МИНОБРНАУКИ РОССИИ
федеральное государственное бюджетное
образовательное учреждение высшего образования
«Череповецкий государственный университет»

Институт (факультет)	Институт информационных технологий
Направление подготовки (специальность)	09.03.04 Программная инженерия
Выпускающая кафедра	Математического и программного обеспечения ЭВМ

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА

Название работы Разработка программного обеспечения скелетизации изобра - жений человека для контроля опасных действий

Студента Богданова Александра Павловича
Ф.И.О.

ДОПУСТИТЬ К ЗАЩИТЕ

Директор института (декан факультета)	Ершов Евгений Валентинович
Заведующий выпускающей кафедрой	Ершов Евгений Валентинович
Руководитель выпускной квалификационной работы	Ершов Евгений Валентинович
Нормоконтролер	Виноградова Людмила Николаевна
Выпускник	Богданов Александр Павлович

МИНОБРНАУКИ РОССИИ
федеральное государственное бюджетное
образовательное учреждение высшего образования
«Череповецкий государственный университет»

АННОТАЦИЯ
выпускной квалификационной работы студента

по теме:
Разработка программного обеспечения скелетизации изображений человека для контроля опасных действий

Студент
Богданов Александр Павлович
фамилия, имя, отчество подпись

Руководитель работы
ФГБОУ ВО «Череповецкий государственные университет»
место работы
профессор
должность
Ершов Евгений Валентинович
фамилия, имя, отчество

(Перечислить основные вопросы, которые рассматривались; результаты работы)

Оглавление

Введение

Отслеживание действий сотрудников промышленных предприятий, работающих в потенциально опасных зонах – крайне сложный и ресурсоёмкий процесс. Отсутствие надлежащего контроля может привести к возникновению чрезвычайных ситуаций, ставящих под угрозу жизнь и здоровье рабочих. Поэтому проблема контроля безопасности на промышленных предприятиях на сегодняшний день остаётся актуальной. В особенности это касается сотрудников, взаимодействующих с заводскими станками и другими видами промышленной техники.

Для контроля безопасности на промышленных предприятиях компания-заказчик использует видеокамеры, способные при регистрации нарушения прервать работу агрегата или подать соответствующий сигнал. Однако, данное решение применимо только к некоторому небольшому числу агрегатов. Следовательно, агрегаты, за которыми не ведётся подобное наблюдение, обладают гораздо меньшим уровнем контроля безопасности. Также, для нового оснащения данное решение может оказаться неприменимым, утратив свою пригодность.

Целью данного проекта является повышение уровня безопасности на предприятии АО «Северсталь – Менеджмент».

1. Основная часть

1.1 Сравнительный анализ отечественных и зарубежных аналогов проектируемой системы

Предлагаемое решение заключается в создании программного обеспечения, которое в режиме реального времени анализировало бы видеопоток, поступающий с видеокамеры и в случае выявления нарушения останавливало работу агрегата, за которым произошло нарушение (если возможно), подавало предупредительный сигнал и фиксировало отчёт.

В ходе анализа были найдены следующие аналоги:

- на выставке «Безопасность и Охрана труда — 2018», компания КРОК представила работу передовых IT-систем для проведения предрейсовых и предсменных осмотров, контроля ношения работниками средств индивидуальной защиты, отслеживания физического состояния и локального позиционирования на объектах, отображения событий на 3D-модели здания и оценки поведения водителей в режиме реального времени;
- внутри компании «Северсталь» в цехе выплавки запущена в работу модель, которая фиксирует нахождение человека в подконвертерной зоне во время продувки. В случае фиксации нарушения модель автоматически отправляет снимок по электронной почте начальнику цеха и мастеру, а также включается сирена в подконвертерной зоне.
- внутри компании «Северсталь» в цехе выплавки запущена в работу модель, которая фиксирует нахождение человека в опасной зоне на машинах подачи кислорода. Система анализирует изображение с видеокамер и в режиме онлайн при помощи специальных алгоритмов определяет, находится ли работник в опасной зоне в то время, когда конвертер не пустой. В случае фиксации нарушения модель автоматически отправляет снимок по электронной почте начальнику цеха и мастеру;
- The industrial machine vision представляет собой комплексную интеграцию оптического, электронного, сенсорного и программного

обеспечения в производственный процесс. Основная цель данной системы заключается в обеспечении безопасности, и проверке качества промышленного продукта.

1.2 Выбор технологии, среды и языка программирования

Для разработки программного обеспечения в первую очередь нужно определиться с подходом к проектирования информационной системы. На данный момент существует два основных подхода: структурный и объектно-ориентированный.

Сущность структурного подхода к разработке информационной системы (ИС) заключается в ее декомпозиции (разбиении) на автоматизируемые функции: система разбивается на функциональные подсистемы, которые в свою очередь делятся на подфункции, подразделяемые на задачи и так далее. Процесс разбиения продолжается вплоть до конкретных процедур. При этом автоматизируемая система сохраняет целостное представление, в котором все составляющие компоненты взаимоувязаны [1].

Объектно-ориентированное программирование (ООП) — методология программирования, основанная на представлении программы в виде совокупности объектов, каждый из которых является экземпляром определённого класса, а классы образуют иерархию наследования [7].

В качестве подхода к разработке программного обеспечения было выбрано Объектно-ориентированное программирование (ООП). Выбор данного подхода обусловлен двумя факторами: относительная простота разработки программного обеспечения и дальнейшая модернизация, C# является объектно-ориентированным языком программирования, что позволит реализовать данный подход без проблем, также, данный подход позволит достаточно легко и точно декомпозировать объекты предметной области [7].

Разработка программного обеспечения предполагает соблюдение стандартов проектирования и написания программной документации и

спецификации. Спецификация разрабатываемого ПО создавалась с применением языка моделирования UML.

UML – унифицированный язык моделирования (англ. Unified Modeling Language) – это система обозначений, которую можно применять для объектно-ориентированного анализа и проектирования [2].

Его можно использовать для визуализации, спецификации, конструирования и документирования программных систем.

Словарь UML включает три вида строительных блоков:

- диаграммы;
- сущности;
- связи.

UML был выбран по следующим причинам:

- данный стандарт позволит максимально подробно описать систему со всех сторон;
- так как UML применяется для объектно-ориентированного анализа и проектирования, то он более всего нам подходит, так как его методология близка к программированию с применением объектного подхода [2].

Разработка программного обеспечения предполагается в соответствии с каскадной моделью жизненного цикла (ЖЦ) с промежуточным контролем (рис 1).

Рис. 1. Каскадная модель жизненного цикла

Выбор данной модели жизненного цикла для разработки системы обусловлен следующими причинами:

- данный тип модели ЖЦ дает план и временной график по всем этапам проекта, упорядочивая, таким образом, ход разработки;
- на каждом этапе разработки формируется законченный набор проектной документации, проверенный на полноту и согласованность;
- в случае изменения требований заказчика позволяет вернуться на любой шаг разработки и начать работу заново.

В данной работе для проектирования системы был выбран следующий стек технологий:

- для обнаружения человека на изображении используется связка библиотеки компьютерного зрения EmguCV и нейронной сети COCO;
- в качестве языка программирования использовался язык C# и платформа .Net;
- пользовательский интерфейс реализован на системе WPF;
- Microsoft Visual Studio 2019 в качестве среды разработки.

Для обработки видеопотока и его вывода на экран использовалась библиотека компьютерного зрения EmguCV. EmguCV — библиотека алгоритмов компьютерного зрения, обработки изображений и численных алгоритмов общего назначения с открытым кодом. Реализована на C/C++, также разрабатывается для Python, Java, Ruby, Matlab, Lua и других языков [4].

Для реализации программного обеспечения использовался язык C#, который работает на платформе .Net. C# — объектно-ориентированный язык программирования. Разработан в 1998—2001 годах группой инженеров компании Microsoft под руководством Андерса Хейлсберга и Скотта Вильтаумота как язык разработки приложений для платформы Microsoft .NET Framework. Впоследствии был стандартизирован как ECMA-334 и ISO/IEC 23270 [5].

Пользовательзкий интерфейс реализован на платформе WPF. Windows Presentation Foundation (WPF) — аналог WinForms, система для построения

клиентских приложений Windows с визуально привлекательными возможностями взаимодействия с пользователем, графическая (презентационная) подсистема в составе .NET Framework (начиная с версии 3.0), использующая язык XAML [6].

В качестве среды разработки ПО использовалась среда Microsoft Visual Studio. Microsoft Visual Studio — линейка продуктов компании Microsoft, включающих интегрированную среду разработки программного обеспечения и ряд других инструментальных средств. Данные продукты позволяют разрабатывать как консольные приложения, так и игры и приложения с графическим интерфейсом, в том числе с поддержкой технологии Windows Forms, а также веб-сайты, веб-приложения, веб-службы как в родном, так и в управляемом кодах для всех платформ, поддерживаемых Windows, Windows Mobile, Windows CE, .NET Framework, Xbox, Windows Phone .NET Compact Framework и Silverlight [6].

1.3 Анализ процесса обработки информации, выбор структур данных для ее хранения, выбор методов и алгоритмов решения задачи

В разработке программного обеспечения (ПО) будет использоваться паттерн MVVM.

Паттерн MVVM (Model-View-ViewModel) позволяет отделить логику приложения от визуальной части (представления). Данный паттерн является архитектурным, то есть он задает общую архитектуру приложения [3].

Данный паттерн был представлен Джоном Госсманом в 2005 году как модификация шаблона Presentation Model и был первоначально нацелен на разработку приложений в WPF.

MVVM состоит из трех компонентов: модели (Model), модели представления (ViewModel) и представления (View).

Модель описывает используемые в приложении данные. Модели могут содержать логику, непосредственно связанную этими данными, например,

логику валидации свойств модели. В то же время модель не должна содержать никакой логики, связанной с отображением данных и взаимодействием с визуальными элементами управления [3].

View или представление определяет визуальный интерфейс, через который пользователь взаимодействует с приложением. Применительно к WPF представление — это код в xaml, который определяет интерфейс в виде кнопок, текстовых полей и прочих визуальных элементов.

Хотя окно (класс Window) в WPF может содержать как интерфейс в xaml, так и привязанный к нему код C#, однако в идеале код C# не должен содержать какой-то логики, кроме разве что конструктора, который вызывает метод InitializeComponent и выполняет начальную инициализацию окна. Вся же основная логика приложения выносится в компонент ViewModel.

Однако иногда в файле связанного кода все может находиться некоторая логика, которую трудно реализовать в рамках паттерна MVVM во ViewModel.

Представление не обрабатывает события за редким исключением, а выполняет действия в основном посредством команд.

ViewModel или модель представления связывает модель и представление через механизм привязки данных. Если в модели изменяются значения свойств, при реализации моделью интерфейса INotifyPropertyChanged автоматически идет изменение отображаемых данных в представлении, хотя напрямую модель и представление не связаны [3].

ViewModel также содержит логику по получению данных из модели, которые потом передаются в представление. И также VewModel определяет логику по обновлению данных в модели [3].

Поскольку элементы представления, то есть визуальные компоненты типа кнопок, не используют события, то представление взаимодействует с ViewModel посредством команд.

Итогом применения паттерна MVVM является функциональное разделение приложения на три компонента, которые проще разрабатывать и тестировать, а также в дальнейшем модифицировать и поддерживать.

Для реализации алгоритма детекции частей тела и интерфейса программы была выбрана среда разработки Microsoft Visual Studio 2019. Языком разработки был выбран c#. По рекомендации заказчика, для работы с нейронными сетями была выбрана библиотека EmguCV. Для использования нейронной сети использовался фреймворк DNN Caffe.

В нашем техническом решении детекции частей тела используется подход «глубокого обучения», который позволяет классифицировать поданное на вход изображение (или сигнал) в соответствии с предварительной настройкой (обучением) нейронной сети [9].

В большинстве технических решений, основанных на глубоком обучении, используются свёрточные нейронные сети (CNN), наиболее известные своей способностью распознавать паттерны, присутствующие на изображениях. На сегодняшний день свёрточные нейронные сети достигли точности, превосходящей человеческий уровень. CNN используют фильтры, чтобы определять, какие особенности, такие как края, присутствуют на всем изображении. Фильтр — это просто матрица значений, называемых весами, которые обучены обнаруживать определенные особенности. Фильтр перемещается по каждой части изображения, чтобы проверить, присутствует ли признак, который он должен обнаруживать. Чтобы предоставить значение, показывающее, насколько достоверно наличие определенного признака, фильтр выполняет операцию свертки, которая представляет собой поэлементное произведение и сумму двух матриц. Если признак присутствует в части изображения, операция свертки между фильтром и этой частью изображения приводит к получению действительного числа с высоким значением. Если признак отсутствует, результирующее значение будет низким [9]. Пример операции свёртки представлен на рис. 2.

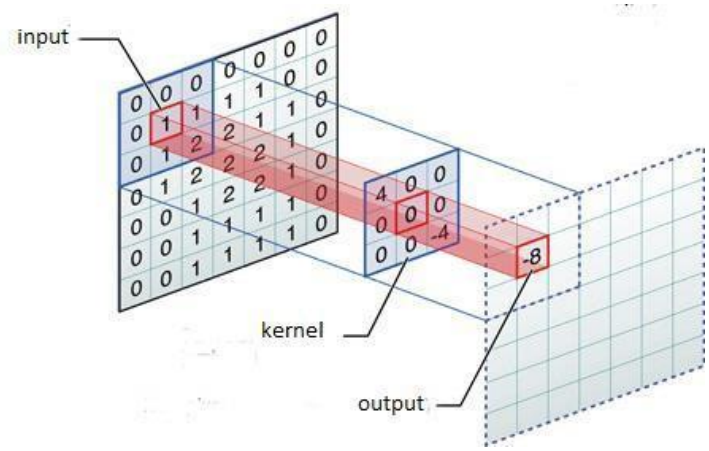


Рис. 2. Операция свёртки

Нейронная сеть OpenPose определяет части тела человека и их расположение в пространстве на видеозаписях и статичных изображениях. Сеть определяет положение туловища, рук, ног и других частей тела через двумерные координаты, не связывая, к какому именно человеку они относятся, а затем присваивает части отдельным людям. Также OpenPose обладает функционалом для распознания ключевых точек лица и рук. По словам разработчиков, OpenPose устойчив к перекрытию частей тела, в том числе при взаимодействии человека с объектом [8]. Пример подобной ситуации представлен на рис. 3.

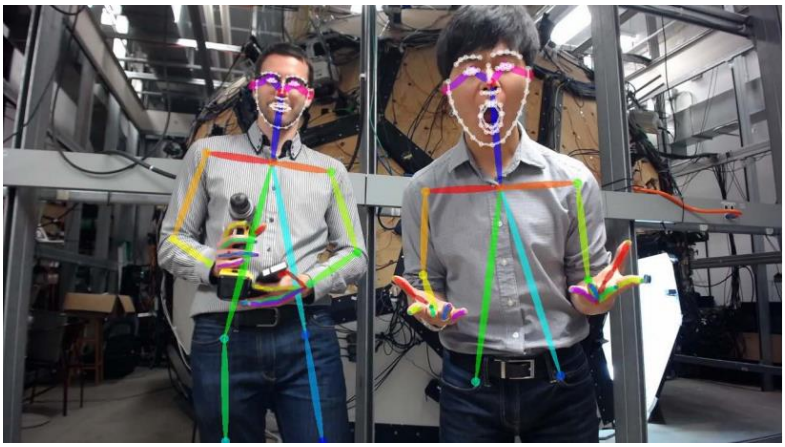


Рис. 3. Пример работы OpenPose

Схема работы OpenPose представлена на рис. 4.

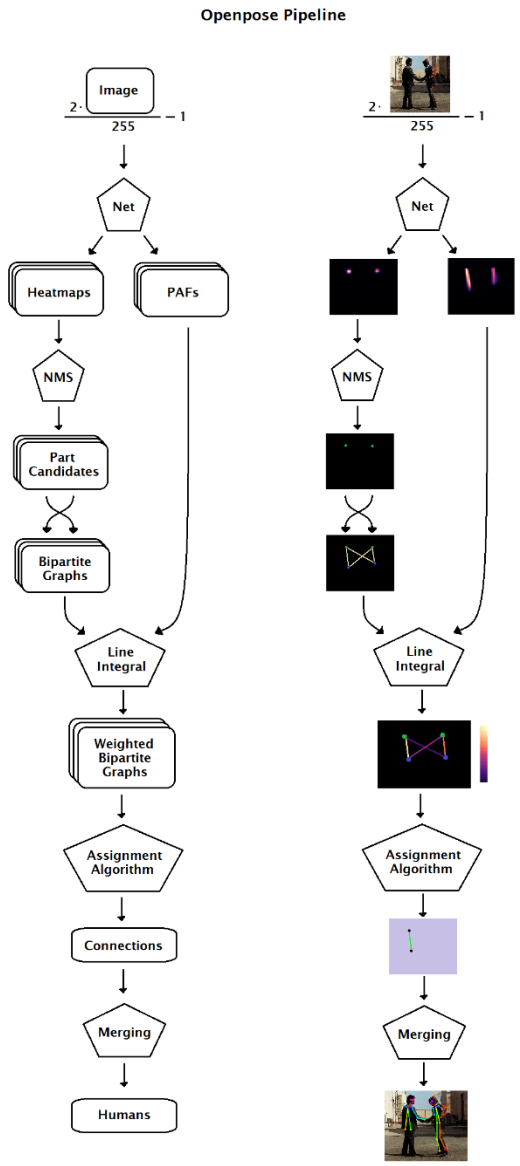


Рис. 4. Схема работы OpenPose

Далее описан алгоритм обнаружения частей тела людей при помощи OpenPose, который применяется в данном решении.

Граф, отмечающий части тела каждого человека, состоит из частей и пар. Часть тела – найденная нейронной сетью точка. Пара – соединение двух точек линией для образования части «скелета» [9].

Тепловая карта представляет собой матрицу, которая показывает уверенность сети в том, что определенный пиксель содержит определенную часть. Есть 18 (+1) тепловых карт, связанных с каждой из частей тела и

проиндексированных. из этих 18 матриц извлекается расположение частей тела.

Поля сходства частей (Part Affinity Fields) — это матрицы, которые дают информацию о положении и ориентации пар. Они существуют парами: для каждой части есть PAF в направлении «x» и PAF в направлении «y». Существует 38 индексированных PAF. Объединение частей в пары происходит благодаря этим 38 матрицам [8].

После создания OpenPose тепловых карт и полей сходства частей происходит извлечение расположения деталей из тепловой карты при помощи алгоритма не максимального подавления (NMS):

- Выбирается первый пиксель тепловой карты;
- Пиксель окружается окном со стороной 5. В этой области находится максимальное значение;
- Значение центрального пикселя области подставляется вместо максимума;
- Окно сдвигается на один пиксель, шаги 1-4 повторяются до полного охвата всей тепловой карты;
- Результат сравнивается с исходной тепловой картой. Пиксели с одинаковым значением являются искомыми пиками. Все остальные пиксели подавляются, получая значение 0;
- После всего процесса ненулевые пиксели обозначают местоположение кандидатов в части тела;
- После нахождения кандидатов для каждой из частей тела, их нужно соединить в пары. Для каждой возможной пары создаётся полный двудольный граф, вершины которого – все возможные кандидаты пары, а рёбра – все возможные соединения [8].

Для нахождения нужных связей в полученном графе необходимо решить «проблему присваивания». Для этого каждому ребру графа нужно присвоить вес при помощи линейного интеграла, представленного на рис. 5.

$$\int_{y_1}^{y_2} \int_{x_1}^{x_2} \begin{bmatrix} \mathbf{PAF_x}(x,y) \\ \mathbf{PAF_y}(x,y) \end{bmatrix} \cdot \begin{bmatrix} \mathbf{v_x} \\ \mathbf{v_y} \end{bmatrix} dxdy$$

Рис. 5. Линейный интеграл оценки соединений

Данный линейный интеграл даёт каждому соединению оценку, исходя из соответствующих этому соединению полей сходства частей PAF. Это позволяет решить задачу присваивания.

Решение задачи присваивания:

- Отсортировать каждое возможное соединение по его баллу;
- Связь с наивысшим баллом действительно является последней связью;
- Перейти к следующему возможному подключению. Если никакие части этого соединения не были назначены окончательному соединению ранее, это окончательное соединение;
- Шаг 3 повторяется до нахождения всех связей.

Пример решения задачи присваивания представлен на рис. 6.

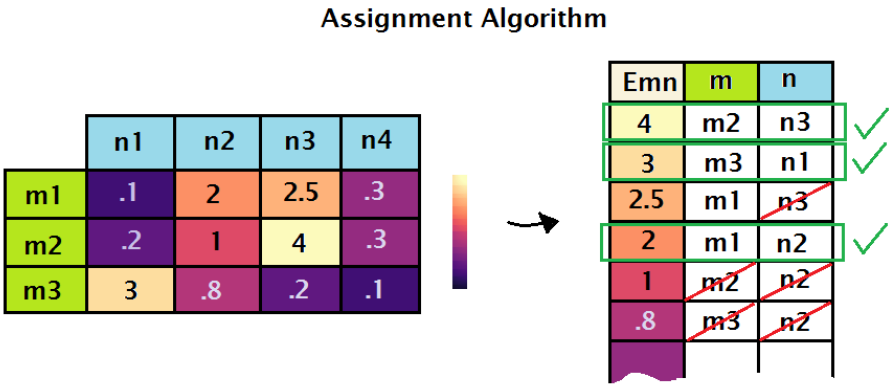


Рис. 6. Пример решения задачи присваивания

Последний шаг – преобразование найденных связей в «скелеты» людей на изображении. Изначально создаётся множество людей H, где каждый человек (набор связей) состоит из одной связи, а число людей равно числу связей. Затем, все люди попарно проверяются, и, если люди H1 и H2 имеют общий индекс части тела с одинаковыми координатами, это значит, что они являются

одним и тем же человеком. Содержимое H2 добавляется в H1, а H2 удаляется из множества. Это происходит до тех пор, пока в множестве не останется людей, использующих одну и ту же часть тела в связях [8].

На выходе каждый человек обозначается, как набор частей, где каждая часть содержит свой индекс, свои относительные координаты и свою оценку.

1.4 Разработка спецификаций проектируемой системы

В основе объектного подхода к разработке программного обеспечения лежит объектная декомпозиция, т. е. представление разрабатываемого программного обеспечения в виде совокупности объектов, в процессе взаимодействия, которых через передачу сообщений и происходит выполнение требуемых функций [1].

Спецификация разрабатываемого программного обеспечения при использовании UML объединяет несколько моделей: использования, логическую, реализации, процессов, развертывания [2].

1.4.1 Построение диаграмм вариантов использования

Одним из стандартных элементов языка UML является диаграмма вариантов использования, которая позволяет наглядно представить ожидаемое поведение системы. Диаграмма вариантов использования отображает взаимодействие между вариантами использования, представляющими функции системы, и действующими лицами, представляющими людей или системы, получающие или передающие информацию в данную систему [2].

Диаграмма вариантов использования для ПО обнаружения опасных действий работников представлена на рис.7.

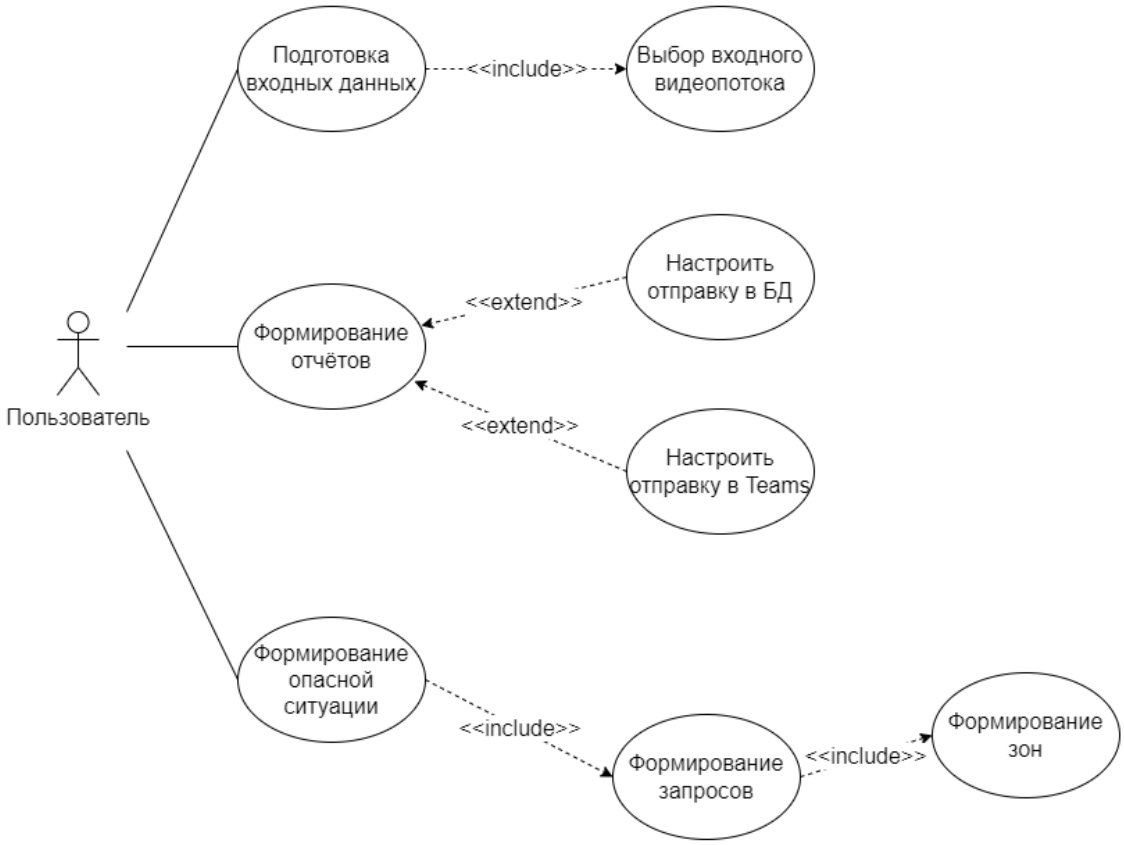


Рис.7. Диаграмма вариантов использования

У данной диаграммы одно действующее лицо – «Пользователь». Оно совершает все действия с программой: подготавливает входные данные, формирует отчёты и опасную ситуацию.

Краткое описание варианта использования «Подготовить входные данные» представлено в табл.1.

Таблица 1

Краткое описание варианта использования «Подготовка входных данных»

Название	Подготовка входных данных
Цель	Подготовить входные данные
Действующие лица	Пользователь
Краткое описание	Пользователь задает начальные параметры для работы системы
Тип варианта	Основой

Типичный ход событий для данного варианта использования представлен в табл.2.

Таблица 2

Типичный ход событий для варианта использования «Подготовка входных данных»

Действие исполнителя	Отклик системы
1 Пользователь обращается к настройкам входным данным	2 Система предоставляет работу с входными данными
3 Пользователь просматривает возможные варианты настройки	4 Система отображает настройку входных данных
5 Пользователь выбирает «Выбор входного видеопотока»	6 Система предоставляет пользователю возможность выбора видеопотока

Альтернатива

1. Если пользователь не настроил входные параметры, то они остаются со значениями по умолчанию.

Краткое описание варианта использования «Выбор входного видеопотока» представлено в табл.3.

Таблица 3

Краткое описание варианта использования «Выбор входного видеопотока»

Название	Выбор входного видеопотока
Цель	Выбрать входной видеопоток
Действующие лица	Пользователь
Краткое описание	Пользователь выбирает видеопоток, на котором будет происходить обнаружение людей в опасности
Тип варианта	Дополнительный

Типичный ход событий для данного варианта использования представлен в табл.4.

Таблица 4

Типичный ход событий для варианта использования «Выбор входного видеопотока»

Действие исполнителя	Отклик системы
1 Пользователь обращается к выбору видеопотока	2 Система предоставляет работу с видеопотоком
3 Пользователь просматривает возможные варианты загрузки видеопотока	4 Система отображает настройку видеопотока
5 Пользователь загружает видеопоток	6 Система принимает видеопоток и загружает его в программу

Альтернатива

5. Если пользователь загружает видеопоток некорректного формата, будет выведено сообщение об ошибке.

Краткое описание варианта использования «Формирование отчёта» представлено в табл.5.

Таблица 5

Краткое описание варианта использования «Формирование отчёта»

Название	Формирование отчёта
Цель	Сформировать отчёт
Действующие лица	Пользователь
Краткое описание	Пользователь выбирает реквизиты для отправки отчёта
Тип варианта	Основой

Типичный ход событий для данного варианта использования представлен в табл.6.

Таблица 6

Типичный ход событий для варианта использования «Формирование отчёта»

Действие исполнителя	Отклик системы
1 Пользователь формирует отчёт	2 Система предоставляет работу с отчётами
3 Пользователь просматривает возможные варианты настройки отчётов	4 Система отображает настройку отчётов
5 Пользователь выбирает «Настроить отправку в БД»	6 Система предоставляет пользователю возможность выбора параметров настройки отправки в БД
7 Пользователь выбирает «Настроить отправку в Teams»	8 Система предоставляет пользователю возможность выбора параметров настройки отправки в Teams

Альтернатива

3. Если пользователь не сформировал отчёт, то отправка в БД останется по умолчанию, а отправка в Teams не будет производиться.

Краткое описание варианта использования «Настроить отправку в БД» представлено в табл.7.

Таблица 7

Краткое описание варианта использования «Настроить отправку в БД»

Название	Настроить отправку в БД
Цель	Настроить параметры для отправки в базу данных
Действующие лица	Пользователь
Краткое описание	Пользователь редактирует параметры, отвечающие за отправление отчёта в базу данных
Тип варианта	Вспомогательный

Типичный ход событий для данного варианта использования представлен в табл.8.

Таблица 8

Типичный ход событий для варианта использования «Настроить отправку в БД»

Действие исполнителя	Отклик системы
1 Пользователь настраивает отправку в БД	2 Система предоставляет работу с настройкой отправки в БД
3 Пользователь просматривает настройку для загрузки строки подключения в БД	4 Система отображает настройку для загрузки строки подключения к БД
5 Пользователь вводит строку подключения к БД	6 Система сохраняет строку подключения в программе и использует ее при отправке отчётов

Краткое описание варианта использования «Настроить отправку в Teams» представлено в табл.9.

Таблица 9

Краткое описание варианта использования «Настроить отправку в Teams»

Название	Настроить отправку в Teams
Цель	Настроить параметры для отправки в Teams
Действующие лица	Пользователь
Краткое описание	Пользователь редактирует параметры, отвечающие за отправление отчёта в Teams
Тип варианта	Вспомогательный

Типичный ход событий для данного варианта использования представлен в табл.10.

Таблица 10

Типичный ход событий для варианта использования «Настроить отправку в Teams»

Действие исполнителя	Отклик системы
1 Пользователь настраивает отправку в Teams	2 Система предоставляет работу с настройкой отправки в Teams
3 Пользователь просматривает настройку для загрузки почты канала Teams	4 Система отображает настройку загрузки почты канала Teams
5 Пользователь вводит почту для отправки отчёта.	6 Система сохраняет почту и использует её при отправке отчётов

Краткое описание варианта использования «Формирование опасной ситуации» представлено в табл.11.

Таблица 11

Краткое описание варианта использования «Формирование опасной ситуации»

Название	Формирование опасной ситуации
Цель	Сформировать опасную ситуацию
Действующие лица	Пользователь
Краткое описание	Пользователь моделирует опасную ситуацию, обнаружив которую, программа среагирует
Тип варианта	Основой

Типичный ход событий для данного варианта использования представлен в табл.12.

Таблица 12

Типичный ход событий для варианта использования «Формирование опасной ситуации»

Действие исполнителя	Отклик системы
1 Пользователь формирует опасную ситуацию	2 Система предоставляет работу с формированием опасной ситуации
3 Пользователь просматривает возможные варианты настройки формирования опасной ситуации	4 Система отображает настройку формирования опасной ситуации
5 Пользователь выбирает «Формирование зон»	6 Система предоставляет пользователю возможность выбора параметров формирования зон
7 Пользователь выбирает «Формирование запросов»	8 Система предоставляет пользователю возможность выбора параметров формирования запросов

Краткое описание варианта использования «Формирование зон» представлено в табл.13.

Таблица 13

Краткое описание варианта использования «Формирование зон»

Название	Формирование зон
Цель	Сформировать зоны
Действующие лица	Пользователь
Краткое описание	Пользователь добавляет, удаляет или редактирует зоны, отвечающие за обнаружение частей тела на кадре в конкретной области
Тип варианта	Дополнительный

Типичный ход событий для данного варианта использования представлен в табл.14.

Таблица 14

Типичный ход событий для варианта использования «Формирование зон»

Действие исполнителя	Отклик системы
1 Пользователь обращается к формированию зон	2 Система предоставляет зону для редактирования её параметров
3 Пользователь настраивает возможные параметры зоны	4 Система сохраняет и применяет параметры зоны

Краткое описание варианта использования «Формирование запросов» представлено в табл.15.

Таблица 15

Краткое описание варианта использования «Формирование запросов»

Название	Формирование запросов
Цель	Сформировать запросы
Действующие лица	Пользователь
Краткое описание	Пользователь добавляет, удаляет или редактирует запросы, отвечающие за формирование логики опасной ситуации путём комбинирования зон
Тип варианта	Дополнительный

Типичный ход событий для данного варианта использования представлен в табл.16.

Таблица 16

Типичный ход событий для варианта использования «Формирование запросов»

Действие исполнителя	Отклик системы
1 Пользователь обращается к формированию запросов	2 система предоставляет запрос для редактирования его параметров
3 Пользователь настраивает возможные параметры запроса	4 Система сохраняет и применяет параметры запроса

1.4.2 Построение контекстных диаграмм классов

Концептуальные диаграммы демонстрируют связи между основными понятиями предметной области. Концептуальные модели в соответствии с определением оперируют понятиями предметной области, атрибутами этих понятий и отношениями между ними. Понятию в предметной области разрабатываемого программного обеспечения могут соответствовать как материальные предметы, так и абстракции, которые применяют специалисты предметной области [1].

Основной сущностью всей системы является Запрос (рис. 8). На его основе происходит формирование опасной ситуации. Каждый запрос содержит одну или несколько зон, которые нужны для выделения ключевых областей на кадре, который обрабатывает нейронная сеть с помощью алгоритма, анализируя видеопоток.

Зоны и запросы формируются пользователем с помощью соответствующих алгоритмов, которые использует система, при получении команд от пользователя. При анализе людей, обнаруженных в определенных областях кадра нейронной сетью, запрос может с помощью алгоритма сформировать отчёт в базу данных (БД) и Teams.

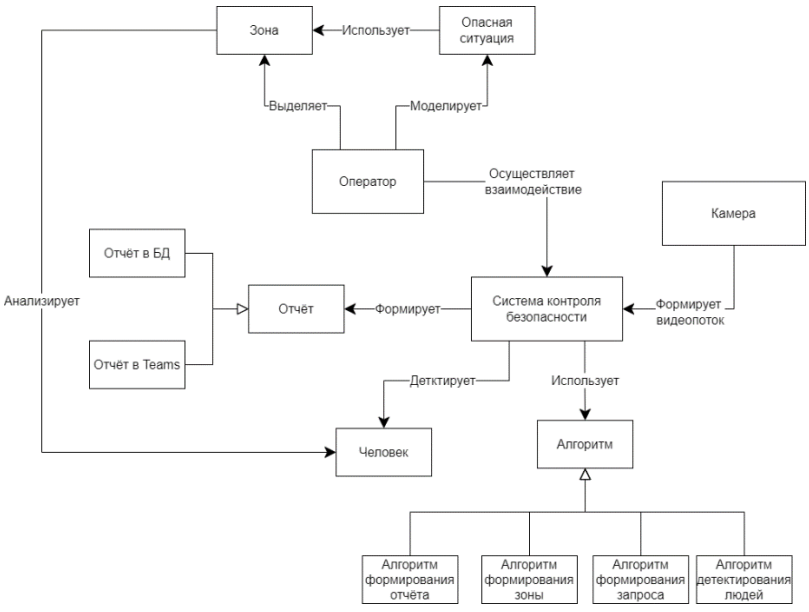


Рис.8. Контекстная диаграмма классов

1.4.3 Построение диаграмм последовательности системы

Диаграмма последовательности — диаграмма UML, на которой для некоторого набора объектов на единой временной оси показан жизненный цикл объекта (создание-деятельность-уничтожение некой сущности) и взаимодействие действующих лиц ПО в рамках прецедента.

Диаграмма последовательности для варианта использования «Подготовка входных данных» представлена на рис. 9. Описания операций представлены в табл. 17-19.

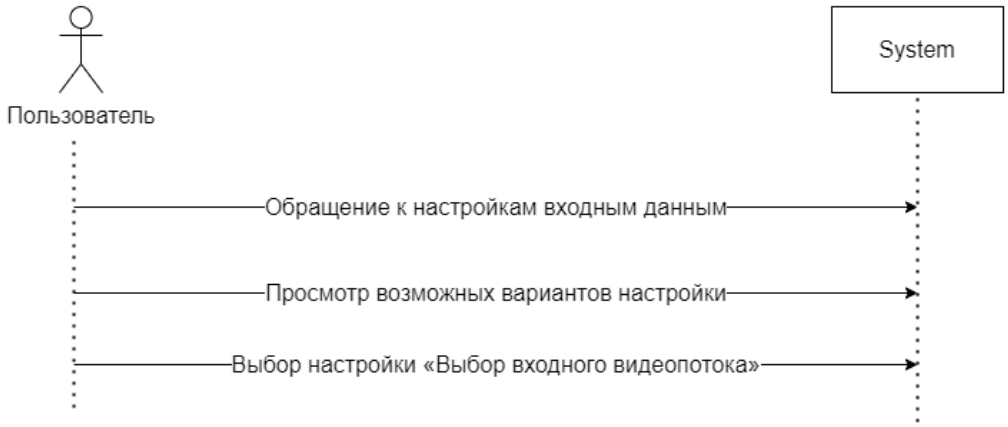


Рис. 9. Диаграмма последовательности для варианта использования «Подготовка входных данных»

Таблица 17

Описание операции «Обращение к настройкам входных данных»

Раздел	Описание
Имя	Обращение к настройкам входных данных
Обязанности	Предоставить настройку входных данных
Тип	Системная
Ссылка	Вариант использования «Подготовка входных данных»
Примечания	-
Исключения	-
Вывод	-
Предусловие	-
Постусловие	-

Таблица 18

Описание операции «Просмотр возможных вариантов настройки»

Раздел	Описание
Имя	Просмотр возможных вариантов настройки
Обязанности	Отобразить возможные варианты настройки входных данных
Тип	Системная
Ссылка	Вариант использования «Подготовка входных данных»
Примечания	-
Исключения	-
Вывод	Варианты настройки системы
Предусловие	Открытая вкладка настроек
Постусловие	-

Таблица 19

Описание операции «Выбор настройки входного видеопотока»

Раздел	Описание
Имя	Выбор настройки видеопотока
Обязанности	Предоставить возможность настройки видеопотока
Тип	Системная
Ссылка	Вариант использования «Подготовка входных данных»
Примечания	-
Исключения	Исключение выбора видеопотока неверного формата
Вывод	-
Предусловие	Открытая вкладка детекции
Постусловие	Начатие обработки видеопотока

Диаграмма последовательности для варианта использования «Выбор входного видеопотока» представлена на рис. 10. Описания операций представлены в табл. 20-22.

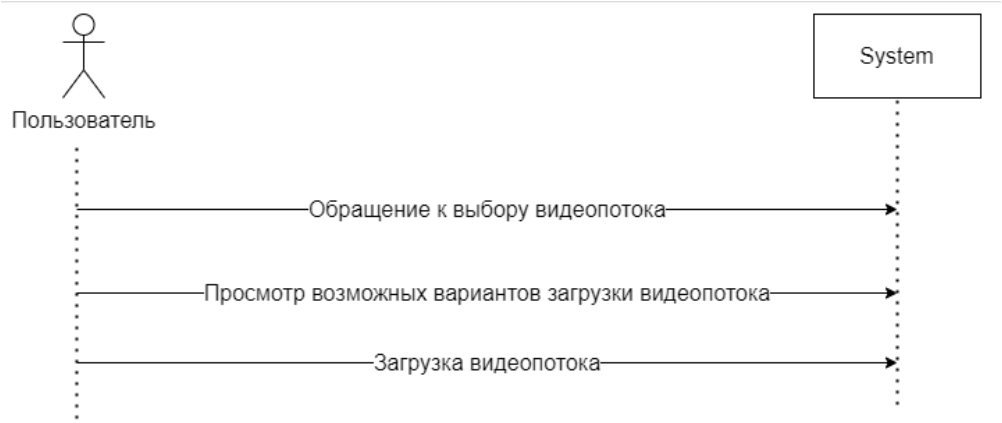


Рис. 10. Диаграмма последовательности для варианта использования «Выбор входного видеопотока»

Таблица 20

Описание операции «Обращение к выбору видеопотока»

Раздел	Описание
Имя	Обращение к выбору видеопотока
Обязанности	Предоставить настройку выбора видеопотока
Тип	Системная
Ссылка	Вариант использования «Выбор входного видеопотока»
Примечания	-
Исключения	-
Вывод	-
Предусловие	Выбрана настройка видеопотока
Постусловие	-

Таблица 21

Описание операции «Просмотр возможных вариантов загрузки видеопотока»

Раздел	Описание
1	2
Имя	Просмотр возможных вариантов загрузки видеопотока
Обязанности	Отобразить возможные варианты настройки видеопотока
Тип	Системная
Ссылка	Вариант использования «Выбор входного видеопотока»
Примечания	-

Продолжение таблицы 21

1	2
Исключения	-
Вывод	Список возможных вариантов загрузки видеопотока
Предусловие	Выбрана настройка видеопотока
Постусловие	-

Таблица 22

Описание операции «Загрузка видеопотока»

Раздел	Описание
Имя	Загрузка видеопотока
Обязанности	Принять от пользователя настройки видеопотока и применить их в системе
Тип	Системная
Ссылка	Вариант использования «Выбор входного видеопотока»
Примечания	-
Исключения	Неверный формат выбранного видеопотока
Вывод	-
Предусловие	Выбрана настройка видеопотока
Постусловие	Начатие обработки видеопотока системой

Диаграмма последовательности для варианта использования «Формирование отчётов» представлена на рис. 11. Описания операций представлены в табл. 23-26.

