбора данных о полных результатах обработки кадра не удалось получить из-за внутренней ошибки сервиса.</response>

	[HttpGet("processedFrameParameters")]

	[ProducesResponseType(StatusCodes.Status200OK, Type = typeof(DumpsProcessorParameters))]

	[ProducesResponseType(StatusCodes.Status404NotFound)]

	[ProducesResponseType(StatusCodes.Status500InternalServerError)]

	public ActionResult GetCurrentProcessedFrameParameters(

		[FromRoute] Guid guidContext) => new JsonResult(_manager.GetDumpSettings(guidContext).GetDumpsProcessedFrameParameters());


	#endregion


	#region PUT ProcessedFrame


	/// <summary>Изменение параметров сбора данных о полных результатах обработки кадра.</summary>

	/// <param name="parametersContext">Параметры сбора данных о полных результатах обработки кадра.</param>

	/// <param name="guidContext">Guid - контекста.</param>

	/// <response code="200">Текущие настройки сбора данных о полных результатах обработки кадра изменены.</response>

	/// <response code="404">Контекст с данным GUID <paramref name="guidContext"/> не найден в системе.</response>

	/// <response code="500">Текущие настройки сбора данных о полных результатах обработки кадра не удалось изменить из-за внутренней ошибки сервиса.</response>

	[HttpPut("processedFrameParameters")]

	[SwaggerRequestExample(typeof(DumpsProcessorParameters), typeof(DumpsProcessorParametersExample))]

	[ProducesResponseType(StatusCodes.Status200OK, Type = typeof(ProcessingResult))]
```

```
      [ProducesResponseType(StatusCodes.Status404NotFound, Type =
typeof(ProcessingResult))]
      [ProducesResponseType(StatusCodes.Status500InternalServerError, Type =
typeof(ProcessingResult))]
      public ActionResult ProcessorFrameParametersPut(
            [FromRoute] Guid guidContext,
            [FromBody] DumpsProcessorParameters parametersContext)
      {
            try
            {

      _manager.GetDumpSettings(guidContext).DupmsProcessedFrameParameterPut(parameters
Context);
                  return new JsonResult(new ProcessingResult
                  {
                        ProcessingSuccess = true
                  })
                  {
                        StatusCode = StatusCodes.Status200OK
                  };
            }
            catch(NotFoundException ex)
            {
                  return new JsonResult(new ProcessingResult
                  {
                        ProcessingSuccess = false,
                        ErrorDescription = ex.Message,
                  })
                  {
                        StatusCode = StatusCodes.Status404NotFound
                  };
            }
            catch(Exception ex)
            {
                  return new JsonResult(new ProcessingResult
                  {
                        ProcessingSuccess = false,
                        ErrorDescription = ex.Message,
```

```
                })
                {
                        StatusCode = StatusCodes.Status500InternalServerError
                };
        }
    }

    #endregion

    #endregion

    #region DetectedEquipment

    #region GET DetectedEquipment

    /// <summary>Возвращает текущие настройки сбора данных о событии детектирования нарушения СИЗ.</summary>
    /// <param name="guidContext">GUID-контекста.</param>
    /// <response code="200">Текущие настройки сбора данных о событии детектирования нарушения СИЗ получены.</response>
    /// <response code="404">Контекст с данным GUID <paramref name="guidContext"/> не найден в системе.</response>
    /// <response code="500">Текущие настройки сбора данных о событии детектирования нарушения СИЗ не удалось получить из-за внутренней ошибки сервиса.</response>
    [HttpGet("detectedEquipmentParameters")]
    [ProducesResponseType(StatusCodes.Status200OK, Type = typeof(DumpsProcessorParameters))]
    [ProducesResponseType(StatusCodes.Status404NotFound)]
    [ProducesResponseType(StatusCodes.Status500InternalServerError)]
    public ActionResult GetCurrentDetectedEquipmentParameters(
          [FromRoute] Guid guidContext) => new JsonResult(_manager.GetDumpSettings(guidContext).GetDumpsDetectedEquipmentParameters());

