МИНОБРНАУКИ РОССИИ
федеральное государственное бюджетное
образовательное учреждение высшего образования
«ЧЕРЕПОВЕЦКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

Институт (факультет) Институт Информационных технологий

Кафедра Математическое и программное обеспечение ЭВМ

КУРСОВОЙ ПРОЕКТ

По дисциплине Управление программными проектами

На тему Проектирование программного обеспечения при объектном подходе

Выполнил студент группы

группа

Программная инженерия

1ПИб-01-41оп

шифр, наименование

Богданов Александр Павлович

фамилия, имя, отчество

Руководитель

Ершов Евгений Валентинович

фамилия, имя, отчество

профессор

должность

Дата представления работы

«_____»_____20 ___ г.

Заключение о допуске к защите

Оценка _____, _____

количество баллов

Подпись преподавателя_____

Череповец, 2022 *год*

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 разработка программного обеспечения системы детекции частей тела для контроля опасных действий работников, основными функциями которого будут:

1. Обнаружение в поступающем с камеры видеопотоке людей с полным распознанием всех частей тела;

2. Разметка и настройка различных зон на изображении пользователем;

3. Определение положения людей в пространстве (определение позы человека и его нахождения в размеченных зонах);

4. Создание запросов для контроля безопасности, содержащих наборы зон и правила для них;

5. Автоматическая отправка отчёта о нарушении человеком правил запроса ответственному лицу через Microsoft Teams.

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 бы работу агрегата, за которым произошло нарушение (если возможно) и подавало бы предупредительный сигнал.

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 [3].

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 [3].

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 [10].

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 [10].

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 [10].

ViewModel также содержит логику по получению данных из модели, которые потом передаются в представление. И также VewModel определяет логику по обновлению данных в модели [10].

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.

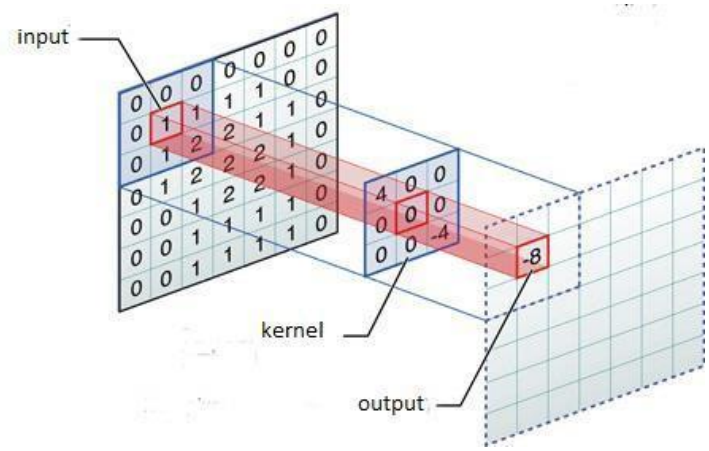


Рис. 2. Операция свёртки

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.

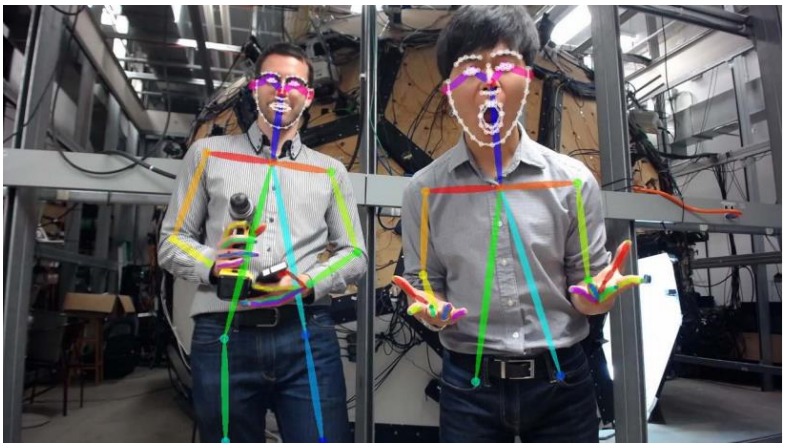


Рис. 3. Пример работы OpenPose

Схема работы OpenPose представлена на рис. 4.