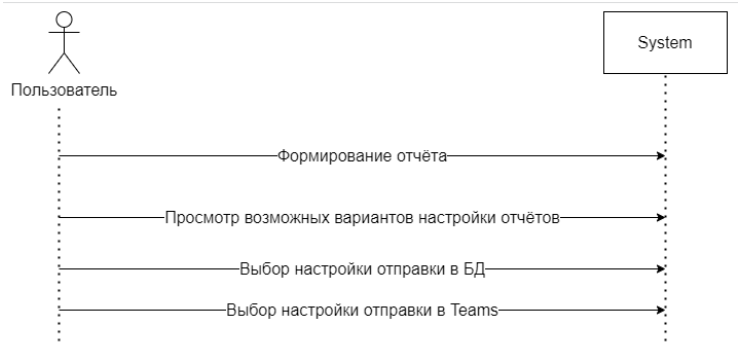


Рис. 11. Диаграмма последовательности для варианта использования «Формирование отчётов»

Таблица 23

Описание операции «Формирование отчёта»

Раздел	Описание
Имя	Формирование отчёта
Обязанности	Предоставить возможность формирования отчёта
Тип	Системная
Ссылка	Вариант использования «Формирование отчётов»
Примечания	-
Исключения	-
Вывод	-
Предусловие	-
Постусловие	-

Таблица 24

Описание операции «Просмотр возможных вариантов настройки отчётов»

Раздел	Описание
Имя	Просмотр возможных вариантов настройки отчётов
Обязанности	Отобразить возможные варианты настройки отчётов
Тип	Системная
Ссылка	Вариант использования «Формирование отчётов»
Примечания	-
Исключения	-
Вывод	Компонент, содержащий возможные настройки отчётов
Предусловие	Открыта настройка формирования отчётов
Постусловие	-

Таблица 25

Описание операции «Выбор настройки отправки в БД»

Раздел	Описание
Имя	Выбор настройки отправки в БД
Обязанности	Предоставить возможность выбора настройки отправки отчёта в базу данных
Тип	Системная
Ссылка	Вариант использования «Формирование отчётов»
Примечания	-
Исключения	Исключение неверной настройки отправки в БД
Вывод	-
Предусловие	Открыта настройка формирования отчётов
Постусловие	Настройка отправки в БД завершена

Таблица 26

Описание операции «Выбор настройки отправки в Teams»

Раздел	Описание
Имя	Выбор настройки отправки в Teams
Обязанности	Предоставить возможность выбора настройки отправки в Teams
Тип	Системная
Ссылка	Вариант использования «Формирование отчётов»
Примечания	-
Исключения	Исключение неверной настройки отправки в Teams
Вывод	-
Предусловие	Открыта настройка формирования отчётов
Постусловие	Настройка отправки в Teams завершена

Диаграмма последовательности для варианта использования «Настроить отправку в БД» представлена на рис. 12. Описания операций представлены в табл. 27-29.

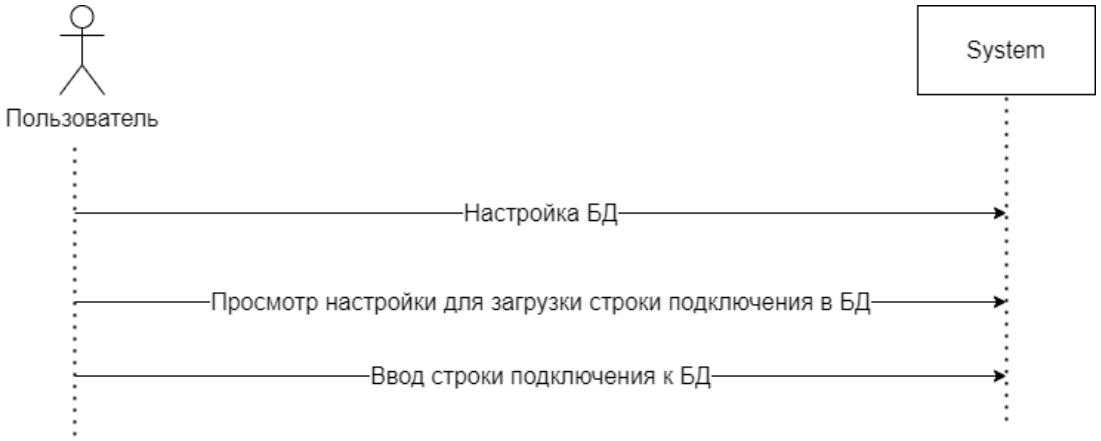


Рис. 12. Диаграмма последовательности для варианта использования «Настроить отправку в БД»

Таблица 27

Описание операции «Настраивание БД»

Раздел	Описание
Имя	Настройка БД
Обязанности	Предоставить возможность настройки базы данных
Тип	Системная
Ссылка	Вариант использования «Настроить отправку в БД»
Примечания	-
Исключения	-
Вывод	-
Предусловие	Открыта настройка отправки в БД
Постусловие	-

Таблица 28

Описание операции «Просмотр настройки для загрузки строки подключения в БД»

Раздел	Описание
1	2
Имя	Просмотр настройки для загрузки строки подключения в БД
Обязанности	Отобразить настройку для загрузки строки подключения к базе данных

Продолжение таблицы 28

1	2
Тип	Системная
Ссылка	Вариант использования «Настроить отправку в БД»
Примечания	-
Исключения	-
Вывод	Возможные варианты настройки параметров для БД
Предусловие	Открыта настройка отправки в БД
Постусловие	-

Таблица 29

Описание операции «Ввод строки подключения к БД»

Раздел	Описание
Имя	Ввод строки подключения к БД
Обязанности	Принять от пользователя строку подключения и применить ее в дальнейшем для отправки отчётов
Тип	Системная
Ссылка	Вариант использования «Настроить отправку в БД»
Примечания	-
Исключения	Исключение неверного формата строки подключения к БД
Вывод	-
Предусловие	Открыта настройка отправки в БД
Постусловие	Настройка отправки в БД завершена

Диаграмма последовательности для варианта использования «Настроить отправку в Teams» представлена на рис. 13. Описания операций представлены в табл. 30-32.

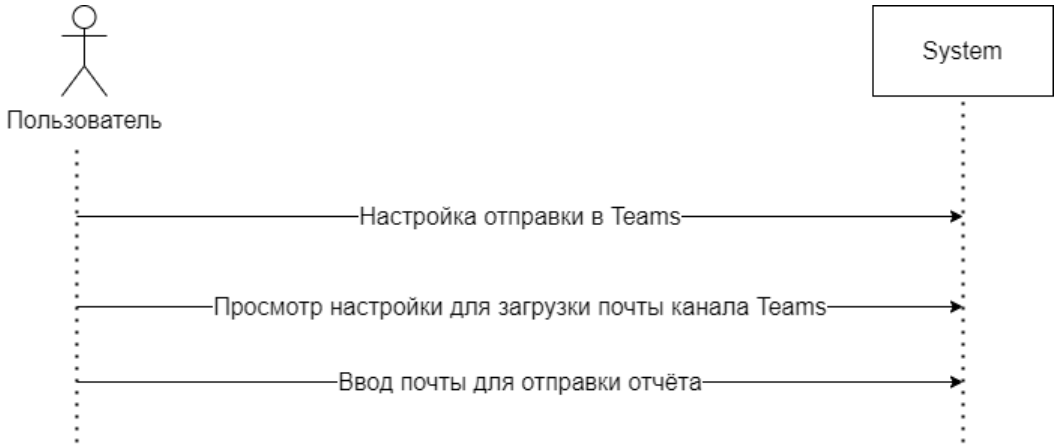


Рис. 13. Диаграмма последовательности для варианта использования «Настроить отправку в Teams»

Таблица 30

Описание операции «Настройка отправки в Teams»

Раздел	Описание
Имя	Настройка отправки в Teams
Обязанности	Предоставить возможность настройки отправки в Teams
Тип	Системная
Ссылка	Вариант использования «Настроить отправку в Teams»
Примечания	-
Исключения	-
Вывод	-
Предусловие	Открыта настройка отправки в Teams
Постусловие	-

Таблица 31

Описание операции «Просмотр настройки для загрузки почты канала Teams»

Раздел	Описание
1	2
Имя	Просмотр настройки для загрузки почты канала Teams
Обязанности	Отобразить настройку для загрузки почты канала Teams
Тип	Системная

Продолжение таблицы 31

1	2
Ссылка	Вариант использования «Настроить отправку в Teams»
Примечания	-
Исключения	-
Вывод	Отображение возможных вариантов настройки отправки в Teams
Предусловие	Открыта настройка отправки в Teams
Постусловие	-

Таблица 32

Описание операции «Ввод почты для отправки отчёта»

Раздел	Описание
Имя	Ввод почты для отправки отчёта
Обязанности	Принять почту канала и применить ее в дальнейшем для отправки отчёта в Teams
Тип	Системная
Ссылка	Вариант использования «Настроить отправку в Teams»
Примечания	-
Исключения	Неверные данные для входа, неверный формат почты канала
Вывод	-
Предусловие	Открыта настройка отправки в Teams
Постусловие	Настройка отправки в Teams завершена

Диаграмма последовательности для варианта использования «Формирование опасной ситуации» представлена на рис. 14. Описания операций представлены в табл. 33-36.

Рис. 14. Диаграмма последовательности для варианта использования «Формирование опасной ситуации»

Таблица 33

Описание операции «Формирование опасной ситуации»

Раздел	Описание
Имя	Формирование опасной ситуации
Обязанности	Предоставить настройку формирования опасной ситуации
Тип	Системная
Ссылка	Вариант использования «Формирование опасной ситуации»
Примечания	-
Исключения	-
Вывод	-
Предусловие	Открыта вкладка детекции
Постусловие	-

Таблица 34

Описание операции «Просмотр настроек формирования опасной ситуации»

Раздел	Описание
Имя	Просмотр настроек формирования опасной ситуации
Обязанности	Отобразить настройку формирования опасной ситуации
Тип	Системная
Ссылка	Вариант использования «Формирование опасной ситуации»
Примечания	-
Исключения	-
Вывод	Отображения конструктора для формирования опасной ситуации
Предусловие	Открыта вкладка детекции
Постусловие	-

Таблица 35

Описание операции «Формирование запросов»

Раздел	Описание
Имя	Формирование запросов
Обязанности	Предоставить возможность формирования запросов
Тип	Системная
Ссылка	Вариант использования «Формирование опасной ситуации»
Примечания	-
Исключения	Неверные параметры при формировании запроса
Вывод	-
Предусловие	Открыта вкладка детекции
Постусловие	Сформированная опасная ситуация

Таблица 36

Описание операции «Формирование зон»

Раздел	Описание
1	2
Имя	Формирование зон

Продолжение таблицы 36

1	2
Обязанности	Предоставить возможность формирования зон
Тип	Системная
Ссылка	Вариант использования «Формирование опасной ситуации»
Примечания	-
Исключения	Неверные параметры при формировании зоны
Вывод	-
Предусловие	Открыта вкладка детекции
Постусловие	Сформированная зона интереса

Диаграмма последовательности для варианта использования «Формирование запросов» представлена на рис. 15. Описания операций представлены в табл. 37-38.

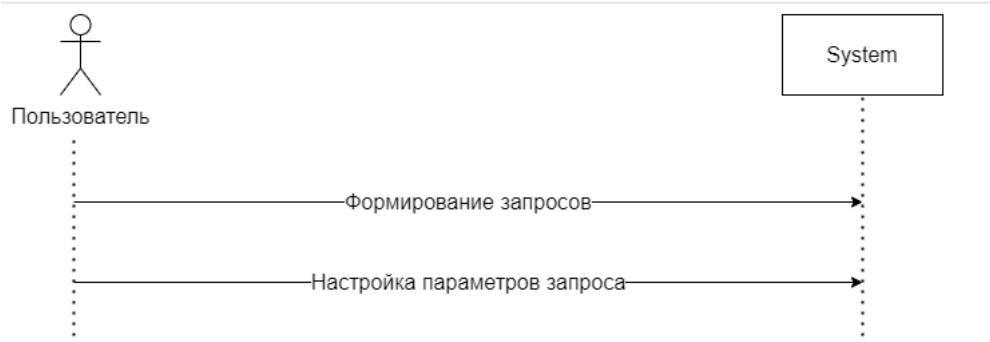


Рис. 15. Диаграмма последовательности для варианта использования «Формирование запросов»

Таблица 37

Описание операции «Формирование запросов»

Раздел	Описание
1	2
Имя	Формирование запросов
Обязанности	Предоставить настройку формирования запросов

Продолжение таблицы 37

1	2
Тип	Системная
Ссылка	Вариант использования «Формирование запросов»
Примечания	-
Исключения	-
Вывод	Отображение возможных вариантов настройки запроса
Предусловие	Открыта вкладка формирования запросов
Постусловие	-

Таблица 38

Описание операции «Настройка параметров запроса»

Раздел	Описание
Имя	Настройка параметров запроса
Обязанности	Принять входные параметра для запроса и использовать их в дальнейшем при формировании опасной ситуации
Тип	Системная
Ссылка	Вариант использования «Формирование запросов»
Примечания	-
Исключения	Выход значения параметров за границы
Вывод	-
Предусловие	Открыта вкладка формирования запросов
Постусловие	Сформированный запрос

Диаграмма последовательности для варианта использования «Выбор видеопотока» представлена на рис. 16. Описания операций представлены в табл. 39-40.

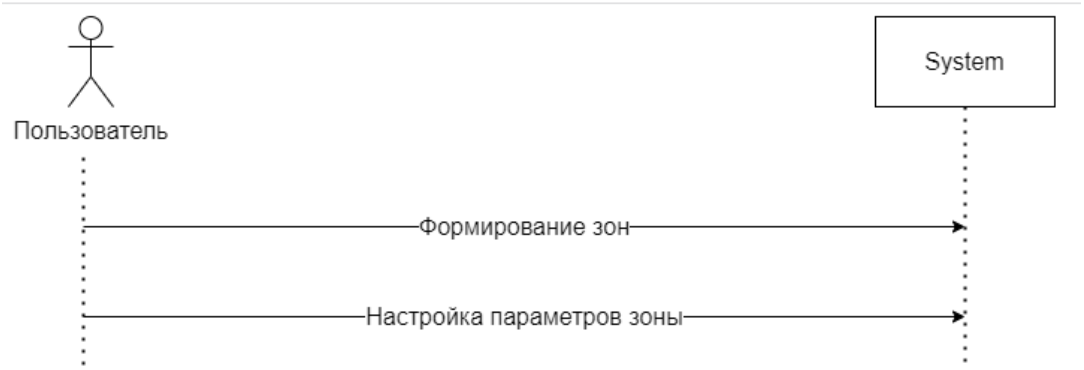


Рис. 16. Диаграмма последовательности для варианта использования «Формирование зон»

Таблица 39

Описание операции «Формирование зон»

Раздел	Описание
Имя	Формирование зон
Обязанности	Предоставить возможность создания зон
Тип	Системная
Ссылка	Вариант использования «Формирование зон»
Примечания	-
Исключения	-
Вывод	Отображение возможных вариантов настройки зоны
Предусловие	Открыта вкладка формирования зон
Постусловие	-

Таблица 40

Описание операции «Настройка параметров зон»

Раздел	Описание
1	2
Имя	Настройка параметров зон
Обязанности	Предоставить возможность редактирования параметров зон и в дальнейшем применения этих параметров для формирования опасной ситуации
Тип	Системная
Ссылка	Вариант использования «Формирование зон»

Продолжение таблицы 41

1	2
Примечания	-
Исключения	Выход значения параметров за границы
Вывод	-
Предусловие	Открыта вкладка формирования зон
Постусловие	Сформированная зона интереса

1.4.4 Построение диаграммы деятельностей варианта использования «Моделирование ситуации»

В зависимости от степени детализации диаграммы деятельностей так же, как диаграммы классов, используют на разных этапах разработки. На этапе анализа требований и уточнения спецификаций диаграммы деятельностей позволяют конкретизировать основные функции разрабатываемого программного обеспечения. Под деятельностью в данном случае понимают задачу (операцию), которую необходимо выполнить вручную или с помощью средств автоматизации. Каждому варианту использования соответствует своя последовательность задач. В теоретическом плане диаграммы деятельности являются обобщенным представлением алгоритма, реализующего анализируемый вариант использования [1].

Последовательность действий пользователя при моделировании опасной ситуации состоит в следующем (рис. 17). Пользователь выбирает запрос, после чего редактирует его параметры. Потом он должен заполнить запрос или уже созданными зонами или создать зону и добавить её в запрос. При просмотре запроса он в реальном времени может изменять его параметры, а также изменять параметры зон, находящихся внутри запрос, до момента пока запрос не будет моделировать опасную ситуацию.

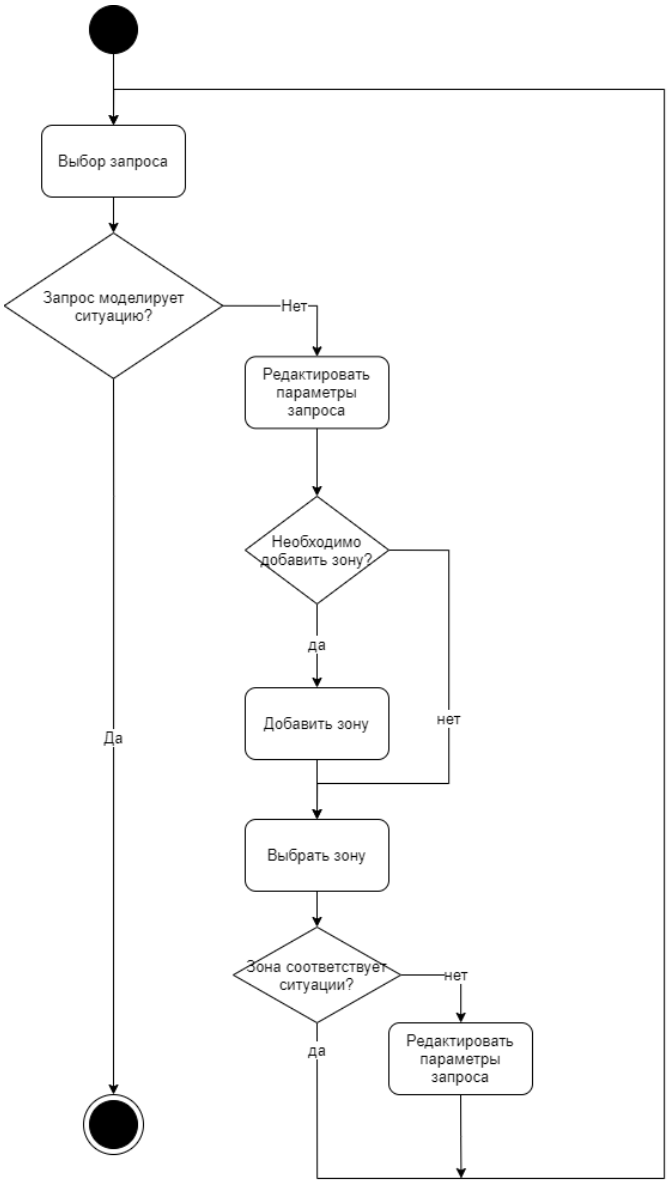


Рис. 17. Диаграмма деятельностей

1.4.5 Построение диаграммы переходов состояния

STD диаграммы используются для моделирования поведения системы при её функционировании во времени. STD позволяют осуществлять декомпозицию управляющих процессов в системе. STD описывают отношения между входными и выходными управляющими потоками на управляющем процессе. STD моделируют последующее функционирование системы на основе ее предыдущего и настоящего функционирования. Данная диаграмма представлена на рис. 18.

Начальное состояние – запуск программы, после запуска система ожидает сигнал о приближении нового проката и при его получении запускает процесс детектирования. После получения кадра, он обрабатывается, на обработанном кадре определяется рабочий. После определения рабочего проверяется его нахождение в зоне в случае, если рабочий вошел в зону, то он проверяется запросом и если запрос обнаружил рабочего, то система отправляет отчёт в БД и Teams. Конечным состоянием является закрытие окна программы.

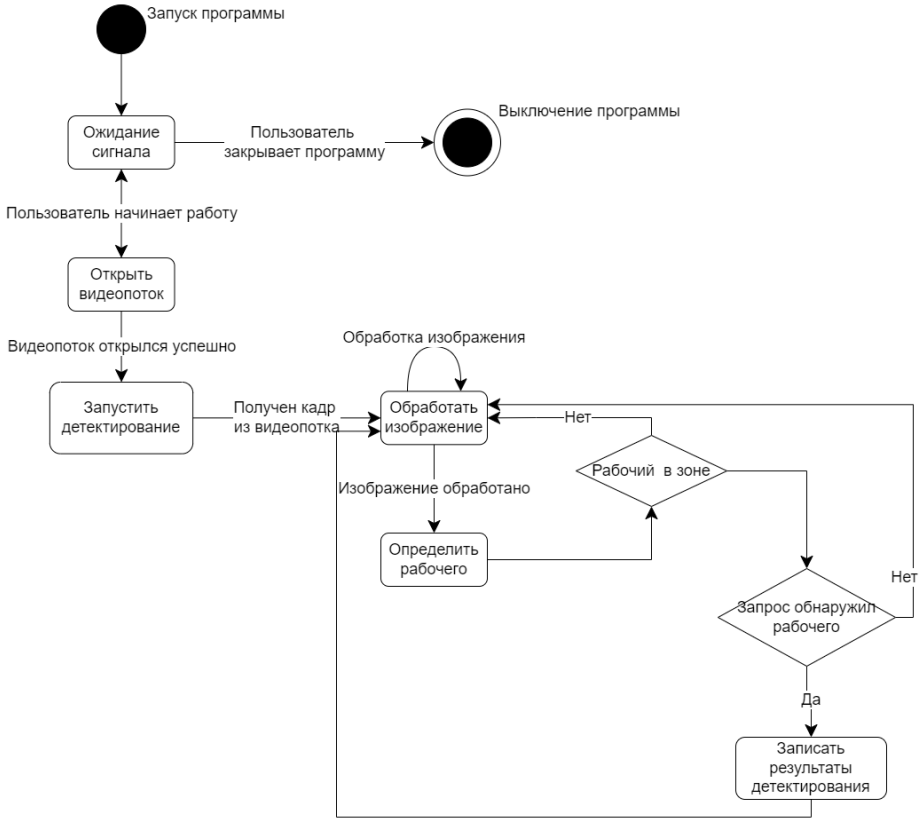


Рис. 18. STD – диаграмма

1.4.6 Построение диаграммы отношений компонентов данных

Практически в любой системе есть данные, которые используются: в работе программы, для сбора статистики, формирования отчетов и другого. Данные нужно где-то хранить, самым удобным способом хранения данных, пожалуй, является база данных.

С помощью CASE-средства ERwin была спроектирована ER-диаграмма (рис. 19), являющаяся описанием схемы объектов рассматриваемой предметной области. В табл. 41 – 42 представлены спецификации таблиц.

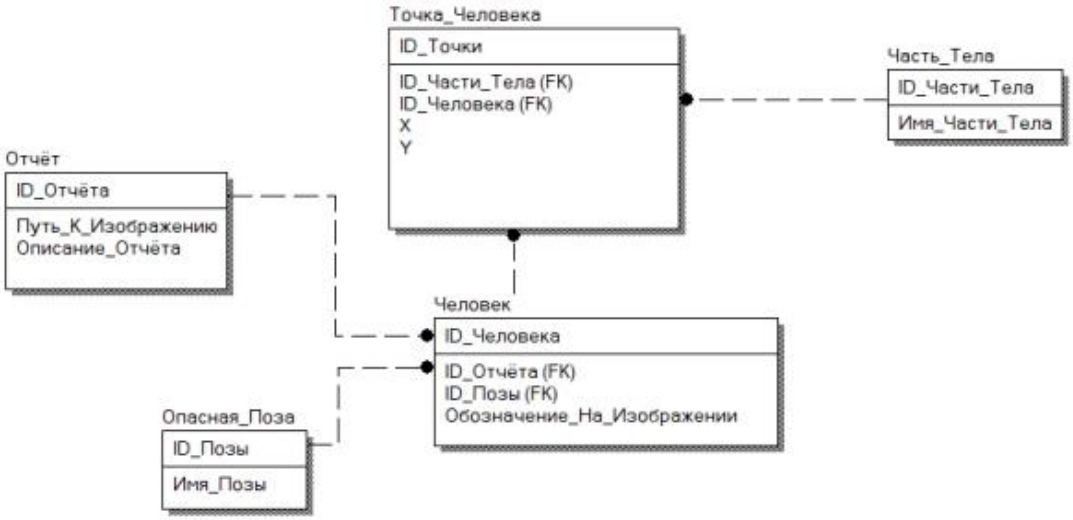


Рис. 19. ER-диаграмма

Таблица 41

Описание сущностей

Сущность	Атрибут	Описание
1	2	3
Отчёт	сущность, характеризующая отчёт о совершенной опасной ситуации	
	ID отчёта	уникальный идентификатор отчёта
	Путь к изображению	путь к файлу, хранящий визуальной подтверждение нарушения
	Описание отчёта	текстовой описание опасной ситуации и людей, нарушивших технику безопасности
Опасная поза	справочник опасных поз людей, которые могут быть обнаружены системой	
	ID позы	уникальный идентификатор опасной позы
	Имя позы	именование опасной позы

Продолжение таблицы 41

1	2	3
Человек	сущность, описывающая человека, совершившего опасное действие	
	ID человека	уникальный идентификатор человека
	ID отчёта	идентификатор отчёта, в котором описано, что именно человек нарушил
	ID позы	идентификатор опасной позы, в ходе которой было составлено обвинение о нарушении техники безопасности
	Обозначение на изображении	обозначение на изображении, фиксирующего опасную ситуацию
Точка человека	сущность, описывающая точку части тела человека на изображении в пикселях	
	ID точки	уникальной идентификатор точки
	ID части тела	идентификатор обнаруженной части тела
	ID человека	идентификатор обнаруженного человека
	X	координата абсцисс, относительно верхнего левого угла изображения
	Y	координата ординат, относительно верхнего левого угла изображения
Часть тела	справочник частей тела, который могут быть обнаружены системой	
	ID части тела	Уникальный идентификатор части тела
	Имя части тела	именование части тела

Таблица 42

Описание полей

Таблица	Поле	Тип данных
1	2	3
Отчёт	ID_Отчёта	Long Integer
	Путь_К_Изображению	Text(40)
	Описание_Отчёта	Text(80)
Опасная поза	ID_Позы	Long Integer
	Имя_Позы	Text(20)

Продолжение таблицы 42

1	2	3
Человек	ID_Человека	Long Integer
	ID_Отчёта	Long Integer
	ID_Позы	Long Integer
	Обозначение_На_Изображении	Text(10)
Точка человека	ID_Точки	Long Integer
	ID_Части_Тела	Long Integer
	ID_Человека	Long Integer
	X	Integer
	Y	Integer
Часть тела	ID_Части_Тела	Long Integer
	Имя_Части_Тела	Text(20)

1.5 Проектирование программного обеспечения

Основной задачей логического проектирования при объектном подходе является разработка классов для реализации объектов, полученных при объектной декомпозиции, что предполагает полное описание полей и методов каждого класса [7].

Физическое проектирование при объектном подходе включает объединение классов и других программных ресурсов в программные компоненты, а также размещение этих компонентов на конкретных вычислительных устройствах [7].

1.5.1 Проектирование структуры системы и построение диаграмм пакетов

Пакетом при объектном подходе называется совокупность описаний классов и других программных ресурсов, в том числе и самих пакетов. Объединение в пакеты используют для удобства создания больших проектов, количество классов в которых велико. При этом в один пакет обычно собирают классы и другие ресурсы одинакового назначения [1].

Диаграмма пакетов показывает, из каких частей состоит проектируемая программная система, и как эти части связаны друг с другом.

Диаграмма пакетов представлена на рис. 20. Описание пакетов представлено в табл. 43.

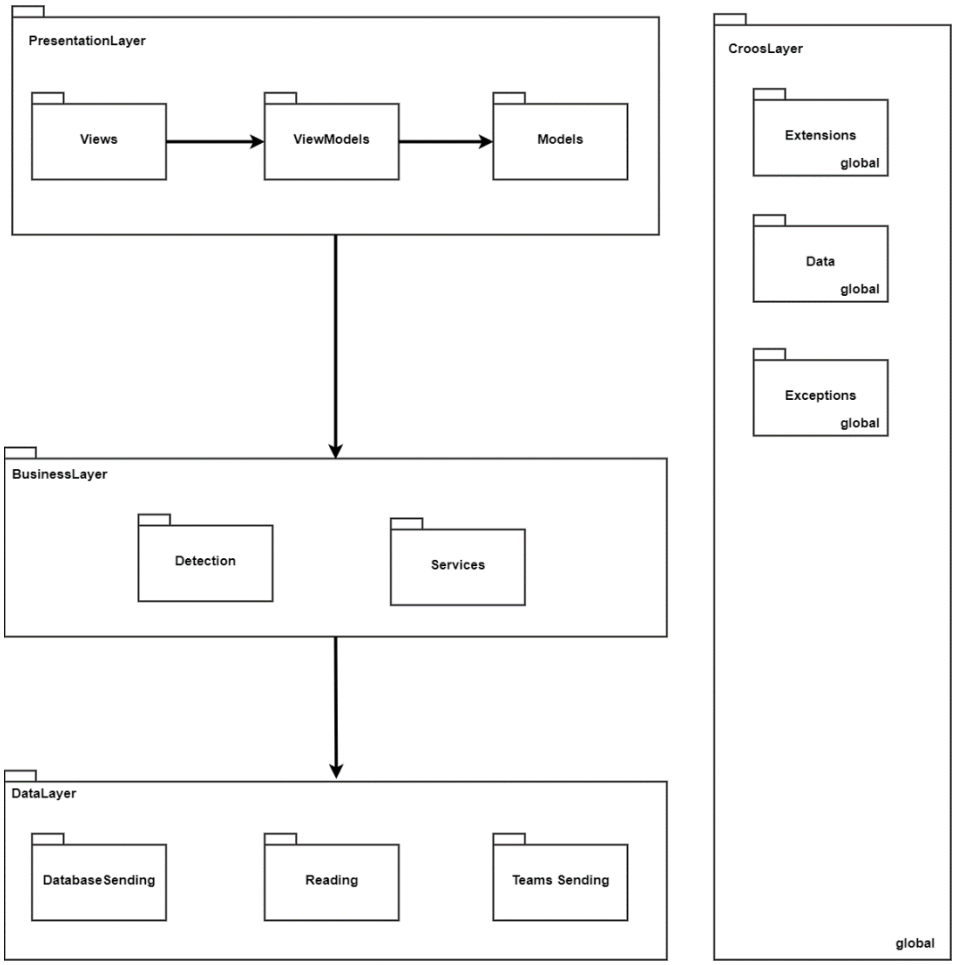


Рис. 20. Диаграмма пакетов

Таблица 43

Описание пакетов

Пакет	Описание
1	2
Views	пакет для классов представления
ViewModels	пакет для классов моделей представления
Models	пакет для классов моделей
Detection	пакет для классов, содержащих логику обнаружения людей

Продолжение таблицы 43

1	2
Services	пакет для классов сервисов
Database Sending	пакет для классов, связанных с базой данных
Reading	пакет для классов, связанных с чтением видеопотока
Teams Sending	пакет для отправки данных в Teams
Extensions	пакет с методами расширения
Data	пакет с общими данными
Exceptions	пакет с классами исключениями

1.5.2 Проектирование классов в пакетах

После определения основных пакетов разрабатываемого программного обеспечения переходят к детальному проектированию классов, входящих в каждый пакет.

Классы-кандидаты, которые предположительно должны войти в конкретный пакет, показывают на диаграмме классов этапа проектирования и уточняют отношения между объектами указанных классов [1].

1.5.2.1 Проектирование классов пакета «Views»

1.5.2.1.1 Исходная диаграмма классов

Исходная диаграмма классов пакета «Views» представлена на рис. 21, а её описание в табл. 44. Данный пакет является библиотекой интерфейсных элементов разметки, в котором основной компонент Shell содержит в себе разметку всего приложения, разделенного на разные классы.

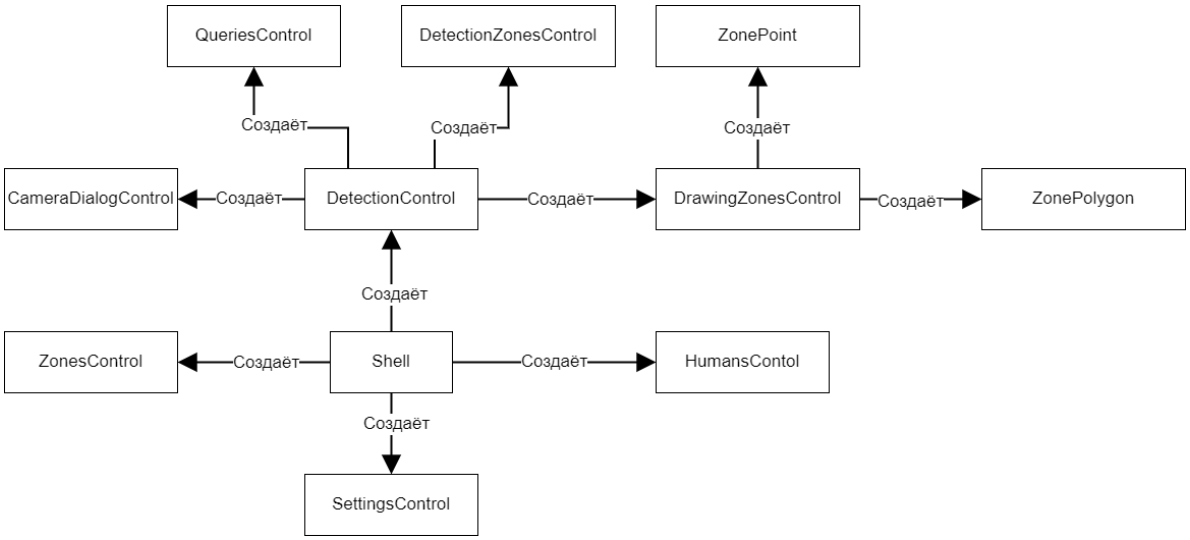


Рис. 21. Исходная диаграмма классов пакета «Views»

Таблица 44

Описание классов пакета «Views»

Класс	Описание
Shell	класс, характеризующий основное окно приложения
DetectionControl	класс, характеризующий окно детекции
QueriesControl	класс, характеризующий форму запросов
DetectionZonesControl	класс, характеризующий форму зон в окне детекции
CameraDialogControl	класс, характеризующий диалог выбора камеры
DrawingZonesControl	класс, характеризующий форму отображения опасных зон
ZoneEllipse	класс, характеризующий визуальную точку зоны
ZonePolygon	класс, характеризующий визуальную область зоны
ZonesControl	класс, характеризующий окно зон
HumansControl	класс, характеризующий окно людей
SettingsControl	класс, характеризующий окно настроек

1.5.2.1.2 Уточнённая диаграмма классов

Уточнённая диаграмма классов пакета «Views» представлена на рис. 22. Данный пакет содержит в себе основной класс Shell, который связан с другими классами с помощью композиции.

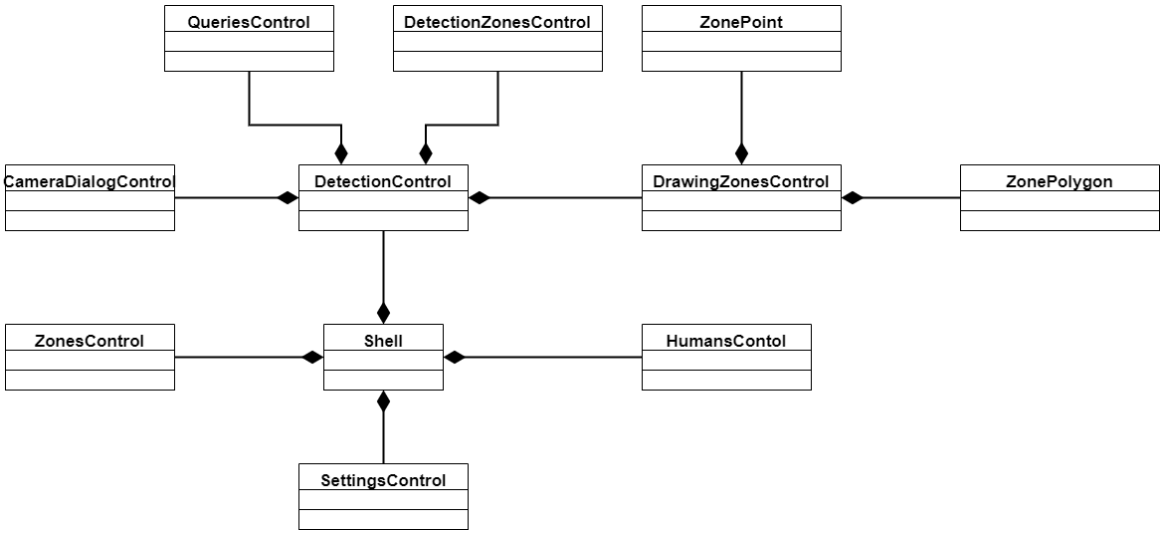


Рис. 22. Уточнённая диаграмма классов пакета «Views»

1.5.2.1.3 Детальная диаграмма классов

Детальная диаграмма классов пакета «Views» представлена на рис. 23. По умолчанию в WPF, у интерфейсных элементов нет никаких методов, а есть лишь одно поле – Content. Описание полей классов пакета «View» (табл. 45-56).

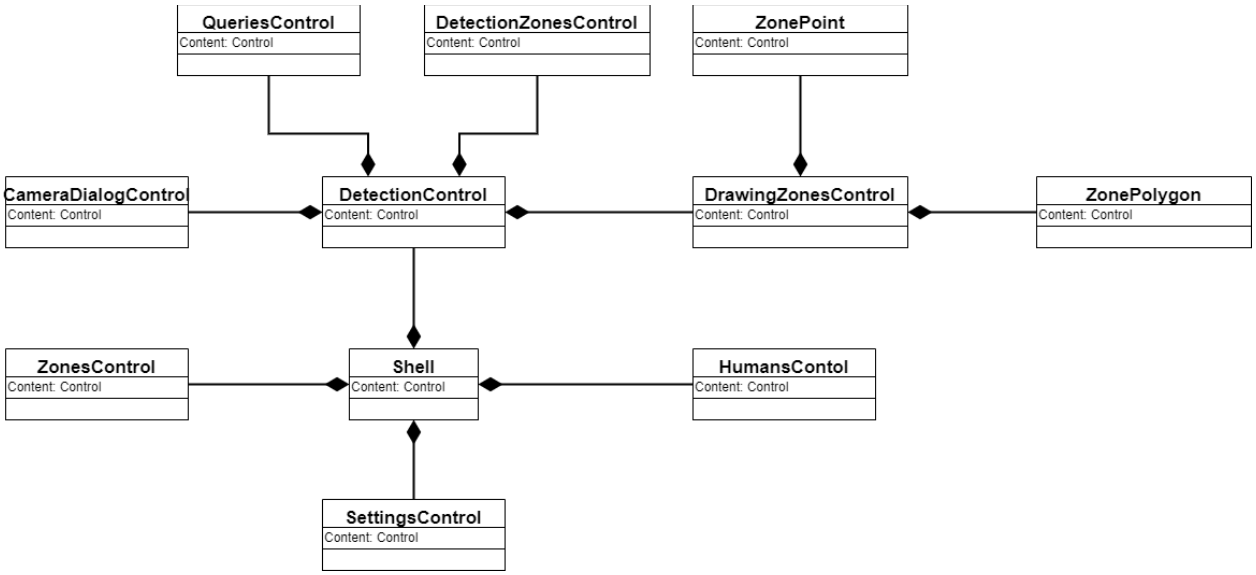


Рис. 23. Детальная диаграмма классов пакета «Views»

Таблица 45

Описание полей класса «Shell»

Название	Тип	Описание
Content	Control	Содержимое основного окна

Таблица 46

Описание полей класса «SettingsControl»

Название	Тип	Описание
Content	Control	Содержимое окна настроек

Таблица 47

Описание полей класса «HumansControl»

Название	Тип	Описание
Content	Control	Содержимое окна с людьми

Таблица 48

Описание полей класса «ZonesControl»

Название	Тип	Описание
Content	Control	Содержимое окна с зонами

Таблица 49

Описание полей класса «DetectionControl»

Название	Тип	Описание
Content	Control	Содержимое основного окна детекции

Таблица 50

Описание полей класса «CameraDialogControl»

Название	Тип	Описание
Content	Control	Содержимое диалогового окна выбора камеры

Таблица 51

Описание полей класса «QueriesControl»

Название	Тип	Описание
Content	Control	Содержимое окна запросов

Таблица 52

Описание полей класса «DetectionZonesControl»

Название	Тип	Описание
Content	Control	Содержимое окна зон в окне детекции

Таблица 53

Описание полей класса «DrawingZonesControl»

Название	Тип	Описание
Content	Control	Содержимое основного окна отрисовки зон

Таблица 54

Описание полей класса «ZonePoint»

Название	Тип	Описание
Content	Control	Содержимое отрисовки точки зоны

Таблица 55

Описание полей класса «ZonePolygon»

Название	Тип	Описание
Content	Control	Содержимое отрисовки области зоны

1.5.2.2 Проектирование классов пакета «ViewModels»

1.5.2.2.1 Исходная диаграмма классов

Исходная диаграмма классов пакета «ViewModels» представлена на рис. 24, а её описание в табл. 56. Данный пакет является библиотекой

«Вьюмоделей», в котором классы никак друг с другом не связаны, они нужны для привязки функционала к классам пакета View.

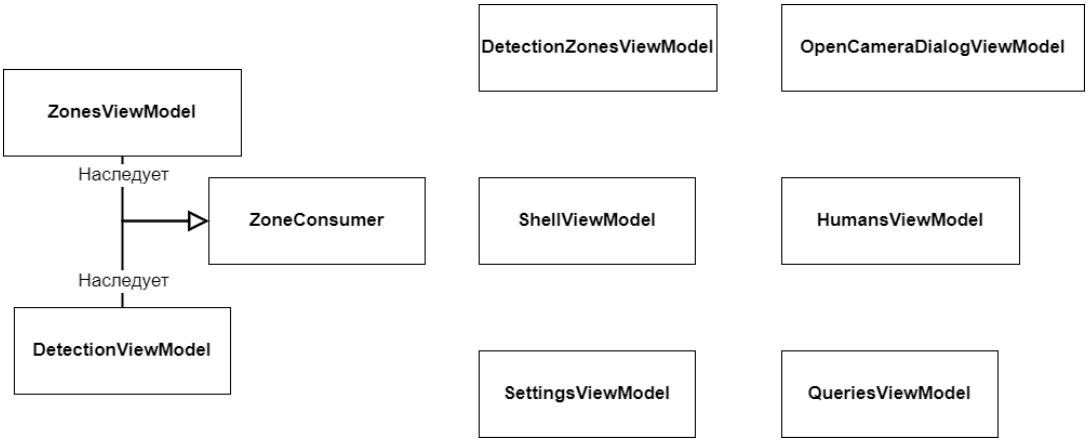


Рис. 24. Исходная диаграмма классов пакета «ViewModels»

Таблица 56

Описание классов пакета «ViewModels»

Класс	Описание
ZonesViewModel	класс, характеризующий модель представления зон
ZoneConsumer	класс, характеризующий модель представления для класса, которому нужно знать о зонах
DetectionViewModel	класс, характеризующий модель представления детекции
DetectionZonesViewModel	класс, характеризующий модель представления зон в детекции
ShellViewModel	класс, характеризующий основную модель представления
SettingsViewModel	класс, характеризующий модель представления настроек
OpenCameraDialogViewModel	класс, характеризующий модель представления для диалога выбора камеры
HumansViewModel	класс, характеризующий модель представления людей
QueriesViewModel	класс, характеризующий модель представления запросов

1.5.2.2.2 Диаграмма последовательностей взаимодействия объектов классов

На рис. 25-26 представлены диаграммы последовательностей взаимодействия объектов классов пакета «ViewModels». Так как пользователь не учувствует в общении объектов данных классов, то нельзя построить диаграмму последовательности действий при прерывании пользователем. Как можно заметит, на диаграмме взаимодействий всего 4 объекта, это связано с тем, что остальные объекты данных классов никак не взаимодействуют друг с другом.