    #endregion

    #region PUT DetectedEquipment
```

```
	/// <summary>Изменение параметров сбора данных о событии детектирования нарушения СИЗ</summary>

	/// <param name="parametersContext">Параметры сбора данных о событии детектирования нарушения СИЗ.</param>

	/// <param name="guidContext">Guid – контекста.</param>

	/// <response code="200">Текущие настройки сбора данных о событии детектирования нарушения СИЗ изменены.</response>

	/// <response code="404">Контекст с данным GUID <paramref name="guidContext"/> не найден в системе.</response>

	/// <response code="500">Текущие настройки сбора данных о событии детектирования нарушения СИЗ не удалось изменить из-за внутренней ошибки сервиса.</response>

	[HttpPut("detectedEquipmentParameters")]

	[SwaggerRequestExample(typeof(DumpsProcessorParameters), typeof(DumpsProcessorParametersExample))]

	[ProducesResponseType(StatusCodes.Status200OK, Type = typeof(ProcessingResult))]

	[ProducesResponseType(StatusCodes.Status404NotFound, Type = typeof(ProcessingResult))]

	[ProducesResponseType(StatusCodes.Status500InternalServerError, Type = typeof(ProcessingResult))]

	public ActionResult DetectedEquipmentParametersPut(

		[FromRoute] Guid guidContext,

		[FromBody] DumpsProcessorParameters parametersContext)

	{

		try

		{


	_manager.GetDumpSettings(guidContext).DupmsDetectedEquipmentParameterPut(parametersContext);

			return new JsonResult(new ProcessingResult

			{

				ProcessingSuccess = true

			})

			{

				StatusCode = StatusCodes.Status200OK

			};

		}

		catch(NotFoundException ex)

		{

			return new JsonResult(new ProcessingResult

			{
```

```
                        ProcessingSuccess = false,
                        ErrorDescription = ex.Message,
                    })
                    {
                        StatusCode = StatusCodes.Status404NotFound
                    };
                }
                catch(Exception ex)
                {
                    return new JsonResult(new ProcessingResult
                    {
                        ProcessingSuccess = false,
                        ErrorDescription = ex.Message,
                    })
                    {
                        StatusCode = StatusCodes.Status500InternalServerError
                    };
                }
            }

            #endregion

            #endregion

            #region ResolvedEquipment

            #region GET ResolvedEquipment

            /// <summary>Возвращает текущие настройки сбора данных о событии устраненных нарушений СИЗ.</summary>
            /// <param name="guidContext">GUID-контекста.</param>
            /// <response code="200">Текущие настройки сбора данных о событии устраненных нарушений СИЗ получены.</response>
            /// <response code="404">Контекст с данным GUID <paramref name="guidContext"/> не найден в системе.</response>
            /// <response code="500">Текущие настройки сбора данных о событии устраненных нарушений СИЗ не удалось получить из-за внутренней ошибки сервиса.</response>
            [HttpGet("resolvedEquipmentParameters")]
```

```
      [ProducesResponseType(StatusCodes.Status200OK, Type =
typeof(DumpsProcessorParameters))]

      [ProducesResponseType(StatusCodes.Status404NotFound)]

      [ProducesResponseType(StatusCodes.Status500InternalServerError)]

      public ActionResult GetCurrentResolvedEquipmentParameters(

            [FromRoute] Guid guidContext) => new
JsonResult(_manager.GetDumpSettings(guidContext).GetDumpsResolvedEquipmentParameters()
);


      #endregion


      #region PUT ResolvedEquipment


      /// <summary>Изменение параметров сбора данных о событии устраненных нарушений
СИЗ.</summary>

      /// <param name="parametersContext">Параметры сбора данных о событии устраненных
нарушений СИЗ.</param>

      /// <param name="guidContext">Guid - контекста.</param>

      /// <response code="200">Текущие настройки сбора данных о событии устраненных
нарушений СИЗ изменены.</response>