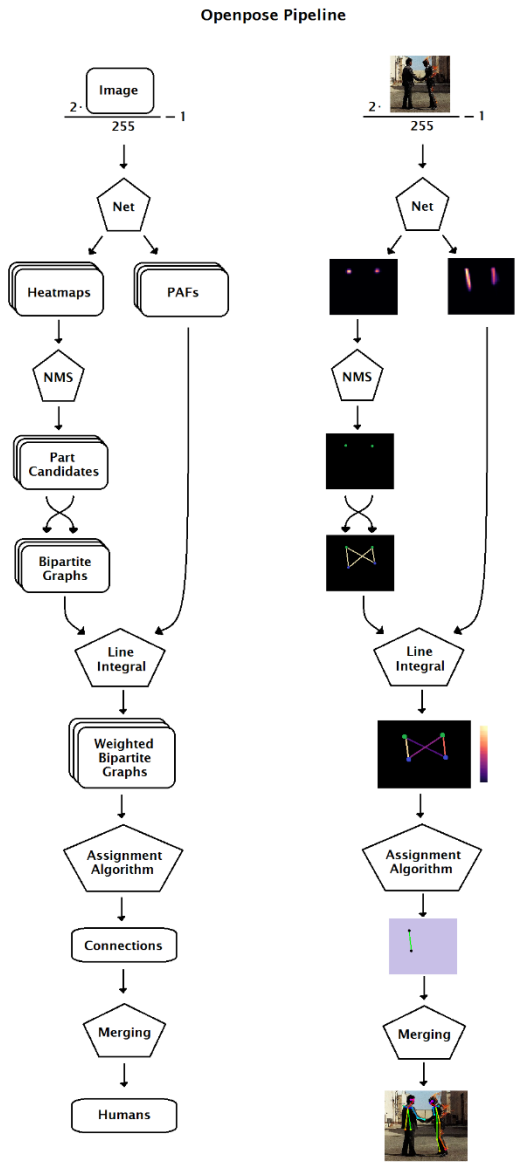


Рис. 4. Схема работы OpenPose

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.

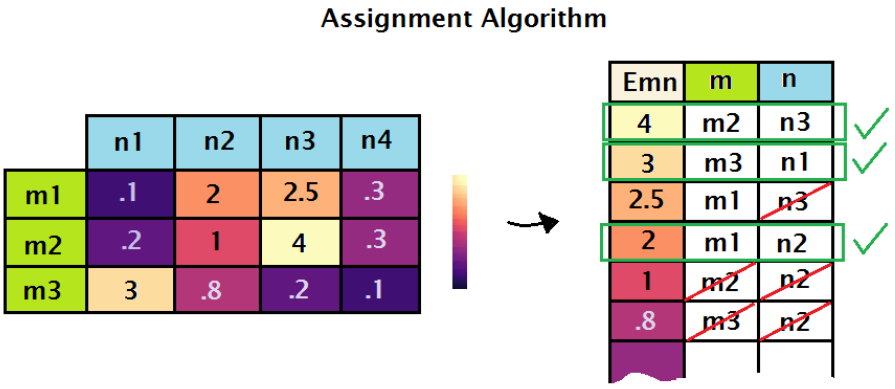


Рис. 6. Пример решения задачи присваивания

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 [3].

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 [3].