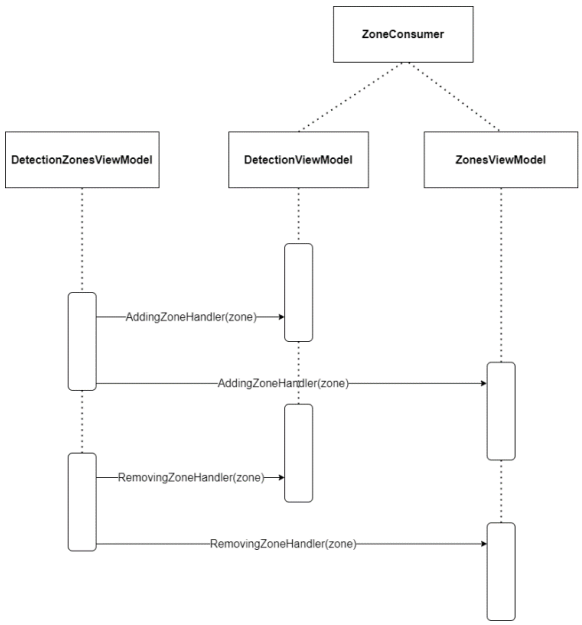


Рис. 25. Диаграмма последовательности действий классов пакета «ViewModels». Нормальный ход событий

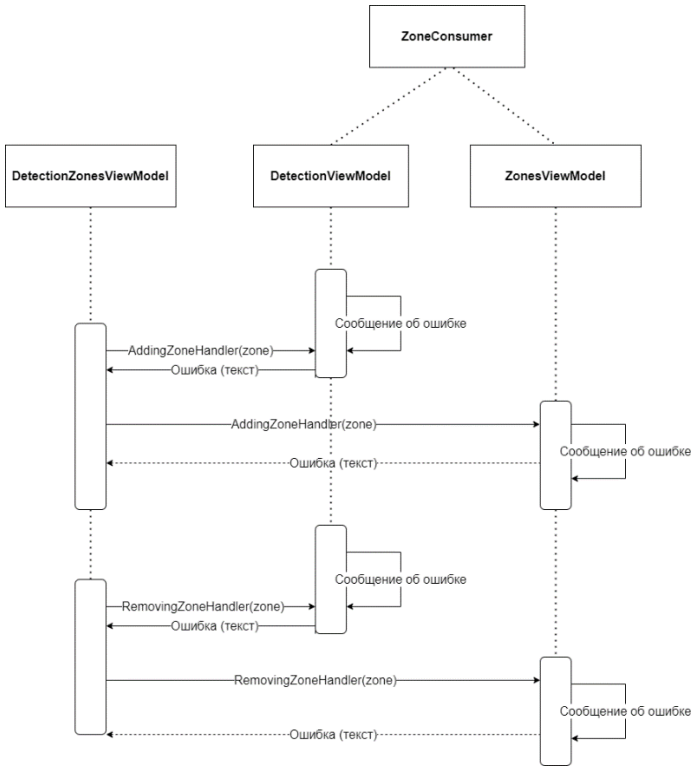


Рис. 26. Диаграмма последовательности действий классов пакета «ViewModels». Прерывание процесса системой

1.5.2.2.3 Уточнённая диаграмма классов

Уточнённая диаграмма классов пакета «ViewModels» представлена на рис. 27. Исходя из методологии паттерна MVVM [5], у «Вьюмоделей» не должно быть никаких открытых методов, поэтому уточненная диаграмма совпадает с исходной.

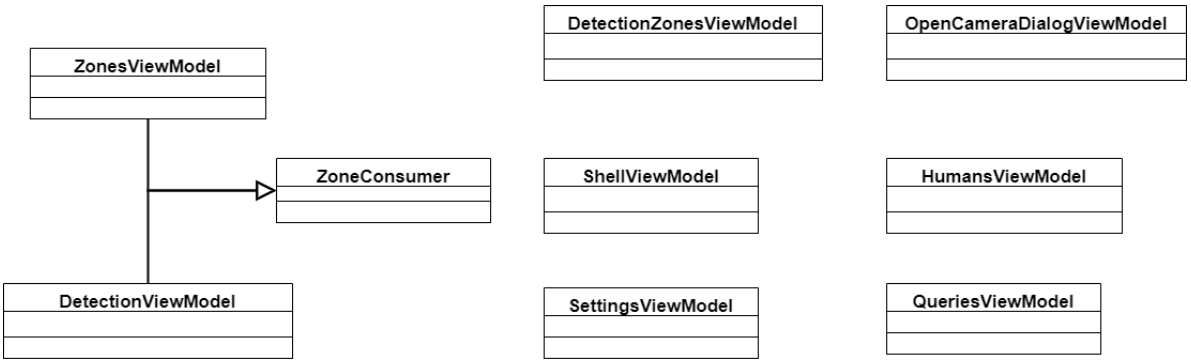


Рис. 27. Уточнённая диаграмма классов пакета «ViewModels»

1.5.2.2.4 Детальная диаграмма классов

Детальная диаграмма классов пакета «ViewModels» представлена на рис. 28. Описание полей и методов классов пакета «ViewModels» (табл. 57-64).

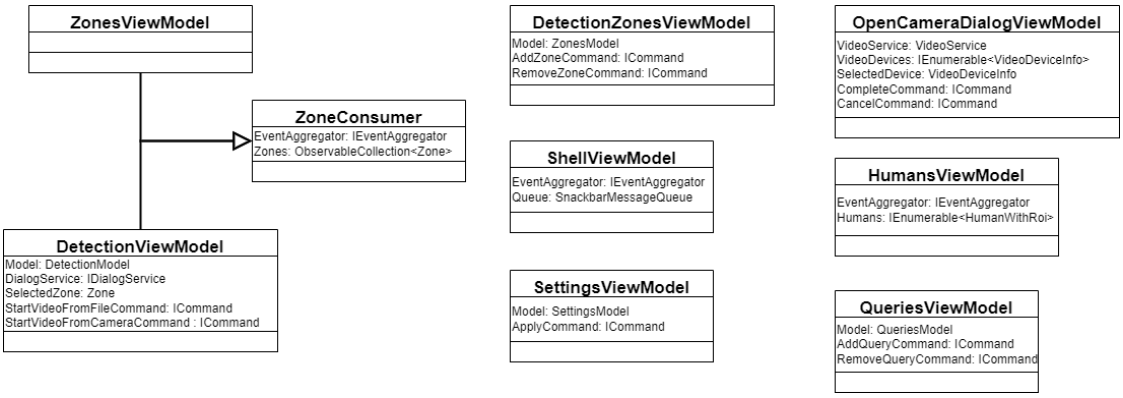


Рис. 28. Детальная диаграмма классов пакета «ViewModels»

Таблица 57

Описание полей класса «ZoneConsumer»

Название	Тип	Описание
EventAggregator	IEventAggregator	Межмодульная шина данных
Zones	ObservableCollection<Zone>	Список зон

Таблица 58

Описание полей класса «DetectionViewModel»

Название	Тип	Описание
Model	DetectionModel	Модель для детекции
DialogService	IDialogService	Сервис для открытия диалога
SelectedZone	Zone	Выбранная зона
StartVideoFromFileCommand	ICommand	Команда для открытия видео из файла
StartVideoFromCameraCommand	ICommand	Команда для открытия видео с камеры

Таблица 59

Описание полей класса «DetectionZonesViewModel»

Название	Тип	Описание
Model	ZonesModel	Модель для зон
AddZoneCommand	ICommand	Команда добавления зоны
RemoveZoneCommand	ICommand	Команда для удаления зоны

Таблица 60

Описание полей класса «ShellViewModel»

Название	Тип	Описание
EventAggregator	IEventAggregator	Межмодульная шина данных
Queue	SnackbarMessageQueue	Очередь сообщений для вывода

Таблица 61

Описание полей класса «SettingsViewModel»

Название	Тип	Описание
Model	SettingsModel	Модель настроек
ApplyCommand	ICommand	Команда для применения настроек

Таблица 62

Описание полей класса «OpenCameraDialogViewModel»

Название	Тип	Описание
VideoService	VideoService	Сервис для работы с видео
VideoDevices	IEnumerable<VideoDeviceInfo>	Список доступных камер
SelectedDevice	VideoDeviceInfo	Выбранная камера
CompleteCommand	ICommand	Команда для успешного закрытия диалога
CancelCommand	ICommand	Команда для отмены диалога

Таблица 63

Описание полей класса «HumansViewModel»

Название	Тип	Описание
EventAggregator	IEventAggregator	Межмодульная шина данных
Humans	IEnumerable<HumanWithRoi>	Список людей

Таблица 64

Описание полей класса «QueriesViewModel»

Название	Тип	Описание
Model	QueriesModel	Модель для запросов
AddQueryCommand	ICommand	Команда для добавления запроса
RemoveQueryCommand	ICommand	Команда для удаления запроса

1.5.2.3 Проектирование классов пакета «Models»

1.5.2.3.1 Исходная диаграмма классов

Исходная диаграмма классов пакета «Models» представлена на рис. 29, а её описание в табл. 65. Данный пакет является библиотекой для моделей элементов, в нём расписана основная логика взаимодействия в приложении.

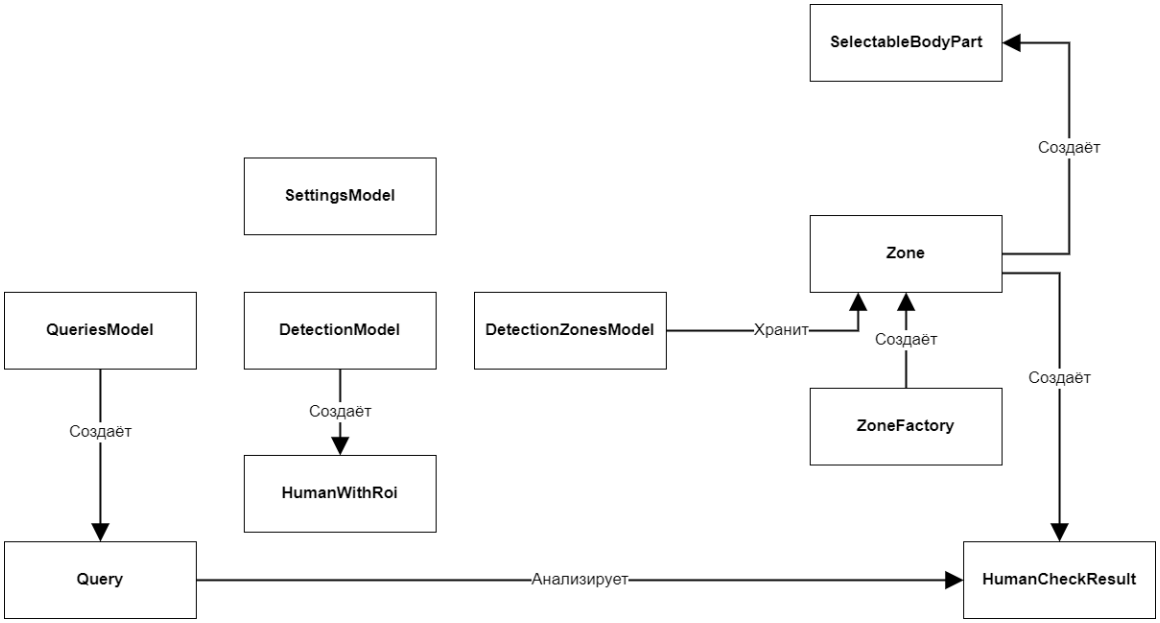


Рис. 29. Исходная диаграмма классов пакета «Models»

Таблица 65

Описание классов пакета «Models»

Класс	Описание
DetectionModel	класс, характеризующий модель детекции
HumanWithRoi	класс, описывающий человека с его областью
SettingsModel	класс, характеризующий модель настроек
QueriesModel	класс, характеризующий модель запросов
Query	класс, описывающий запрос
DetectionZonesModel	класс, характеризующий модель зон в детекции
Zone	класс, описывающий зону
ZoneFactory	класс, описывающий логику создания новой зоны
SelectableBodyPart	класс, характеризующий выбранную часть тела
HumansCheckResult	класс, описывающий результат проверки человека

1.5.2.3.2 Диаграмма последовательностей взаимодействия объектов классов

На рис. 30-32 представлены диаграммы последовательностей взаимодействия объектов классов пакета «Models».

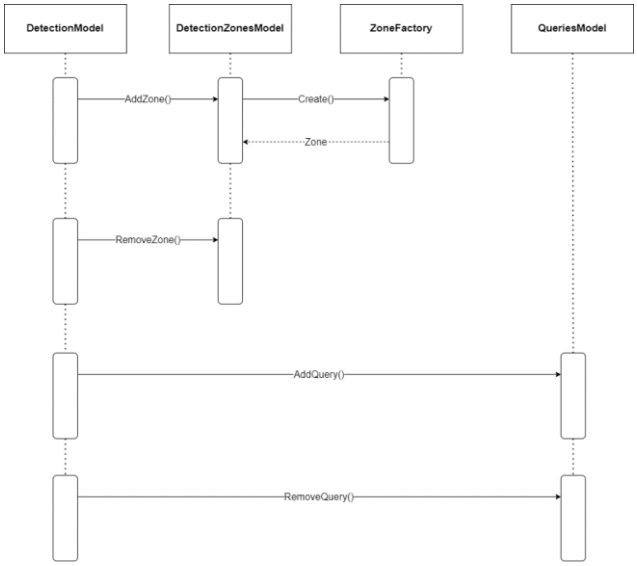


Рис. 30. Диаграмма последовательности действий классов пакета «Models». Нормальный ход событий

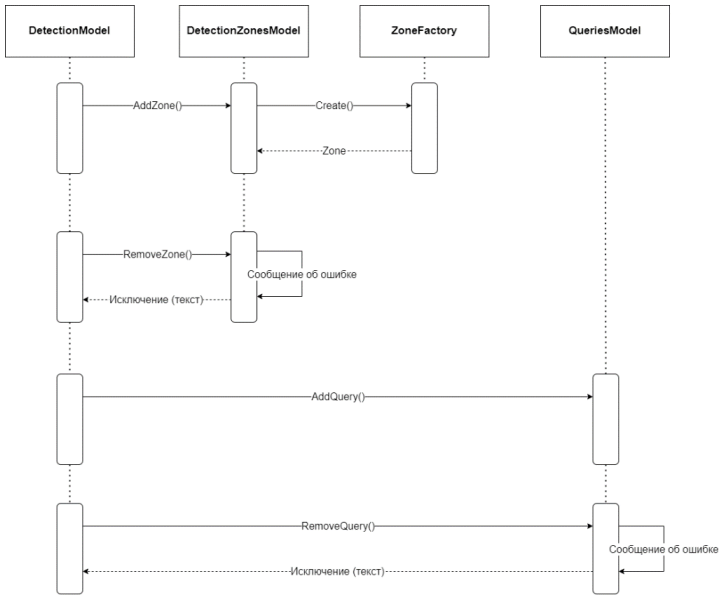


Рис. 31. Диаграмма последовательности действий классов пакета «Models». Прерывание процесса системой

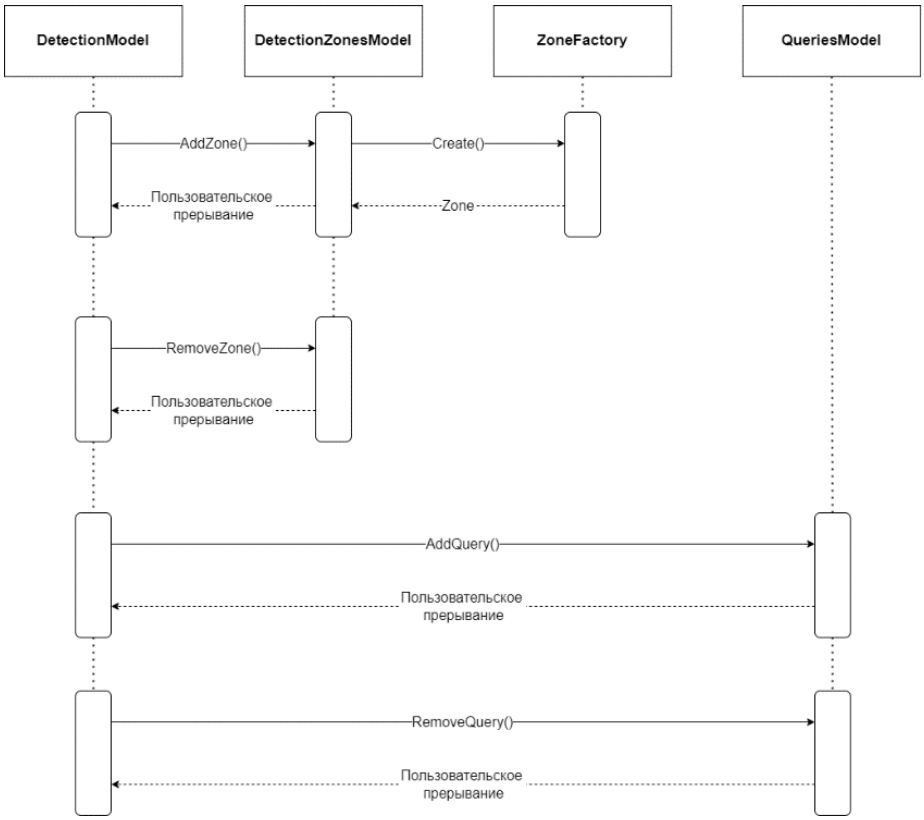


Рис. 32. Диаграмма последовательности действий классов пакета «Models». Прерывание процесса пользователем

1.5.2.3.3 Уточнённая диаграмма классов

Уточнённая диаграмма классов пакета «Models» представлена на рис. 33.

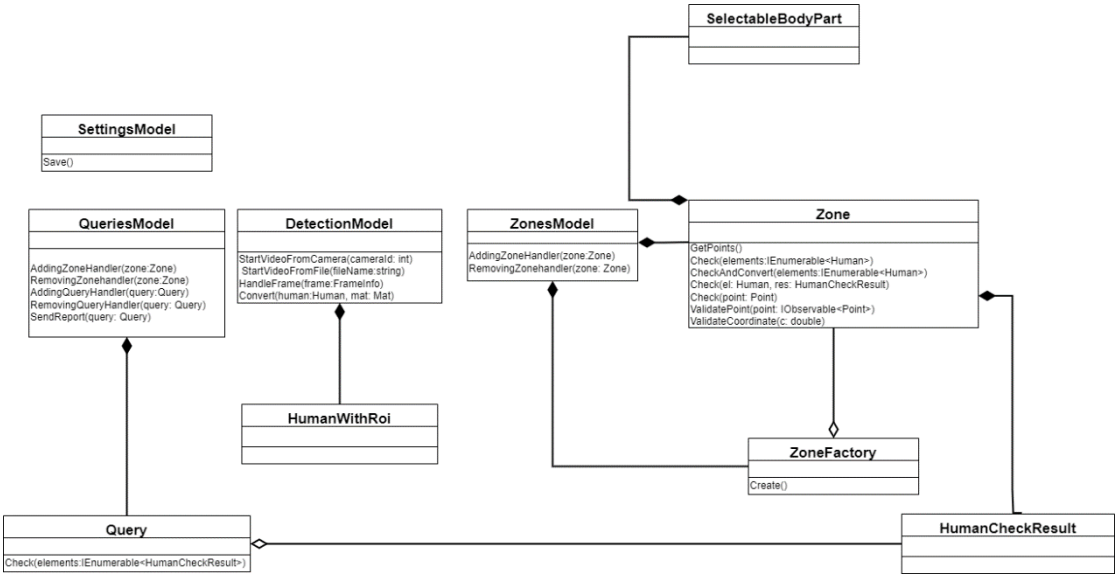


Рис. 33. Уточнённая диаграмма классов пакета «Models»

1.5.2.3.4 Детальная диаграмма классов

Детальная диаграмма классов пакета «Models» представлена на рис. 34. Описание полей и методов классов пакета «Models» (табл. 66-82).

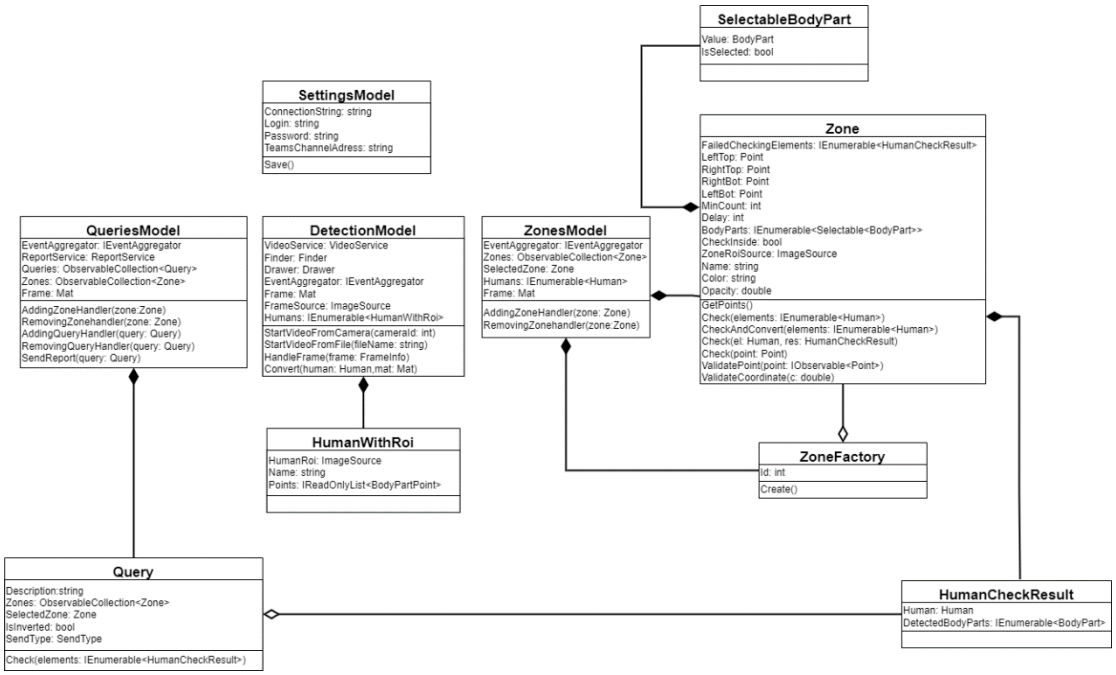


Рис. 34. Детальная диаграмма классов пакета «Models»

Таблица 66

Описание полей класса «SettingsModel»

Название	Тип	Описание
ConnectionString	string	Строка подключения к БД
Login	string	Логин для почты
Password	string	Пароль для почты
TeamsChannelAdress	string	Почта канала

Таблица 67

Описание методов класса «SettingsModel»

Название	Параметры	Возвращаемое значение	Описание
Save	-	-	Метод для сохранения настроек

Таблица 68

Описание полей класса «DetectionModel»

Название	Тип	Описание
VideoService	VideoService	Видео сервис
Finder	Finder	Конечный обработчик нейронной сети
Drawer	Drawer	Объект, для отрисовки людей на кадре
EventAggregator	IEventAggregato	Межмодульная шина данных
Frame	Mat	Текущий кадр
FrameSource	ImageSource	Объект для отображения текущего кадра
Humans	IEnumerable<HumanWithRoi>	Найденные люди

Таблица 69

Описание методов класса «DetectionModel»

Название	Параметры	Возвращаемое значение	Описание
StartVideoFromCamera	cameraId: int	-	Метод для открытия видеопотока с камеры
StartVideoFromFile	fileName: string	-	Метод для открытия видеопотока с файла
HandleFrame	frame: FrameInfo	-	Метод обработки кадров
Convert	human: Human, mat: Mat	HumanWithRoi	Метод конвертирования человека

Таблица 70

Описание полей класса «Query»

Название	Тип	Описание
Description	string	Описание
Zones	ObservableCollection<Zone>	Зоны запроса
SelectedZone	Zone	Выбранная зона
IsInverted	bool	Инвертированность запроса
SendType	SendType	Тип отправки отчёта

Таблица 71

Описание методов класса «Query»

Название	Параметры	Возвращаемое значение	Описание
Check	elements: IEnumerable<HumanCheckResult>	-	Проверка людей на подтверждение запроса

Таблица 72

Описание полей класса «HumanWithRoi»

Название	Тип	Описание
HumanRoi	ImageSource	Часть кадра, в которой находится человек
Name	string	Отображаемое имя
Points	IReadOnlyList<BodyPartPoint>	Найденные точки частей тела

Таблица 73

Описание полей класса «QueriesModel»

Название	Тип	Описание
EventAggregator	IEventAggregator	Межмодульная шина данных
ReportService	ReportService	Сервис для отправки отчётов
Queries	ObservableCollection<Query>	Общий список запросов
Zones	ObservableCollection<Zone>	Общий список зон
Frame	Mat	Текущий кадр

Таблица 74

Описание методов класса «QueriesModel»

Название	Параметры	Возвращаемое значение	Описание
AddingZoneHandler	zone: Zone	-	Обработчик добавления зоны
RemovingZonehandler	zone: Zone	-	Обработчик удаления зоны
AddingQueryHandler	query: Query	-	Обработчик добавления запроса
RemovingQueryHandler	query: Query	-	Обработчик удаления запроса
SendReport	query: Query	-	Метод отправки запроса

Таблица 75

Описание полей класса «ZonesModel»

Название	Тип	Описание
EventAggregator	IEventAggregator	Межмодульная шина данных
Zones	ObservableCollection<Zone>	Общий список зон
SelectedZone	Zone	Выбранная зона
Humans	IEnumerable<Human>	Список людей
Frame	Mat	Текущий кадр

Таблица 76

Описание методов класса «ZonesModel»

Название	Параметры	Возвращаемое значение	Описание
AddingZoneHandler	zone: Zone	-	Обработчик добавления зоны
RemovingZonehandler	zone: Zone	-	Обработчик удаления зоны

Таблица 77

Описание полей класса «Zone»

Название	Тип	Описание
FailedCheckingElements	IEnumerable<HumanCheckResult>	Найденные зоной люди
LeftTop	Point	Левая верхняя точка зоны
RightTop	Point	Правая верхняя точка зоны
RightBot	Point	Правая нижняя точка зоны
LeftBot	Point	Левая нижняя точка зоны
MinCount	int	Минимальное количество людей, которое нужно для обнаружения
Delay	int	Минимальное время, после прохождения которого, зона обнаружит людей
BodyParts	IEnumerable<Selectable>	Список частей тела, которые обнаруживает зоны
CheckInside	bool	Флаг о том, что обнаружение идёт внутри зоны
ZoneRoiSource	ImageSource	Часть кадра, находящаяся внутри точек зоны
Name	string	Имя зоны
Color	string	Цвет зоны
Opacity	double	Прозрачность зоны

Таблица 78

Описание методов класса «Zone»

Название	Параметры	Возвращаемое значение	Описание
1	2	3	4
GetPoints	-	IEnumerable<Point>	Метод получения точек зоны в виде списка
Check	elements: IEnumerable<Human>	-	Метод проверки людей

Продолжение таблицы 78

1	2	3	4
CheckAndConvert	elements: IEnumerable<Human>	IEnumerable<HumanCheckResult>	Метод получения людей, которые находятся в зоне
Check	el: Human, res: HumanCheckResult	bool	Метод проверки человека на нахождение зоне
Check	point: Point	bool	Метод проверки точки на нахождение зоне
ValidatePoint	point: IObservable<Point>	IObservable<Point>	Метод валидации точки
ValidateCoordinate	c: double	double	Метод валидации координаты

Таблица 79

Описание полей класса «SelectableBodyPart»

Название	Тип	Описание
Value	BodyPart	Часть тела
IsSelected	bool	Флаг того, что элемент выбран

Таблица 80

Описание полей класса «ZoneFactory»

Название	Тип	Описание
Id	int	Идентификатор зоны

Таблица 81

Описание методов класса «ZoneFactory»

Название	Параметры	Возвращаемое значение	Описание
Create	-	Zone	

Таблица 82

Описание полей класса «HumanCheckResult»

Название	Тип	Описание
Human	Human	Человек
DetectedBodyParts	IEnumerable<BodyPart>	Обнаруженные зоной части тела

1.5.2.4 Проектирование классов пакета «Detection»

1.5.2.4.1 Исходная диаграмма классов

Исходная диаграмма классов пакета «Detection» представлена на рис. 35, а её описание в табл. 83. Данный пакет отвечает за обнаружение частей тела людей и за возможную настройку нейронной сети.

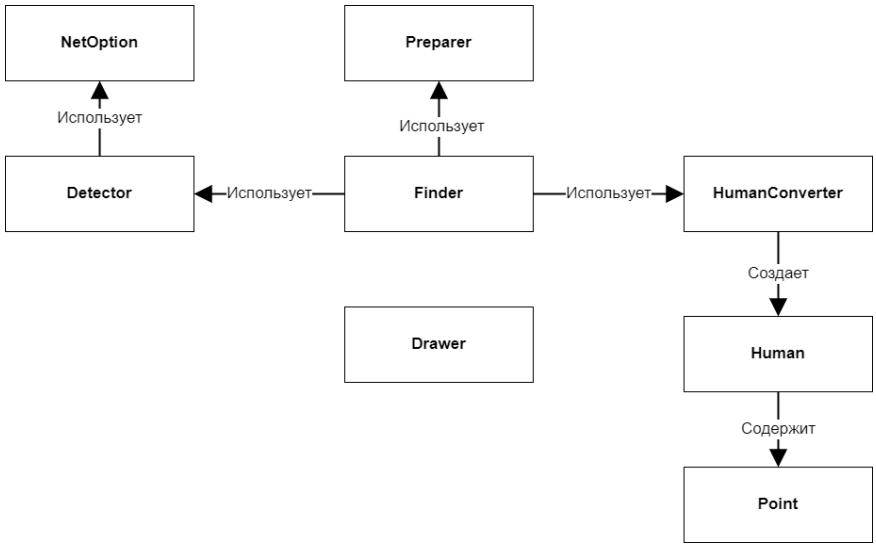


Рис. 35. Исходная диаграмма классов пакета «Detection»

Таблица 83

Описание классов пакета «Detection»

Класс	Описание
NetOption	класс, описывающий конфигурацию нейронной сети
Detector	класс, необходимый для детектирования точек частей тела
Finder	класс, необходимый для детектирования людей
Drawer	класс, отвечающий за логику отрисовки людей на кадре
Preparer	класс, отвечающий за логику преобразования найденных точек частей тела
HumansConverter	класс, отвечающий за логику преобразования найденных точек частей тела в людей
Human	класс, описывающий обнаруженного человека
Point	класс, описывающий точку с относительными координатами

1.5.2.4.2 Диаграмма последовательностей взаимодействия объектов классов

На рис. 36-38 представлены диаграммы последовательностей взаимодействия объектов классов пакета «Detection». При прерывании процесса системой, происходит обработка исключений, а при прерывании пользователем – процесс детекции останавливается и не возникает выброса исключений.

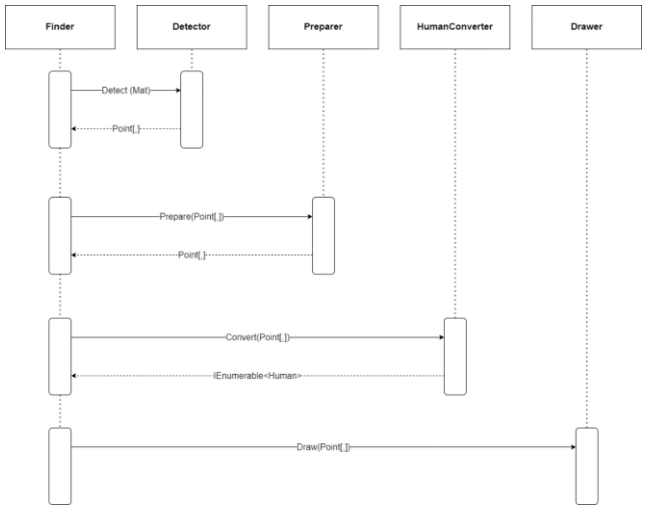


Рис. 36. Диаграмма последовательности действий классов пакета «Detection». Нормальный ход событий

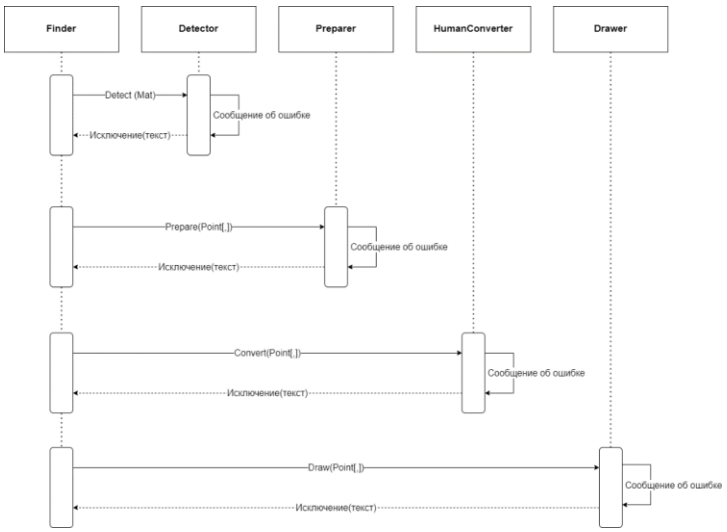


Рис. 37. Диаграмма последовательности действий классов пакета «Detection». Прерывание процесса системой

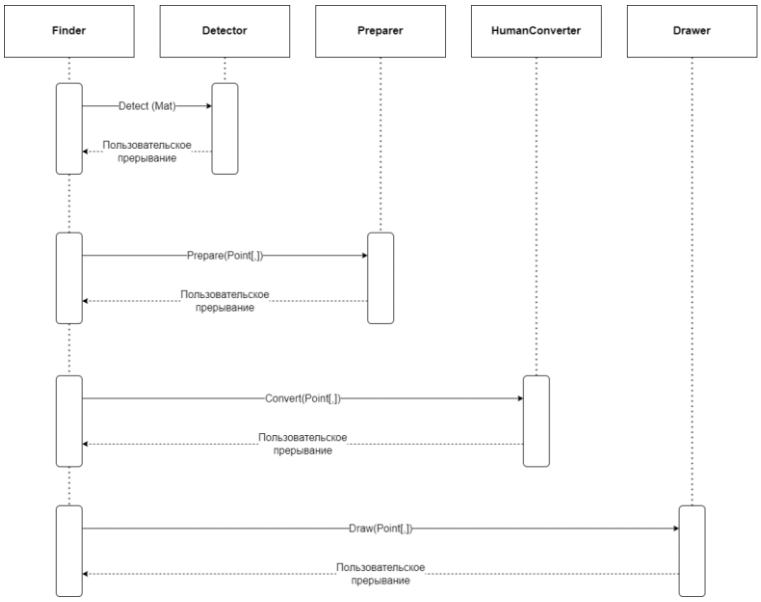


Рис. 38. Диаграмма последовательности действий классов пакета «Detection». Прерывание процесса пользователем

1.5.2.4.3 Уточнённая диаграмма классов

Уточнённая диаграмма классов пакета «Detection» представлена на рис. 39. Данный пакет основан на основном классе Finder, который связан с другими классами с помощью композиции и лишь Detector связан с настройками нейронной сети (классом NetOption) с помощью агрегации.

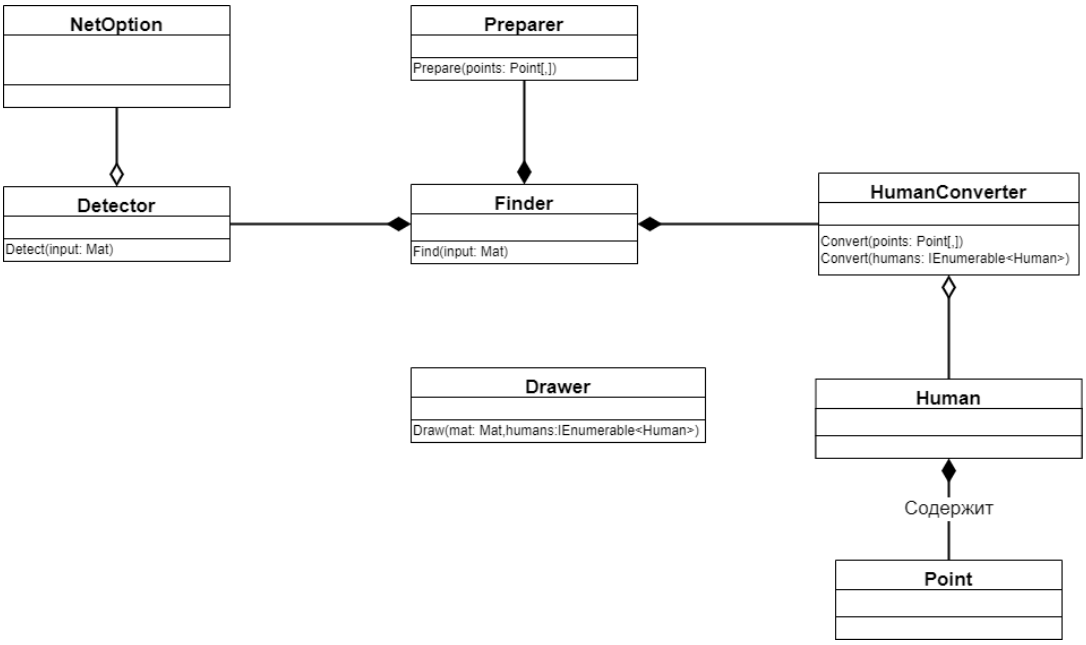


Рис. 39. Уточнённая диаграмма классов пакета «Detection»

1.5.2.4.4 Детальная диаграмма классов

Детальная диаграмма классов пакета «Detection» представлена на рис. 40. Описание полей и методов классов пакета «Detection» (табл. 84-93).

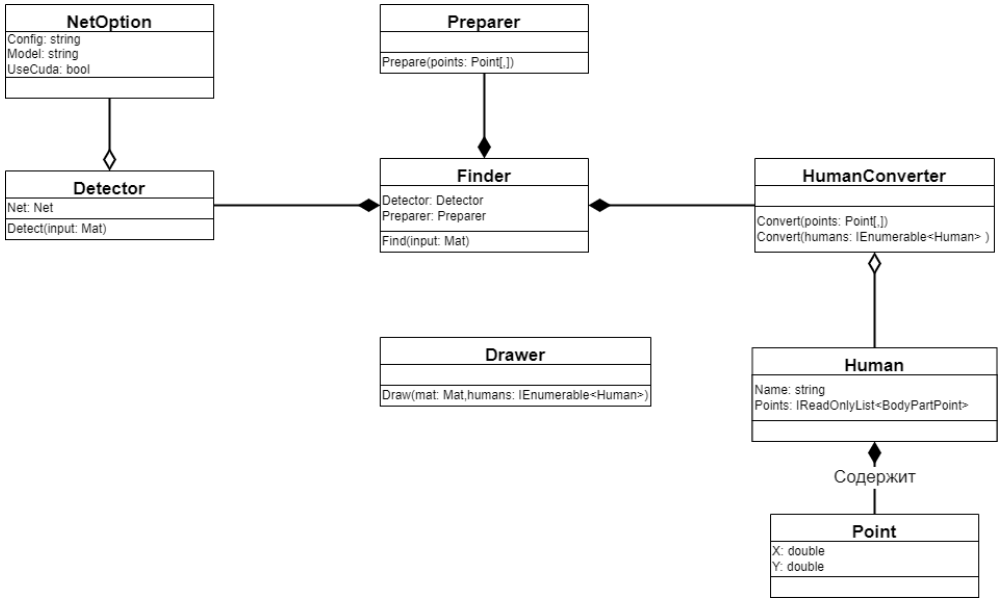


Рис. 40. Детальная диаграмма классов пакета «Detection»

Таблица 84

Описание полей класса «Finder»

Название	Тип	Описание
Detector	Detector	Объект для обнаружения частей тела в формате точек
Preparer	Preparer	Объект для изменения структуры выходных точек

Таблица 85

Описание методов класса «Finder»

Название	Параметры	Возвращаемое значение	Описание
Find	input: Mat	IReadOnlyList<Human>	Метод получения объектов людей, найденных на кадре

Таблица 86

Описание полей класса «NetOption»

Название	Тип	Описание
Config	string	Путь для файла конфигурации нейронной сети
Model	string	Путь для файла весов нейронной сети
UseCuda	bool	Флаг для использования видеокарты при обработке нейронной сети

Таблица 87

Описание полей класса «Detector»

Название	Тип	Описание
Net	Net	Объект, характеризующий нейронную сеть

Таблица 88

Описание методов класса «Detector»

Название	Параметры	Возвращаемое значение	Описание
Detect	input: Mat	Point[,]	Метод получения точек частей тела, найденных на кадре

Таблица 89

Описание методов класса «Preparer»

Название	Параметры	Возвращаемое значение	Описание
Find	points: Point[,]	Point[,]	Метод для преобразования найденных точек частей тела

Таблица 90

Описание методов класса «HumanConverter»

Название	Параметры	Возвращаемое значение	Описание
Convert	points: Point[,]	IEnumerable<Human>	Метод для преобразования найденных точек частей тела в список людей
Convert	humans: IEnumerable<Human>	Point[,]	Метод для преобразования найденных людей в точки частей тела

Таблица 91

Описание методов класса «Drawer»

Название	Параметры	Возвращаемое значение	Описание
Draw	mat: Mat, humans: IEnumerable<Human>	-	Метод для отрисовки найденных людей на кадре

Таблица 92

Описание полей класса «Human»

Название	Тип	Описание
Name	string	Отображаемое имя человека
Points	IReadOnlyList<BodyPartPoint>	Точки частей тела человека

Таблица 93

Описание полей класса «Point»

Название	Тип	Описание
X	double	Координата по оси абсцисс
Y	double	Координата по оси ординат

1.5.2.5 Проектирование классов пакета «Services»

1.5.2.5.1 Исходная диаграмма классов

Исходная диаграмма классов пакета «Services» представлена на рис. 41, а её описание в табл. 94. Данный пакет является библиотекой для обработки суперпакета DataLayer и хранит в себе 2 сервиса, которые будет удобно использовать в приложении.