      /// <response code="404">Контекст с данным GUID <paramref name="guidContext"/>
не найден в системе.</response>

      /// <response code="500">Текущие настройки сбора данных о событии устраненных
нарушений СИЗ не удалось изменить из-за внутренней ошибки сервиса.</response>

      [HttpPut("resolvedEquipmentParameters")]

      [SwaggerRequestExample(typeof(DumpsProcessorParameters),
typeof(DumpsProcessorParametersExample))]

      [ProducesResponseType(StatusCodes.Status200OK, Type = typeof(ProcessingResult))]

      [ProducesResponseType(StatusCodes.Status404NotFound, Type =
typeof(ProcessingResult))]

      [ProducesResponseType(StatusCodes.Status500InternalServerError, Type =
typeof(ProcessingResult))]

      public ActionResult ResolvedEquipmentParametersPut(

            [FromRoute] Guid guidContext,

            [FromBody] DumpsProcessorParameters parametersContext)

      {

            try

            {


      _manager.GetDumpSettings(guidContext).DupmsResolvedEquipmentParameterPut(paramet
ersContext);
```

```
            return new JsonResult(new ProcessingResult
            {
                ProcessingSuccess = true
            })
            {
                StatusCode = StatusCodes.Status200OK
            };
        }
        catch(NotFoundException ex)
        {
            return new JsonResult(new ProcessingResult
            {
                ProcessingSuccess = false,
                ErrorDescription = ex.Message,
            })
            {
                StatusCode = StatusCodes.Status404NotFound
            };
        }
        catch(Exception ex)
        {
            return new JsonResult(new ProcessingResult
            {
                ProcessingSuccess = false,
                ErrorDescription = ex.Message,
            })
            {
                StatusCode = StatusCodes.Status500InternalServerError
            };
        }
    }

    #endregion

    #endregion
```

```
	#region UnresolvedEquipment


	#region GET UnresolvedEquipment


	/// <summary>Возвращает текущие настройки сбора данных о событии неустраненных
нарушений СИЗ из-за пропажи объекта из кадра.</summary>

	/// <param name="guidContext">GUID-контекста.</param>

	/// <response code="200">Текущие настройки сбора данных о событии неустраненных
нарушений СИЗ из-за пропажи объекта из кадра получены.</response>

	/// <response code="404">Контекст с данным GUID <paramref name="guidContext"/>
не найден в системе.</response>

	/// <response code="500">Текущие настройки сбора данных о событии неустраненных
нарушений СИЗ из-за пропажи объекта из кадра не удалось получить из-за внутренней
ошибки сервиса.</response>

	[HttpGet("unresolvedEquipmentParameters")]

	[ProducesResponseType(StatusCodes.Status200OK, Type =
typeof(DumpsProcessorParameters))]

	[ProducesResponseType(StatusCodes.Status404NotFound)]

	[ProducesResponseType(StatusCodes.Status500InternalServerError)]

	public ActionResult GetUnresolvedEquipmentParameters(

		[FromRoute] Guid guidContext) => new
JsonResult(_manager.GetDumpSettings(guidContext).GetDumpsUnresolvedEquipmentParameters
());


	#endregion


	#region PUT UnresolvedEquipment


	/// <summary>Изменение параметров сбора данных о событии неустраненных нарушений
СИЗ из-за пропажи объекта из кадра.</summary>

	/// <param name="parametersContext">Параметры сбора данных о событии
неустраненных нарушений СИЗ из-за пропажи объекта из кадра.</param>

	/// <param name="guidContext">Guid - контекста.</param>

	/// <response code="200">Текущие настройки сбора данных о событии неустраненных
нарушений СИЗ из-за пропажи объекта из кадра изменены.</response>

	/// <response code="404">Контекст с данным GUID <paramref name="guidContext"/>
не найден в системе.</response>