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

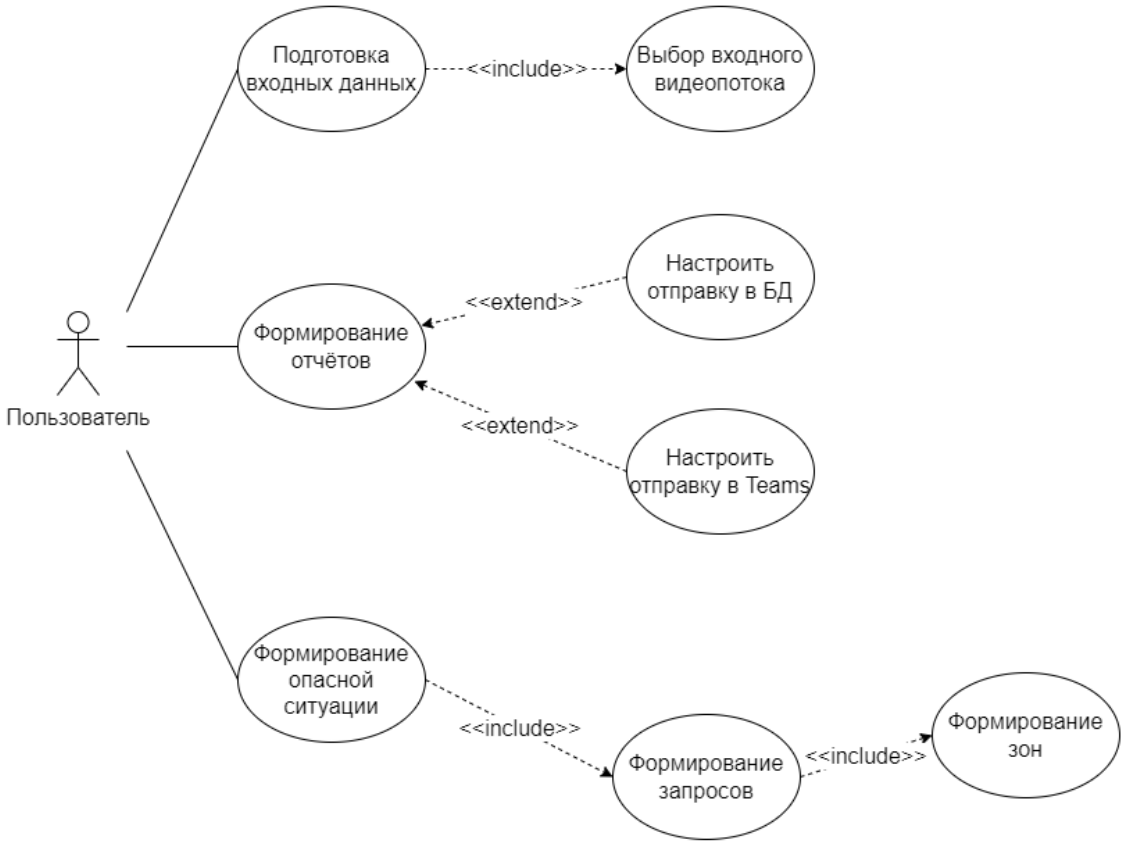


Рис.7. Диаграмма вариантов использования

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	4 Система отображает настройку

для загрузки почты канала Teams	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	8 Система предоставляет пользователю

запросов»	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. При анализе людей, обнаруженных в определенных областях кадра нейронной сетью, запрос может с помощью алгоритма сформировать отчёт в базу данных (БД) и Teams.