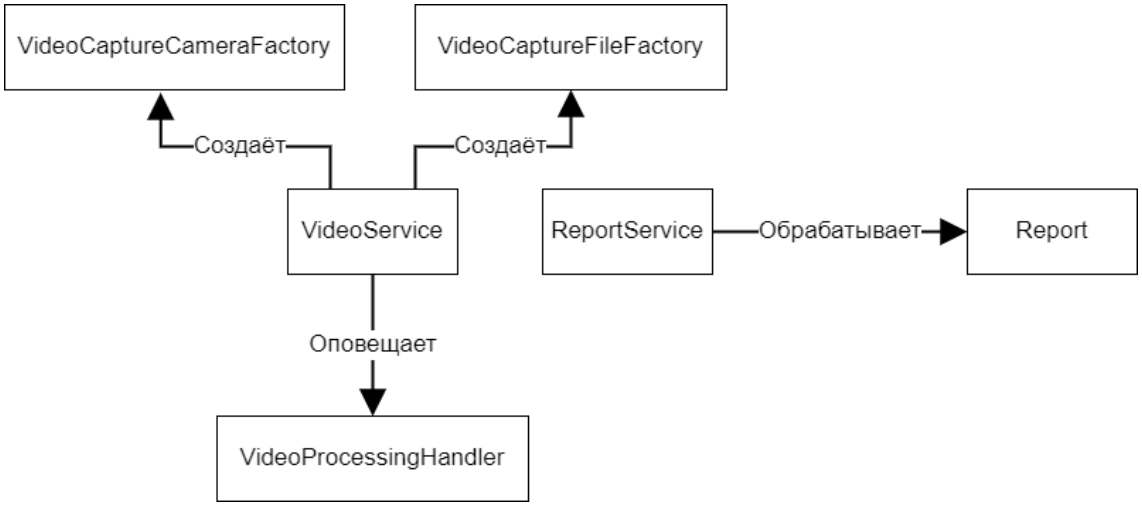


Рис. 41. Исходная диаграмма классов пакета «Services»

Таблица 94

Описание классов пакета «Services»

Класс	Описание
VideoCaptureCameraFactory	класс, описывающий логику открытия видеопотока с камеры
VideoCaptureFileFactory	класс, описывающий логику открытия видеопотока из файла
VideoService	класс, содержащий логику работы с видео
VideoProcessingHandler	класс, описывающий обработчика нейронной сети
ReportService	класс, содержащий логику работы с отправкой отчетов
Report	класс, описывающий отчёт

1.5.2.5.2 Диаграмма последовательностей взаимодействия объектов классов

На рис. 42-44 представлены диаграммы последовательностей взаимодействия объектов классов пакета «Services».

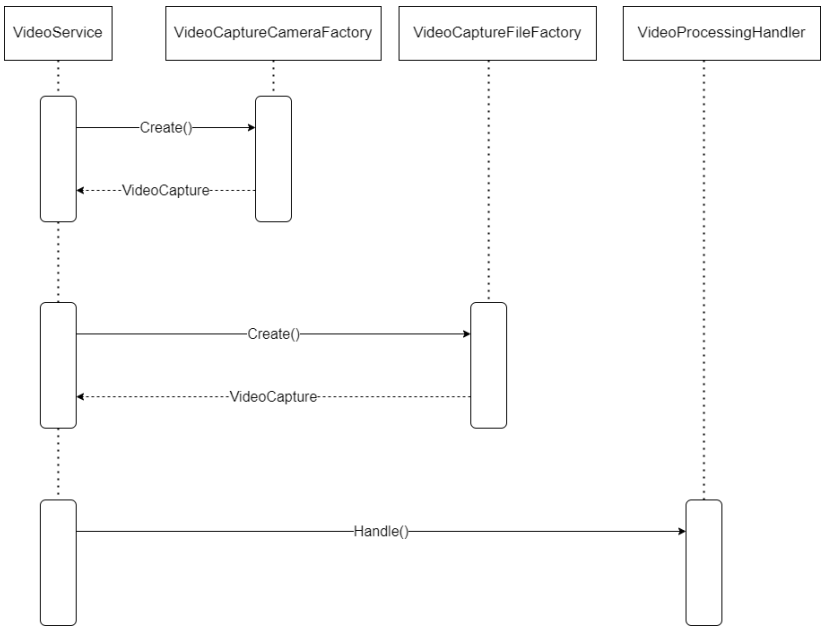


Рис. 42. Диаграмма последовательности действий классов пакета «Services». Нормальный ход событий

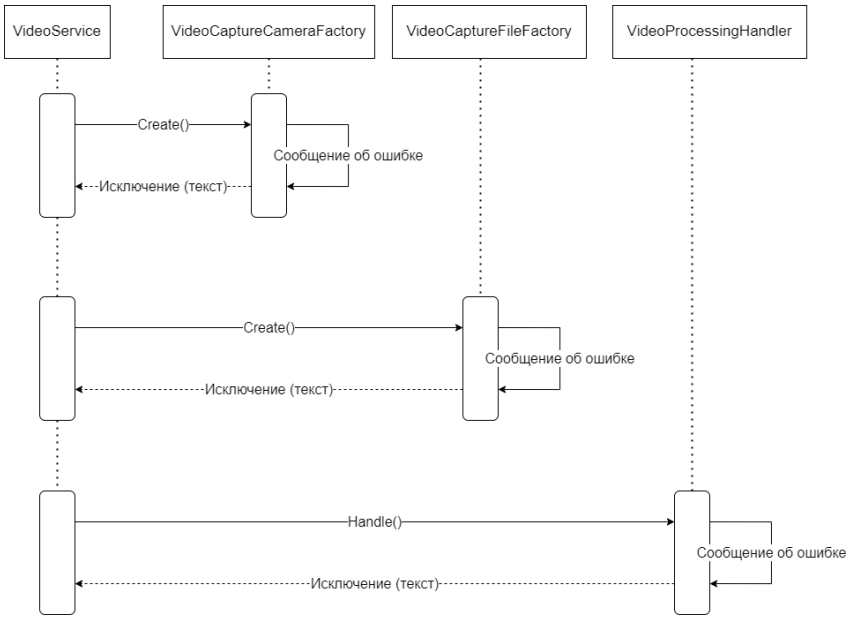


Рис. 43. Диаграмма последовательности действий классов пакета «Services». Прерывание процесса системой

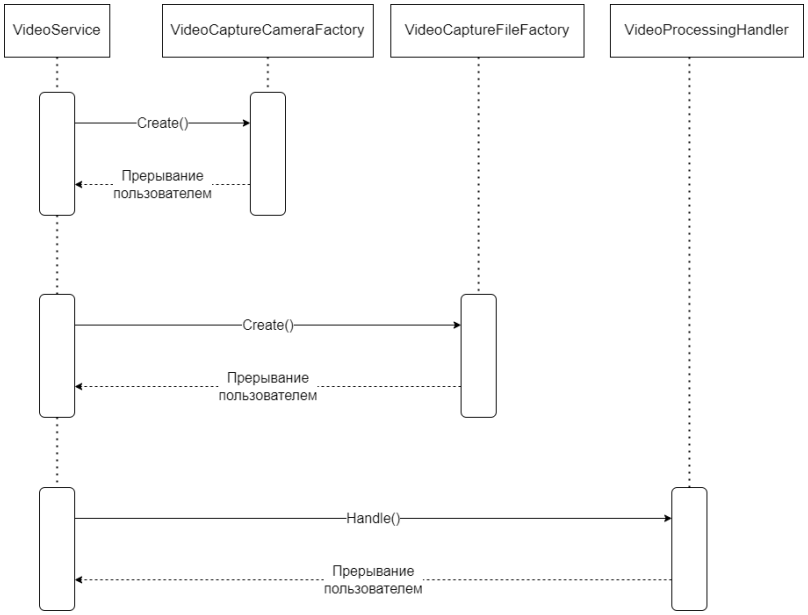


Рис. 44. Диаграмма последовательности действий классов пакета «Services». Прерывание процесса пользователем

1.5.2.5.3 Уточнённая диаграмма классов

Уточнённая диаграмма классов пакета «Services» представлена на рис. 45.

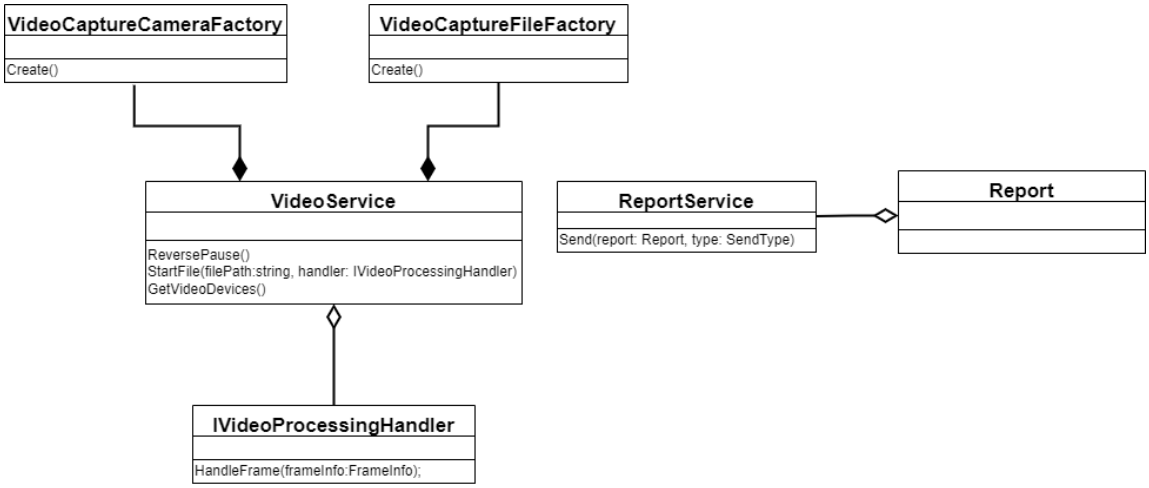


Рис. 45. Уточнённая диаграмма классов пакета «Services»

1.5.2.5.4 Детальная диаграмма классов

Детальная диаграмма классов пакета «Services» представлена на рис. 46. Описание полей и методов классов пакета «Services» (табл. 95-104).

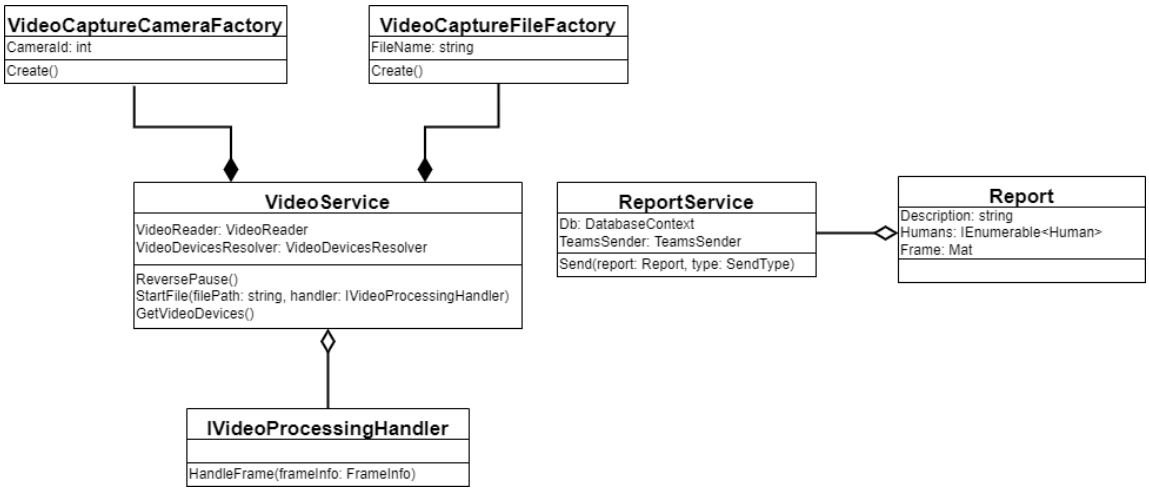


Рис. 46. Детальная диаграмма классов пакета «Services»

Таблица 95

Описание полей класса «VideoCaptureCameraFactory»

Название	Тип	Описание
CameraId	int	Идентификатор камеры

Таблица 96

Описание методов класса «VideoCaptureCameraFactory»

Название	Параметры	Возвращаемое значение	Описание
Create	-	VideoCapture	Метод создания видеопотока

Таблица 97

Описание полей класса «VideoCaptureFileFactory»

Название	Тип	Описание
FileName	string	Путь для файла

Таблица 98

Описание методов класса «VideoCaptureFileFactory»

Название	Параметры	Возвращаемое значение	Описание
Create	-	VideoCapture	Метод создания видеопотока

Таблица 99

Описание полей класса «VideoService»

Название	Тип	Описание
VideoReader	VideoReader	Объект с уровня данных для работы с видеопотоком
VideoDevicesResolver	VideoDevicesResolver	Объект для получения информации по доступным видеокамерам

Таблица 100

Описание методов класса «VideoService»

Название	Параметры	Возвращаемое значение	Описание
ReversePause	-	bool	Метод для изменения состояния паузы
StartFile	filePath: string, handler: IVideoProcessingHandler	-	Метод для старта видеопотока с файла
StartVideo	id: int, handler: IVideoProcessingHandler	-	Метод для старта видеопотока из файла
GetVideoDevices	-	IEnumerable<VideoDeviceInfo>	Метод для получения информация о доступных камеры

Таблица 101

Описание методов класса «IVideoProcessingHandler»

Название	Параметры	Возвращаемое значение	Описание
HandleFrame	frameInfo: FrameInfo	-	Метод обработки кадра

Таблица 102

Описание полей класса «ReportService»

Название	Тип	Описание
Db	DatabaseContext	Объект для работы с базой данных
TeamsSender	TeamsSender	Объект для отправки сообщений на канал в Teams

Таблица 103

Описание методов класса «ReportService»

Название	Параметры	Возвращаемое значение	Описание
Send	report: Report, type: SendType	-	Метод для отправки отчёта, согласно опции

Таблица 104

Описание полей класса «Report»

Название	Тип	Описание
Description	string	Описание отчёта
Humans	IEnumerable<Human>	Люди отчёта
Frame	Mat	Кадр для отчета

1.5.2.6 Проектирование классов пакета «Database Sending»

1.5.2.6.1 Исходная диаграмма классов

Исходная диаграмма классов пакета «Database Sending» представлена на рис. 47, а её описание в табл. 105.

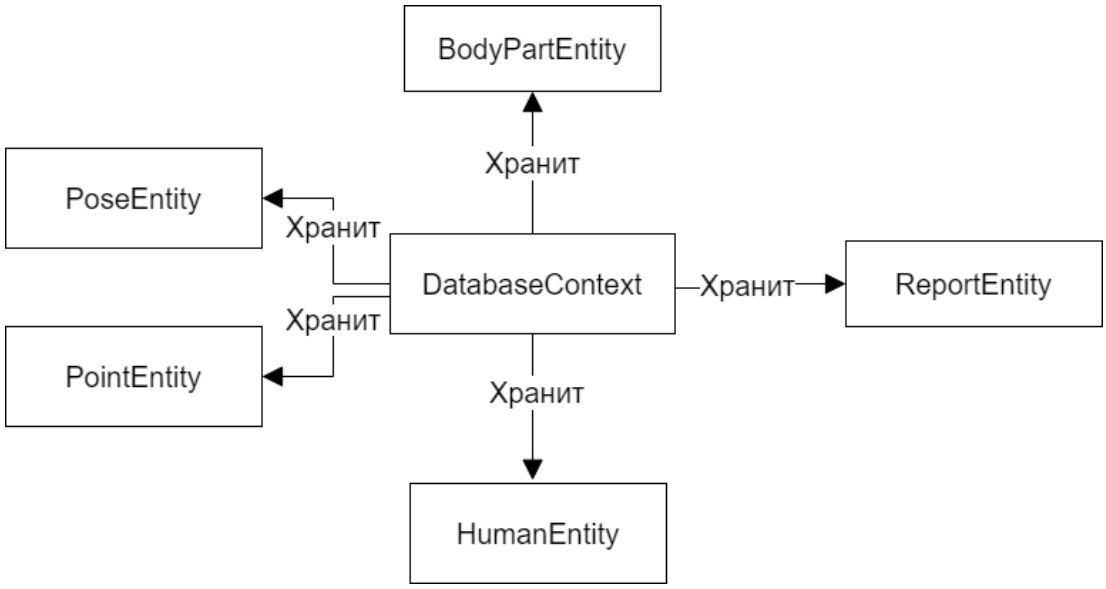


Рис. 47. Исходная диаграмма классов пакета «Database Sending»

Таблица 105

Описание классов пакета «Database Sending»

Класс	Описание
DatabaseContext	класс, описывающий схему базы данных
PoseEntity	класс, описывающий таблицу поз
PointEntity	класс, описывающий таблицу точек
HumanEntity	класс, описывающий таблицу людей
ReportEntity	класс, описывающий таблицу отчётов
BodyPartEntity	класс, описывающий таблицу частей тела

1.5.2.6.2 Уточнённая диаграмма классов

Уточнённая диаграмма классов пакета «Database Sending» представлена на рис. 48.

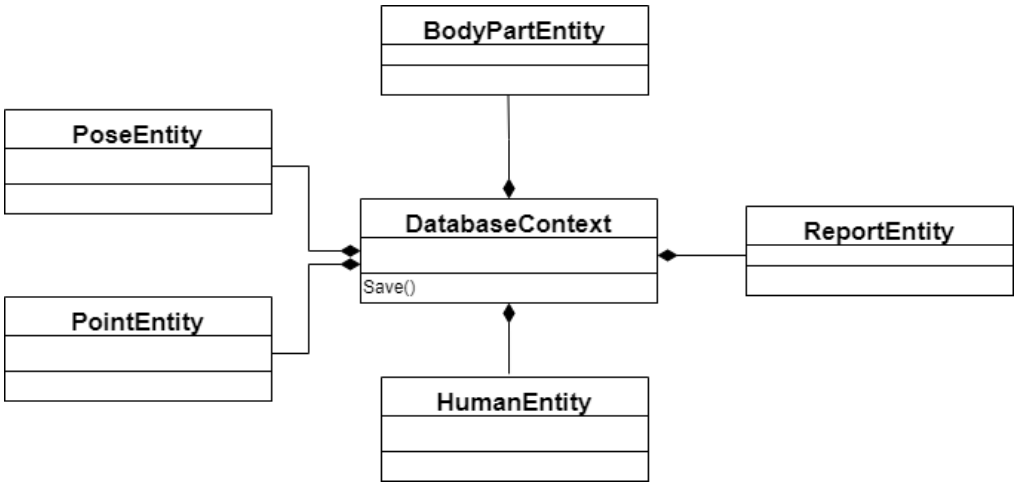


Рис. 48. Уточнённая диаграмма классов пакета «Database Sending»

1.5.2.6.3 Детальная диаграмма классов

Детальная диаграмма классов пакета «Database Sending» представлена на рис. 49. Описание полей и методов классов пакета «Database Sending» (табл. 106-112).

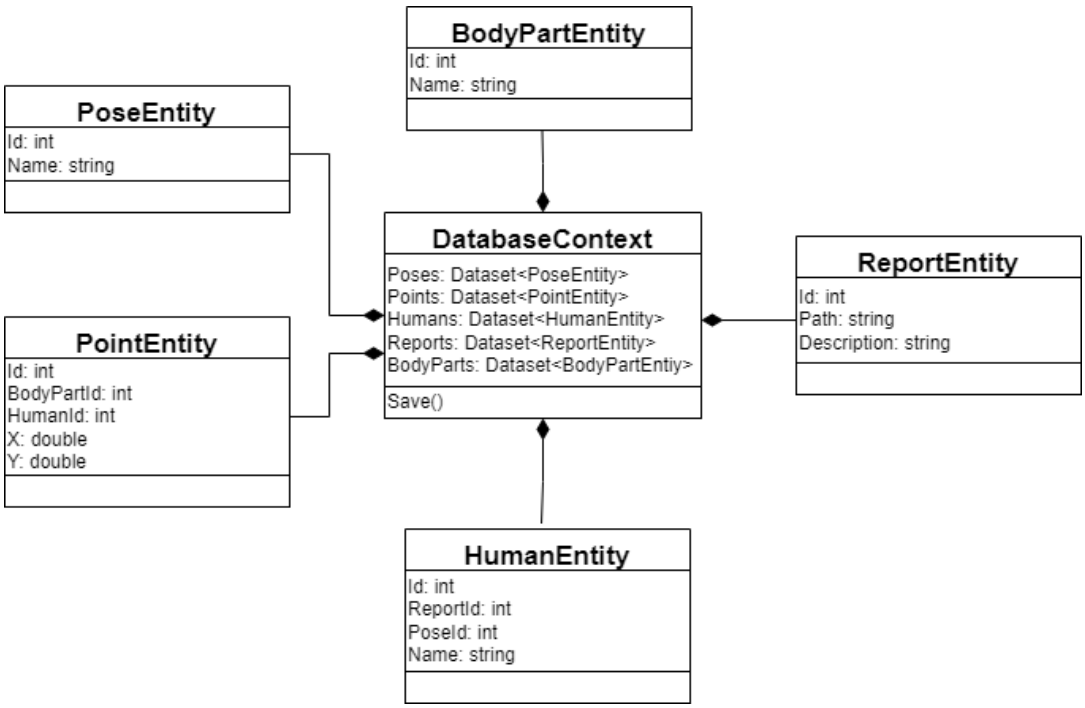


Рис. 49. Детальная диаграмма классов пакета «Database Sending»

Таблица 106

Описание полей класса «DatabaseContext»

Название	Тип	Описание
Poses	Dataset<PoseEntity>	Репозиторий поз
Points	Dataset<PointEntity>	Репозиторий точек
Humans	Dataset<HumanEntity>	Репозиторий человек
Reports	Dataset<ReportEntity>	Репозиторий отчётов
BodyParts	Dataset<BodyPartEntiy>	Репозиторий частей тела

Таблица 107

Описание методов класса «DatabaseContext»

Название	Параметры	Возвращаемое значение	Описание
Save	-	-	Метод для сохранения данных в БД

Таблица 108

Описание полей класса «PoseEntity»

Название	Тип	Описание
Id	int	Идентификатор позы
Name	string	Имя позы

Таблица 109

Описание полей класса «PointEntity»

Название	Тип	Описание
Id	int	Идентификатор точки
BodyPartId	int	Идентифкатор части тела
HumanId	int	Идентификатор человека
X	double	Координата по оси абсцисс
Y	double	Координата по оси ординат

Таблица 110

Описание полей класса «HumanEntity»

Название	Тип	Описание
Id	int	Идентифкатор человека
ReportId	int	Идентификатор отчёта
PoseId	int	Идентифактор позы
Name	string	Отображаемое имя

Таблица 111

Описание полей класса «ReportEntity»

Название	Тип	Описание
Id	int	Идентфикатор отчёта
Path	string	Путь для кадра
Description	string	Описание отчёта

Таблица 112

Описание полей класса «BodyPartEntity»

Название	Тип	Описание
Id	int	Идентификатор части тела
Name	string	Имя части тела

1.5.2.7 Проектирование классов пакета «Teams Sending»

1.5.2.7.1 Исходная диаграмма классов

Исходная диаграмма классов пакета «Teams Sending» представлена на рис. 50, а её описание в табл. 113.

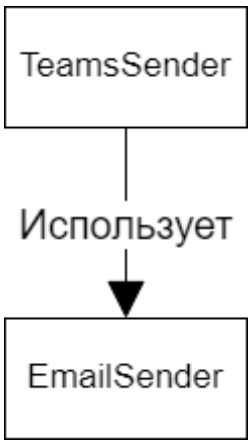


Рис. 50. Исходная диаграмма классов пакета «Teams Sending»

Таблица 113

Описание классов пакета «Teams Sending»

Класс	Описание
TeamsSender	класс, содержащий логику для отправки данных на канал в Teams
EmailSender	класс, содержащий логику для отправки данных на почтовый ящик

1.5.2.7.2 Диаграмма последовательностей взаимодействия объектов классов

На рис. 51-53 представлены диаграммы последовательностей взаимодействия объектов классов пакета «Teams Sending».

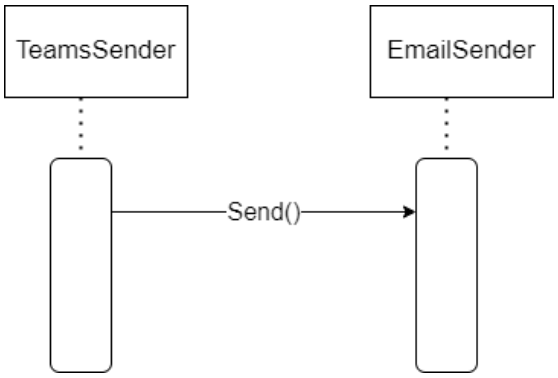


Рис. 51. Диаграмма последовательности действий классов пакета «Teams Sending». Нормальный ход событий

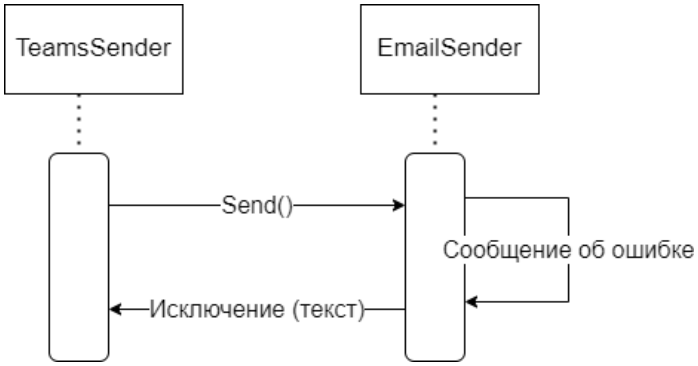


Рис. 52. Диаграмма последовательности действий классов пакета «Teams Sending». Прерывание процесса системой

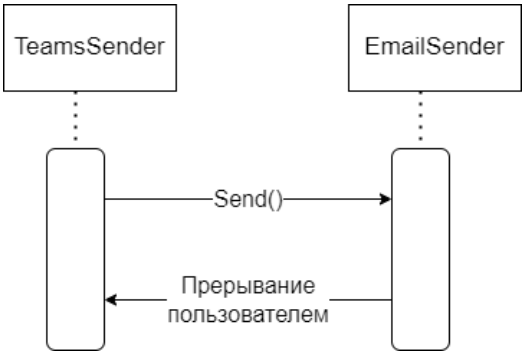


Рис. 53. Диаграмма последовательности действий классов пакета «Teams Sending». Прерывание процесса пользователем

1.5.2.7.3 Уточнённая диаграмма классов

Уточнённая диаграмма классов пакета «Teams Sending» представлена на рис. 54.

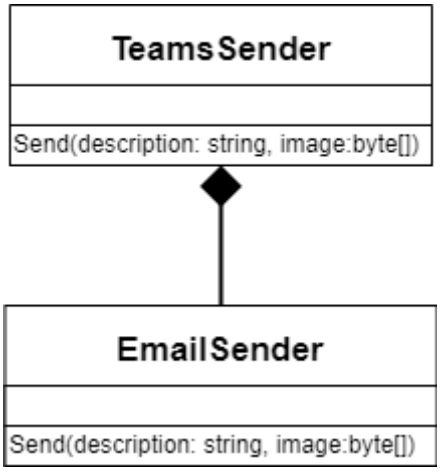


Рис. 54. Уточнённая диаграмма классов пакета «Teams Sending»

1.5.2.7.4 Детальная диаграмма классов

Детальная диаграмма классов пакета «Teams Sending» представлена на рис. 55. Описание полей и методов классов пакета «Teams Sending» (табл. 114-116).

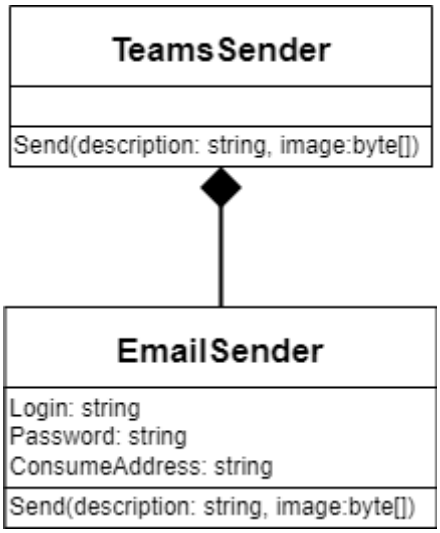


Рис. 55. Детальная диаграмма классов пакета «Teams Sending»

Таблица 114

Описание полей класса «EmailSender»

Название	Тип	Описание
Login	string	Логин для почты
Password	string	Пароль для почты
ConsumeAddress	string	Адрес получателя

Таблица 115

Описание методов класса «EmailSender»

Название	Параметры	Возвращаемое значение	Описание
Send	description: string, image: byte[]	-	Метод отправки отчёта на почту

Таблица 116

Описание методов класса «TeamsSender»

Название	Параметры	Возвращаемое значение	Описание
Send	description: string, image: byte[]	-	Метод отправки отчёта на канал в Teams

1.5.2.8 Проектирование классов пакета «Reading»

1.5.2.8.1 Исходная диаграмма классов

Исходная диаграмма классов пакета «Reading» представлена на рис. 56, а её описание в табл. 117.

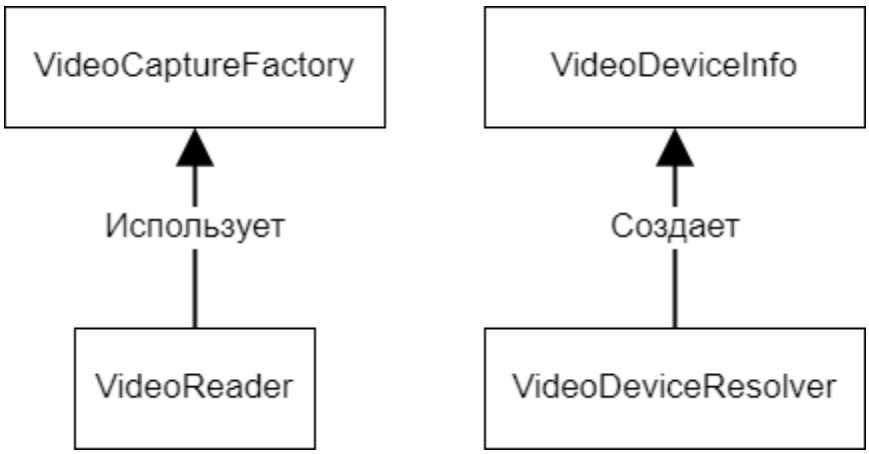


Рис. 56. Исходная диаграмма классов пакета «Reading»

Таблица 117

Описание классов пакета «Reading»

Класс	Описание
VideoReader	класс, содержащий логику чтения видеопотока в отдельном потоке
VideoCaptureFactory	класс, описывающий логику создания видеопотока
VideoDeviceResolver	класс, содержащий логику получения информации о доступных видеокамерах
VideoDeviceInfo	класс, описывающий информацию о видеокамере

1.5.2.8.2 Уточнённая диаграмма классов

Уточнённая диаграмма классов пакета «Reading» представлена на рис. 57.

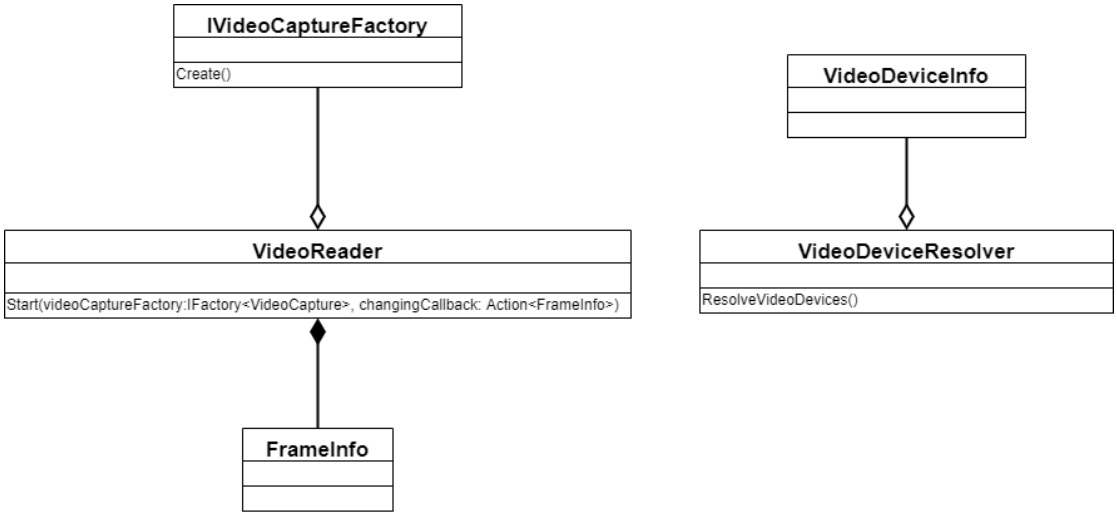


Рис. 57. Уточнённая диаграмма классов пакета «Reading»

1.5.2.8.3 Детальная диаграмма классов

Детальная диаграмма классов пакета «Reading» представлена на рис. 58. Описание полей и методов классов пакета «Reading» (табл. 118-124).

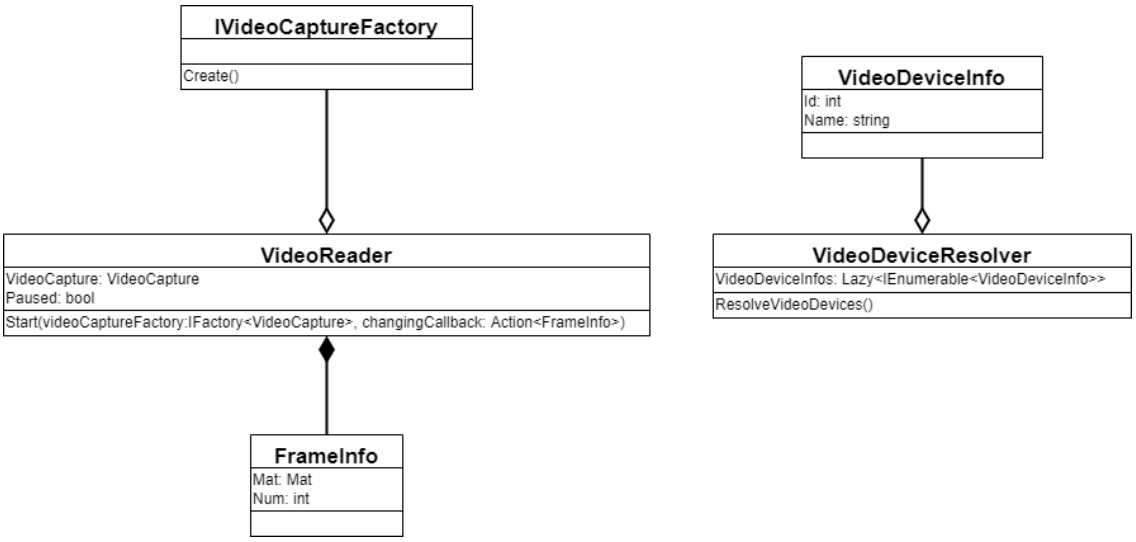


Рис. 58. Детальная диаграмма классов пакета «Reading»

Таблица 118

Описание полей класса «VideoReader»

Название	Тип	Описание
VideoCapture	VideoCapture	Объект для получения видеопотока
Paused	bool	Находится ли чтение видео в состоянии паузы

Таблица 119

Описание методов класса «VideoReader»

Название	Параметры	Возвращаемое значение	Описание
Start	videoCaptureFactory: IVideoCaptureFactory, changingCallback: Action<FrameInfo>	-	Метод для начала обработки видеопотока

Таблица 120

Описание полей класса «VideoDeviceResolver»

Название	Тип	Описание
VideoDeviceInfos	Lazy<IEnumerable<VideoDeviceInfo>>	Список загруженных видеокамер в ленивой загрузке

Таблица 121

Описание методов класса «VideoDeviceResolver»

Название	Параметры	Возвращаемое значение	Описание
ResolveVideoDevices	-	IEnumerable<VideoDeviceInfo>	Метод для получения информации о доступных камерах

Таблица 122

Описание полей класса «VideoDeviceInfo»

Название	Тип	Описание
Id	int	Идентификатор камеры
Name	string	Имя камеры

Таблица 123

Описание полей класса «FrameInfo»

Название	Тип	Описание
Mat	Mat	Кадр
Num	int	Номер кадра

Таблица 124

Описание методов класса «IVideoCaptureFactory»

Название	Параметры	Возвращаемое значение	Описание
Create	-	VideoCapture	Метод для создания объекта видеопотка

1.5.2.9 Проектирование классов пакета «Extensions»

1.5.2.9.1 Исходная диаграмма классов

Исходная диаграмма классов пакета «Extensions» представлена на рис. 59, а её описание в табл. 125.

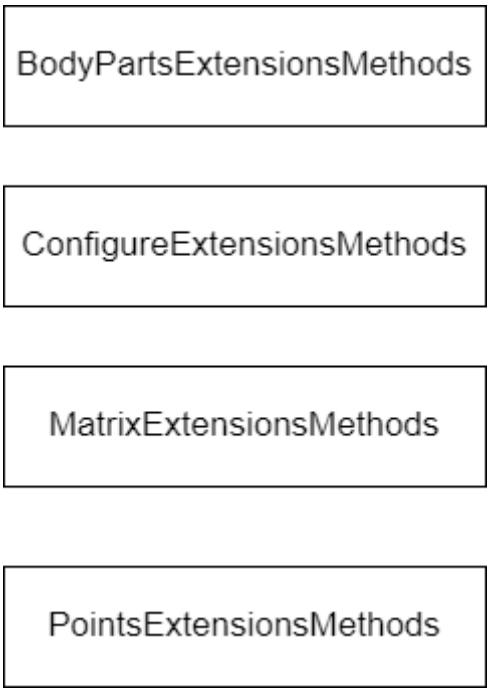


Рис. 59. Исходная диаграмма классов пакета «Extensions»

Таблица 125

Описание классов пакета «Extensions»

Класс	Описание
BodyPartsExtensionsMethod	Класс, содержащий методы для работы с частями чела
ConfigureExtensionsMethod	Класс, содержащий методы для работы с конфигурацией
MatrixExtensionsMethod	Класс, содержащий методы для работы с матрицами
PointsExtensionsMethod	Класс, содержащий методы для работы с точками

1.5.2.9.2 Уточнённая диаграмма классов

Уточнённая диаграмма классов пакета «Extensions» представлена на рис. 60.

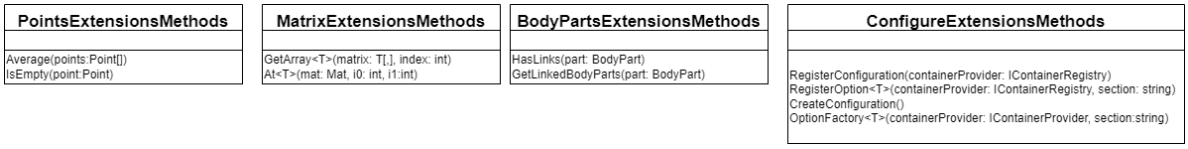


Рис. 60. Уточнённая диаграмма классов пакета «Extensions»

1.5.2.9.3 Детальная диаграмма классов

Детальная диаграмма классов пакета «Extensions» представлена на рис. 61. Описание полей и методов классов пакета «Extensions» (табл. 126-129).

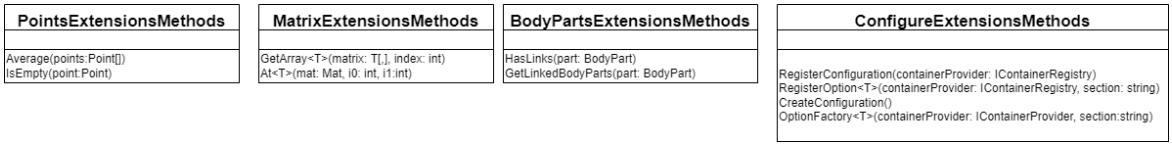


Рис. 61. Детальная диаграмма классов пакета «Extensions»

Таблица 126

Описание методов класса «PointsExtensionsMethods»

Название	Параметры	Возвращаемое значение	Описание
Average	points: Point[]	Point	Метод для получения средней точки
IsEmpty	point: Point	bool	Метод для проверки точки на пустоту

Таблица 127

Описание методов класса «MatrixExtensionsMethods»

Название	Параметры	Возвращаемое значение	Описание
GetArray<T>()	matrix: T[,], index: int	T[]	Метод получения массива из матрица
At<T>	mat: Mat, i0: int, i1: int	T	Метод получения пикселя по координатам на кадре

Таблица 128

Описание методов класса «BodyPartsExtensionsMethods»

Название	Параметры	Возвращаемое значение	Описание
HasLinks	part: BodyPart	bool	Метод получения информации о том имеет ли часть тела связи с другими
GetLinkedBodyParts	part: BodyPart	IEnumerable<BodyPart>	Метод получения смежныъ частей тела

Таблица 129

Описание методов класса «BodyPartsExtensionsMethods»

Название	Параметры	Возвращаемое значение	Описание
RegisterConfiguration	containerProvider: IContainerRegistry	-	Метод для регистрации конфигурации
RegisterOption<T>	containerProvider: IContainerRegistry, section: string	-	Метод регистрации опции
CreateConfiguration	-	-	Метод для создания конфигурации
OptionFactory<T>	containerProvider: IContainerProvider, section: string	-	Метод фабрики для опции

1.5.2.10 Проектирование классов пакета «Data»

1.5.2.10.1 Исходная диаграмма классов

Исходная диаграмма классов пакета «Data» представлена на рис. 62, а её описание в табл. 130.

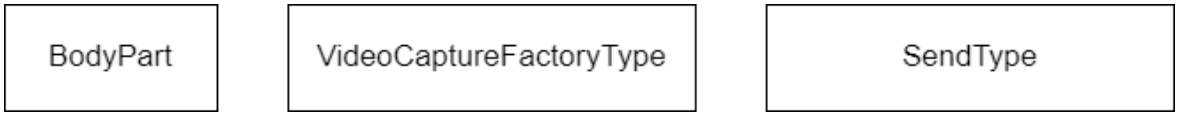


Рис. 62. Исходная диаграмма классов пакета «Data»

Таблица 130

Описание классов пакета «Data»

Класс	Описание
BodyPart	класс, описывающий все возможные части тела
VideoCaptureFactoryType	класс, описывающий все возможные фабрики создания видеопотока
SendType	класс, описывающий все возможные варианты отправки отчёта

1.5.2.10.2 Уточнённая диаграмма классов

Уточнённая диаграмма классов пакета «Data» представлена на рис. 63.

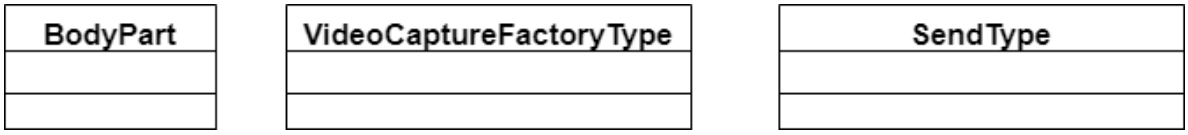


Рис. 63. Уточнённая диаграмма классов пакета «Data»

1.5.2.10.3 Детальная диаграмма классов

Детальная диаграмма классов пакета «Data» представлена на рис. 64. Описание полей и методов классов пакета «Data» (табл. 131-133).