	/// <response code="500">Текущие настройки сбора данных о событии неустраненных
нарушений СИЗ из-за пропажи объекта из кадра не удалось изменить из-за внутренней
ошибки сервиса.</response>

	[HttpPut("unresolvedEquipmentParameters")]
```

```
      [SwaggerRequestExample(typeof(DumpsProcessorParameters),
typeof(DumpsProcessorParametersExample))]

      [ProducesResponseType(StatusCodes.Status200OK, Type = typeof(ProcessingResult))]

      [ProducesResponseType(StatusCodes.Status404NotFound, Type =
typeof(ProcessingResult))]

      [ProducesResponseType(StatusCodes.Status500InternalServerError, Type =
typeof(ProcessingResult))]

      public ActionResult UnresolvedEquipmentParametersPut(

            [FromRoute] Guid guidContext,

            [FromBody] DumpsProcessorParameters parametersContext)

      {

            try

            {


      _manager.GetDumpSettings(guidContext).DupmsUnresolvedEquipmentParameterPut(param
etersContext);

                  return new JsonResult(new ProcessingResult

                  {

                        ProcessingSuccess = true

                  })

                  {

                        StatusCode = StatusCodes.Status200OK

                  };

            }

            catch(NotFoundException ex)

            {

                  return new JsonResult(new ProcessingResult

                  {

                        ProcessingSuccess = false,

                        ErrorDescription = ex.Message,

                  })

                  {

                        StatusCode = StatusCodes.Status404NotFound

                  };

            }

            catch(Exception ex)

            {

                  return new JsonResult(new ProcessingResult
```

```
                {
                        ProcessingSuccess = false,
                        ErrorDescription = ex.Message,
                })
                {
                        StatusCode = StatusCodes.Status500InternalServerError
                };
        }
    }

    #endregion

    #endregion

    #region DetectedValidationsEquipment

    #region GET DetectedValidatioonsEquipment

    /// <summary>Возвращает текущие настройки сбора данных о событии детектирования нарушения валидационных СИЗ.</summary>
    /// <param name="guidContext">GUID-контекста.</param>
    /// <response code="200">Текущие настройки сбора данных о событии детектирования нарушения валидационных СИЗ получены.</response>
    /// <response code="404">Контекст с данным GUID <paramref name="guidContext"/> не найден в системе.</response>
    /// <response code="500">Текущие настройки сбора данных о событии детектирования нарушения валижационных СИЗ не удалось получить из-за внутренней ошибки сервиса.</response>
    [HttpGet("detectedValidationsEquipmentParameters")]
    [ProducesResponseType(StatusCodes.Status200OK, Type = typeof(DumpsProcessorParameters))]
    [ProducesResponseType(StatusCodes.Status404NotFound)]
    [ProducesResponseType(StatusCodes.Status500InternalServerError)]
    public ActionResult GetCurrentDetectedValidationsEquipmentParameters(
        [FromRoute] Guid guidContext) => new JsonResult(_manager.GetDumpSettings(guidContext).GetDumpsDetectedValidationsEquipmentParameters());

    #endregion
```

```
	#region PUT DetectedValidationsEquipment


	/// <summary>Изменение параметров сбора данных о событии детектирования нарушения валидационных СИЗ</summary>

	/// <param name="parametersContext">Параметры сбора данных о событии детектирования нарушения валидационных СИЗ.</param>

	/// <param name="guidContext">Guid – контекста.</param>

	/// <response code="200">Текущие настройки сбора данных о событии детектирования нарушения валидационных СИЗ изменены.</response>

	/// <response code="404">Контекст с данным GUID <paramref name="guidContext"/> не найден в системе.</response>

	/// <response code="500">Текущие настройки сбора данных о событии детектирования нарушения валидационных СИЗ не удалось изменить из-за внутренней ошибки сервиса.</response>

	[HttpPut("detectedValidationsEquipmentParameters")]

	[SwaggerRequestExample(typeof(DumpsProcessorParameters), typeof(DumpsProcessorParametersExample))]