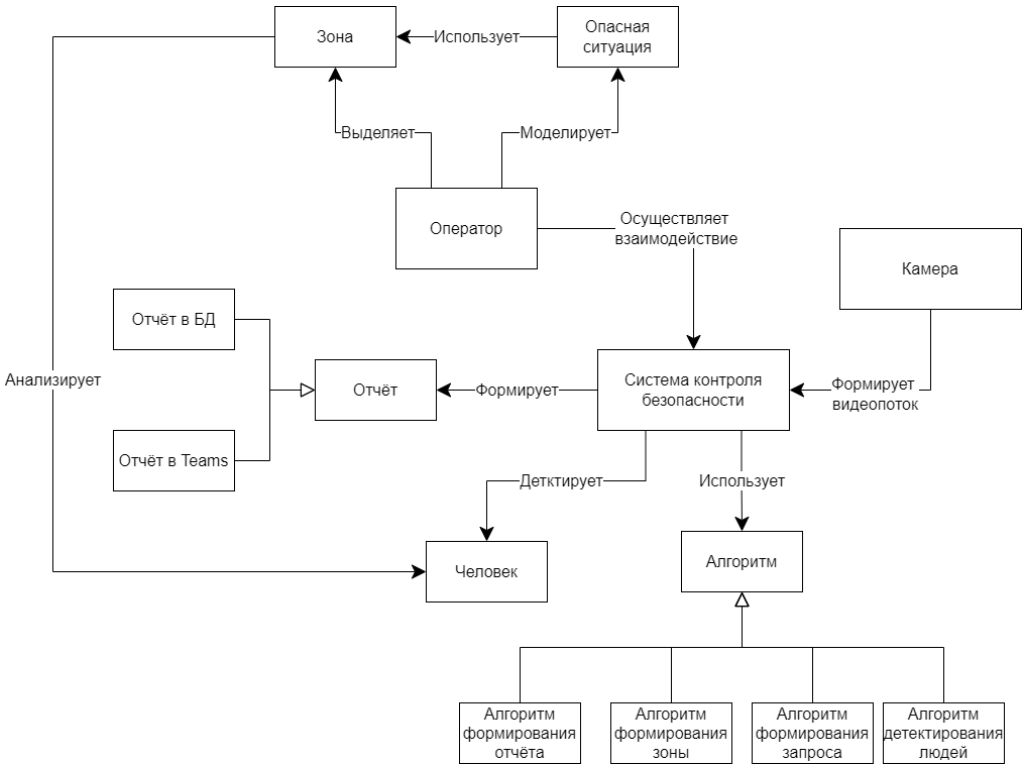


Рис.8. Контекстная диаграмма классов

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.

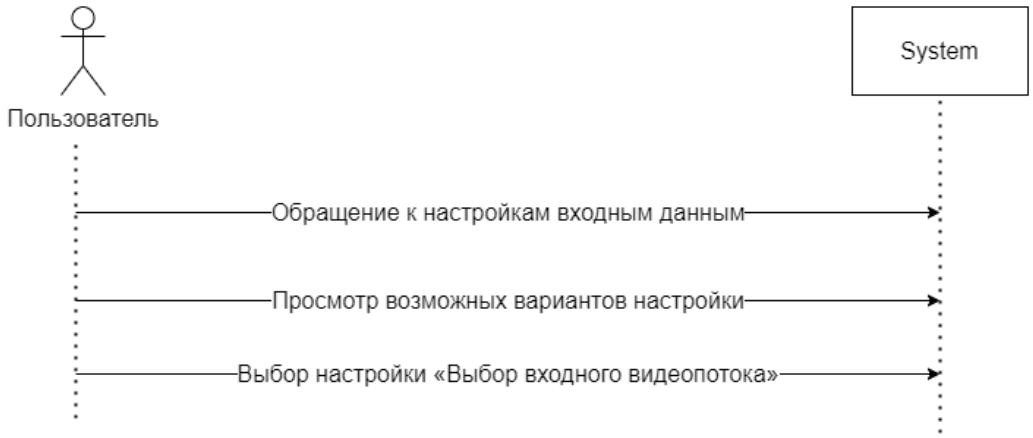


Рис. 9. Диаграмма последовательности для варианта использования «Подготовка входных данных»

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	-
Предусловие	-
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	-
Вывод	-
Предусловие	-
Постусловие	-

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

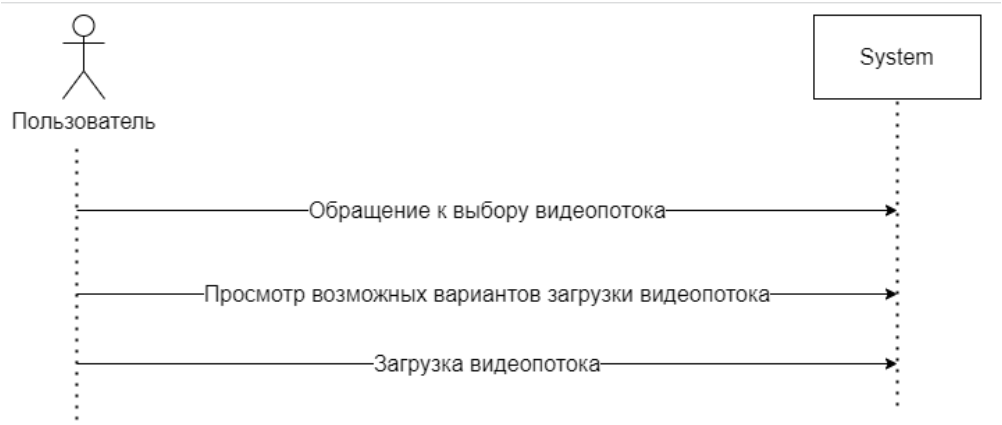


Рис. 10. Диаграмма последовательности для варианта использования «Выбор входного видеопотока»