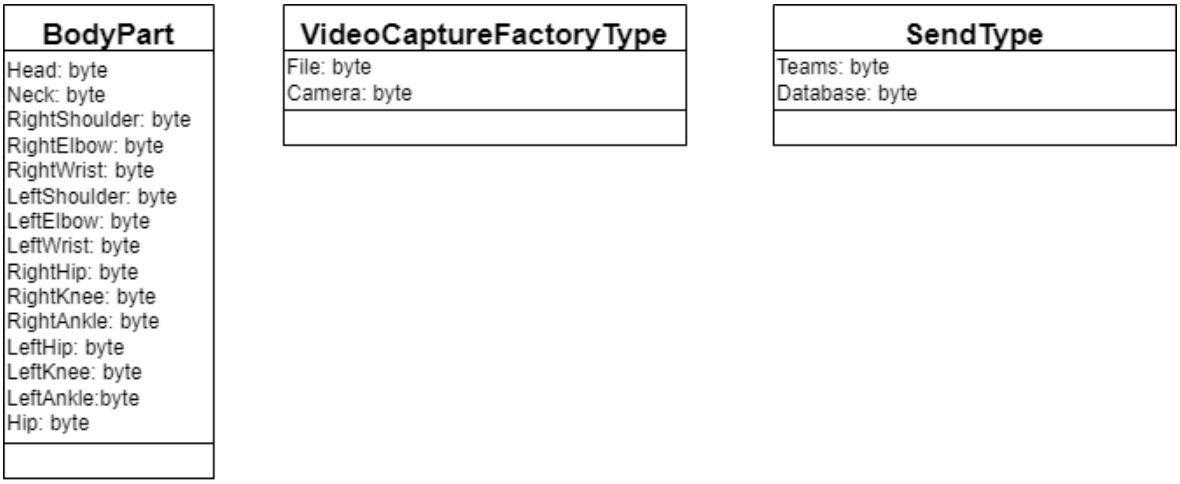


Рис. 64. Детальная диаграмма классов пакета «Data»

Таблица 131

Описание полей класса «BodyPart»

Название	Описание
Head	Голова
Neck	Шея
RightShoulder	Правое плечо
RightElbow	Правый локоть
RightWrist	Правая кисть
LeftShoulder	Левое плечо
LeftElbow	Левый локоть
LeftWrist	Левая кисть
RightHip	Правое бедро
RightKnee	Правое колено

RightAnkle	Правая ступня
LeftHip	Левое бедро
LeftKnee	Левое колено
LeftAnkle	Левая ступня
Hip	Таз

Таблица 132

Описание полей класса «VideoCaptureFactoryType»

Название	Описание
File	Файл
Camera	Камера

Таблица 133

Описание полей класса «SendType»

Название	Описание
Teams	Тимс
Database	База данных

1.5.2.11 Проектирование классов пакета «Exceptions»

1.5.2.11.1 Исходная диаграмма классов

Исходная диаграмма классов пакета «Exceptions» представлена на рис. 65, а её описание в табл. 134.

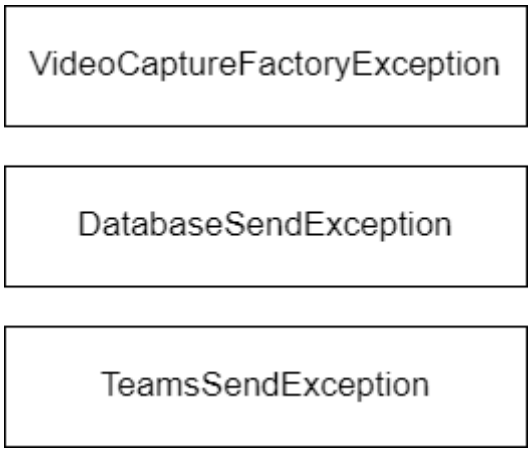


Рис. 65. Исходная диаграмма классов пакета «Exceptions»

Таблица 134

Описание классов пакета «Data»

Класс	Описание
VideoCaptureFactoryException	класс, описывающий исключение фабрики создания видеопотка
DatabaseSendException	класс, описывающий исключение отправки данных в базу данных
TeamsSendException	класс, описывающий исключение отправки данных в Teams

1.5.2.11.2 Уточнённая диаграмма классов

Уточнённая диаграмма классов пакета «Exceptions» представлена на рис. 66.

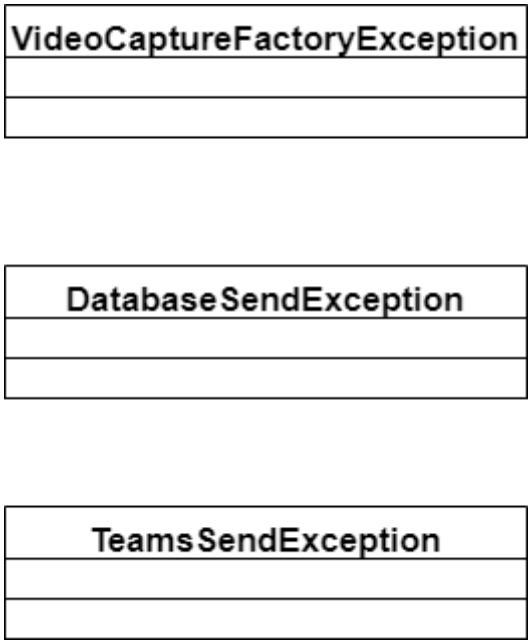


Рис. 66. Уточнённая диаграмма классов пакета «Exceptions»

1.5.2.11.3 Детальная диаграмма классов

Детальная диаграмма классов пакета «Exceptions» представлена на рис. 67. Описание полей и методов классов пакета «Exceptions» (табл. 135-137).

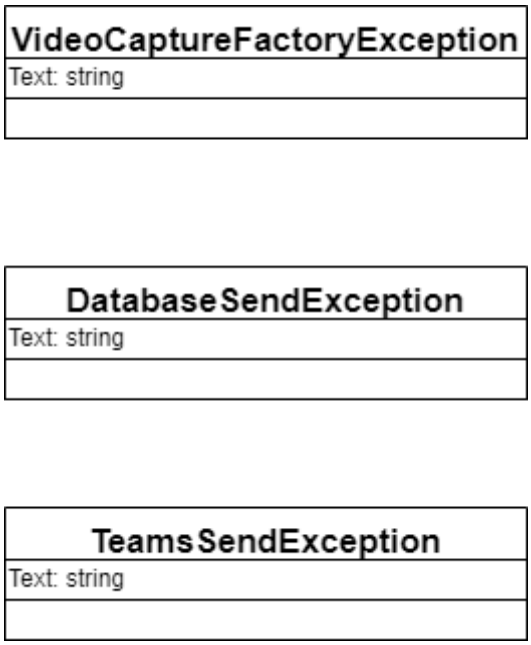


Рис. 67. Детальная диаграмма классов пакета «Exceptions»

Таблица 135

Описание полей класса «VideoCaptureFactoryException»

Название	Тип	Описание
Text	string	Сообщение исключения

Таблица 136

Описание полей класса «DatabaseSendException»

Название	Тип	Описание
Text	string	Сообщение исключения

Таблица 137

Описание полей класса «TeamsSendException»

Название	Тип	Описание
Text	string	Сообщение исключения

1.5.3 Построение диаграммы компонентов

Диаграммы компонентов применяют при проектировании физической структуры разрабатываемо программного обеспечения. Эти диаграммы показывают, как выглядит программное обеспечение на физическом уровне, т. е. из каких частей оно состоит и как эти части связаны между собой [1].

Диаграммы компонентов оперируют понятиями компонент и зависимость. Под компонентами при этом понимают физические заменяемые части программного обеспечения, которые соответствуют некоторому набору интерфейсов и обеспечивают их реализацию. [1]

Разрабатываемая программа содержит одну систему, которая взаимодействует с БД и с Teams. Ее диаграмма компонентов представлена на рис. 68.

Рис. 68. Диаграмма компонентов системы

Описание компонентов предоставлено в табл. 138.

Таблица 138

Описание компонентов системы

Наименование	Назначение	Входные данные	Выходные данные
Human Pose Estimator	Позволяет вести обработку поступающего с камеры видеопотока, а именно: ● Настраивать рабочие области. ● Обнаруживать части тела рабочих. ● Проверять рабочих на соблюдении техники безопасности ● Создавать отчет.	Видеопоток	Записи в БД и в канале Teams (журнал) и отчет о нахождении рабочего в зоне
PosgreSQL	База данных, предназначенная для хранения записей о нахождении рабочего, не соблюдающего технику безопасности	Информация, подлежащая хранению (записи)	Записи о нахождении рабочего в зоне
Microsoft Teams	Предназначена для хранения отчетов о нахождении рабочего, не соблюдающего технику безопасности	Информация, подлежащая хранению (текст и кадр)	Записи о нахождении рабочего в зоне
EmguCV.dll	Библиотека для работы с матрицами	-	Классы библиотеки
EmguCV .Runtime.dll	Расширение библиотеки для работы с нейронной сетью с помощью видеокарты	-	Классы библиотеки
ReactiveUI.dll	Библиотека, содержащая классы для реактивного программирования и для архитектурного паттерна MVVM	-	Классы библиотеки
Serilog.dll	Библиотека для логирования	-	Классы библиотеки
pose_deploy _linevec.prototxt	Входной файл конфигурации нейронной сети	-	Конфигурация нейронной сети
pose_iter_440000 .caffemodel	Входной файл с файлом весов для нейронной сети	-	Веса нейронной сети

Модульная структура приложения представлена на рис. 69.

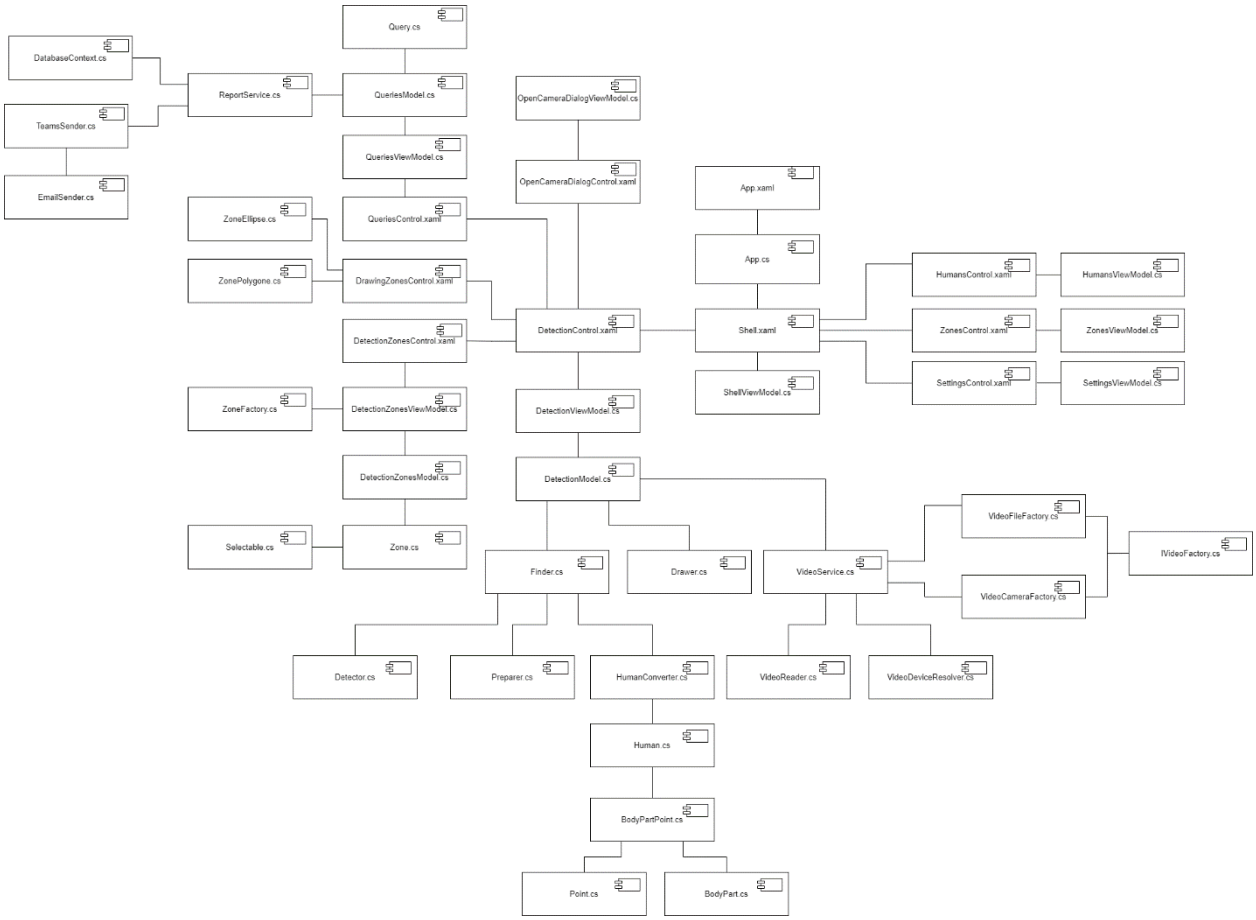


Рис. 69. Модульная структура программы

Краткие спецификации на разработанное приложение расположены в прил. 3. Подробные спецификации расположены в табл. 139 – 185.

Таблица 139

Спецификации компонента «DatabaseContext.cs»

Функция или свойство	Назначение
Poses	Репозиторий поз
Points	Репозиторий точек
Humans	Репозиторий человек
Reports	Репозиторий отчётов
BodyParts	Репозиторий частей тела
Save	Метод для сохранения данных в БД

Таблица 140

Спецификации компонента «TeamsSender.cs»

Функция или свойство	Назначение
Send	Метод отправки отчёта на канал в Teams

Таблица 141

Спецификации компонента «EmailSender.cs»

Функция или свойство	Назначение
Login	Логин для почты
Password	Пароль для почты
ConsumeAddress	Адрес получателя
Send	Метод отправки отчёта на почту

Таблица 142

Спецификации компонента «ReportService.cs»

Функция или свойство	Назначение
Db	Объект для работы с базой данных
TeamsSender	Объект для отправки сообщений на канал в Teams
Send	Метод для отправки отчёта, согласно опции

Таблица 143

Спецификации компонента «Query.cs»

Функция или свойство	Назначение
Description	Описание
Zones	Зоны запроса
SelectedZone	Выбранная зона
IsInverted	Инвертированность запроса
SendType	Тип отправки отчёта
Check	Проверка людей на подтверждение запроса

Таблица 144

Спецификации компонента «QueriesModel.cs»

Функция или свойство	Назначение
EventAggregator	Межмодульная шина данных
ReportService	Сервис для отправки отчётов
Queries	Общий список запросов
Zones	Общий список зон
Frame	Текущий кадр
AddingZoneHandler	Обработчик добавления зоны
RemovingZonehandler	Обработчик удаления зоны
AddingQueryHandler	Обработчик добавления запроса
RemovingQueryHandler	Обработчик удаления запроса
SendReport	Метод отправки запроса

Таблица 145

Спецификации компонента «QueriesViewModel.cs»

Функция или свойство	Назначение
Model	Модель для запросов
AddQueryCommand	Команда для добавления запроса
RemoveQueryCommand	Команда для удаления запроса

Таблица 146

Спецификации компонента «QueriesControl.xaml»

Функция или свойство	Назначение
Content	Содержимое окна запросов

Таблица 147

Спецификации компонента «DetectionControl.xaml»

Функция или свойство	Назначение
Content	Содержимое основного окна детекции

Таблица 148

Спецификации компонента «OpenCameraDialogControl.xaml»

Функция или свойство	Назначение
Content	Содержимое диалогового окна выбора видеопотока

Таблица 149

Спецификации компонента «OpenCameraDialogViewModel.cs»

Функция или свойство	Назначение
VideoService	Сервис для работы с видео
VideoDevices	Список доступных камер
SelectedDevice	Выбранная камера
CompleteCommand	Команда для успешного закрытия диалога
CancelCommand	Команда для отмены диалога

Таблица 150

Спецификации компонента «DrawingZonesControl.xaml»

Функция или свойство	Назначение
Content	Содержимое основного окна отрисовки зон

Таблица 151

Спецификации компонента «ZoneEllipse.cs»

Функция или свойство	Назначение
Content	Содержимое отрисовки точки зоны

Таблица 152

Спецификации компонента «ZonePolygone.cs»

Функция или свойство	Назначение
Content	Содержимое отрисовки области зоны

Таблица 153

Спецификации компонента «DetectionZonesControl.xaml»

Функция или свойство	Назначение
Content	Содержимое окна зон в окне детекции

Таблица 154

Спецификации компонента «DetectionZonesViewModel.cs»

Функция или свойство	Назначение
Model	Модель для зон
AddZoneCommand	Команда добавления зоны
RemoveZoneCommand	Команда для удаления зоны

Таблица 155

Спецификации компонента «ZoneFactory.cs»

Функция или свойство	Назначение
Id	Идентификатор зоны
Create	Метод для создания нового объекта зоны

Таблица 156

Спецификации компонента «DetectionZonesModel.cs»

Функция или свойство	Назначение
EventAggregator	Межмодульная шина данных
Zones	Общий список зон
SelectedZone	Выбранная зона
Humans	Список людей
Frame	Текущий кадр
AddingZoneHandler	Обработчик добавления зоны
RemovingZonehandler	Обработчик удаления зоны

Таблица 157

Спецификации компонента «Zone.cs»

Функция или свойство	Назначение
FailedCheckingElements	Найденные зоной люди
LeftTop	Левая верхняя точка зоны
RightTop	Правая верхняя точка зоны
RightBot	Правая нижняя точка зоны
LeftBot	Левая нижняя точка зоны
MinCount	Минимальное количество людей, которое нужно для обнаружения
Delay	Минимальное время, после прохождения которого, зона обнаружит людей
BodyParts	Список частей тела, которые обнаруживает зоны
CheckInside	Флаг о том, что обнаружение идёт внутри зоны
ZoneRoiSource	Часть кадра, находящаяся внутри точек зоны
Name	Имя зоны
Color	Цвет зоны
Opacity	Прозрачность зоны
GetPoints	Метод получения точек зоны в виде списка
Check	Метод проверки людей
CheckAndConvert	Метод получения людей, которые находятся в зоне
Check	Метод проверки человека на нахождение зоне
Check	Метод проверки точки на нахождение зоне
ValidatePoint	Метод валидации точки
ValidateCoordinate	Метод валидации координаты

Таблица 158

Спецификации компонента «Selectable.cs»

Функция или свойство	Назначение
Value	Часть тела
IsSelected	Флаг того, что элемент выбран

Таблица 159

Спецификации компонента «DetectionViewModel.cs»

Функция или свойство	Назначение
Model	Модель для детекции
DialogService	Сервис для открытия диалога
SelectedZone	Выбранная зона
StartVideoFromFileCommand	Команда для открытия видео из файла
StartVideoFromCameraCommand	Команда для открытия видео с камеры

Таблица 160

Спецификации компонента «DetectionModel.cs»

Функция или свойство	Назначение
VideoService	Видео сервис
Finder	Конечный обработчик нейронной сети
Drawer	Объект, для отрисовки людей на кадре
EventAggregator	Межмодульная шина данных
Frame	Текущий кадр
FrameSource	Объект для отображения текущего кадра
Humans	Найденные люди
StartVideoFromCamera	Метод для открытия видеопотока с камеры
StartVideoFromFile	Метод для открытия видеопотока с файла
HandleFrame	Метод обработки кадров
Convert	Метод конвертирования человека

Таблица 161

Спецификации компонента «Finder.cs»

Функция или свойство	Назначение
Detector	Объект для обнаружения частей тела в формате точек
Preparer	Объект для изменения структуры выходных точек
Find	Метод получения объектов людей, найденных на кадре

Таблица 162

Спецификации компонента «Detector.cs»

Функция или свойство	Назначение
Net	Объект, характеризующий нейронную сеть
Detect	Метод получения точек частей тела, найденных на кадре

Таблица 163

Спецификации компонента «Preparer.cs»

Функция или свойство	Назначение
Prepare	Метод для преобразования найденных точек частей тела

Таблица 164

Спецификации компонента «HumanConverter.cs»

Функция или свойство	Назначение
Convert	Метод для преобразования найденных точек частей тела в список людей
Convert	Метод для преобразования найденных людей в точки частей тела

Таблица 165

Спецификации компонента «Human.cs»

Функция или свойство	Назначение
Name	Отображаемое имя человека
Points	Точки частей тела человека

Таблица 166

Спецификации компонента «BodyPartPoint.cs»

Функция или свойство	Назначение
Point	Точка части тела на кадре
BodyPart	Тип части тела

Таблица 167

Спецификации компонента «Point.cs»

Функция или свойство	Назначение
X	Координата по оси абсцисс
Y	Координата по оси ординат

Таблица 168

Спецификации компонента «BodyPart.cs»

Функция или свойство	Назначение
1	2
Head	Голова
Neck	Шея
RightShoulder	Правое плечо
RightElbow	Правый локоть
RightWrist	Правая кисть
LeftShoulder	Левое плечо
LeftElbow	Левый локоть
LeftWrist	Левая кисть
RightHip	Правое бедро
RightKnee	Правое колено

Продолжение таблицы 168

1	2
RightAnkle	Правая ступня
LeftHip	Левое бедро
LeftKnee	Левое колено
LeftAnkle	Левая ступня
Hip	Таз

Таблица 169

Спецификации компонента «Drawer.cs»

Функция или свойство	Назначение
Draw	Метод для отрисовки найденных людей на кадре

Таблица 170

Спецификации компонента «VideoService.cs»

Функция или свойство	Назначение
VideoReader	Объект с уровня данных для работы с видеопотоком
VideoDevicesResolver	Объект для получения информации по доступным видеокамерам
ReversePause	Метод для изменения состояния паузы
StartFile	Метод для старта видеопотока с файла
StartVideo	Метод для старта видеопотока из файла
GetVideoDevices	Метод для получения информация о доступных камеры

Таблица 171

Спецификации компонента «VideoReader.cs»

Функция или свойство	Назначение
VideoCapture	Объект для получения видеопотока
Paused	Находится ли чтение видео в состоянии паузы
Start	Метод для начала обработки видеопотока

Таблица 172

Спецификации компонента «VideoDeviceResolver.cs»

Функция или свойство	Назначение
VideoDeviceInfos	Список загруженных видеокамер в ленивой загрузке
ResolveVideoDevices	Метод для получения информации о доступных камерах

Таблица 173

Спецификации компонента «VideoFileFactory.cs»

Функция или свойство	Назначение
FileName	Путь для файла
Create	Метод создания видеопотока

Таблица 174

Спецификации компонента «VideoCameraFactory.cs»

Функция или свойство	Назначение
CameraId	Идентификатор камеры
Create	Метод создания видеопотока

Таблица 175

Спецификации компонента «IVideoFactory.cs»

Функция или свойство	Назначение
Create	Метод для создания объекта видеопотока

Таблица 176

Спецификации компонента «Shell.xaml»

Функция или свойство	Назначение
Content	Содержимое основного окна

Таблица 177

Спецификации компонента «ShellViewModel.cs»

Функция или свойство	Назначение
EventAggregator	Межмодульная шина данных
Queue	Очередь сообщений для вывода

Таблица 178

Спецификации компонента «App.cs»

Функция или свойство	Назначение
Main	Метод запуска приложения

Таблица 179

Спецификации компонента «App.xaml»

Функция или свойство	Назначение
CreateShell	Метод создания основного окна

Таблица 180

Спецификации компонента «HumansControl.xaml»

Функция или свойство	Назначение
Content	Содержимое окна с людми

Таблица 181

Спецификации компонента «ZonesControl.xaml»

Функция или свойство	Назначение
Content	Содержимое окна с зонами

Таблица 182

Спецификации компонента «SettingsControl.xaml»

Функция или свойство	Назначение
Content	Содержимое окна настроек

Таблица 183

Спецификации компонента «HumansViewModel.cs»

Функция или свойство	Назначение
EventAggregator	Межмодульная шина данных
Humans	Список людей

Таблица 184

Спецификации компонента «ZonesViewModel.cs»

Функция или свойство	Назначение
EventAggregator	Межмодульная шина данных
Zones	Список зон

Таблица 185

Спецификации компонента «SettingsViewModel.cs»

Функция или свойство	Назначение
Model	Модель настроек
ApplyCommand	Команда для применения настроек

1.5.4 Построение диаграммы размещения

При физическом проектировании распределенных программных систем необходимо определить наиболее оптимальный вариант размещения программных компонентов на реальном оборудовании в локальной или глобальной сетях. Для этого используют специальную модель UML - диаграмму размещения.

Диаграмма размещения отражает физические взаимосвязи между программными и аппаратными компонентами системы. Каждой части аппаратных средств системы, например, компьютеру или датчику, на диаграмме размещения соответствует узел.

Диаграмма размещения системы представлена на рис. 70.

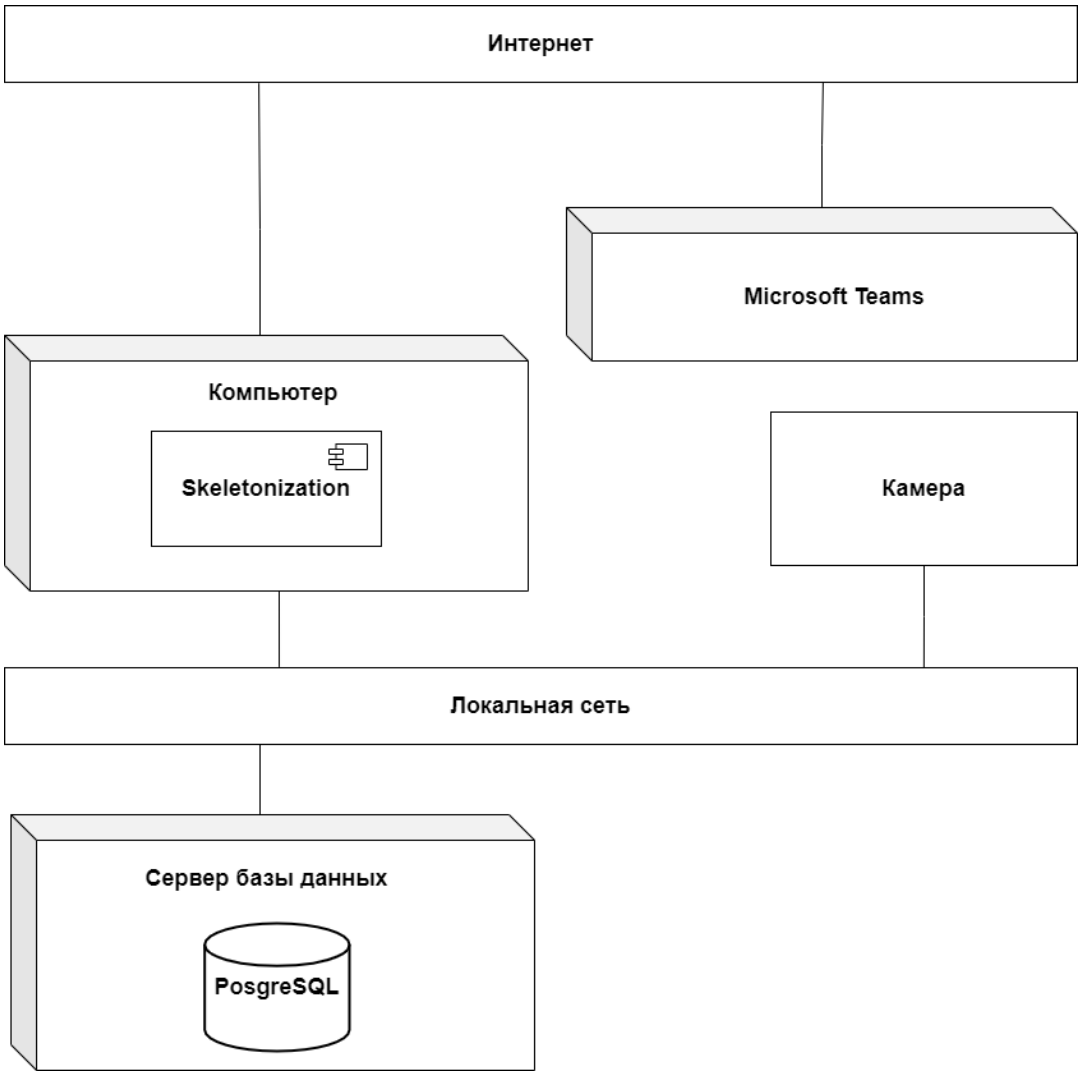


Рис. 70. Диаграмма размещения

1.6 Проектирование интерфейса пользователя

В данном разделе будет рассматриваться проектирование пользовательского интерфейса разработанной программы.

1.6.1 Построение графа диалога

Диалог — это процесс обмена информацией между пользователем и программной системой, осуществляемый через интерактивный терминал и по определенным правилам [1].

Тип диалога определяет, кто из «собеседников» управляет процессом обмена информацией. Соответственно различают два типа диалога: управляемые программой и управляемые пользователем.

Диалог, управляемый программой, предусматривает наличие жесткого, линейного или древовидного, т. е. включающего возможные альтернативные варианты, сценария диалога, заложенного в программное обеспечение. Такой диалог обычно сопровождают большим количеством подсказок, которые уточняют, какую информацию необходимо вводить на каждом шаге.

Диалог, управляемый пользователем, подразумевает, что сценарий диалога зависит от пользователя, который применяет систему для выполнения необходимых ему операций. При этом система обеспечивает возможность реализации различных пользовательских сценариев.

Граф диалога - ориентированный взвешенный граф, каждой вершине которого соответствует определенное состояние диалога, характеризующееся набором доступных пользователю действий. Дуги, исходящие из вершин, показывают возможные изменения состояний при выполнении пользователем указанных действий. Таким образом, граф представляет собой набор состояний системы, между которыми в ходе диалога при определенных условиях осуществляются переходы [1].

Разработка графа диалога позволяет выявить и устранить возможные тупиковые ситуации, выбрать рациональный путь перехода из текущего

состояния системы в требуемое, выявить неоднозначные ситуации, когда для пользователя требуется дополнительная помощь.

Интерфейс пользователя можно упростить, снизив степень неопределенности действий пользователя. Для этого можно применить смешанную структуру диалога, ограничив при необходимости свободу выбора пользователя, используя меню или другие элементы и контролировать вводимую пользователем информацию, принимать только допустимые данные.

Граф диалога программы представлен на рис. 71. После того, как программа запуститься, пользователь должен загрузить видео, он может это сделать либо загрузив видеофайл, либо получив доступ к установленной камере и ее видеопотоку. Также пользователь может, открыв специальное окно, настроить адрес почты для отправки отчета, путь для сохранения видео.

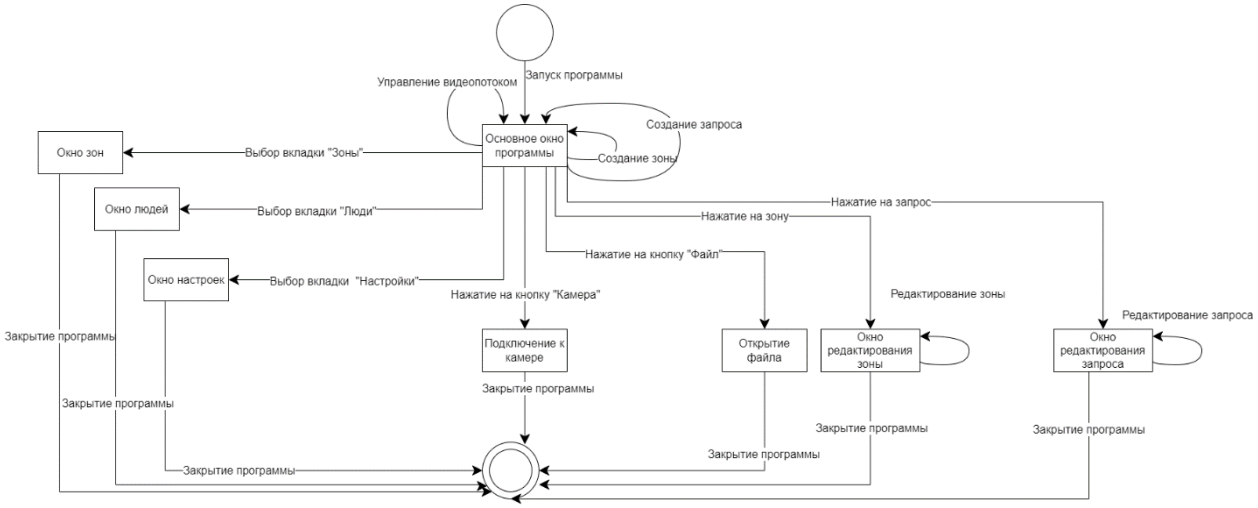


Рис. 71. Граф диалога

1.6.2 Разработка форм ввода-вывода информации

После открытия приложения, перед пользователем открывается главное окно (рис. 72), на котором размещена вся информация.

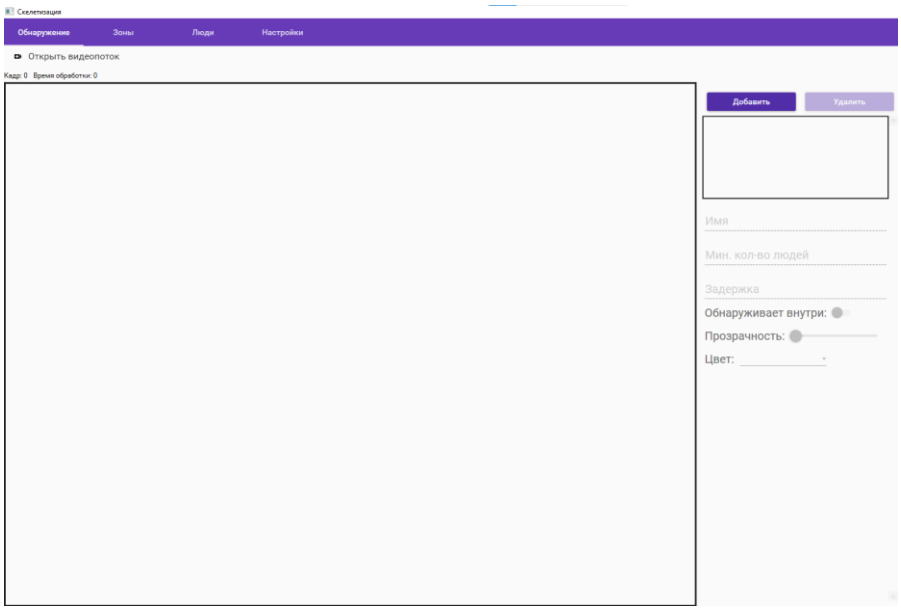


Рис. 72. Главное окно приложения

На главном окне пользователю доступно открытие видеопотока (рис. 73).

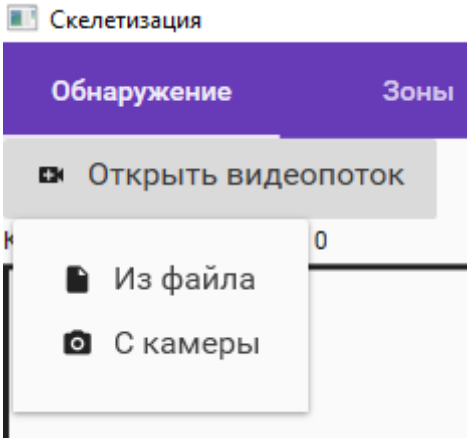


Рис. 73. Меню открытия видеопотка

Открытие из файла реализовано средствами операционной системы (рис. 74), открытие с камеры реализовано с помощью диалогового окна (рис. 75).

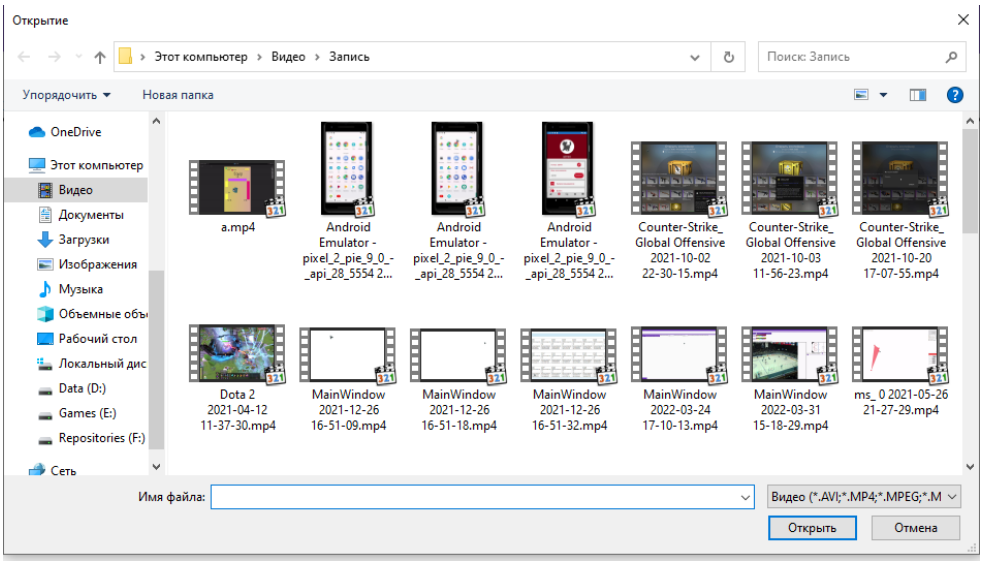


Рис. 74. Открытие файла

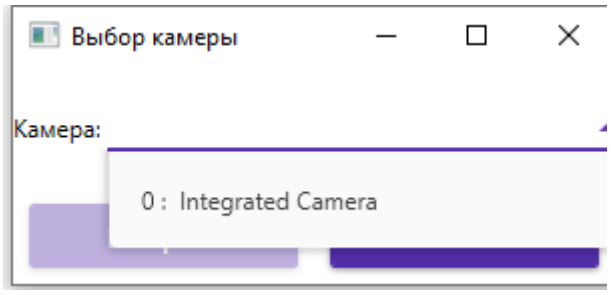


Рис. 75. Выбор камеры

После открытия видеопотока программа начнет процесс детектирования (рис. 76)

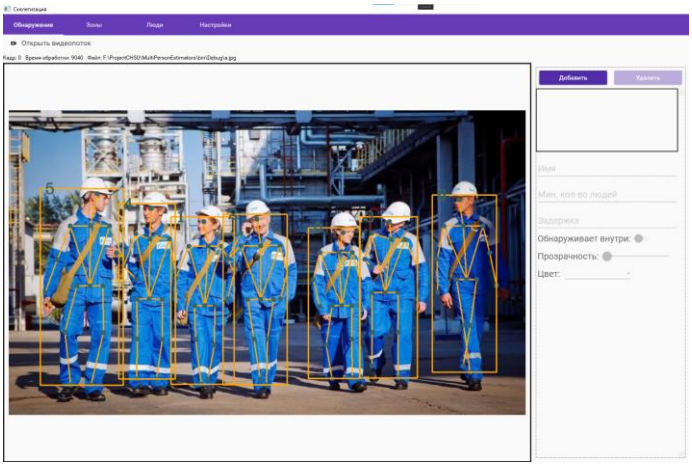


Рис. 76. Детектирование

Существует возможность добавлять зоны и редактировать её параметры, для этого создана отдельная форма (рис. 77)

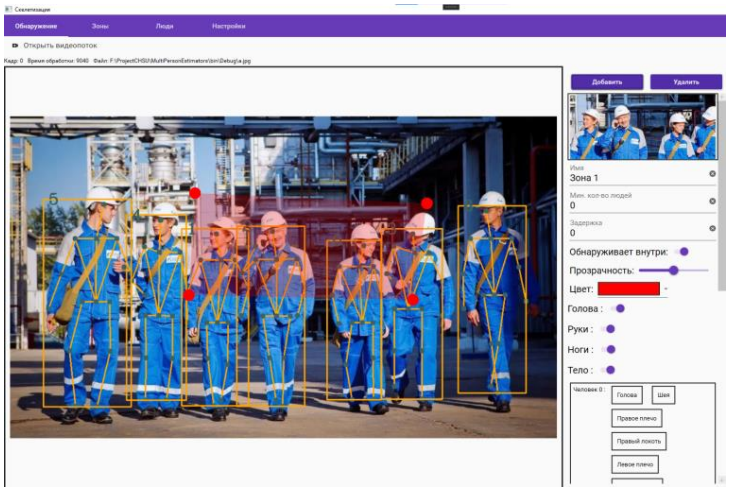


Рис. 77. Работа с зонами

Все зоны отображаются на вкладке «Зоны» (рис. 78)

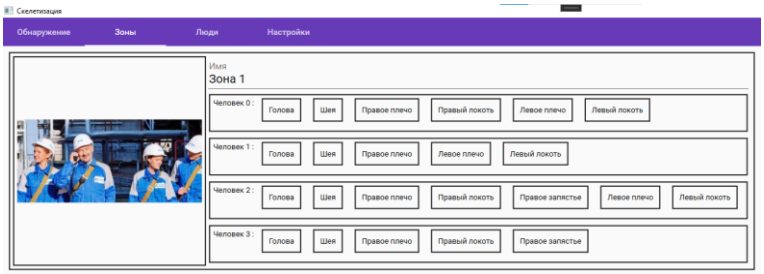


Рис. 78. Окно зон

Все люди отображаются на вкладке «Люди» (рис. 79)

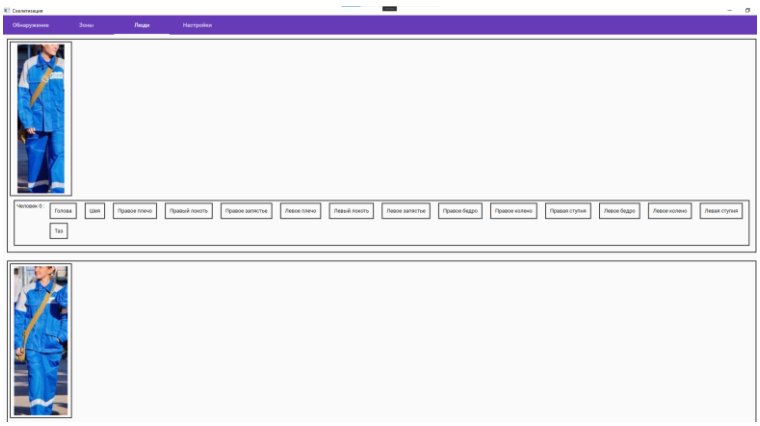


Рис. 79. Окно людей

Конечный интерфейс программы можно увидеть в прил.4.

1.7 Выбор стратегии тестирования, разработки тестов, программа и методика испытаний

В данном разделе описано и представлено тестирование разработанного программного обеспечения.

1.7.1 Объект и цель испытаний

Объектом испытаний является система скелетизации изображений человека для контроля опасных действий.

Цели испытаний:

- нахождение ошибок в программе;
- проверка правильности работы отдельных функций программы;
- проверка соответствия, разрабатываемого ПО требованиям, заявленным в прил. 1.