	[ProducesResponseType(StatusCodes.Status200OK, Type = typeof(ProcessingResult))]

	[ProducesResponseType(StatusCodes.Status404NotFound, Type = typeof(ProcessingResult))]

	[ProducesResponseType(StatusCodes.Status500InternalServerError, Type = typeof(ProcessingResult))]

	public ActionResult DetectedValidationsEquipmentParametersPut(

		[FromRoute] Guid guidContext,

		[FromBody] DumpsProcessorParameters parametersContext)

	{

		try

		{


	_manager.GetDumpSettings(guidContext).DupmsDetectedValidationsEquipmentParameterPut(parametersContext);

			return new JsonResult(new ProcessingResult

			{

				ProcessingSuccess = true

			})

			{

				StatusCode = StatusCodes.Status200OK

			};

		}

		catch(NotFoundException ex)
```

```
            {
                  return new JsonResult(new ProcessingResult
                  {
                        ProcessingSuccess = false,
                        ErrorDescription = ex.Message,
                  })
                  {
                        StatusCode = StatusCodes.Status404NotFound
                  };
            }
            catch(Exception ex)
            {
                  return new JsonResult(new ProcessingResult
                  {
                        ProcessingSuccess = false,
                        ErrorDescription = ex.Message,
                  })
                  {
                        StatusCode = StatusCodes.Status500InternalServerError
                  };
            }
      }

      #endregion

      #endregion

      #region ResolvedValidationsEquipment

      #region GET ResolvedValidationsEquipment

      /// <summary>Возвращает текущие настройки сбора данных о событии устраненных
нарушений валидационных СИЗ.</summary>
      /// <param name="guidContext">GUID-контекста.</param>
      /// <response code="200">Текущие настройки сбора данных о событии устраненных
нарушений валидационных СИЗ получены.</response>
```

```
      /// <response code="404">Контекст с данным GUID <paramref name="guidContext"/>
не найден в системе.</response>

      /// <response code="500">Текущие настройки сбора данных о событии устраненных
нарушений валидационных СИЗ не удалось получить из-за внутренней ошибки
сервиса.</response>

      [HttpGet("resolvedValidationsEquipmentParameters")]

      [ProducesResponseType(StatusCodes.Status200OK, Type =
typeof(DumpsProcessorParameters))]

      [ProducesResponseType(StatusCodes.Status404NotFound)]

      [ProducesResponseType(StatusCodes.Status500InternalServerError)]

      public ActionResult GetCurrentResolvedValidationsEquipmentParameters(

            [FromRoute] Guid guidContext) => new
JsonResult(_manager.GetDumpSettings(guidContext).GetDumpsResolvedValidationsEquipmentP
arameters());



      #endregion



      #region PUT ResolvedValidationsEquipment



      /// <summary>Изменение параметров сбора данных о событии устраненных нарушений
валидационных СИЗ.</summary>

      /// <param name="parametersContext">Параметры сбора данных о событии устраненных
нарушений валидационных СИЗ.</param>

      /// <param name="guidContext">Guid – контекста.</param>

      /// <response code="200">Текущие настройки сбора данных о событии устраненных
нарушений валидационных СИЗ изменены.</response>

      /// <response code="404">Контекст с данным GUID <paramref name="guidContext"/>
не найден в системе.</response>

      /// <response code="500">Текущие настройки сбора данных о событии устраненных
нарушений валидационных СИЗ не удалось изменить из-за внутренней ошибки
сервиса.</response>

      [HttpPut("resolvedValidationsEquipmentParameters")]

      [SwaggerRequestExample(typeof(DumpsProcessorParameters),
typeof(DumpsProcessorParametersExample))]

      [ProducesResponseType(StatusCodes.Status200OK, Type = typeof(ProcessingResult))]

      [ProducesResponseType(StatusCodes.Status404NotFound, Type =
typeof(ProcessingResult))]