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	-
Постусловие	-

Таблица 21

Описание операции «Просмотр возможных вариантов загрузки видеопотока»

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

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

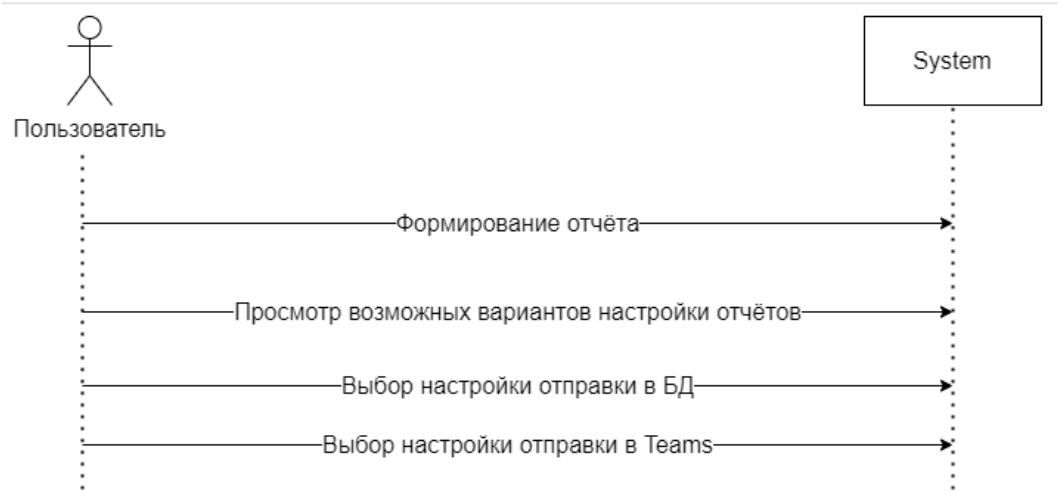


Рис. 11. Диаграмма последовательности для варианта использования «Формирование отчётов»

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	Отсутствие подключения к серверу. Сообщение об отсутствии связи с сервером
Вывод	Компоненты Web-страницы
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	-
Предусловие	-
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	-
Вывод	-
Предусловие	-
Постусловие	-

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	-
Вывод	-
Предусловие	-
Постусловие	-