1.7.2 Требования к информационному, аппаратно-программному обеспечению

1.7.2.1 Требования к функциональным характеристикам

Программное обеспечение представляет собой десктопное приложение, реализующее следующий функционал:

- отслеживание положение работников;
- отслеживание отдельные части тела работников;
- определение позу работников;
- наличие предусмотренной возможности добавления нескольких зон опасности;
- наличие настройки у каждой зоны обнаружения конкретных частей тела;
- наличие инструмента комбинирования зон для формирования сложной опасной ситуации.

1.7.2.2 Требования к надежности

К разрабатываемому программному обеспечению предъявляются следующие требования к надежности:

- система должна иметь защиту от некорректных действий оператора и ошибочных исходных данных;
- система не должна во время работы модифицировать свой код или коды других программ;
- при удалении и передаче зафиксированных данных из базы данных система должна запрашивать подтверждения действия;
- логгирование;
- корректный вывод данных на экран, видео должно иметь качество не менее HD;
- проверка вводимых в настройках пользователем данных на корректность;
- система должна обеспечивать контроль целостности структур баз данных, нарушение которой возможно после аппаратных сбоев.

1.7.2.3 Требования к составу и параметрам технических средств

Компьютер, на котором будет установлена система должен обладать следующими требованиями:

- процессора с тактовой частотой не менее 3 ГГц;
- объем ОЗУ не менее 16 Гб;
- видеокарта компании NVidia серии GTX 1060 или выше для быстрой работы нейронной сети
- объём жёсткого диска не менее 500 Гб;
- монитор, поддерживающие разрешение 1920x1080 точек.

1.7.2.4 Требования к программной документации

Программная документация должна включать в себя следующие элементы:

1. Расчетно-пояснительная записка.
2. Техническое задание.
3. Схемы и/или диаграммы.
4. Текст программы.
5. Спецификации.
6. Руководство пользователя.

Документация оформляется на листах формата А4 по действующим стандартам на создание документации к программному обеспечению (ЕСПД).

1.7.3 Состав, порядок и методы испытаний

Тестирование разработанного ПО проводилось согласно следующего состава мероприятий по тестированию:

- тестирование разработанных модулей;
- тестирование программы на соответствие функциональных требований, изложенных в прил. 1;
- тестирование надежности программы на соответствие требований к надежности, изложенных в прил. 1;
- проверка программной документации на соответствие требований ЕСПД.

Тестирование ПО проводилось на персональном компьютере, имеющем следующие характеристики:

- процессор Ryzen 7 2700 (8 ядер, 16 потоков, тактовая частота 4.2 ГГц);
- видеокарта Gigabyte GeForce GTX 1660Ti;
- объем оперативной памяти 16 Гб;
- твердотельный накопитель объемом 1024 Гб;
- операционная система Windows 10 Домашняя;

- монитор с разрешением 1920x1080.

Для тестирования программной документации использовался метод ручного контроля. Для тестирования программного обеспечения использовалось два метода: функциональное тестирование и системное тестирование.

Для системного тестирования применялся метод ручного контроля. Ручной контроль обычно используется на ранних этапах разработки, так как с его помощью можно находить от 30 до 70 % ошибок логического проектирования и кодирования. Исходными данными для таких проверок являются: техническое задание, спецификации, структурная и функциональная схемы программного продукта, схемы отдельных компонентов.

Из методов ручного контроля был выбран метод проверки за столом. Этот метод не требует наличия группы специалистов. Проверка исходного текста проводится одним человеком, который читает текст программы и проверяет его на наличие возможных ошибок по списку наиболее часто встречающихся ошибок.

Для тестирования модулей был выбран один из методов функционального тестирования – метод граничных значений. Исходными данными для этого метода являются: программные модули, спецификации на компоненты, правильные и неправильные данные для модулей.

1.7.4 Результаты проведения испытаний

В результате проверки программной документации методом ручного контроля были исправлены найденные грамматические и пунктуационные ошибки, было проверено соответствие документации требованиям ЕСПД. Также были проверены построенные схемы и диаграммы на соответствие ГОСТ.

Результаты функционального тестирования компонентов программного обеспечения представлены в табл. 186.

Таблица 186

Результаты тестирования компонентов программного обеспечения

Дата	Тестируемый модуль или подпрограмма	Кто проводил тестирование	Способ тестирования	Результат тестирования
1	2	3	4	5
10.05.2022	Query.cs CheckPerson	Тестировщик	Функциональное	Неудача, люди, не подходящие под запрос, были помечены как нарушители
10.05.2022	Query.cs CheckPerson	Тестировщик	Функциональное	Успех
10.05.2022	Query.cs AddZone	Тестировщик	Функциональное	Успех
10.05.2022	Query.cs RemoveZone	Тестировщик	Функциональное	Неудача, удалена зона не была удалена из запроса
10.05.2022	Query.cs RemoveZone	Тестировщик	Функциональное	Успех
10.05.2022	Query.cs CreateReport	Тестировщик	Функциональное	Неудача, в отчёте неправильно сформировалось описание ситуации
10.05.2022	Query.cs CreateReport	Тестировщик	Функциональное	Успех

Продолжение таблицы 186

1	2	3	4	5
10.05.2022	Zone.cs CheckPoint	Тестировщик	Функциональное	Неудача, зона обнаруживает все точки, находящиеся не внутри, а снаружи
10.05.2022	Zone.cs CheckPoint	Тестировщик	Функциональное	Успех
10.05.2022	Zone.cs AddPart	Тестировщик	Функциональное	Успех
10.05.2022	Zone.cs RemovePart	Тестировщик	Функциональное	Успех
10.05.2022	Zone.cs SetMaxHumanCount	Тестировщик	Функциональное	Неудача, удалось задать отрицательное количество
11.05.2022	Zone.cs SetMaxHumanCount	Тестировщик	Функциональное	Успех
11.05.2022	Zone.cs SetDetectionInside	Тестировщик	Функциональное	Неудача, изменение параметра работает инвертировано
11.05.2022	Zone.cs SetDetectionInside	Тестировщик	Функциональное	Успех
11.05.2022	Report.cs Send	Тестировщик	Функциональное	Неудача, не удалось отправить отчёт в Teams
11.05.2022	Zone.cs Send	Тестировщик	Функциональное	Успех

После функционального тестирования, было проведено тестирование всей системы в целом на выполнение требований ТЗ (см. прил. 1), т.е. было проведено системное тестирование.

Было проведено тестирование разработанного ПО на выполнение требований к функциональным характеристикам. Результаты данного тестирования представлены в табл. 187.

Таблица 187

Тестирование разработанного ПО на выполнение требований к функциональным характеристикам

Дата	Тестируемое требование из ТЗ	Кто проводил тестирование	Способ тестирования	Результат
1	2	3	4	5
11.05.2022	отслеживание положение работников	Тестировщик	Системное тестирование	Успех
11.05.2022	отслеживание отдельных частей тела работников	Тестировщик	Системное тестирование	Неудача, некорректно преобразуются точки глаз и ушей
12.05.2022	отслеживание отдельных частей тела работников	Тестировщик	Системное тестирование	Успех
12.05.2022	определять позу работников	Тестировщик	Системное тестирование	Неудача, некорректное определение лежачего положения
12.05.2022	определять позу работников	Тестировщик	Системное тестирование	Успех

Продолжение таблицы 187

1	2	3	4	5
12.05.2022	добавление нескольких зон опасности	Тестировщик	Системное тестирование	Успех
12.05.2022	обнаружение конкретных частей тела	Тестировщик	Системное тестирование	Успех
12.05.2022	тестирование инструмента для комбинирования зон для формирования сложной опасной ситуации.	Тестировщик	Системное тестирование	Неудача, комбинирование зон работает только для обнаруженных внутри людей, а не снаружи
12.05.2022	тестирование инструмента для комбинирования зон для формирования сложной опасной ситуации.	Тестировщик	Системное тестирование	Успех

Следующим было проведено тестирование разработанного ПО на выполнение требований к надежности. Результаты данного тестирования представлены в табл. 188.

Таблица 188

Тестирование разработанного ПО на выполнение требований к надежности

Дата и время тестирования	Тестируемое требование из ТЗ	Кто проводил тестирование	Способ тестирования	Результаты тестирования
1	2	3	4	5
13.05.2022	Система должна иметь защиту от некорректных действий оператора и ошибочных исходных данных	Тестировщик	Системное тестирование	Неудача, не был запрещен ввод букв в настройках времени нахождения в зоне
13.05.2022	Система должна иметь защиту от некорректных действий оператора и ошибочных исходных данных	Тестировщик	Системное тестирование	Успех
13.05.2022	Система не должна во время работы модифицировать свой код или коды других программ	Тестировщик	Системное тестирование	Успех

Продолжение таблицы 188

1	2	3	4	5
13.05.2022	При удалении и передаче зафиксированных данных из базы данных система должна запрашивать подтверждения действия	Тестировщик	Системное тестирование	Успех
13.05.2022	Логгирование	Тестировщик	Системное тестирование	Неудача, лог-файл сохранялся по неверному пути
13.05.2022	Логгирование	Тестировщик	Системное тестирование	Успех
13.05.2022	Корректный вывод данных на экран, видео должно иметь качество не менее HD	Тестировщик	Системное тестирование	Успех
13.05.2022	контроль целостности структур баз данных, нарушение которой возможно после аппаратных сбоев	Тестировщик	Системное тестирование	Успех

Так как тестирование ПО проводилось на персональном компьютере, имеющем следующие характеристики:

- процессор Ryzen 7 2700 (8 ядер, 16 потоков, тактовая частота 4.2 ГГц);
- видеокарта Gigabyte GeForce GTX 1660Ti;
- объем оперативной памяти 16 Гб;
- твердотельный накопитель объемом 512 Гб;
- операционная система Windows 10 Домашняя;
- монитор с разрешением 1920x1080.

данные характеристики соответствуют минимальным системным требованиям, предъявляемым к оборудованию.

В результате функционального тестирования компонентов ПО были выявлены их недостатки и ошибки в работе, которые были устранены в результате доработки и повторного тестирования. В ходе системного тестирования также были устранены недостатки в работе системы. В результате программное обеспечение выполняет все заявленные требования в полном объеме.

2 Технико-экономическое обоснование выполняемой разработки

2.1 Организация работ

Для выполнения проектирования системы были составлены следующие задачи с определенной иерархической структурой (рис. 80).

Task Name	Длительност	Начало	Окончание
1 Диплом	**185 дней**	**Пт 10.09.21**	**Чт 26.05.22**
1.1 Разработка ТЗ	10 дней	Пт 10.09.21	Чт 23.09.21
1.2 Анализ отечественных и зарубежных аналогов	5 дней	Пт 24.09.21	Чт 30.09.21
1.3 Анализ предметной области	4 дней	Пт 01.10.21	Ср 06.10.21
1.4 Выбор технологии, среды и языка программирования	5 дней	Чт 07.10.21	Ср 13.10.21
1.5 Анализ процесса обработки информации, выбор структур данных для ее хранения, выбор методов и алгоритмов решения	5 дней	Чт 14.10.21	Ср 20.10.21
1.6 Разработка спецификации и архитектуры ПО	**42 дней**	**Чт 21.10.21**	**Пт 17.12.21**
1.6.1 Построение ДВИ	6 дней	Чт 21.10.21	Чт 28.10.21
1.6.2 Построение контекстной диаграммы классов	5 дней	Пт 29.10.21	Чт 04.11.21
1.6.3 Построение диаграмм последовательности системы для каждого варианта использования	7 дней	Пт 05.11.21	Пн 15.11.21
1.6.4 Построение диаграмм деятельностей сценариев вариантов использования	4 дней	Вт 16.11.21	Пт 19.11.21
1.6.5 Проектирование структур данных и построение диаграмм отношений компонентов данных	15 дней	Пн 22.11.21	Пт 10.12.21
1.6.6 Доработка разработанных диаграмм	5 дней	Пн 13.12.21	Пт 17.12.21
1.7 Проектирование системы и пользовательского интерфейса	**23 дней**	**Пн 20.12.21**	**Ср 19.01.22**
1.7.1 Проектирование системы	15 дней	Пн 20.12.21	Пт 07.01.22
1.7.2 Проектирование пользовательского интерфейса	8 дней	Пн 10.01.22	Ср 19.01.22

Рис. 80. Список задач

80. Продолжение

1.8 Разработка программного обеспечения	**85 дней**	**Чт 20.01.22**	**Ср 18.05.22**
1.8.1 Разработка пакета Presentation Layer	**27 дней**	**Чт 20.01.22**	**Пт 25.02.22**
1.8.1.1 Разработка пакета ViewModels	5 дней	Чт 20.01.22	Ср 26.01.22
1.8.1.2 Разработка пакета Vie	7 дней	Чт 27.01.22	Пт 04.02.22
1.8.1.3 Разработка пакета Models	15 дней	Пн 07.02.22	Пт 25.02.22
1.8.2 Разработка пакета Business Layer	**17 дней**	**Пн 28.02.22**	**Вт 22.03.22**
1.8.2.1 Разработка пакета Detection	7 дней	Пн 28.02.22	Вт 08.03.22
1.8.2.2 Разработка пакета Services	10 дней	Ср 09.03.22	Вт 22.03.22
1.8.3 Разработка пакета DataLay	**26 дней**	**Ср 23.03.22**	**Ср 27.04.22**
1.8.3.1 Разработка пакета Re	5 дней	Ср 23.03.22	Вт 29.03.22
1.8.3.2 Разработка пакета DataBase Sending	10 дней	Ср 30.03.22	Вт 12.04.22
1.8.3.3 Разработка пакета Teams Sending	11 дней	Ср 13.04.22	Ср 27.04.22
1.8.4 Разработка пакета CrossLa	**15 дней**	**Чт 28.04.22**	**Ср 18.05.22**
1.8.4.1 Разработка пакета Extensions	4 дней	Чт 28.04.22	Вт 03.05.22
1.8.4.2 Разработка пакета Da	5 дней	Ср 04.05.22	Вт 10.05.22
1.8.4.3 Разработка пакета Exceptions	6 дней	Ср 11.05.22	Ср 18.05.22
1.9 Тестирование	5 дней	Чт 19.05.22	Ср 25.05.22

2.2 Работа с ресурсами

Для выполнения работ были задействованы следующие ресурсы (рис. 81).

Название ресурса	Тип	Единицы измерения материалс	Краткое названи	Группа	Макс. единиц	Стандартна ставка	Ставка сверхурочных	Затраты на использ.
Microsoft Word	Материальны		MW			0,00 ₽		10 000,00 ₽
Rational Rose	Материальны		RR			0,00 ₽		0,00 ₽
Интернет	Материальны	Гб	И			100,00 ₽		0,00 ₽
Draw IO	Материальны		DI			0,00 ₽		0,00 ₽
Visual Studio 2019	Материальны		VS19			0,00 ₽		0,00 ₽
Богданов А.П.	Трудовой		БАП		100%	250,00 ₽/час	500,00 ₽/час	0,00 ₽
Нейронная сеть	Материальны		НС			0,00 ₽		0,00 ₽
Рабочая станция	Материальны		РС			0,00 ₽		0,00 ₽
Тестировщик	Трудовой		Т		100%	200,00 ₽/час	400,00 ₽/час	0,00 ₽
Учебная	Материальны	Книг	УЛ			1 000,00 ₽		0,00 ₽

Рис. 81. Используемые ресурсы

Теперь необходимо назначить ресурсы по задачам (рис. 82).

Task Name	Длительност	Начало	Окончание	П	Названия ресурсов
1 Диплом	**185 дней**	**Пт 10.09.21**	**Чт 26.05.22**		
1.1 Разработка ТЗ	10 дней	Пт 10.09.21	Чт 23.09.21		Microsoft Word[1];Богданов А.П.
1.2 Анализ отечественных и зарубежных аналогов	5 дней	Пт 24.09.21	Чт 30.09.21	2	Microsoft Word[1];Богданов А.П.;Интернет[1 Гб]
1.3 Анализ предметной области	4 дней	Пт 01.10.21	Ср 06.10.21	3	Microsoft Word[1];Богданов А.П.;Интернет[1 Гб];У
1.4 Выбор технологии, среды и языка программирования	5 дней	Чт 07.10.21	Ср 13.10.21	4	Microsoft Word[1];Богданов А.П.;Интернет[1 Гб];Учебная литература[1 Книг]
1.5 Анализ процесса обработки информации, выбор структур данных для ее хранения, выбор методов и алгоритмов решения	5 дней	Чт 14.10.21	Ср 20.10.21	5	Microsoft Word[1];Богданов А.П.;Интернет[1 Гб];Учебная литература[1 Книг]
1.6 Разработка спецификации и архитектуры ПО	**42 дней**	**Чт 21.10.21**	**Пт 17.12.21**	**6**	
1.6.1 Построение ДВИ	6 дней	Чт 21.10.21	Чт 28.10.21		Rational Rose[1];Богданов А.П.;Интернет[1 Гб];Учебная литература[1 Книг]
1.6.2 Построение контекстной диаграммы классов	5 дней	Пт 29.10.21	Чт 04.11.21	8	Rational Rose[1];Богданов А.П.;Интернет[1 Гб];Учебная литература[1 Книг]
1.6.3 Построение диаграмм последовательности системы для каждого варианта использования	7 дней	Пт 05.11.21	Пн 15.11.21	9	Rational Rose[1];Богданов А.П.;Интернет[1 Гб];Учебная литература[1 Книг]
1.6.4 Построение диаграмм деятельностей сценариев вариантов использования	4 дней	Вт 16.11.21	Пт 19.11.21	10	Rational Rose[1];Богданов А.П.;Интернет[1 Гб];Учебная литература[1 Книг]
1.6.5 Проектирование структур данных и построение диаграмм отношений компонентов данных	15 дней	Пн 22.11.21	Пт 10.12.21	11	Rational Rose[1];Богданов А.П.;Интернет[1 Гб];Учебная литература[1 Книг]
1.6.6 Доработка разработанных диаграмм	5 дней	Пн 13.12.21	Пт 17.12.21	12	Rational Rose[1];Богданов А.П.;Интернет[1 Гб];Учебная литература[1 Книг]
1.7 Проектирование системы и пользовательского интерфейса	**23 дней**	**Пн 20.12.21**	**Ср 19.01.22**	**7**	
1.7.1 Проектирование системы	15 дней	Пн 20.12.21	Пт 07.01.22		Draw IO[1];Богданов А.П.;Интернет[1 Гб];Учебная литература[1 Книг]
1.7.2 Проектирование пользовательского интерфейса	8 дней	Пн 10.01.22	Ср 19.01.22	15	Draw IO[1];Богданов А.П.;Учебная литература[1 Книг];Интернет[1 Гб]

Рис. 82. Назначенные ресурсы

82. Продолжение

Task Name	Длительност	Начало	Окончание	П	Названия ресурсов
1.8 Разработка программного обеспечения	**85 дней**	**Чт 20.01.22**	**Ср 18.05.22**	**14**	
1.8.1 Разработка пакета Presentation Layer	**27 дней**	**Чт 20.01.22**	**Пт 25.02.22**		
1.8.1.1 Разработка пакета ViewModels	5 дней	Чт 20.01.22	Ср 26.01.22		Visual Studio 2019[1];Богданов А.П.;Интернет[1 Гб];Учебная литература[1 Книг];Рабочая
1.8.1.2 Разработка пакета Vie	7 дней	Чт 27.01.22	Пт 04.02.22	19	Visual Studio 2019[1];Богданов А.П.;Интернет[1 Гб
1.8.1.3 Разработка пакета Models	15 дней	Пн 07.02.22	Пт 25.02.22	20	Microsoft Word[1];Visual Studio 2019[1];Богданов А.П.;Интернет[1 Гб];Учебная литература[1
1.8.2 Разработка пакета Business Layer	**17 дней**	**Пн 28.02.22**	**Вт 22.03.22**	**18**	
1.8.2.1 Разработка пакета Detection	7 дней	Пн 28.02.22	Вт 08.03.22		Visual Studio 2019[1];Богданов А.П.;Интернет[1 Гб];Рабочая станция[1];Учебная литература[1 Книг];Нейронная сеть[1]
1.8.2.2 Разработка пакета Services	10 дней	Ср 09.03.22	Вт 22.03.22	23	Visual Studio 2019[1];Богданов А.П.;Интернет[1 Гб];Рабочая станция[1];Учебная литература[1
1.8.3 Разработка пакета DataLay	**26 дней**	**Ср 23.03.22**	**Ср 27.04.22**	**22**	
1.8.3.1 Разработка пакета Re	5 дней	Ср 23.03.22	Вт 29.03.22		Visual Studio 2019[1];Богданов А.П.;Интернет[1 Гб
1.8.3.2 Разработка пакета DataBase Sending	10 дней	Ср 30.03.22	Вт 12.04.22	26	Visual Studio 2019[1];Богданов А.П.;Интернет[1 Гб];Рабочая станция[1];Учебная литература[1
1.8.3.3 Разработка пакета Teams Sending	11 дней	Ср 13.04.22	Ср 27.04.22	27	Visual Studio 2019[1];Богданов А.П.;Интернет[1 Гб];Рабочая станция[1];Учебная литература[1
1.8.4 Разработка пакета CrossLa	**15 дней**	**Чт 28.04.22**	**Ср 18.05.22**	**25**	
1.8.4.1 Разработка пакета Extensions	4 дней	Чт 28.04.22	Вт 03.05.22		Visual Studio 2019[1];Богданов А.П.;Интернет[1 Гб];Рабочая станция[1];Учебная литература[1
1.8.4.2 Разработка пакета Da	5 дней	Ср 04.05.22	Вт 10.05.22	30	Visual Studio 2019[1];Богданов А.П.;Интернет[1 Гб
1.8.4.3 Разработка пакета Exceptions	6 дней	Ср 11.05.22	Ср 18.05.22	31	Visual Studio 2019[1];Богданов А.П.;Рабочая станция[1];Учебная литература[1 Книг]
1.9 Тестирование	5 дней	Чт 19.05.22	Ср 25.05.22	17	Интернет[1 Гб];Нейронная сеть[1];Рабочая станция[1];Тестировщик

2.3 Критический путь проекта

Критический путь — это последовательность связанных задач, от которых непосредственно зависит дата окончания проекта. Если какая-либо задача на критическом пути выполняется с опозданием, задерживается весь проект.

Критический путь данного проекта представлен на рис. 83.

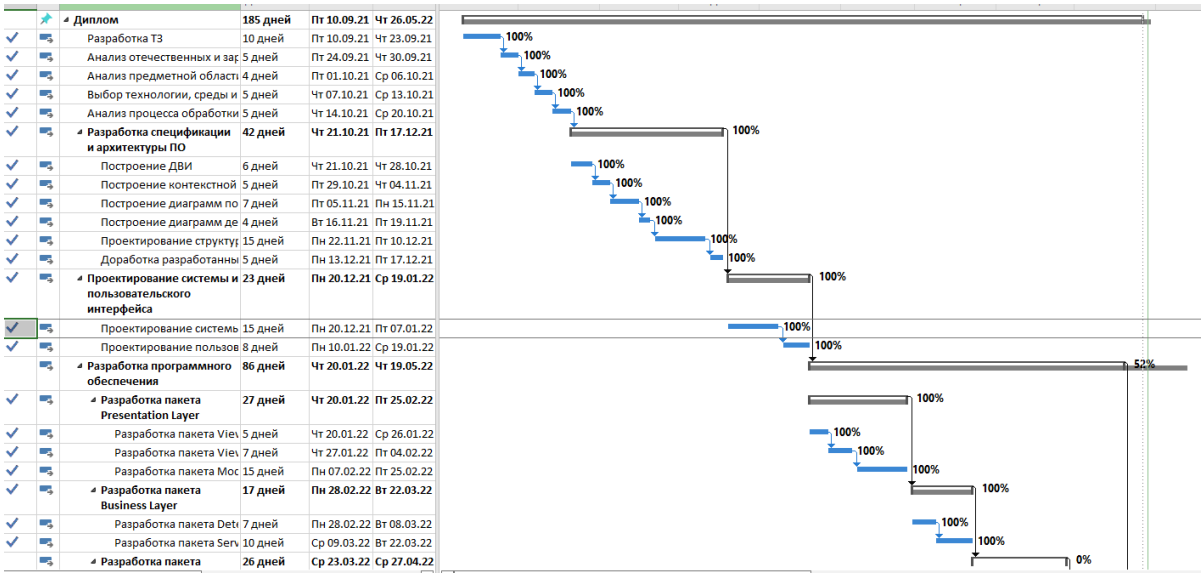
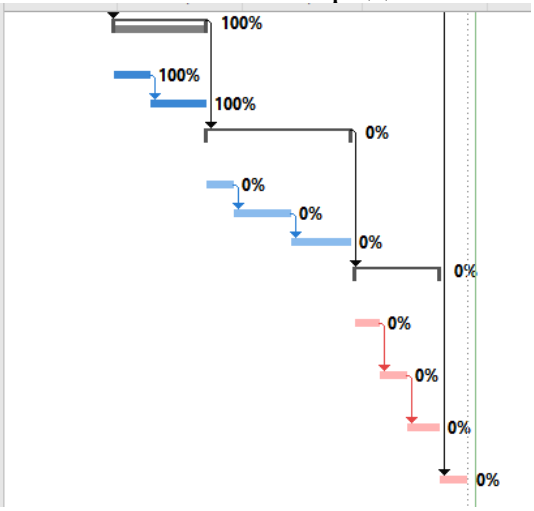


Рис. 83. Критический путь

83. Продолжение

✓		**Разработка пакета Business Layer**	**17 дней**	**Пн 28.02.22**	**Вт 22.03.22**
✓		Разработка пакета Dete	7 дней	Пн 28.02.22	Вт 08.03.22
✓		Разработка пакета Serv	10 дней	Ср 09.03.22	Вт 22.03.22
		Разработка пакета DataLayer	**26 дней**	**Ср 23.03.22**	**Ср 27.04.22**
		Разработка пакета Rea	5 дней	Ср 23.03.22	Вт 29.03.22
		Разработка пакета Data	10 дней	Ср 30.03.22	Вт 12.04.22
		Разработка пакета Tea	11 дней	Ср 13.04.22	Ср 27.04.22
		Разработка пакета CrossLayer	**15 дней**	**Пт 29.04.22**	**Чт 19.05.22**
		Разработка пакета Extensions	4 дней	Пт 29.04.22	Ср 04.05.22
		Разработка пакета Data	5 дней	Чт 05.05.22	Ср 11.05.22
		Разработка пакета Exceptions	6 дней	Чт 12.05.22	Чт 19.05.22
		Тестирование	5 дней	Пт 20.05.22	Чт 26.05.22



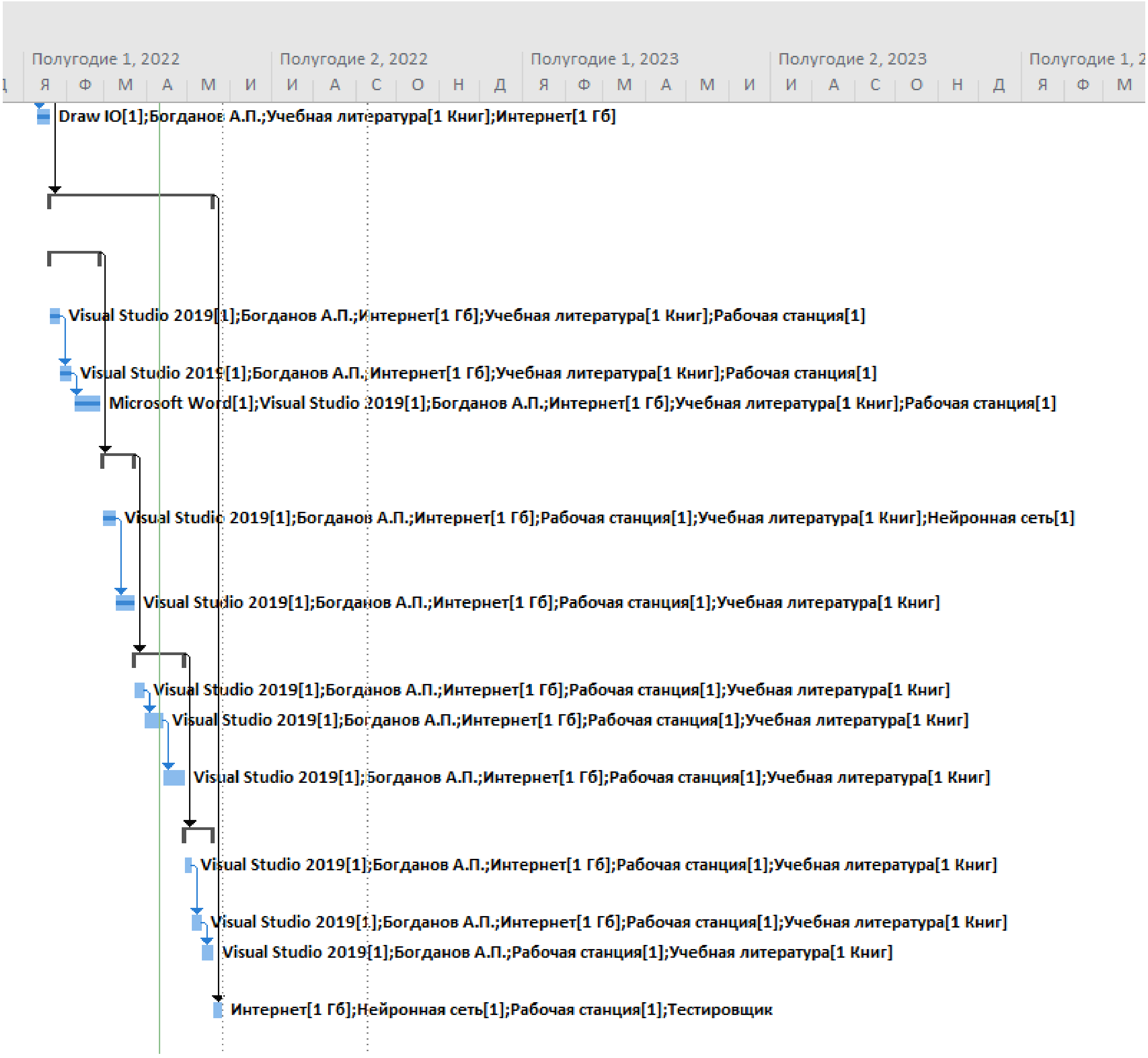
2.4 Диаграмма Ганта

Диаграмма Ганта – способ представления календарного плана проекта в виде горизонтальной гистограммы, где по вертикальной оси располагаются задачи, а по горизонтальной – даты.

Диаграмма Ганта представлена на рис. 84.

Рис. 84. Диаграмма Ганта

84. Продолжение



2.5 Расчет себестоимости продукта

Состав разработчиков: программист-дипломник, руководитель ВКР.

Затраты на оплату труда при разработке программного продукта вычисляются по формуле:

$$З_{тр} = (З_{общ} + Отч) * Т_{н}, \text{ - с окладом} \quad (1)$$

где $З_{общ}$ – общая зарплата работника за час;

Отч – отчисления с зарплаты, %;

$Т_{н}$ – время написания программы.

Заработная плата программиста за час определяется по следующей формуле:

$$З_{пр} = \frac{Ст_{пр}}{Ф_{вм}} \quad (2)$$

где $Ст_{пр}$ – ставка программиста;

$Ф_{вм}$ – фонд рабочего времени в месяц, ч.

Заработная плата дополнительная определяется по следующей формуле:

$$З_{доп} = \frac{З_{пр} \cdot Н_{доп}}{100\%} \quad (3)$$

где $З_{пр}$ – заработная плата программиста;

$Н_{доп}$ – норма отчислений на дополнительную зарплату (10%).

Зарплата общая вычисляется по следующей формуле:

$$З_{общ} = З_{пр} + З_{доп} \quad (4)$$

Отчисления на соцстрах, фонд занятости и пенсионный фонд вычисляются по следующей формуле:

$$Отч = О_{сс} + О_{фз} + О_{пф} \quad (5)$$

где $О_{сс}$ – отчисления на соцстрах (0,5% от $З_{общ}$);

$О_{фз}$ – отчисления в фонд занятости (0,5% от $З_{общ}$);

$О_{пф}$ – отчисления в пенсионный фонд (2% от $З_{общ}$).

Ставка программиста разработчика равна 29000. Ставка руководителя ВКР равна 31700р. По формуле (2) рассчитываются общие заработные платы за час программиста-дипломника и руководителя ВКР:

$$З_{пр} = \frac{29000}{160} = 181{,}25 \text{ руб.}$$

$$З_{пр} = \frac{31700}{160} = 198{,}125 \text{ руб.}$$

По формуле (3) рассчитываются дополнительные заработные платы:

$$З_{доп} = \frac{181{,}25 \cdot 0}{100\%} = 0 \text{ руб.}$$

$$З_{доп} = \frac{198{,}125 \cdot 0}{100\%} = 0 \text{ руб.}$$

Зарплаты общие вычисляются по формуле (4):

$$З_{общ} = 29000 \text{ руб.}$$

$$З_{общ} = 31700 \text{ руб.}$$

Отчисления на соцстрах, фонд занятости и пенсионный фонд вычисляются по формуле (5):

$$Отч = 145 + 145 + 580 = 870 \text{ руб.}$$

$$Отч = 158{,}5 + 158{,}5 + 634 = 951 \text{ руб.}$$

По формуле (1) рассчитываются затраты на оплату труда при разработке программного продукта:

$$З_{тр.\,прогр} = (29000 + 870) * 4{,}5 = 134415 \text{ руб.}$$

$$З_{тр.\,рук} = (31700 + 951) * 4{,}5 = 146929{,}5 \text{ руб.}$$

Все данные по заработной плате сводятся в табл. 189.

Таблица 189

Данные по заработной плате

Должность разработчика	Разряд	Время работы, мес.	$Ст_{пр}$, руб.	$З_{пр}$, руб.	$З_{доп}$, руб.	$З_{общ}$, руб.	Отч, руб.	$З_{тр}$, руб.
Программист-дипломник		4,5	29000	181,25	0	29000	870	134415
Руководитель ВКР		4,5	31700	198,125	0	31700	951	146929,5

Затраты на использование машинного времени вычисляются по формуле:

$$З_{м.вр} = С_{м.вр} \cdot Вр_{в.т} \quad (6)$$

где Зм.вр – затраты на использование машинного времени, руб.;

См.вр – стоимость одного часа машинного времени, руб./ч;

Врв.т – время использования вычислительной техники, ч.

Стоимость одного часа машинного времени рассчитывается по формуле:

$$С_{м.вр} = \frac{Ц_к}{С_{сл.к} * К_{р.д} * Вр_с} + Ст_э * М_{вс} \quad (7)$$

где См.вр – стоимость одного часа машинного времени, руб./ч;

Цк – покупная цена компьютера, руб.;

Ссл.к – срок службы компьютера, год;

Кр.д – количество рабочих дней в году;

Врс – время работы компьютера в течение суток, ч;

Стэ – стоимость одного кВт*ч электроэнергии, руб.;

Мвс – мощность вычислительной системы, кВт.

Время использования вычислительной техники рассчитывается по следующей формуле:

$$Вр_{в.т} = К_{д.р} \cdot Вр_с \quad (8)$$

где Врв.т – время использования вычислительной техники, ч;

Кд.р – количество дней разработки ПО.

Затраты на носители информации принимаются в размере 2 % от цены вычислительной техники $З_{н.и}$.

$$З_{н.и} = 800 \text{ руб.}$$

Затраты на текущий и профилактический ремонт принимаются в размере 4 % от цены вычислительной техники $З_{рем}$.

$$З_{рем} = 1600 \text{ руб.}$$

Прочие эксплуатационные расходы включают в себя затраты на освещение, отопление, охрану, уборку и текущий ремонт помещений. Они принимаются в размере 10 % от стоимости помещения (или его аренды), где происходит разработка программного продукта $З_{пр}$.

$$З_{пр} = 600 \text{ руб.}$$

Себестоимость программного продукта рассчитывается по формуле:

$$С_{пп} = З_{тр} + З_{м.вр} + З_{н.и} + З_{рем} + З_{пр} \quad (9)$$

По формуле (7) вычисляется стоимость одного часа машинного времени:

$$С_{м.вр} = \frac{40000}{4*247*9} + 5{,}11*0{,}65 = 7{,}82 \text{ руб./ч.}$$

Время использования вычислительной техники рассчитывается по формуле (8):

$$Вр_{в.т} = К_{д.р} \cdot Вр_{с} = 120 * 9 = 1080 \text{ ч.}$$

По формуле (6) вычисляются затраты на использование машинного времени:

$$З_{м.вр} = 7{,}82 \cdot 1080 = 8\,445{,}6 \text{ руб.}$$

Себестоимость разработки рассчитывается по формуле (9):

$$С_{пп} = 134415 + 146929{,}5 + 8\,445{,}6 + 800 + 1600 + 600 = 292\,790{,}1 \text{ руб.}$$

2.6 Расчёт цены программного продукта

Для определения минимальной цены, ниже которой разработчику будет невыгодно продавать программный продукт, следующая формула:

$$Ц_{п.п} = С_{п.п} \cdot (1 + Н_{пр}) \quad (10)$$

где Цп.п – цена программного продукта, руб.;

Сп.п – себестоимость программного продукта, руб.;

Нпр – норматив прибыли (20 %, в формуле Нпр = 0,2).

$$Ц_{п.п} = 292\,790{,}1 \cdot (1 + 0{,}2) = 351\,348{,}12 \text{ руб.}$$

2.7 Расчёт экономической эффективности

Стоимость одного часа машинного времени рассчитывается по формуле (7):

$$С_{м.вр} = \frac{70000}{4*365*24} + 3{,}11*0{,}65 = 4{,}019 \text{ руб./час.}$$

Объем машинного времени в течении года, необходимый для решения данной задачи с использованием программы рассчитывается по формуле (8):

$$Вр_{в.т} = 365 \cdot 24 = 8760 \text{ ч.}$$

Расходы потребителя, связанные с эксплуатацией программы, определяются по следующей формуле:

$$Р_{э.п} = Вр_{п.п} \cdot С_{м.вр} + Ц_{п.п}/С_{сл} \quad (11)$$

где $Р_{э.п}$ – эксплуатационные расходы потребителя, руб.;

$Вр_{п.п}$ – объем машинного времени в течение года, необходимый для решения данной задачи с использованием программы, ч;

$С_{м.вр}$ – стоимость одного часа машинного времени, руб./ч;

$Ц_{п.п}$ – цена программного продукта, руб.;

$С_{сл}$ – срок службы программного продукта, год. Обычно составляет 1 – 2 года, затем выпускается новая версия программного продукта.

Расходы потребителя, связанные с эксплуатацией программы рассчитываются по формуле (11):

$$Р_{э.п} = 8760 \cdot 4{,}019 + 351348{,}12/2 = 210880{,}5 \text{ руб.}$$

Капитальные затраты на вычислительную технику рассчитываются по формуле:

$$К_{ЭВМ} = Ц_{ЭВМ} + Р_{п.п} \quad (12)$$

где ЦЭВМ – цена вычислительной техники, руб.;

Рп.п – прочие расходы потребителя, связанные с помещением (отопление, освещение, уборка и т.д.), принимаются в размере 10 % от стоимости помещения потребителя (или его аренды), руб.

Капитальные затраты на вычислительную технику рассчитываются по формуле (12):

$$К_{ЭВМ} = 70000 + 6000 = 76000 \text{ руб.}$$

Капитальные затраты рассчитываются по формуле:

$$Р_{кап} = \frac{Вр_{п.п} \cdot К_{ЭВМ}}{Ф_{вр}} + Ц_{п.п} \quad (13)$$

где Ркап – капитальные расходы потребителя, руб.;

Фвр – полезный годовой фонд времени работы вычислительной техники, принимается условно 2000 ч в год;

$К_{ЭВМ}$ – капитальные затраты на вычислительную технику, для которой предназначена программа, руб.

Капитальные затраты высчитываются по формуле (13):

$$Р_{кап} = \frac{8760 \cdot 76000}{2000} + 351\,348{,}12 = 684\,228{,}12 \text{ руб.}$$

Для расчета годовой экономии эксплуатационных расходов потребителя вычисляются эксплуатационные затраты потребителя при решении задачи вручную:

$$Р_{э.руч} = 1{,}21 \cdot ФЗП \cdot 12 \quad (14)$$

где $Р_{э.руч}$ – эксплуатационные расходы потребителя при решении задачи вручную, руб.;

ФЗП – фонд заработной платы персонала, обслуживающего решение задачи вручную, руб.; 12 – количество месяцев в году; 1,21 – поправочный коэффициент.

Для расчета годовой экономии эксплуатационных расходов потребителя вычисляются эксплуатационные затраты потребителя при решении задачи вручную по формуле (14):

$$Р_{э.руч} = 1{,}21 \cdot 60000 \cdot 12 = 871200 \text{ руб.}$$

Годовая экономия эксплуатационных расходов у одного потребителя рассчитывается по формуле:

$$Э = Р_{э.руч} - Р_{э.п.} \quad (15)$$

Годовая экономия эксплуатационных расходов у одного потребителя рассчитывается по формуле (15):

$$Э = 871200 - 210880{,}5 = 660\,319{,}5 \text{ руб.}$$

Срок окупаемости программного продукта рассчитывается по формуле:

$$T_{ок} = \frac{P_{кап}}{Э} \quad (16)$$

Срок окупаемости программного продукта рассчитывается по формуле (16):

$$Т_{ок} = \frac{684\,228{,}12}{660\,319{,}5} = 1{,}036207 \text{ год.}$$

Годовой экономический эффект, получаемый одним потребителем, рассчитывается по формуле:

$$ЭЭ = Э - Е_{н} \cdot Р_{кап} \quad (17)$$

где $Е_н$ – нормативный коэффициент эффективности дополнительных капитальных вложений, равный 0,15.

Годовой экономический эффект, получаемый одним потребителем, рассчитывается по формуле (17):

$$ЭЭ = 660\ 319,5 – 0,15 \cdot 684\ 228,12 = 557\ 685,282 \text{ руб.}$$

Заключение

В результате выполнения выпускной квалификационной работы было создано программное обеспечения скелетизации изображения человека для контроля опасных действий. В ходе выполнения были выполнены следующие этапы при создании системы:

1. анализ зарубежных и отечественных аналогов;
2. проектирование программного обеспечения;
3. тестирование модулей, требований и надежности;
4. создание плана организации работ;
5. расчёт стоимости, себестоимости и срока окупаемости программного обеспечения для заказчика АО «Северсталь - Менеджмент».

На данный момент проект передан заказчику для внедрения на производство. По данному проекту написано две работы в сборник статей Череповецкого Государственного Университета.

Точность распознавания частей тела и определение положения работников равна 85%. После внедрения количество опасных ситуаций снизится на 20%.