      [ProducesResponseType(StatusCodes.Status500InternalServerError, Type =
typeof(ProcessingResult))]

      public ActionResult ResolvedValidationsEquipmentParametersPut(

            [FromRoute] Guid guidContext,

            [FromBody] DumpsProcessorParameters parametersContext)
```

```
	{
		try
		{

	_manager.GetDumpSettings(guidContext).DupmsResolvedValidationsEquipmentParameter
Put(parametersContext);
			return new JsonResult(new ProcessingResult
			{
				ProcessingSuccess = true
			})
			{
				StatusCode = StatusCodes.Status200OK
			};
		}
		catch(NotFoundException ex)
		{
			return new JsonResult(new ProcessingResult
			{
				ProcessingSuccess = false,
				ErrorDescription = ex.Message,
			})
			{
				StatusCode = StatusCodes.Status404NotFound
			};
		}
		catch(Exception ex)
		{
			return new JsonResult(new ProcessingResult
			{
				ProcessingSuccess = false,
				ErrorDescription = ex.Message,
			})
			{
				StatusCode = StatusCodes.Status500InternalServerError
			};
		}
	}
```

```
        #endregion


        #endregion


        #region UnresolvedValidationsEquipment


        #region GET UnresolvedValidationsEquipment


        /// <summary>Возвращает текущие настройки сбора данных о событии неустраненных
нарушений валидационных СИЗ из-за пропажи объекта из кадра.</summary>

        /// <param name="guidContext">GUID-контекста.</param>

        /// <response code="200">Текущие настройки сбора данных о событии неустраненных
нарушений валидационных СИЗ из-за пропажи объекта из кадра получены.</response>

        /// <response code="404">Контекст с данным GUID <paramref name="guidContext"/>
не найден в системе.</response>

        /// <response code="500">Текущие настройки сбора данных о событии неустраненных
нарушений валидационных СИЗ из-за пропажи объекта из кадра не удалось получить из-за
внутренней ошибки сервиса.</response>

        [HttpGet("unresolvedValidationsEquipmentParameters")]

        [ProducesResponseType(StatusCodes.Status200OK, Type =
typeof(DumpsProcessorParameters))]

        [ProducesResponseType(StatusCodes.Status404NotFound)]

        [ProducesResponseType(StatusCodes.Status500InternalServerError)]

        public ActionResult GetUnresolvedValidationsEquipmentParameters(

              [FromRoute] Guid guidContext) => new
JsonResult(_manager.GetDumpSettings(guidContext).GetDumpsUnresolvedValidationsEquipmen
tParameters());


        #endregion


        #region PUT UnresolvedValidationsEquipment


        /// <summary>Изменение параметров сбора данных о событии неустраненных нарушений
валидационных СИЗ из-за пропажи объекта из кадра.</summary>

        /// <param name="parametersContext">Параметры сбора данных о событии
неустраненных нарушений валидационных СИЗ из-за пропажи объекта из кадра.</param>

        /// <param name="guidContext">Guid - контекста.</param>

        /// <response code="200">Текущие настройки сбора данных о событии неустраненных
нарушений СИЗ из-за пропажи объекта из кадра изменены.</response>
```

```
      /// <response code="404">Контекст с данным GUID <paramref name="guidContext"/>
не найден в системе.</response>

      /// <response code="500">Текущие настройки сбора данных о событии неустраненных
нарушений валидационных СИЗ из-за пропажи объекта из кадра не удалось изменить из-за
внутренней ошибки сервиса.</response>

      [HttpPut("unresolvedValidationsEquipmentParameters")]

      [SwaggerRequestExample(typeof(DumpsProcessorParameters),
typeof(DumpsProcessorParametersExample))]

      [ProducesResponseType(StatusCodes.Status200OK, Type = typeof(ProcessingResult))]

      [ProducesResponseType(StatusCodes.Status404NotFound, Type =
typeof(ProcessingResult))]