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

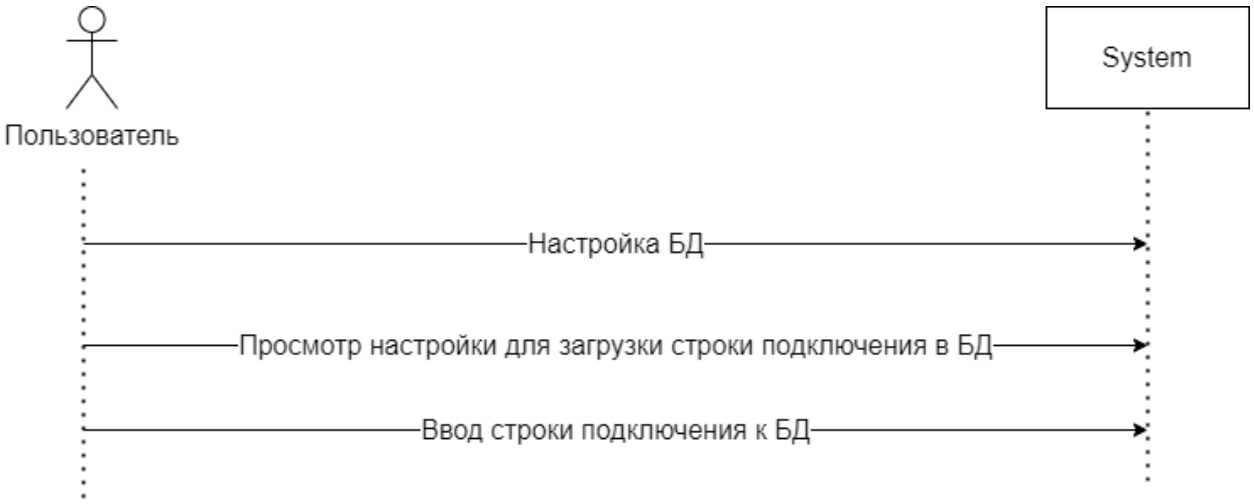


Рис. 12. Диаграмма последовательности для варианта использования «Настроить отправку в БД»

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	-
Постусловие	-

Таблица 28

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

Раздел	Описание
Имя	Просмотр настройки для загрузки строки подключения в БД
Обязанности	Отобразить настройку для загрузки строки подключения к базе данных
Тип	Системная
Ссылка	Вариант использования «Настроить отправку в БД»
Примечания	-
Исключения	-
Вывод	-
Предусловие	-
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	-
Вывод	-
Предусловие	-
Постусловие	-

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

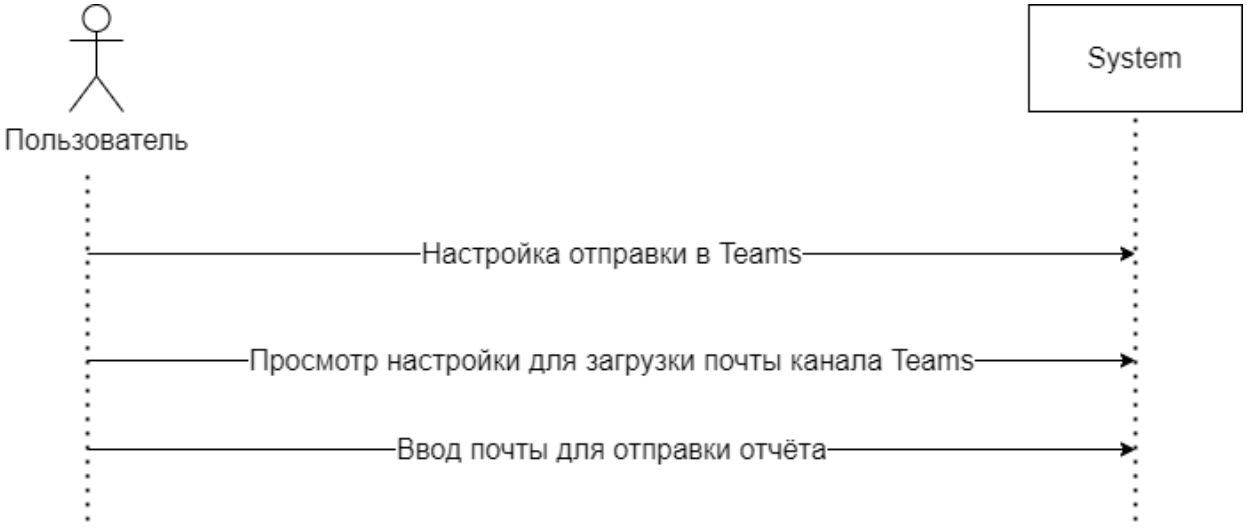


Рис. 13. Диаграмма последовательности для варианта использования «Настроить отправку в Teams»

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	-
Постусловие	-

Таблица 31

Описание операции «Просмотр настройки для загрузки почты канала Teams»

Раздел	Описание
Имя	Просмотр настройки для загрузки почты канала Teams
Обязанности	Отобразить настройку для загрузки почты канала Teams
Тип	Системная
Ссылка	Вариант использования «Настроить отправку в Teams»
Примечания	-
Исключения	-
Вывод	-
Предусловие	-
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	-
Вывод	-
Предусловие	-
Постусловие	-