В ходе работы над выпускной квалификационной работой были приобретены следующие компетенции:

- владение навыками использования различных технологий разработки программного обеспечения;
- владение концепциями и атрибутами качества программного обеспечения (надежности, безопасности, удобства использования), в том числе, роли людей, процессов, методов, инструментов и технологий обеспечения качества;
- владение классическими концепциями и моделями менеджмента в управлении проектам;
- способность к формализации в своей предметной области с учетом ограничений используемых методов исследования;

- готовность к использованию методов и инструментальных средств исследования объектов профессиональной деятельности;
- готовность обосновать принимаемые проектные решения, осуществлять постановку и выполнение экспериментов по проверке их корректности и эффективности;
- способность готовить презентации, оформлять научно-технические отчеты по результатам выполненной работы, публиковать результаты исследований в виде статей и докладов на научно-технических конференциях;
- способность формализовать предметную область программного проекта и разработать спецификации для компонентов программного продукта;
- способность выполнить начальную оценку степени трудности, рисков, затрат и сформировать рабочий график;
- способность готовить коммерческие предложения с вариантами решения.

Список литературы

1. Иванова, Г.С. Технология программирования: Учебник для вузов

2. Буч, Г., Язык UML. Руководство пользователя / Грейди Буч, Джеймс Рамбо, Айвар Джекобсон. – М.: ДМК, 2015. – 432 с.

3. Паттерн MVVM Определение паттерна MVVM [Электронный ресурс] URL: https://metanit.com/sharp/wpf/22.1.php (дата обращения: 13.12.2021)

4. Кейлер Адриан, Брадски Гари. Learning OpenCV: Computer Vision with the OpenCV Library. O'Reilly Media, 2008. – 556 с.

5. Герберт Шилдт. C# 4.0: полное руководство C# 4.0 The Complete Reference. Издательство — «Вильямс», 2010. — С. 1056.

6. Мэтью Мак-Дональд. WPF: Windows Presentation Foundation в .NET 4.5 с примерами на C# 5.0 для профессионалов, 4-е издание. Издательство - «Вильямс», 2013. — 1024 с.

7. Барков И.А. Объектно-ориентированное программирование. Лань, 2019 г. 700 с.

8. Human pose estimation using OpenPose with TensorFlow (part 2) [Электронный ресурс]. URL: https://arvrjourney.com/human-pose-estimation-using-openpose-with-tensorflow-part-2-e78ab9104fc8 / (дата обращения: 7.12.2021).

9. Escontrela, A. Convolutional Neural Networks from the ground up, 2018 [Электронный ресурс]. URL: https://towardsdatascience.com/convolutional-neural-networks-from-theground-up-c67bb41454e1 (дата обращения: 8.12.2021).

10. Ершов Е.В., д-р техн. наук, проф.; Виноградова Л.Н. и др. Методика и организация самостоятельной работы студентов − Коллектив авторов, ФГБОУ ВПО «Череповецкий государственный университет», 2012. −208 с.

11. Studfile [Электронный ресурс]. URL: https://studfile.net/preview/3545270/page:11/ (дата обращения: 15.12.2021)

Приложение 1

МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования

«ЧЕРЕПОВЕЦКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

Институт информационных технологий

наименование института (факультета)

Математического и программного обеспечения ЭВМ

наименование кафедры

УТВЕРЖДАЮ

Зав. кафедрой МПО ЭВМ

д.т.н., профессор____Ершов Е.В.

«__»_____20__ г.

Разработка программного обеспечения скелетизации изображений человека для контроля опасных действий

Техническое задание на выпускную квалификационную работу

Листов 6

Руководитель: Ершов Е.В.

Исполнитель: студент гр. 1ПИб-01-41оп

Богданов А.П.

Череповец, 2022 год

Введение

Производственные процессы в рабочей сфере являются крайне опасными, поэтому к обеспечению безопасности относятся всё серьёзнее. На компании “Северсталь” уже имеется множество различных способов и методик профилактики травматизма среди работников. Для того, чтобы обезопасить работу сотрудников используются видеокамеры, которые при регистрации нарушения прерывают работу агрегата или подают соответствующий сигнал. На данный момент подобные системы используются только на отдельных агрегатах. Предлагаемое решение позволит автоматически регулировать безопасность действий работников, поможет обнаружить и предотвратить деятельность в опасных зонах.

1. Основания для разработки

Основанием для разработки является задание на выпускную квалификационную работу, выданное на кафедре Математического и программного обеспечения ЭВМ Института информационных технологий по запросу заказчика АО "Северсталь менеджмент".

Дата утверждения: 10.09.2021.

Название темы разработки: «Разработка программного обеспечения скелетизации изображений человека для контроля опасных действий»

2. Назначение разработки

Проектируемое программное обеспечение предназначено повышения уровня безопасности на предприятии компании «Северсталь» путём предотвращения работы агрегата или подачи звукового сигнала при контроле передвижения и состояния сотрудников.

3. Требование к разработке

3.1 Требования к функциональным характеристикам

Необходимо разработать информационную систему контроля безопасности в реальном времени на производстве компании «Северсталь».

Она должна функционировать при помощи архитектуры толстого клиента, в которой приложение напрямую связано с базой данных. Система должна иметь поддержку нескольких типов баз данных. Обработка данных должна происходить с видеопотока в реальном времени, но для тестирования буду предоставлены специальные видеофайлы, следовательно необходимо произвести обработку с разных типов источников. Необходимо предусмотреть обработку нейронной сети для обнаружения частей тела работников с помощью видеокарты для быстроты работы.

Сформированный отчёт в MS Teams должен иметь подробное и понятное описание с прикрепленный изображением для того, чтобы в будущем можно было сделать вывод о корректности выговора.

К функциональным характеристикам разрабатываемого ПО предъявлены следующие краткие требования:

- обнаружение частей тела работников и их положение с точностью не менее 70%;
- определение позы работников с точностью не менее 90%;
- добавление нескольких зон интереса, которые можно настроить в двух вариантах: обязующая (рабочие должны находиться внутри) и опасная (рабочие должны находиться снаружи).
- наличие у каждой зоны возможности обнаружения только конкретных частей тела;
- наличие инструмента комбинирования зон для формирования сложной опасной ситуации.

3.2 Требования к надёжности

С целью предотвращения ошибок во время работы информационной системы должны быть предусмотрены следующие обработчики исключительных ситуаций:

- система должна иметь проверку от ввода некоренных данных;
- система должна иметь проверку на вывод корректных данных, т. е если произошел сбой при взаимодействии с базой данных и данные не могут быть выведены, об этом сообщается пользователю;
- все элементы пользовательского интерфейса должны корректно отображаться на экране;
- визуальная часть должна масштабироваться под размеры экрана.

3.3 Условия эксплуатации

Компьютеры и сервер предназначены для работы в закрытом отапливаемом помещении при следующих условиях:

- температура окружающего воздуха от +10°С до +35°С;
- относительная влажность воздуха не более 80%;
- запыленность воздуха не более 0,75 мг/м³;
- атмосферное давление от 630 до 800 мм ртутного столба;
- при работе с монитором расстояние от глаз должно быть 50-75 см;
- уровень шума не должен превышать 50 дБ;
- электропитание оборудования осуществляется от сети переменного тока напряжением 220 В и частотой 50 Гц.

3.4 Требование к составу и параметрам технических средств

Рекомендуемая конфигурация:

- процессора с тактовой частотой не менее 3 ГГц;
- объем ОЗУ не менее 16 Гб;

- видеокарта компании NVidia серии GTX 1060 или выше для быстрой работы нейронной сети
- объём жёсткого диска не менее 500 Гб;
- монитор, поддерживающие разрешение 1920x1080 точек.

Требования к оборудованию, формирующему видеопоток:

- разрешение видеопотока не должно быть менее 640x480 точек;
- видеокамера должна находиться непосредственно подключенной к компьютеру или быть доступна в локальной сети.

3.5 Требования к информационной и программной совместимости

Программное обеспечение должно быть разработано при помощи языка программирования C#, использовать функционал OpenCV и использовать базу данных и канал в MS Teams для фиксации нарушений техники безопасности.

Для стабильного функционирования программного обеспечения необходимо наличие операционной системы Windows 10, фреймворка .NET 5 и современного Интернет-браузера.

4. Требования к программной документации

Программная документация должна содержать расчётно-пояснительную записку (РПЗ) с содержанием: текст программы (прил. 2), спецификации (прил. 3), руководство пользователя (прил. 4).

Документация оформляется на листах формата А4 по действующим стандартам на создание документации к программному обеспечению.

5. Стадии и этапы разработки

Стадии и этапы разработки программного обеспечения представлены в табл. П1.1.

Таблица П1.1

Стадии и этапы разработки

Наименование этапа разработки	Сроки разработки	Результат выполнения	Отметка о выполнении
Разработка технического задания	12.09.2021	Готовое техническое задание	
Изучение предметной области	18.09.2021	Предметная область изучена	
Проведение сравнительного анализа аналогов проектируемого ПО	25.09.2021	Выявлены преимущества и недостатки аналогов	
Выбор технологии, среды и языка программирования	05.10.2021	Выбраны технологии, среда и языки программирования	
Анализ процесса обработки информации, выбор методов и алгоритмов для решения поставленной задачи	24.10.2021	Составлен алгоритм решения поставленной задачи	
Разработка спецификаций проектируемого ПО	20.11.2021	Разработаны спецификации проектируемого ПО	
Проектирование ПО	10.01.2022	Спроектировано ПО	
Организация работ	03.02.2022	Выполнена организация работ	
Разработка первой версии ПО	02.03.2022	Разработана первая версия ПО	
Выбор методики тестирования и тестирование первой версии ПО	10.03.2022	Протестированная первая версия ПО	
Разработка итоговой версии ПО	20.03.2022	Разработанное ПО	
Выбор методики тестирования и тестирование ПО	15.04.2022	Протестированное ПО	
Оформление документации	25.05.2022	Оформлена РПЗ со всеми приложениями	

6. Порядок контроля и приемки

Контроль выполнения работы осуществляется преподавателем в соответствии с графиком, представленным в табл. П1.2.

Таблица П1.2

Порядок контроля и приёмки

Наименование контрольного этапа выполнения курсового проекта	Сроки контроля	Результат выполнения	Отметка о приемке результата контрольного этапа
Проверка технического задания	12.09.2021	Техническое задание утверждено	
Демонстрация спроектированного ПО	10.01.2022	Спроектированное ПО согласованно	
Демонстрация финальной версии ПО	27.03.2022	Финальная версия ПО утверждена	
Демонстрация стратегии тестирования и проведённых тестов	25.04.2022	Стратегии тестирования утверждены, ПО работает исправно	
Подготовка документации	30.05.2022	Расчётно-пояснительная записка прошла норм контроль и утверждена	
Защита выпускной квалификационной работы	16.06.2022	Выпускная квалификационная работа защищена	

Приложение 2

Текст программы

Текст класса DetectionModel представлен на рис. П2.1.

```
using Emgu.CV;
using Prism.Events;
using ReactiveUI;
using ReactiveUI.Fody.Helpers;
using Skeletonization.BusinessLayer.Abstractions;
using Skeletonization.BusinessLayer.Data;
using Skeletonization.DataLayer.Data;
using
Skeletonization.PresentationLayer.Detection.Models.Abstractions;
using Skeletonization.PresentationLayer.Shared.Data;
using Skeletonization.PresentationLayer.Shared.Extensions;
using Skeletonization.PresentationLayer.Shared.Prism;
using Skeletonization.PresentationLayer.Shared.Reactive;
using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.Linq;
using System.Reactive.Linq;
using System.Threading.Tasks;
using System.Windows;
using System.Windows.Media;

namespace
Skeletonization.PresentationLayer.Detection.Models.Implementation
s
{
   internal class DetectionModel : ReactiveObject, IDetectionModel
   {
      public IVideoService VideoService { get; }
      public IFinder Finder { get; }
      public IDrawer Drawer { get; }
      public IEventAggregator EventAggregator { get; set; }

      [Reactive] public int FrameNum { get; set; }
      [Reactive] public long FrameHandlingTime { get; set; }
      [Reactive] public string VideoDescription { get; set; }

      [Reactive] public Mat Frame { get; set; }
      [Reactive] public Mat DrawedFrame { get; set; }
      [Reactive] public ImageSource FrameSource { get; set; }

      [Reactive] public IEnumerable<HumanWithRoi> Humans {
get; set; }
      public DetectionModel(IVideoService videoService,
                            IFinder finder,
                            IDrawer drawer,
                            IEventAggregator eventAggregator,

ataLayer.Implementations.DatabaseSending.SkeletonizationContext
context)
      {
         VideoService = videoService;
         Finder = finder;
         Drawer = drawer;
         EventAggregator = eventAggregator;
         var a = context.Reports.ToList();
         this.WhenAnyValue(x => x.DrawedFrame)
            .WhereNotNull()
            .Subscribe(x => FrameSource = x.ToImageSource())
            .Cashe();

         this.WhenAnyValue(x => x.Frame)
            .WhereNotNull()
.Subscribe(EventAggregator.GetEvent<FrameChanged>().Publish)
.Cashe();

         this.WhenAnyValue(x => x.Humans)
            .WhereNotNull()

.Subscribe(EventAggregator.GetEvent<HumansChanged>().Publish)
            .Cashe();
      }

      public void StartVideoFromCamera(int cameraId)
      {
         VideoService.StartCamera(cameraId, this);
         VideoDescription = $"Камера: {cameraId}";
      }

      public void StartVideoFromFile(string fileName)
      {
         VideoService.StartFile(fileName, this);
         VideoDescription = $"Файл: {fileName}";
      }

      public Task HandleFrame(FrameInfo frame)
      {
         return Application.Current?.Dispatcher?.Invoke(async () =>
         {
            var st = Stopwatch.StartNew();
            var humans = await Finder.Find(frame.Mat);

            Frame?.Dispose();
            Frame = null;
            DrawedFrame?.Dispose();
            DrawedFrame = null;
            Frame = frame.Mat;
            var copy = frame.Mat.Clone();
            if (humans.Count > 0)
            {                  Drawer.Draw(copy, humans);
            }
            DrawedFrame = copy;

            Humans = humans.Select(x => Convert(x, Frame))
                     .ToList();

            FrameNum = frame.Num;
            FrameHandlingTime = st.ElapsedMilliseconds;
         });       }       private static HumanWithRoi Convert(Human
human, Mat mat)       {
         var points = human.Points.Select
                     (bp => bp with
                     {                              Point = new
                        (
                           bp.Point.X / mat.Width,
                           bp.Point.Y / mat.Height
                        )                          }).ToList();
         return new HumanWithRoi(mat.GetRoi(points.Select(x =>
x.Point))?.ToImageSource(),
                           human.Name,
                           points);
      }       public void HandleVideoInformation(VideoInfo
videoInfo)       {       }
   }}
```

Рис. П2.1. Текст класса DetectionModel

Текст класса ZonesModel представлен на рис. П2.2.

```
using Emgu.CV;
using Prism.Events;
using ReactiveUI;
using ReactiveUI.Fody.Helpers;
using Skeletonization.BusinessLayer.Data;
using
Skeletonization.PresentationLayer.Detection.Models.Abstractions;
using Skeletonization.PresentationLayer.Shared.Data;
using Skeletonization.PresentationLayer.Shared.Extensions;
using Skeletonization.PresentationLayer.Shared.Prism;
using Skeletonization.PresentationLayer.Shared.Reactive;
using System;
using System.Collections.Concurrent;
using System.Collections.Generic;
using System.Collections.ObjectModel;
using System.Collections.Specialized;
using System.Drawing;
using System.Linq;
using System.Reactive.Disposables;
using System.Reactive.Linq;

namespace
Skeletonization.PresentationLayer.Detection.Models.Implementation
s
{
   internal class ZonesModel : ReactiveObject, IZonesModel
   {
      public IEventAggregator EventAggregator { get; }

      public ObservableCollection<Zone> Zones { get; } = new();
      [Reactive] public Zone SelectedZone { get; set; }

      [Reactive] public IEnumerable<Human> Humans { get; set; }
      private IEnumerable<Human> _humansCashe;

      [Reactive] private Mat Frame { get; set; }
      private ConcurrentDictionary<Zone, IDisposable>
_frameRoiSubs = new();

      public ZonesModel(IEventAggregator eventAggregator)
      {
         EventAggregator = eventAggregator;

         Zones.ToObservable()

.Subscribe(EventAggregator.GetEvent<ZonesChanged>().Publish)
            .Cashe();

         Zones.ToObservable(NotifyCollectionChangedAction.Add)
            .Subscribe(AddingZoneHandler)
            .Cashe();


Zones.ToObservable(NotifyCollectionChangedAction.Remove)
            .Do(RemovingZonehandler)
            .Where(x => x == SelectedZone)
            .Subscribe(_ => SelectedZone = null)
            .Cashe();

         EventAggregator.GetEvent<ZoneSelected>()
                  .ToObservable()
                  .Subscribe(zone => SelectedZone = zone)
                  .Cashe();

         EventAggregator.GetEvent<FrameChanged>()
                  .ToObservable()
                  .Do(_ => Frame = null)
                  .Subscribe(frame => Frame = frame)
                  .Cashe();

         EventAggregator.GetEvent<HumansChanged>()
                  .ToObservable()
                  .WhereNotNull()
                  .Do(x => _humansCashe = x)
                  .Subscribe(x => Humans = x)
                  .Cashe();
      }

      private void AddingZoneHandler(Zone zone)
      {
         var frameChanged = this.WhenAnyValue(x => x.Frame);

         var zonePointsFrameChanged = zone.WhenAnyValue(x =>
x.Points)

.Throttle(TimeSpan.FromMilliseconds(100))
                           .ObserveOnDispatcher()
                           .Select(_ => Frame);

         var sub = frameChanged.Merge(zonePointsFrameChanged)
                        .WhereNotNull()
                        .Select(frame =>
frame.GetRoi(zone.GetPoints()))
                        .Select(x => x?.ToImageSource())
                        .Subscribe(image => zone.ZoneRoiSource =
image);

         var zoneParametersChanged = zone.WhenAnyValue(x =>
x.Name, x => x.MinCount, x => x.Delay, x => x.CheckInside)
                              .Select(_ => _humansCashe);

         var zonePointsHumanChanged = zone.WhenAnyValue(x =>
x.Points)

.Throttle(TimeSpan.FromMilliseconds(100))
                              .ObserveOnDispatcher()
                              .Select(_ => _humansCashe);

         var humansChangedWithZoneParameters =
zoneParametersChanged.Merge(zonePointsHumanChanged);
         foreach (var selectable in zone.BodyParts)
         {
            var selectChanged = selectable.WhenAnyValue(x =>
x.IsSelected)
                              .Select(_ => _humansCashe);

            humansChangedWithZoneParameters =
humansChangedWithZoneParameters.Merge(selectChanged);
         }

         var humansChanged = this.WhenAnyValue(x => x.Humans);
         humansChangedWithZoneParameters =
humansChangedWithZoneParameters.Merge(humansChanged)

.Throttle(TimeSpan.FromMilliseconds(1))

.ObserveOnDispatcher()
                                                .WhereNotNull();

         var humansSub =
humansChangedWithZoneParameters.Subscribe(zone.Check);
         _frameRoiSubs.TryAdd(zone, new
CompositeDisposable(sub, humansSub));
      }

      private void RemovingZonehandler(Zone zone)
      {
         if (_frameRoiSubs.TryRemove(zone, out var sub))
         {
            sub.Dispose();
         }
      }
   }
}
```

Рис. П2.2. Текст класса ZonesModel

Текст класса DetectionViewModel представлен на рис. П2.3.

```
using Microsoft.Win32;
using Prism.Events;
using Prism.Services.Dialogs;
using ReactiveUI;
using ReactiveUI.Fody.Helpers;
using
Skeletonization.PresentationLayer.Detection.Models.Abstractions;
using Skeletonization.PresentationLayer.Shared.Data;
using Skeletonization.PresentationLayer.Shared.Extensions;
using Skeletonization.PresentationLayer.Shared.Prism;
using Skeletonization.PresentationLayer.Shared.Reactive;
using System;
using System.Reactive.Linq;
using System.Windows.Input;

namespace
Skeletonization.PresentationLayer.Detection.ViewModels
{
   internal class DetectionViewModel : ZonesConsumer,
IReactiveObject
   {
      public IDetectionModel Model { get; }
      public IDialogService DialogService { get; }
      [Reactive] public Zone SelectedZone { get; set; }

      public ICommand StartVideoFromFileCommand { get; }
      public ICommand StartVideoFromCameraCommand { get; }

      public DetectionViewModel(IDetectionModel model,
                        IEventAggregator eventAggregator,
                        IDialogService dialogService)
         : base(eventAggregator)
      {
         Model = model;
         DialogService = dialogService;
         this.WhenAnyValue(x => x.SelectedZone)
            .WhereNotNull()
            .Subscribe(x =>
            {
               EventAggregator.GetEvent<ZoneSelected>().Publish(x);
               SelectedZone = null;
            })
            .Cashe();
         StartVideoFromFileCommand = ReactiveCommand.Create(()
=>          {              OpenFileDialog openFileDialog = new();
            string videoExtensions =
"Видео".ConcatExtensions("AVI", "MP4", "MPEG", "MOV");
            string imageExtensions =
"Изображения".ConcatExtensions("BMP", "JPEG", "JPG", "PNG");
            openFileDialog.Filter = string.Join('|', videoExtensions,
imageExtensions);
            if (openFileDialog.ShowDialog() == true)
            {
               Model.StartVideoFromFile(openFileDialog.FileName);
            }
         });
         StartVideoFromCameraCommand =
ReactiveCommand.CreateFromTask(async () =>
         {
            var (ok, device) = await
DialogService.OpenCameraChooseDialog();
            if (ok)
            {
               Model.StartVideoFromCamera(device.Id);
            }
         });
      }
   }
}
```

Рис. П2.3. Текст класса DetectionViewModel

Текст класса OpenCameraDialogViewModel представлен на рис. П2.4.

```
using Prism.Services.Dialogs;
using ReactiveUI;
using ReactiveUI.Fody.Helpers;
using Skeletonization.BusinessLayer.Abstractions;
using Skeletonization.DataLayer.Abstractions;
using Skeletonization.DataLayer.Data;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Reactive.Linq;
using System.Windows.Input;
namespace
Skeletonization.PresentationLayer.Detection.ViewModels
{    internal class OpenCameraDialogViewModel : ReactiveObject,
IDialogAware
   {       public event Action<IDialogResult> RequestClose;
      public string Title => "Выбор камеры";

      public IVideoService VideoService { get; }
      [Reactive] public IEnumerable<VideoDeviceInfo>
VideoDevices { get; set; }
      [Reactive] public VideoDeviceInfo SelectedDevice { get; set; }
      public ICommand CompleteCommand { get; }
      public ICommand CancelCommand { get; }

      public OpenCameraDialogViewModel(IVideoService
videoService)
      {
         VideoService = videoService;
         VideoDevices = VideoService.GetVideoDevices();

         CompleteCommand = ReactiveCommand.Create(() =>
         {
            DialogParameters parameters = new();
            parameters.Add("device", SelectedDevice);
            RequestClose?.Invoke(new
DialogResult(ButtonResult.OK, parameters));
         }, this.WhenAnyValue(x => x.SelectedDevice).Select(x => x
is not null));
         CancelCommand = ReactiveCommand.Create(() =>
RequestClose?.Invoke(new DialogResult(ButtonResult.Cancel)));
      }
      public bool CanCloseDialog()
      {          return true;
      }       p
ublic void OnDialogClosed()
      {
      }
      public void OnDialogOpened(IDialogParameters parameters)
      {       }
   }
}
```

Рис. П2.4. Текст класса OpenCameraDialogViewModel

Текст класса QueriesViewModel представлен на рис. П2.5.

```
using Prism.Events;
using ReactiveUI;
using
Skeletonization.PresentationLayer.Detection.Models.Abstractions;
using Skeletonization.PresentationLayer.Shared.Data;
using Skeletonization.PresentationLayer.Shared.Extensions;
using Skeletonization.PresentationLayer.Shared.Prism;
using System;
using System.Windows.Input;

namespace
Skeletonization.PresentationLayer.Detection.ViewModels
{
   internal class QueriesViewModel : ZonesConsumer
   {
      public IQueriesModel Model { get; }

      public ICommand AddQueryCommand { get; }
      public ICommand RemoveQueryCommand { get; }
      public ICommand AddZoneToQueryCommand { get; }
      public ICommand RemoveZoneFromQueryCommand { get; }

      public QueriesViewModel(IQueriesModel model,
IEventAggregator eventAggregator)
         : base(eventAggregator)
      {
         Model = model;

         AddQueryCommand = ReactiveCommand.Create(() =>
Model.Queries.Add(new()));
         RemoveQueryCommand =
ReactiveCommand.Create<Query>(x =>
Model.Queries.Remove(x));

         AddZoneToQueryCommand =
ReactiveCommand.Create<(Zone, Query)>
         (
            x =>
            {
               if (x.Item1 is null ||
x.Item2.QueriesZones.Contains(x.Item1))
               {
                  EventAggregator.GetEvent<NotificationSended>()
                        .Publish("Выбранная зона пустая или уже
используется запросом");
                  return;
               }

               x.Item2.QueriesZones.Add(x.Item1);
            }
         );

         RemoveZoneFromQueryCommand =
ReactiveCommand.Create<(Zone, Query)>(x =>
x.Item2.QueriesZones.Remove(x.Item1));
      }
   }
}
```

Рис. П2.5. Текст класса QueriesViewModel

Текст класса ZonesViewModel представлен на рис. П2.6.

```
using ReactiveUI;
using Skeletonization.CrossfulLayer.Abstractions;
using
Skeletonization.PresentationLayer.Detection.Models.Abstractions;
using Skeletonization.PresentationLayer.Shared.Data;
using System.Reactive.Linq;
using System.Windows.Input;

namespace
Skeletonization.PresentationLayer.Detection.ViewModels
{
   internal class ZonesViewModel : ReactiveObject
   {
      public IZonesModel Model { get; }
      public IFactory<Zone> ZoneFactory { get; }

      public ICommand AddZoneCommand { get; }
      public ICommand RemoveZoneCommand { get; }

      public ZonesViewModel(IZonesModel model, IFactory<Zone>
zoneFactory)
      {
         Model = model;
         ZoneFactory = zoneFactory;

         AddZoneCommand = ReactiveCommand.Create(() =>
         {
            var zone = ZoneFactory.Create();
            Model.Zones.Add(zone);
            Model.SelectedZone = zone;
         });

         RemoveZoneCommand = ReactiveCommand.Create
         (
            () => Model.Zones.Remove(Model.SelectedZone),
            Model.WhenAnyValue(x => x.SelectedZone)
               .Select(x => x is not null)
         );
      }
   }
}
```

Рис. П2.6. Текст класса ZonesViewModel

Текст класса DetectionControl представлен на рис. П2.7.

```
<UserControl

x:Class="Skeletonization.PresentationLayer.Detection.Views.DetectionControl"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
   xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
   xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
   xmlns:local="clr-namespace:Skeletonization.PresentationLayer.Detection.Views"
   xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
   xmlns:md="http://materialdesigninxaml.net/winfx/xaml/themes"
   xmlns:prism="http://prismlibrary.com/"
   xmlns:sh="clr-namespace:Skeletonization.PresentationLayer.Shared.Converters;assembly=Skeletonization.Shared"
   xmlns:shD="clr-namespace:Skeletonization.PresentationLayer.Shared.Data;assembly=Skeletonization.Shared"
   xmlns:vm="clr-namespace:Skeletonization.PresentationLayer.Detection.ViewModels"
   x:Name="control"
   d:DataContext="{d:DesignInstance Type=vm:DetectionViewModel}"
   d:DesignHeight="450"
   d:DesignWidth="800"
   mc:Ignorable="d">
   <Grid>
      <Grid.RowDefinitions>
         <RowDefinition Height="auto" />
         <RowDefinition />
      </Grid.RowDefinitions>
      <Grid.ColumnDefinitions>
         <ColumnDefinition MinWidth="100" />
         <ColumnDefinition Width="360" />
         <ColumnDefinition Width="360" />
      </Grid.ColumnDefinitions>
      <StackPanel
         Grid.Row="0"
         Grid.Column="0"
         Grid.ColumnSpan="3">
         <Menu Height="40">
            <Menu.Resources>
               <Style BasedOn="{StaticResource MaterialDesignBody1TextBlock}" TargetType="TextBlock">
                  <Setter Property="Margin" Value="10,-6,0,0" />
               </Style>
            </Menu.Resources>
            <MenuItem>
               <MenuItem.Header>
                  <StackPanel Orientation="Horizontal">
                     <md:PackIcon Kind="VideoPlus" />
                     <TextBlock Text="Открыть видеопоток" />
                  </StackPanel>
               </MenuItem.Header>               <MenuItem Command="{Binding StartVideoFromFileCommand}">
<MenuItem.Header>                  <StackPanel Orientation="Horizontal">
                     <md:PackIcon Kind="File" />
                     <TextBlock Text="Из файла" />
                  </StackPanel>
               </MenuItem.Header>
            </MenuItem>
            <MenuItem Command="{Binding StartVideoFromCameraCommand}">
               <MenuItem.Header>
                  <StackPanel Orientation="Horizontal">
                     <md:PackIcon Kind="Camera" />
                     <TextBlock Text="С камеры" />
                  </StackPanel>
               </MenuItem.Header>
            </MenuItem>
         </Menu>
         <WrapPanel>
            <WrapPanel.Resources>
               <Style BasedOn="{StaticResource MaterialDesignCaptionTextBlock}" TargetType="TextBlock">
                  <Setter Property="Margin" Value="0,0,10,0" />
                  <Setter Property="VerticalAlignment" Value="Center" />
               </Style>
            </WrapPanel.Resources>
            <TextBlock Text="{Binding Model.FrameNum, StringFormat='Кадр: {0}'}" />
            <TextBlock Text="{Binding Model.FrameHandlingTime, StringFormat='Время обработки: {0}'}" />
            <TextBlock Text="{Binding Model.VideoDescription}" />
         </WrapPanel>
      </StackPanel>
      <Border
         Grid.Row="1"
         Grid.Column="0"
         BorderBrush="{StaticResource MaterialDesignBody}"
         BorderThickness="3">
         <Image
            x:Name="image"
            Grid.Row="1"
            Grid.Column="0"
            Margin="10"
            HorizontalAlignment="Center"
            VerticalAlignment="Center"
            Source="{Binding Model.FrameSource}"
            Stretch="Uniform" />
      </Border>
      <Canvas
         Grid.Row="1"
         Grid.Column="0"
         Width="{Binding ActualWidth, ElementName=image, Mode=OneWay}"
         Height="{Binding ActualHeight, ElementName=image, Mode=OneWay}"
         Margin="{Binding Margin, ElementName=image, Mode=OneWay}"
         HorizontalAlignment="{Binding HorizontalAlignment, ElementName=image, Mode=OneWay}"
         VerticalAlignment="{Binding VerticalAlignment, ElementName=image, Mode=OneWay}"
         Background="Transparent">
         <Canvas.Resources>
            <Style TargetType="local:ZonePoint">
               <Setter Property="Width" Value="25" />
               <Setter Property="Height" Value="25" />
               <Setter Property="Fill" Value="{Binding Color, Converter={sh:ColorConverter}}" />
            </Style>
            <Style TargetType="local:ZonePolygon">
               <Setter Property="Opacity" Value="{Binding Opacity}" />               <Setter Property="Fill" Value="{Binding Color, Converter={sh:ColorConverter}}" />
            </Style>
         </Canvas.Resources>
         <local:ZonesMapper
            x:Name="mapper"
            SelectedZone="{Binding SelectedZone, Mode=TwoWay}"
            Zones="{Binding Zones}">
            <local:ZonesMapper.DataTemplates>
               <DataTemplate>
                  <local:ZonePolygon d:DataContext="{d:DesignInstance Type=shD:Zone}" Zone="{Binding}" />
               </DataTemplate>
               <DataTemplate>
                  <local:ZonePoint d:DataContext="{d:DesignInstance Type=shD:Zone}" Point="{Binding LeftTop,
```

Рис. П2.7. Текст класса DetectionControl

П2.7. Продолжение

```
Converter={sh:PointConverter}, Mode=TwoWay}" />
                </DataTemplate>
                <DataTemplate>
                    <local:ZonePoint d:DataContext="{d:DesignInstance
Type=shD:Zone}" Point="{Binding RightTop,
Converter={sh:PointConverter}, Mode=TwoWay}" />
                </DataTemplate>
                <DataTemplate>
                    <local:ZonePoint d:DataContext="{d:DesignInstance
Type=shD:Zone}" Point="{Binding RightBot,
Converter={sh:PointConverter}, Mode=TwoWay}" />
                </DataTemplate>
                <DataTemplate>
                    <local:ZonePoint d:DataContext="{d:DesignInstance
Type=shD:Zone}" Point="{Binding LeftBot,
Converter={sh:PointConverter}, Mode=TwoWay}" />
                </DataTemplate>
            </local:ZonesMapper.DataTemplates>
        </local:ZonesMapper>
    </Canvas>
    <ContentControl
        Grid.Row="1"
        Grid.Column="1"
        Margin="10"
        prism:RegionManager.RegionName="{x:Static
local:Regions.Zones}" />
    <ContentControl
        Grid.Row="1"
        Grid.Column="2"
        Margin="10"
        prism:RegionManager.RegionName="{x:Static
local:Regions.Queries}" />
  </Grid>
</UserControl>
```

Текст класса OpenCameraDialogControl представлен на рис. П2.8.

```
<UserControl

x:Class="Skeletonization.PresentationLayer.Detection.Views.OpenC
ameraDialogControl"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation
"
  xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
  xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
  xmlns:local="clr-
namespace:Skeletonization.PresentationLayer.Detection.Views"
  xmlns:mc="http://schemas.openxmlformats.org/markup-
compatibility/2006"
  xmlns:vm="clr-
namespace:Skeletonization.PresentationLayer.Detection.ViewModel
s"
  Width="300"
  Height="auto"
  d:DataContext="{d:DesignInstance
Type=vm:OpenCameraDialogViewModel}"
  mc:Ignorable="d">
  <Grid>
    <Grid.RowDefinitions>
      <RowDefinition Height="60" />
      <RowDefinition Height="auto" />
    </Grid.RowDefinitions>
    <Grid.ColumnDefinitions>
      <ColumnDefinition Width="auto" />
      <ColumnDefinition />
    </Grid.ColumnDefinitions>
    <TextBlock
      Grid.Row="0"
      Grid.Column="0"
      VerticalAlignment="Center"
      Text="Камера: " />
    <ComboBox
      Grid.Row="0"
      Grid.Column="1"
      VerticalAlignment="Center"
      IsDropDownOpen="True"
      ItemsSource="{Binding VideoDevices}"
      SelectedItem="{Binding SelectedDevice}">
      <ComboBox.ItemTemplate>
        <DataTemplate>
          <TextBlock>
            <Run Text="{Binding Id}" />
            <Run Text=": " />
            <Run Text="{Binding Name}" />
          </TextBlock>
        </DataTemplate>
      </ComboBox.ItemTemplate>
    </ComboBox>
    <UniformGrid
      Grid.Row="1"
      Grid.Column="0"
      Grid.ColumnSpan="2"
      Rows="1">
      <Button Command="{Binding CompleteCommand}"
Content="Выбрать" />
      <Button Command="{Binding CancelCommand}"
Content="Отменить" />
    </UniformGrid>
  </Grid>
</UserControl>
```

Рис. П2.8. Текст класса OpenCameraDialogControl

Текст класса QueriesControl представлен на рис. П2.8.

```
<UserControl

x:Class="Skeletonization.PresentationLayer.Detection.Views.Querie
sControl"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation
"
  xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
  xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
  xmlns:local="clr-
namespace:Skeletonization.PresentationLayer.Detection.Views"
```

Рис. П2.8. Текст класса QueriesControl

П2.8. Продолжение

```
xmlns:mc="http://schemas.openxmlformats.org/markup-
compatibility/2006"
   xmlns:md="http://materialdesigninxaml.net/winfx/xaml/themes"
   xmlns:sh="clr-
namespace:Skeletonization.PresentationLayer.Shared.Converters;ass
embly=Skeletonization.Shared"
   xmlns:vm="clr-
namespace:Skeletonization.PresentationLayer.Detection.ViewModel
s"
   x:Name="parent"
   d:DataContext="{d:DesignInstance
Type=vm:QueriesViewModel}"
   d:DesignHeight="500.467"
   d:DesignWidth="350"
   mc:Ignorable="d">
   <UserControl.Resources>
      <Style BasedOn="{StaticResource
MaterialDesignFloatingHintTextBox}" TargetType="TextBox">
         <Setter Property="FontSize" Value="20" />
         <Setter Property="Margin" Value="4" />
         <Setter Property="md:TextFieldAssist.HasClearButton"
Value="True" />
      </Style>
      <Style BasedOn="{StaticResource
MaterialDesignBody1TextBlock}" TargetType="TextBlock">
         <Setter Property="FontSize" Value="20" />
         <Setter Property="Margin" Value="4,4,10,4" />
      </Style>
   </UserControl.Resources>
   <Grid>
      <Grid.RowDefinitions>
         <RowDefinition Height="auto" />
         <RowDefinition />
      </Grid.RowDefinitions>
      <Button
         Grid.Row="0"
         HorizontalAlignment="Right"
         Command="{Binding AddQueryCommand}">
         <md:PackIcon Kind="AddCircle" />
      </Button>
      <ScrollViewer Grid.Row="1">
         <ItemsControl ItemsSource="{Binding Model.Queries}">
            <ItemsControl.ItemTemplate>
               <DataTemplate>
                  <Border
                     Margin="0,0,0,10"
                     BorderBrush="{StaticResource
MaterialDesignBody}"
                     BorderThickness="3">
                     <StackPanel x:Name="queryPanel">
                        <Button
                           HorizontalAlignment="Right"
                           Command="{Binding
DataContext.RemoveQueryCommand, ElementName=parent}"
                           CommandParameter="{Binding}">
                           <md:PackIcon Kind="DeleteCircle" />
                        </Button>
                        <WrapPanel>
                           <TextBlock Text="Инвертрование:" />
                           <ToggleButton IsChecked="{Binding
IsInverted}" />
                        </WrapPanel>
                        <WrapPanel>
                           <TextBlock Text="Поиск в зоне:" />
                           <ToggleButton Name="checkTogle"
IsChecked="{Binding CheckInZone}" />
                           <ComboBox
                              MinWidth="100"
                              IsEnabled="{Binding IsChecked,
ElementName=checkTogle}"
                              ItemsSource="{Binding QueriesZones}"
                              SelectedItem="{Binding CheckingZone}">

                              <ComboBox.ItemTemplate>
                                 <DataTemplate>
                                    <TextBlock Text="{Binding Name}"
/>
                                 </DataTemplate>
                              </ComboBox.ItemTemplate>
                           </ComboBox>
                        </WrapPanel>
                        <TextBlock Text="Зоны запроса:" />
                        <ItemsControl ItemsSource="{Binding
QueriesZones}">
                           <ItemsControl.ItemTemplate>
                              <DataTemplate>
                                 <WrapPanel x:Name="zonePanel">
                                    <TextBlock Text="{Binding Name}"
/>
                                    <Button Command="{Binding
DataContext.RemoveZoneFromQueryCommand,
ElementName=parent}">
                                       <md:PackIcon Kind="AddCall" />
                                       <Button.CommandParameter>
                                          <MultiBinding
Converter="{sh:QueryZoneConverter}">
                                             <Binding />
                                             <Binding
ElementName="queryPanel" Path="DataContext" />
                                          </MultiBinding>
                                       </Button.CommandParameter>
                                    </Button>
                                 </WrapPanel>
                              </DataTemplate>
                           </ItemsControl.ItemTemplate>
                        </ItemsControl>
                        <WrapPanel>
                           <TextBlock Text="Добавление зоны" />
                           <ComboBox
                              MinWidth="100"
                              ItemsSource="{Binding
DataContext.Zones, ElementName=parent}"
                              SelectedItem="{Binding SelectedZone}">
                              <ComboBox.ItemTemplate>
                                 <DataTemplate>
                                    <TextBlock Text="{Binding Name}"
/>
                                 </DataTemplate>
                              </ComboBox.ItemTemplate>
                           </ComboBox>
                           <Button Command="{Binding
DataContext.AddZoneToQueryCommand, ElementName=parent}"
CommandParameter="{Binding}">
                              <md:PackIcon Kind="AddCall" />
                           </Button>
                        </WrapPanel>
                     </StackPanel>
                  </Border>
               </DataTemplate>
            </ItemsControl.ItemTemplate>
         </ItemsControl>
      </ScrollViewer>
   </Grid>
</UserControl>
```

Текст класса ZoneControl представлен на рис. П2.9.

```
<UserControl
x:Class="Skeletonization.PresentationLayer.Detection.Views.ZoneC
ontrol"
xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation
xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
   xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
   xmlns:local="clr-
namespace:Skeletonization.PresentationLayer.Detection.Views"
   xmlns:mc="http://schemas.openxmlformats.org/markup-
compatibility/2006"
   xmlns:md="http://materialdesigninxaml.net/winfx/xaml/themes"
   xmlns:sh="clr-
namespace:Skeletonization.PresentationLayer.Shared.Data;assembly
=Skeletonization.Shared"
   xmlns:shC="clr-
namespace:Skeletonization.PresentationLayer.Shared.Converters;ass
embly=Skeletonization.Shared"
   xmlns:shEx="clr-
namespace:Skeletonization.PresentationLayer.Shared.Extensions;ass
embly=Skeletonization.Shared"
   xmlns:shV="clr-
namespace:Skeletonization.PresentationLayer.Shared.Views;assembl
y=Skeletonization.Shared"
   d:DataContext="{d:DesignInstance Type=sh:Zone}"
   d:DesignHeight="500.467"
   d:DesignWidth="350"
   mc:Ignorable="d">
   <UserControl.Resources>
      <Style BasedOn="{StaticResource
MaterialDesignFloatingHintTextBox}" TargetType="TextBox">
         <Setter Property="FontSize" Value="20" />
         <Setter Property="Margin" Value="4" />
         <Setter Property="md:TextFieldAssist.HasClearButton"
Value="True" />
      </Style>
      <Style BasedOn="{StaticResource
MaterialDesignBody1TextBlock}" TargetType="TextBlock">
         <Setter Property="FontSize" Value="20" />
         <Setter Property="Margin" Value="4,4,10,4" />
      </Style>
   </UserControl.Resources>
   <ScrollViewer>
      <StackPanel>
         <Border BorderBrush="{StaticResource
MaterialDesignBody}" BorderThickness="2">
            <Image
               Height="140"                Source="{Binding
ZoneRoiSource}"
               Stretch="Uniform" />          </Border>
<TextBox md:HintAssist.Hint="Имя" Text="{Binding Name,
Mode=TwoWay, UpdateSourceTrigger=PropertyChanged}" />
         <TextBox md:HintAssist.Hint="Мин. кол-во людей"
Text="{Binding MinCount}" />
         <TextBox md:HintAssist.Hint="Задержка" Text="{Binding
Delay}" />          <WrapPanel>
            <TextBlock VerticalAlignment="Center"
Text="Обнаруживает внутри:" />
            <ToggleButton
               HorizontalAlignment="Left"
               VerticalAlignment="Center"
               IsChecked="{Binding CheckInside}" />
         </WrapPanel>
         <WrapPanel>
            <TextBlock VerticalAlignment="Center"
Text="Прозрачность:" />
            <Slider Width="150"     VerticalAlignment="Center"
               Maximum="1"  Minimum="0"
TickFrequency="0.1"
               Value="{Binding Opacity}" />
         </WrapPanel>
         <WrapPanel>
            <TextBlock VerticalAlignment="Center" Text="Цвет:" />
            <ComboBox
               Width="150"
               ItemsSource="{Binding Source={x:Static
shEx:ColorExtensionsMethods.Colors}}"
               SelectedValue="{Binding Color,
Converter={shC:ColorConverter}}">
               <ComboBox.ItemTemplate>
                  <DataTemplate>
                     <Border BorderBrush="{StaticResource
MaterialDesignBody}" BorderThickness="2">
                        <Rectangle Height="25" Fill="{Binding}" />
                     </Border>
                  </DataTemplate>
               </ComboBox.ItemTemplate>
            </ComboBox>
         </WrapPanel>
         <shV:BodyPartGrouper BodyParts="{Binding BodyParts}"
/>
         <ItemsControl ItemsSource="{Binding
FailedCheckingElements}">
            <ItemsControl.ItemsPanel>
               <ItemsPanelTemplate>
                  <StackPanel />                </ItemsPanelTemplate>
</ItemsControl.ItemsPanel>             <ItemsControl.ItemTemplate>
<DataTemplate>
                  <shV:HumanResultControl BodyParts="{Binding
DetectedBodyParts}" Human="{Binding Human}" />
               </DataTemplate>
</ItemsControl.ItemTemplate>          </ItemsControl>
</StackPanel> </ScrollViewer></UserControl>
```