      [ProducesResponseType(StatusCodes.Status500InternalServerError, Type =
typeof(ProcessingResult))]

      public ActionResult UnresolvedEquipmentValidationsParametersPut(

            [FromRoute] Guid guidContext,

            [FromBody] DumpsProcessorParameters parametersContext)

      {

            try

            {


      _manager.GetDumpSettings(guidContext).DupmsUnresolvedValidationsEquipmentParamet
erPut(parametersContext);

                  return new JsonResult(new ProcessingResult

                  {

                        ProcessingSuccess = true

                  })

                  {

                        StatusCode = StatusCodes.Status200OK

                  };

            }

            catch(NotFoundException ex)

            {

                  return new JsonResult(new ProcessingResult

                  {

                        ProcessingSuccess = false,

                        ErrorDescription = ex.Message,

                  })

                  {

                        StatusCode = StatusCodes.Status404NotFound
```

```
                };
            }
            catch(Exception ex)
            {
                return new JsonResult(new ProcessingResult
                {
                    ProcessingSuccess = false,
                    ErrorDescription = ex.Message,
                })
                {
                    StatusCode = StatusCodes.Status500InternalServerError
                };
            }
        }

        #endregion

        #endregion
}
```

ПРИЛОЖЕНИЕ 2

(обязательное)

Код тестового интерфейса

```
using System;
using System.Threading;
using System.Threading.Tasks;
using System.Windows.Forms;

using Mallenom;
using Mallenom.Diagnostics.Logs;
using Mallenom.FFmpeg;
using Mallenom.Framework;
using Mallenom.Imaging;
using Mallenom.Imaging.Processing;
using Mallenom.Video;
using Mallenom.Video.FFmpeg;

using EyeCont.Framework;
using EyeCont.Service.Client;
using EyeCont.Service.Contracts;
using EyeCont.TestApplication.Configuration;
using EyeCont.Service.Client.General;
using EyeCont.TestApplication.UI.Forms.General;

namespace EyeCont.TestApplication.Processing;

/// <summary>Процесс обработки видео.</summary>
public class BaseVideoProcessor<TSubscribtionClient, TProcessImageClient, TResults> :
IVideoProcessor<TSubscribtionClient, TProcessImageClient>
      where TSubscribtionClient : BaseSubscriptionClient
      where TProcessImageClient : BaseProcessImage<TResults>
{

      #region Static

      private static readonly ILog Log = LogManager.GetCurrentClassLog();

      #endregion

      #region Fields

      private IDisposable _imageSubscription;
      private FrameQueue<Reference<ImageData>> _frameQueue;
      private TProcessImageClient _imageClient;

      #endregion

      #region Properties

      public IImageSource VideoSource => Parameters.VideoSourceParameters.ImageSource;

      public TSubscribtionClient SubscribtionClient { get; set; }

      public CameraParameters Parameters { get; private set; }

      public VideoParameters ParametersVideoSource { get; private set; } = new();

      public int CameraId { get; set; }

      #endregion
```

```
        #region .ctor

        /// <summary>.ctor.</summary>
        /// <param name="parameters">Настройки камеры.</param>
        public BaseVideoProcessor(CameraParameters parameters)
        {
                Verify.Argument.IsNotNull(parameters, nameof(parameters));

                Parameters = parameters;
        }

        #endregion

        #region Methods

        /// <summary>Запуск обработки изображений.</summary>
        /// <returns></returns>
        public async Task Start()
        {
                try
                {
                        if(VideoSource == null || VideoSource.State.Type ==
ImageSourceStateType.Running) return;

                        if(_imageClient == null)
                        {
                                MessageBox.Show("Не выбран контекст", "Ошибка");
                                return;
                        }

                        if(_imageSubscription == null)
                        {
                                _imageSubscription = VideoSource.Images.Subscribe(
                                        (imageDataRef, metadata) =>
                                        {
                                                var newRef = new Reference<ImageData>();
                                                imageDataRef.CopyTo(newRef);
                                                _frameQueue.Enqueue(newRef);
                                        });
                        }
                        await VideoSource.ChangeStateAsync(ImageSourceTargetState.Running);
                }
                catch(FFmpegException)
                {
                        MessageBox.Show("Проверьте строку подключения", "Ошибка
подключения");
                }
                catch(Exception ex)
                {
                        Log.Warn(ex.Message);
                }