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

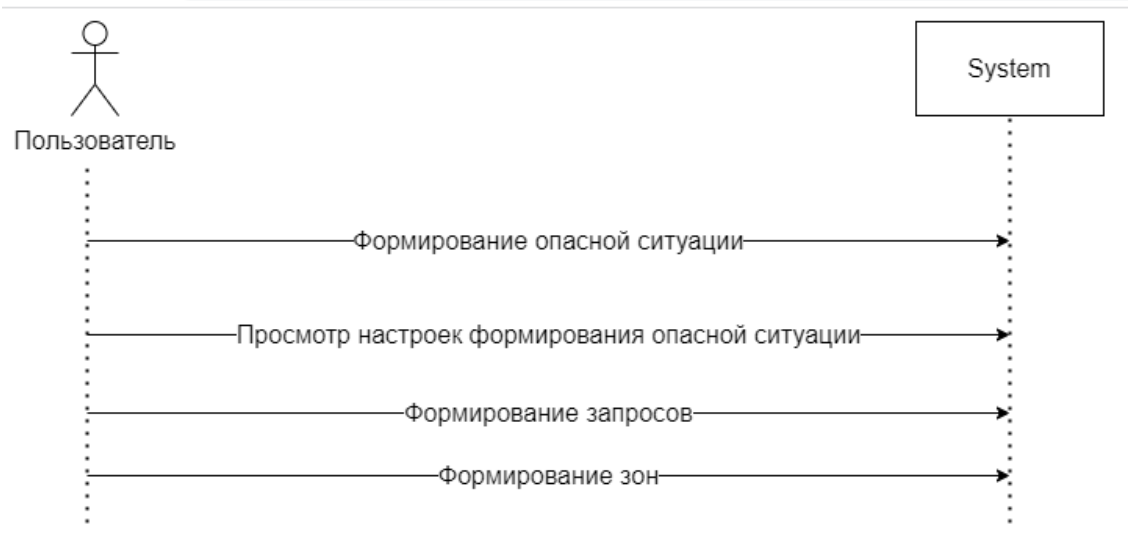


Рис. 14. Диаграмма последовательности для варианта использования «Формирование опасной ситуации»

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	-
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	-
Предусловие	-
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	-
Вывод	-
Предусловие	-
Постусловие	-

Таблица 36

Описание операции «Формирование зон»

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

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

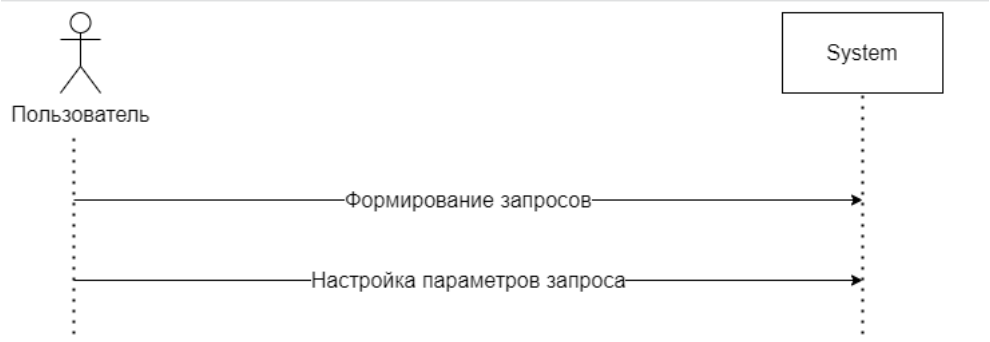


Рис. 15. Диаграмма последовательности для варианта использования «Формирование запросов»

Таблица 37

Описание операции «Формирование запросов»