Рис. П2.9. Текст класса ZoneControl

Текст класса ZonesControl представлен на рис. П2.10.

```
<UserControl
x:Class="Skeletonization.PresentationLayer.Detection.Views.Zones
Control"
xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation
"    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
   xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
   xmlns:local="clr-
namespace:Skeletonization.PresentationLayer.Detection.Views"
   xmlns:mc="http://schemas.openxmlformats.org/markup-
compatibility/2006"    xmlns:sh="clr-
namespace:Skeletonization.PresentationLayer.Shared.Converters;ass
embly=Skeletonization.Shared"    xmlns:vm="clr-
namespace:Skeletonization.PresentationLayer.Detection.ViewModel
s"    d:DataContext="{d:DesignInstance
Type=vm:ZonesViewModel}"    d:DesignHeight="708.5"
   d:DesignWidth="485"
   mc:Ignorable="d">    <Grid>
      <Grid.RowDefinitions>
         <RowDefinition Height="auto" />
         <RowDefinition />
      </Grid.RowDefinitions>
      <UniformGrid Grid.Row="0" Rows="1">
         <Button Command="{Binding AddZoneCommand}"
Content="Добавить" />
         <Button Command="{Binding RemoveZoneCommand}"
Content="Удалить" />       </UniformGrid>       <local:ZoneControl
Grid.Row="1"          DataContext="{Binding
Model.SelectedZone}"
         IsEnabled="{Binding Converter={sh:IsNullableConverter}}"
/>    </Grid></UserControl>
```

Рис. П2.10. Текст класса ZonesControl

Текст класса ZonePoint представлен на рис. П2.11.

```
using Skeletonization.PresentationLayer.Shared.Reactive;
using System;
using System.Reactive.Disposables;
using System.Reactive.Linq;
using System.Windows;
using System.Windows.Controls;

namespace Skeletonization.PresentationLayer.Detection.Views
{
   public class ZonePoint : ZoneShapeBase
   {
      public Point Point
      {
         get => (Point)GetValue(PointProperty);
         set => SetValue(PointProperty, value);
      }

      public static readonly DependencyProperty PointProperty =
            DependencyProperty.Register(nameof(Point),
                                        typeof(Point),
                                        typeof(ZonePoint),
                                        new(PointChangedCallback));

      private IDisposable _sub;
      private bool _isDragging;

      static ZonePoint()
      {

DefaultStyleKeyProperty.OverrideMetadata(typeof(ZonePoint), new
FrameworkPropertyMetadata(typeof(ZonePoint)));
      }

      protected override void OnLoaded(object sender,
RoutedEventArgs e)
      {
         var parent = Parent as Panel;

         var parentSizeChanged = parent.SizeChangedObservable()
                                       .Subscribe(_ => DrawPoint());

         var movingSub = parent.MouseMoveObservable()
                               .Where(_ => _isDragging)
                               .Select(x => x.GetPosition(parent))
                               .Select(FromParent)
                               .Subscribe(p => Point = p);

         var parentMouseUp = parent.MouseUpObservable()
                                   .Select(_ => false);

         var mouseDown = this.MouseDownObservable()
                             .Select(_ => true);

         var draggingSub = parentMouseUp.Merge(mouseDown)
                                        .Subscribe(x => _isDragging = x);

         _sub = new CompositeDisposable(parentSizeChanged,
movingSub, draggingSub);
      }

      protected override void OnUnloaded(object sender,
RoutedEventArgs e)
      {
         _sub.Dispose();
      }

      private static void PointChangedCallback(DependencyObject d,
DependencyPropertyChangedEventArgs e)
      {
         (d as ZonePoint).DrawPoint();
      }

      private void DrawPoint()
      {
         var point = ToParent(Point);
         Canvas.SetLeft(this, point.X - Width / 2);
         Canvas.SetTop(this, point.Y - Height / 2);
      }
   }
}
```

Рис. П2.11. Текст класса ZonePoint

Текст класса ZonePolygone представлен на рис. П2.12.

```
using ReactiveUI;
using Skeletonization.PresentationLayer.Shared.Data;
using Skeletonization.PresentationLayer.Shared.Reactive;
using System;
using System.Linq;
using System.Reactive.Disposables;
using System.Reactive.Linq;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Media;
using D = Skeletonization.BusinessLayer.Data;

namespace Skeletonization.PresentationLayer.Detection.Views
{
   public class ZonePolygon : ZoneShapeBase
   {
      public Zone Zone
      {
         get => (Zone)GetValue(ZoneProperty);
         set => SetValue(ZoneProperty, value);
      }

      public PointCollection Points
      {
         get => (PointCollection)GetValue(PointsProperty);
         private set => SetValue(PointsProperty, value);
      }

      public static readonly DependencyProperty ZoneProperty =
        DependencyProperty.Register(nameof(Zone),
                            typeof(Zone),
                            typeof(ZonePolygon));

      public static readonly DependencyProperty PointsProperty =
       DependencyProperty.Register(nameof(Points),
                           typeof(PointCollection),
                           typeof(ZonePolygon));

      private IDisposable _sub;
      private bool _isDragging;
      private Point _lastPosition;
      static ZonePolygon()
      {

DefaultStyleKeyProperty.OverrideMetadata(typeof(ZonePolygon),
          new FrameworkPropertyMetadata(typeof(ZonePolygon)));
      }
      protected override void OnLoaded(object sender,
RoutedEventArgs e)
      {
```

Рис. П2.12. Текст класса ZonePolygone

П2.12. Продолжение

```
 var resizing = (Parent as Panel).SizeChangedObservable()
                                 .Select(_ => Zone.Points);
        _sub?.Dispose();


var pointsSub = Zone.WhenAnyValue(x => x.Points)
                        .Merge(resizing)
                        .Select(x => x.Select(x => new Point(x.X, x.Y)))
                        .Select(x => x.Select(ToParent))
                        .Select(x => new PointCollection(x))
                        .Subscribe(x => Points = x);

        var parent = Parent as Panel;
        var movingSub = parent.MouseMoveObservable()
                        .Where(_ => _isDragging)
                        .Select(x =>
                        {
                            var lastPosition = _lastPosition;
                            _lastPosition = x.GetPosition(parent);
                            return _lastPosition - lastPosition;
                        })                              .Select(v => new Point(v.X,
v.Y))                           .Select(FromParent)
                        .Select(v => new D.Point(v.X, v.Y))
                        .Subscribe(v =>
                        {                                   Zone.LeftTop += v;
                            Zone.RightTop += v;
                            Zone.RightBot += v;
                            Zone.LeftBot += v;
                        });

        var parentMouseUp = parent.MouseUpObservable()
                                  .Select(_ => false);

        var mouseDown = this.MouseDownObservable()
                            .Select(x => x.GetPosition(parent))
                            .Do(x => _lastPosition = x)
                            .Select(_ => true);

        var draggingSub = parentMouseUp.Merge(mouseDown)
                                       .Subscribe(x => _isDragging = x);

        _sub = new CompositeDisposable(pointsSub, movingSub,
draggingSub);
    }

    protected override void OnUnloaded(object sender,
RoutedEventArgs e)
    {
        _sub?.Dispose();
    }
  }
}
```

Текст класса SkeletonizationContext представлен на рис. П2.13.

```
using Microsoft.EntityFrameworkCore;
using Microsoft.Extensions.Configuration;

namespace
Skeletonization.DataLayer.Implementations.DatabaseSending
{

  public class SkeletonizationContext : DbContext
  {
    private readonly IConfiguration _configuration;

    public DbSet<BodyPart> BodyParts { get; set; }
    public DbSet<Human> Humans { get; set; }
    public DbSet<Point> Points { get; set; }
    public DbSet<Pose> Poses { get; set; }
    public DbSet<Report> Reports { get; set; }

    public SkeletonizationContext(IConfiguration configuration)
    {
      _configuration = configuration;
      Database.EnsureCreated();
    }

    protected override void
OnConfiguring(DbContextOptionsBuilder optionsBuilder)
    {

optionsBuilder.UseNpgsql(_configuration.GetConnectionString("Ske
letonization"));
    }
  }
}
```

Рис. П2.13. Текст класса SkeletonizationContext

Текст класса VideoReader представлен на рис. П2.14.

```
using Emgu.CV;
using Skeletonization.CrossfulLayer.Abstractions;
using Skeletonization.DataLayer.Abstractions;
using Skeletonization.DataLayer.Data;
using System;
using System.Threading;
using System.Threading.Tasks;

namespace Skeletonization.DataLayer.Implementations.Reading
{
  internal class VideoReader : IVideoReader
  {
    private VideoCapture _videoCapture;
    public bool Paused
    {
      get
      {
        lock (this)
        {
          return _paused;
        }
      }
      set
      {
        lock (this)
        {
          _paused = value;
        }
      }    }
    private bool _paused;
    public async void Start(IFactory<VideoCapture>
videoCaptureFactory, Func<FrameInfo, Task> changingCallback,
Action<VideoInfo> captureLoaded)
    {        _videoCapture = videoCaptureFactory.Create();
```

Рис. П2.14. Текст класса VideoReader

П2.14. Продолжение

```
        captureLoaded(new(_videoCapture.Width,
_videoCapture.Height));

        await Task.Run(async () =>
        {
            int frameNum = 0;
            while (true)
            {
                if (Paused)
                {
                    Thread.Sleep(100);
                    continue;
                }

    Mat frame = new();

                if (!_videoCapture.Read(frame))
                {
                    break;
                }

                await changingCallback?.Invoke(new FrameInfo(frame,
frameNum++));
            }
        });
        _videoCapture.Dispose();
    }
  }
}
```

Текст класса VideoDevicesResolver представлен на рис. П2.15.

```
using DirectShowLib;
using Skeletonization.DataLayer.Abstractions;
using Skeletonization.DataLayer.Data;
using System;
using System.Collections.Generic;
using System.Linq;

namespace Skeletonization.DataLayer.Implementations.Reading
{
  internal class VideoDevicesResolver : IVideoDevicesResolver
  {
    private readonly Lazy<IEnumerable<VideoDeviceInfo>>
_videoDeviceInfos;

    public VideoDevicesResolver()
    {
      _videoDeviceInfos = new(() =>
LoadVideoDevicesInfo().ToList());
    }

    public IEnumerable<VideoDeviceInfo> ResolveVideoDevices()
    {
      return _videoDeviceInfos.Value;
    }

    private static IEnumerable<VideoDeviceInfo>
LoadVideoDevicesInfo()
    {
      var captureDevices =
DsDevice.GetDevicesOfCat(FilterCategory.VideoInputDevice);

      for (int idx = 0; idx < captureDevices.Length; idx++)
      {
        yield return new(idx, captureDevices[idx].Name);
      }
    }
  }
}
```

Рис. П2.15. Текст класса VideoDevicesResolver

Текст класса Finder представлен на рис. П2.16.

```
using Emgu.CV;
using Skeletonization.BusinessLayer.Abstractions;
using Skeletonization.BusinessLayer.Data;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;

namespace Skeletonization.BusinessLayer.Implementation.Detection
{
  internal class Finder : IFinder
  {
    public IDetector Detector { get; }
    public IPreparer Preparer { get; }
    public IHumanConverter HumanConverter { get; }

    public Finder(IDetector detector, IPreparer preparer,
IHumanConverter humanConverter)
    {
      Detector = detector;
      Preparer = preparer;
      HumanConverter = humanConverter;
    }

    public async Task<IReadOnlyList<Human>> Find(Mat input)
    {
      var points = Preparer.Prepare(await Task.Run(() =>
Detector.Detect(input)));
      return HumanConverter.Convert(points).ToList();
    }
  }
}
```

Рис. П2.16. Текст класса Finder

Текст класса Detector представлен на рис. П2.17.

```
using Emgu.CV;
using Emgu.CV.CvEnum;
using Emgu.CV.Dnn;
using Emgu.CV.Structure;
using Emgu.CV.Util;
using Skeletonization.BusinessLayer.Abstractions;
using Skeletonization.CrossLayer.Extensions;
using System;
using System.Collections.Generic;
using System.Drawing;
using System.Linq;

namespace Skeletonization.BusinessLayer.Implementation.Detection
{
   public class Detector : IDetector
   {
      private const int nPoints = 18;
      private readonly Net _net;

      private static (int first, int second)[] mapIdx =
      {
         (31,32), (39,40), (33,34), (35,36), (41,42), (43,44),
         (19,20), (21,22), (23,24), (25,26), (27,28), (29,30),
         (47,48), (49,50), (53,54), (51,52), (55,56), (37,38),
         (45,46)
      };

      private static (int first, int second)[] posePairs =
      {
         (1,2), (1,5), (2,3), (3,4), (5,6), (6,7),
         (1,8), (8,9), (9,10), (1,11), (11,12), (12,13),
         (1,0), (0,14), (14,16), (0,15), (15,17), (2,17),
         (5,16)
      };

      public Detector(Net net)
      {
         _net = net;
      }

      private static void GetKeyPoints(Mat probMap, double
threshold, out KeyPoint[] keyPointsResult)
      {
         var smoothProbMap = new Mat();
         CvInvoke.GaussianBlur(probMap, smoothProbMap, new
Size(3, 3), 0, 0);

         var maskedProbMap = new Mat();
         CvInvoke.Threshold(smoothProbMap, maskedProbMap,
threshold, 255, ThresholdType.Binary);

         maskedProbMap.ConvertTo(maskedProbMap,
DepthType.Cv8U, 1);

         var contours = new VectorOfVectorOfPoint();
         var empty = new Mat();
         CvInvoke.FindContours(maskedProbMap, contours, empty,
RetrType.Tree, ChainApproxMethod.ChainApproxSimple);

         var keyPoints = new List<KeyPoint>();

         for (int i = 0; i < contours.Size; ++i)
         {
            var blobMask = Mat.Zeros(smoothProbMap.Rows,
smoothProbMap.Cols, smoothProbMap.Depth,
smoothProbMap.NumberOfChannels);

            CvInvoke.FillConvexPoly(blobMask, contours[i], new
MCvScalar(1));
            double _ = 0;
            double __ = 0;
            Point ___ = default;
            Point maxLoc = default;

            var mult = new Mat();
            CvInvoke.Multiply(smoothProbMap, blobMask, mult);
            CvInvoke.MinMaxLoc(mult, ref _, ref __, ref ___, ref
maxLoc);

            keyPoints.Add(new KeyPoint(maxLoc,
probMap.At<float>(maxLoc.Y, maxLoc.X)));
         }

         keyPointsResult = keyPoints.ToArray();
      }

      private static void SplitNetOutputBlobToParts(Mat
netOutputBlob, Size targetSize, out Mat[] netOutputsPartsResult)
      {
         int nparts = netOutputBlob.SizeOfDimension[1];
         int h = netOutputBlob.SizeOfDimension[2];
         int w = netOutputBlob.SizeOfDimension[3];

         var netOutputsParts = new List<Mat>();
         for (int i = 0; i < nparts; ++i)
         {
            var part = new Mat(new[] { h, w }, DepthType.Cv32F,
netOutputBlob.GetDataPointer(0, i));
            var resizedMat = new Mat();
            CvInvoke.Resize(part, resizedMat, targetSize);
            netOutputsParts.Add(resizedMat);
         }

         netOutputsPartsResult = netOutputsParts.ToArray();
      }

      private static void PopulateInterpPoints(Point a, Point b, int
numPoints, out Point[] interpCoordsResult)
      {

         var interpCoords = new List<Point>();

         float xStep = (b.X - a.X) / (float)(numPoints - 1);
         float yStep = (b.Y - a.Y) / (float)(numPoints - 1);

         interpCoords.Add(a);

         for (int i = 1; i < numPoints - 1; ++i)
         {
            interpCoords.Add(new Point((int)(a.X + xStep * i),
(int)(a.Y + yStep * i)));
         }

         interpCoords.Add(b);

         interpCoordsResult = interpCoords.ToArray();
      }

      private static void GetValidPairs(Mat[] netOutputParts,
                        KeyPoint[][] detectedKeyPoints,
                        out ValidPair[][] validPairsResult,
                        out int[] invalidPairsResult)
      {
         int nInterpSamples = 10;
         float pafScoreTh = 0.1f;
         float confTh = 0.7f;

         var validPairs = new List<List<ValidPair>>();
         var invalidPairs = new SortedSet<int>();

         for (int k = 0; k < mapIdx.Length; ++k)
         {
            var pafA = netOutputParts[mapIdx[k].first];
            var pafB = netOutputParts[mapIdx[k].second];

            var candA = detectedKeyPoints[posePairs[k].first];
            var candB = detectedKeyPoints[posePairs[k].second];
            int nA = candA.Length;
            int nB = candB.Length;
            if (nA != 0 && nB != 0)
            {                  var localValidPairs = new List<ValidPair>();
```

Рис. П2.17. Текст класса Detector

П2.17. Продолжение

```
            for (int i = 0; i < nA; ++i)
            {
                int maxJ = -1;
                float maxScore = -1;
                bool found = false;

                for (int j = 0; j < nB; ++j)
                {
                    (float first, float second) = (candB[j].Point.X -
candA[i].Point.X, candB[j].Point.Y - candA[i].Point.Y);
                    float norm = MathF.Sqrt(first * first + second *
second);

                    if (norm == 0)
                    {
                        continue;
                    }

                    first /= norm;
                    second /= norm;

                    PopulateInterpPoints(candA[i].Point,
candB[j].Point, nInterpSamples, out var interpCoords);

                    var pafInterp = new List<(float first, float
second)>();

                    for (int l = 0; l < interpCoords.Length; ++l)
                    {
                        pafInterp.Add
                            ((pafA.At<float>(interpCoords[l].Y,
interpCoords[l].X),
                              pafB.At<float>(interpCoords[l].Y,
interpCoords[l].X)));
                    }

                    var pafScores = new List<float>();
                    float sumOfPafScores = 0;
                    int numOverTh = 0;
                    for (int l = 0; l < pafInterp.Count; ++l)
                    {
                        float score = pafInterp[l].first * first +
pafInterp[l].second * second;
                        sumOfPafScores += score;
                        if (score > pafScoreTh)
                        {
                            ++numOverTh;
                        }

                        pafScores.Add(score);
                    }

                    float avgPafScore = sumOfPafScores /
pafInterp.Count;

                    if (numOverTh / (float)nInterpSamples > confTh)
                    {
                        if (avgPafScore > maxScore)
                        {
                            maxJ = j;
                            maxScore = avgPafScore;
                            found = true;
                        }
                    }
                }//j

                if (found)
                {
                    localValidPairs.Add(new ValidPair(candA[i].Id,
candB[maxJ].Id, maxScore));
                }
            }//i

            validPairs.Add(localValidPairs);
        }
        else
        {
            invalidPairs.Add(k);
            validPairs.Add(new List<ValidPair>());
        }
    }//k

    validPairsResult = validPairs.Select(x =>
x.ToArray()).ToArray();
    invalidPairsResult = invalidPairs.ToArray();
}

private static void GetPersonwiseKeypoints(ValidPair[][]
validPairs,
                                           int[] invalidPairs,
                                           out int[][] personwiseKeypointsResult)
{
    var personwiseKeypoints = new List<List<int>>();

    for (int k = 0; k < mapIdx.Length; ++k)
    {
        if (invalidPairs.Contains(k))
        {
            continue;
        }

        var localValidPairs = new List<ValidPair>(validPairs[k]);

        int indexA = posePairs[k].first;
        int indexB = posePairs[k].second;

        for (int i = 0; i < localValidPairs.Count; ++i)
        {
            bool found = false;
            int personIdx = -1;

            for (int j = 0; !found && j <
personwiseKeypoints.Count; ++j)
            {
                if (indexA < personwiseKeypoints[j].Count &&
                    personwiseKeypoints[j][indexA] ==
localValidPairs[i].Aid)
                {
                    personIdx = j;
                    found = true;
                }
            }

            if (found)
            {
                personwiseKeypoints[personIdx][indexB] =
localValidPairs[i].Bid;
            }
            else if (k < 17)
            {
                var lpkp = new List<int>(Enumerable.Range(0,
18).Select(_ => -1))
                {
                    [indexA] = localValidPairs[i].Aid,
                    [indexB] = localValidPairs[i].Bid
                };

                personwiseKeypoints.Add(lpkp);
            }
        }
    }

    personwiseKeypointsResult = personwiseKeypoints.Select(x
=> x.ToArray()).ToArray();
}

public Point[,] Detect(Mat input)
{
    var inputBlob = DnnInvoke.BlobFromImage(input, 1.0 /
255.0, new Size(368 * input.Cols / input.Rows, 368), new
MCvScalar(0, 0, 0), false, false);
```

П2.17. Продолжение

```
        _net.SetInput(inputBlob);

        var netOutputBlob = _net.Forward();
        SplitNetOutputBlobToParts(netOutputBlob, new
Size(input.Cols, input.Rows), out var netOutputParts);

        int keyPointId = 0;
        var detectedKeypoints = new KeyPoint[nPoints][];
        var keyPointsList = new List<KeyPoint>();

        for (int i = 0; i < nPoints; ++i)
        {
            GetKeyPoints(netOutputParts[i], 0.1, out var keyPoints);
            for (int j = 0; j < keyPoints.Length; j++)
            {
                keyPoints[j].Id = keyPointId++;
            }

            detectedKeypoints[i] = keyPoints;
            keyPointsList.AddRange(keyPoints);
        }

        GetValidPairs(netOutputParts, detectedKeypoints, out var
validPairs, out int[] invalidPairs);
        GetPersonwiseKeypoints(validPairs, invalidPairs, out int[][]
personwiseKeypoints);

        var persons = new Point[personwiseKeypoints.Length,
nPoints];

        for (int i = 0; i < persons.GetLength(0); i++)
        {
            for (int j = 0; j < persons.GetLength(1); j++)
            {
                persons[i, j] = new Point(-1, -1);
            }
        }

        for (int i = 0; i < nPoints - 1; ++i)
        {
            for (int n = 0; n < personwiseKeypoints.Length; ++n)
            {
                var (first, second) = posePairs[i];
                int indexA = personwiseKeypoints[n][first];
                int indexB = personwiseKeypoints[n][second];

                if (indexA == -1 || indexB == -1)
                {
                    continue;
                }

                var kpA = keyPointsList[indexA];
                var kpB = keyPointsList[indexB];

                persons[n, first] = kpA.Point;
                persons[n, second] = kpB.Point;
            }
        }


        return persons;
    }
  }
}
```

Текст класса ConfigurationExtensionsMethods представлен на рис. П2.18.

```
using Microsoft.Extensions.Configuration;
using Prism.Ioc;
using System.IO;

namespace Skeletonization.CrossLayer.Extensions
{
   public static class ConfigurationExtensionsMethods
   {
      public static IContainerRegistry RegisterConfiguration(this
IContainerRegistry containerProvider)
      {

containerProvider.RegisterSingleton<IConfiguration>(CreateConfigu
ration);
         return containerProvider;
      }

      public static IContainerRegistry RegisterOption<T>(this
IContainerRegistry containerProvider, string section)
      {
         containerProvider.RegisterSingleton<T>(c =>
c.OptionFactory<T>(section));
         return containerProvider;
      }

      private static IConfiguration CreateConfiguration()
      {
         return new ConfigurationBuilder()
               .SetBasePath(Directory.GetCurrentDirectory())
               .AddJsonFile("appsettings.json")
               .Build();
      }

      private static T OptionFactory<T>(this IContainerProvider
containerProvider, string section)
      {
         return containerProvider.Resolve<IConfiguration>()
                        .GetSection(section)
                        .Get<T>();
      }
   }
}
```

Рис. П2.18. Текст класса ConfigurationExtensionsMethods

Приложение 3

Спецификации

Настоящее приложение содержит спецификации на разработанную программную документацию и компоненты программного обеспечения. Они приведены в табл. П3.1.

Табл. П3.1

Спецификации

Обозначение	Описание	Примечание
1	2	3
Shell.xaml	компонент, характеризующий основное окно приложения	
DetectionControl.xaml	компонент, характеризующий окно детекции	
QueriesControl.xaml	компонент, характеризующий форму запросов	
DetectionZonesControl.xaml	компонент, характеризующий форму зон в окне детекции	
CameraDialogControl.xaml	компонент, характеризующий диалог выбора камеры	
DrawingZonesControl.xaml	компонент, характеризующий форму отображения опасных зон	
ZoneEllipse.cs	компонент, характеризующий визуальную точку зоны	
ZonePolygon.cs	компонент, характеризующий визуальную область зоны	
ZonesControl.xaml	компонент, характеризующий окно зон	
HumansControl.xaml	компонент, характеризующий окно людей	
SettingsControl.xaml	компонент, характеризующий окно настроек	

Продолжение таблицы П3.1

1	2	3
ZonesViewModel.cs	класс, характеризующий модель представления зон	
ZoneConsumer.cs	класс, характеризующий модель представления для класса, которому нужно знать о зонах	
DetectionViewModel.cs	класс, характеризующий модель представления детекции	
DetectionZonesViewModel.cs	класс, характеризующий модель представления зон в детекции	
ShellViewModel.cs	класс, характеризующий основную модель представления	
SettingsViewModel.cs	класс, характеризующий модель представления настроек	
OpenCameraDialogViewModel.cs	класс, характеризующий модель представления для диалога выбора камеры	
HumansViewModel.cs	класс, характеризующий модель представления людей	
QueriesViewModel.cs	класс, характеризующий модель представления запросов	
DetectionModel.cs	класс, характеризующий модель детекции	
HumanWithRoi.cs	класс, описывающий человека с его областью	
SettingsModel.cs	класс, характеризующий модель настроек	
QueriesModel.cs	класс, характеризующий модель запросов	
Query.cs	класс, описывающий запрос	

Продолжение таблицы П3.1

1	2	3
DetectionZonesModel.cs	класс, характеризующий модель зон в детекции	
Zone.cs	класс, описывающий зону	
ZoneFactory.cs	класс, описывающий логику создания новой зоны	
SelectableBodyPart.cs	класс, характеризующий выбранную часть тела	
HumansCheckResult.cs	класс, описывающий результат проверки человека	
NetOption.cs	класс, описывающий конфигурацию нейронной сети	
Detector.cs	класс, необходимый для детектирования точек частей тела	
Finder.cs	класс, необходимый для детектирования людей	
Drawer.cs	класс, отвечающий за логику отрисовки людей на кадре	
Preparer.cs	класс, отвечающий за логику преобразования найденных точек частей тела	
HumansConverter.cs	класс, отвечающий за логику преобразования найденных точек частей тела в людей	
Human.cs	класс, описывающий обнаруженного человека	
Point.cs	класс, описывающий точку с относительными координатами	
VideoCaptureCameraFactory.cs	класс, описывающий логику открытия видеопотока с камеры	

Продолжение таблицы П3.1

1	2	3
VideoCaptureFileFactory.cs	класс, описывающий логику открытия видеопотока из файла	
VideoService.cs	класс, содержащий логику работы с видео	
VideoProcessingHandler.cs	класс, описывающий обработчика нейронной сети	
ReportService.cs	класс, содержащий логику работы с отправкой отчетов	
Report.cs	класс, описывающий отчёт	
DatabaseContext.cs	класс, описывающий схему базы данных	
PoseEntity.cs	класс, описывающий таблицу поз	
PointEntity.cs	класс, описывающий таблицу точек	
HumanEntity.cs	класс, описывающий таблицу людей	
ReportEntity.cs	класс, описывающий таблицу отчётов	
BodyPartEntity.cs	класс, описывающий таблицу частей тела	
TeamsSender.cs	класс, содержащий логику для отправки данных на канал в Teams	
EmailSender.cs	класс, содержащий логику для отправки данных на почтовый ящик	
VideoReader.cs	класс, содержащий логику чтения видеопотока в отдельном потоке	
VideoCaptureFactory.cs	класс, описывающий логику создания видеопотока	
VideoDeviceResolver.cs	класс, содержащий логику получения информации о доступных видеокамерах	

Продолжение таблицы П3.1

1	2	3
VideoDeviceInfo.cs	класс, описывающий информацию о видеокамере	
BodyPartsExtensionsMethod.cs	Класс, содержащий методы для работы с частями чела	
ConfigureExtensionsMethod.cs	Класс, содержащий методы для работы с конфигурацией	
MatrixExtensionsMethod.cs	Класс, содержащий методы для работы с матрицами	
PointsExtensionsMethod.cs	Класс, содержащий методы для работы с точками	
BodyPart.cs	класс, описывающий все возможные части тела	
VideoCaptureFactoryType.cs	класс, описывающий все возможные фабрики создания видеопотока	
SendType.cs	класс, описывающий все возможные варианты отправки отчёта	
VideoCaptureFactoryException.cs	класс, описывающий исключение фабрики создания видеопотка	
DatabaseSendException.cs	класс, описывающий исключение отправки данных в базу данных	
TeamsSendException.cs	класс, описывающий исключение отправки данных в Teams	

Приложение 4

Руководство пользователя

1. Общие сведения о программе

Данная программа представляет собой приложение для работы с нейронной сетью по обнаружению людей и их частей тела на изображении с видеопотока и контролю безопасности их передвижения.

Программное обеспечение выполняет следующие функции:

- отслеживание положения работников;
- отслеживание отдельных частей тела работников;
- определение позы работников;
- добавления нескольких зон опасности и их настройка;
- добавления комбинирующих логику зон запросов;
- отправление отчетов в MS Teams.

2. Описание установки

Вставьте диск в оптический привод компьютера. Откройте папку Skeletonization с содержимым диска. Выберите папку Program, которая содержит в себе исполняемый файл Skeletonization.exe.

3. Описание запуска

Для запуска программы необходимо открыть файл Skeletonization.exe (рис. П4.1). Будет открыта основная вкладка приложения (рис. П4.2).

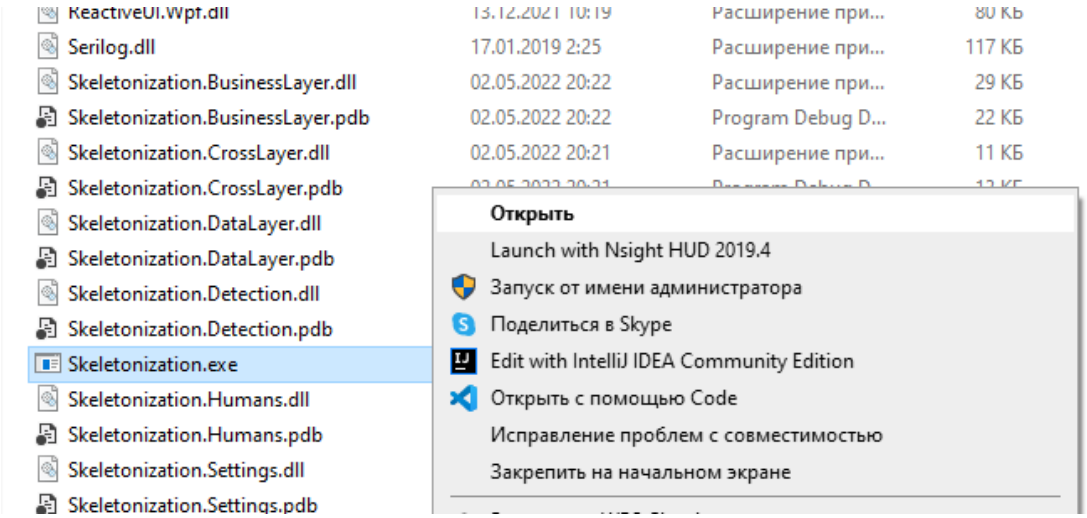


Рис. П4.1. Открытие программы

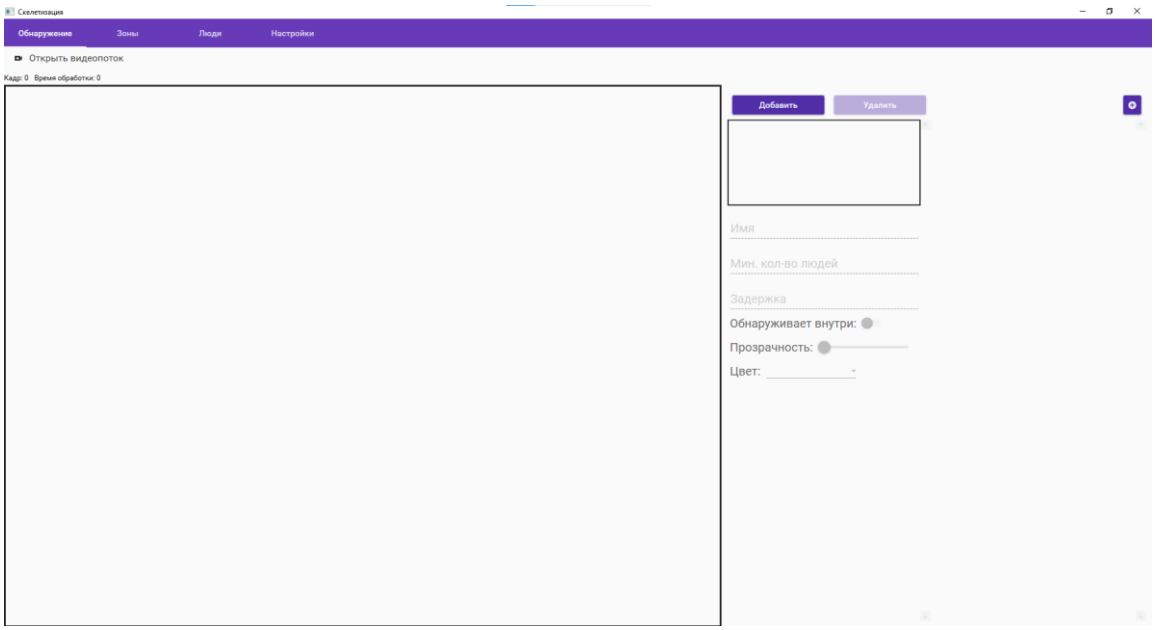


Рис. П4.2. Основная вкладка приложения

4. Инструкции по работе

Для того, чтобы начать обработку, необходимо открыть любой видеофайл или видео непосредственно с камеры (рис. П4.3), нажав соответствующие кнопки в левой части окна (рис. П4.4). После некоторого времени (для первого кадра нейросеть загружается в видеокарту) будет произведена обработка и в центре будут отображены найденные на кадре люди.

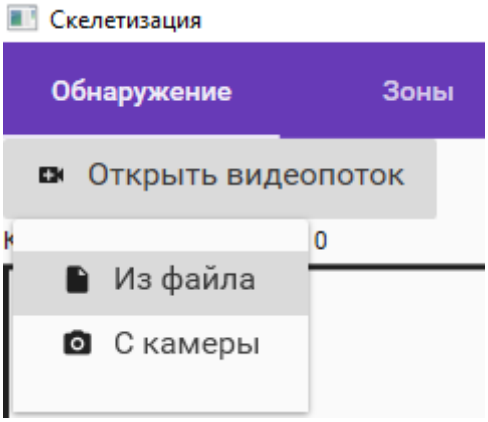


Рис. П4.3. Открытие видеопотока

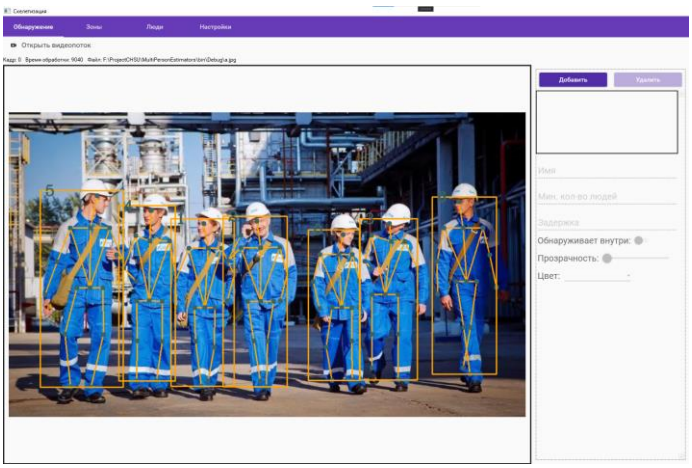


Рис. П4.4. Детектирование

Существует возможность добавлять зоны и редактировать её параметры, для этого создана отдельная форма (рис. П4.5)

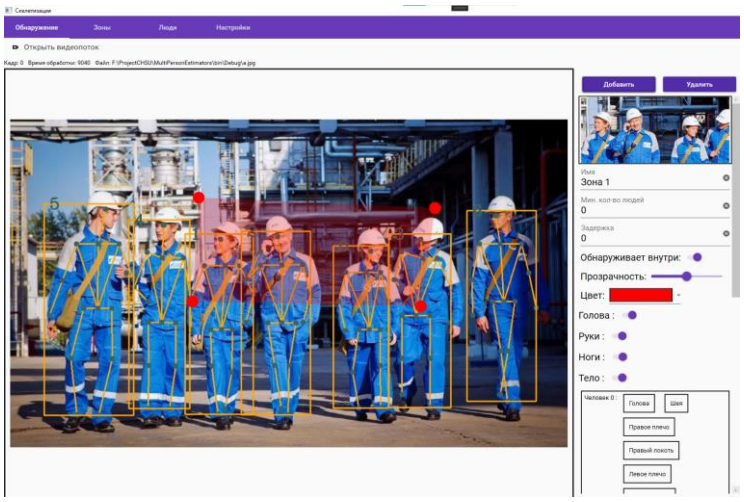


Рис. П4.5. Работа с зонами

Перейдя на вкладку «Люди», будет представлена вся информация о найденных людях (рис. П4.6).

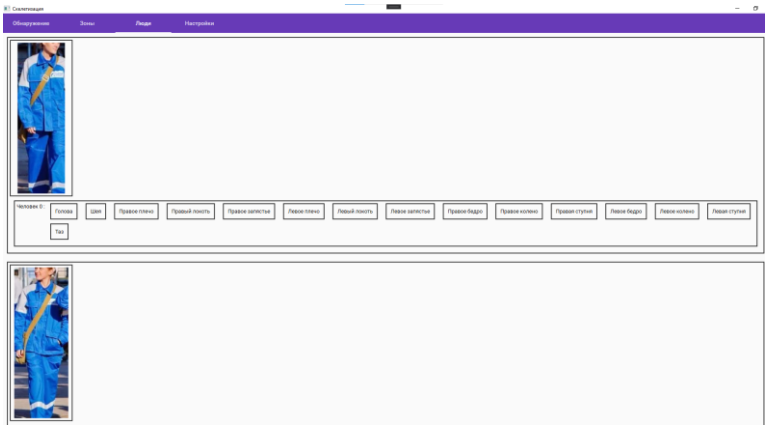


Рис. П4.6. Окно людей

Аналогичным образом можно получить информацию о зонах на одноименной вкладке «Зоны (рис. П4.7).

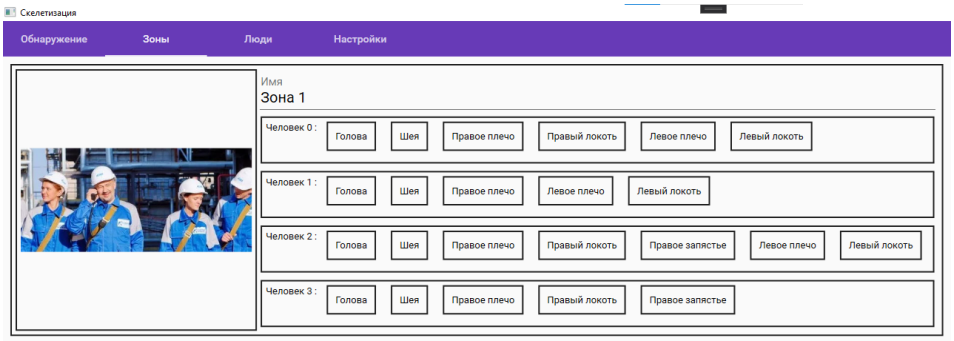


Рис. П4.7. Окно зон

После создания всех зон будет производится контроль опасных действий и отправка отчётов в базу данных и в MS Teams (рис. П4.8).

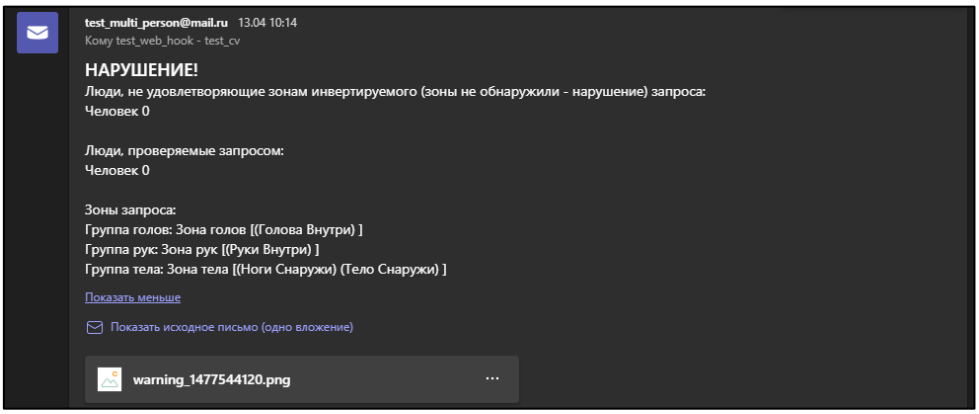


Рис. П4.8. Отчёт в MS Teams

Для закрытия приложения в правом верхнем углу необходимо нажать на крестик (рис. П4.9).

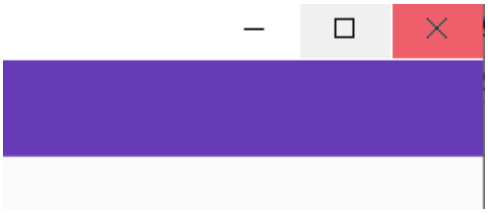


Рис. П4.9. Закрытие программы