                _frameQueue = new FrameQueue<Reference<ImageData>>(100);

                var thread = new Thread(async () =>
                {
                        while(_frameQueue.TryDequeue(out var frame))
                        {
                                if(frame.HasValue)
                                {
                                        if(_imageClient != null)
                                        {
                                                try
                                                {
```

```
                                                await _imageClient.ProcessImageRaw(frame,
new ImageDescription
                                                {
                                                      Width = frame.Value.Width,
                                                      Height = frame.Value.Height,
                                                      Stride = frame.Value.Slice0.Stride,
                                                      ImagePixelFormat =
frame.Value.Format.ToImagePixelFormat()
                                                });
                                          }
                                          catch(System.IO.IOException ex)
                                          {
                                                Stop();

      Log.Error($"{nameof(BaseVideoProcessor<TSubscribtionClient, TProcessImageClient,
TResults>)}.Start(): a task was canceled. {ex.Message}");
                                          }
                                          catch(System.Net.Http.HttpRequestException ex)
                                          {
                                                Stop();

      Log.Error($"{nameof(BaseVideoProcessor<TSubscribtionClient, TProcessImageClient,
TResults>)}.Start(): neural network service not running. {ex.Message}");
                                          }
                                          catch(TaskCanceledException ex)
                                          {
                                                Stop();

      Log.Error($"{nameof(BaseVideoProcessor<TSubscribtionClient, TProcessImageClient,
TResults>)}.Start(): timeout exception. {ex.Message}");
                                          }
                                    }
                                    if(frame is Reference<ImageData> reference)
                                    {
                                          reference.UnreferenceValue();
                                    }
                              }
                              else
                              {
                                    Log.Error("Wrong image");
                              }
                        }
                  })
                  {
                        IsBackground = false,
                        Name = "Processing",
                        Priority = ThreadPriority.AboveNormal
                  };


                  thread.Start();
            }

            /// <summary>Остановка обработки изображений.</summary>
            /// <returns></returns>
            public async void Stop()
            {
                  if(_imageSubscription != null)
                  {
                        _imageSubscription.Dispose();
                        _imageSubscription = null;
                  }
                  _frameQueue?.Close();
                  await VideoSource.ChangeStateAsync(ImageSourceTargetState.Closed);
            }
```

```
        /// <summary>Настройка видеоисточника.</summary>
        /// <returns></returns>
        public async Task SetupVideoSource()
        {
                using(var setupForm = new SetupForm(ParametersVideoSource))
                {
                        if(setupForm.ShowDialog() == DialogResult.Yes)
                        {
                                ((FFmpegImageSource)VideoSource).Parameters = new
FFmpegImageSourceParameters(
                                        ParametersVideoSource.FileName,
                                        TimeSpan.FromSeconds(10),
                                        ParametersVideoSource.AutoRepeat,
                                        ParametersVideoSource.TimeStamps);
                        }
                }
        }

        /// <summary>Настройка клиента изображений.</summary>
        /// <param name="guid">Канал для видеообработки.</param>
        /// <param name="imageClient">Клиент предоставляющий доступ к контроллеру
изображений</param>
        public void SetupImageClient(
                Guid guid,
                TProcessImageClient imageClient)
        {
                if(VideoSource.State == ImageSourceState.Running) Stop();

                if(imageClient != null)
                {
                        //_imageClient.Dispose();
                        _imageClient = imageClient;
                }
        }

        /// <summary>Настройка подписки.</summary>
        /// <param name="guid">Канал для видеообработки.</param>
        public virtual void SetupSubscriptionClient(Guid guid)
        {

        }

        #endregion
}
```