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

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

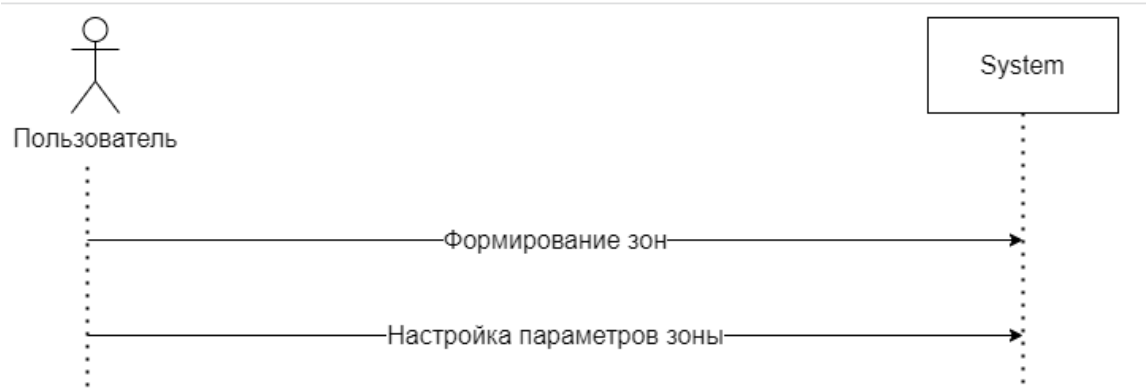


Рис. 16. Диаграмма последовательности для варианта использования «Формирование зон»

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	-
Предусловие	-
Постусловие	-

Таблица 40

Описание операции «Настройка параметров зон»

Раздел	Описание
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»
Примечания	-

Исключения	-
Вывод	-
Предусловие	-
Постусловие	-

1.4.4 Построение диаграмм 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.

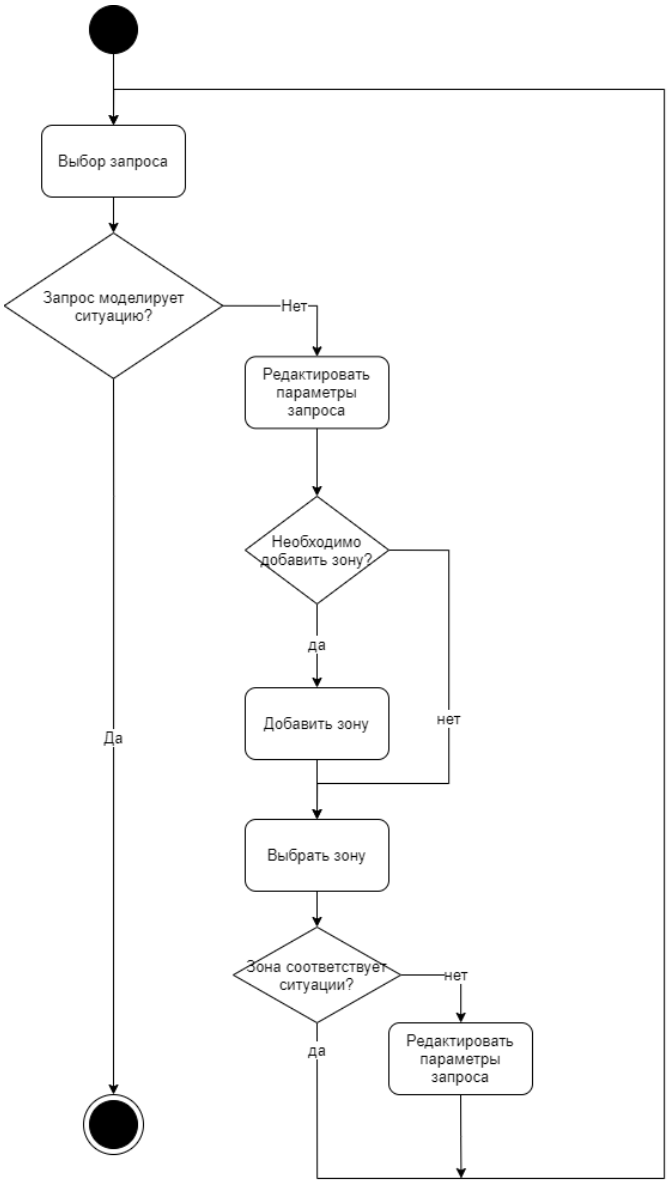


Рис. 17. Диаграмма деятельностей

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.

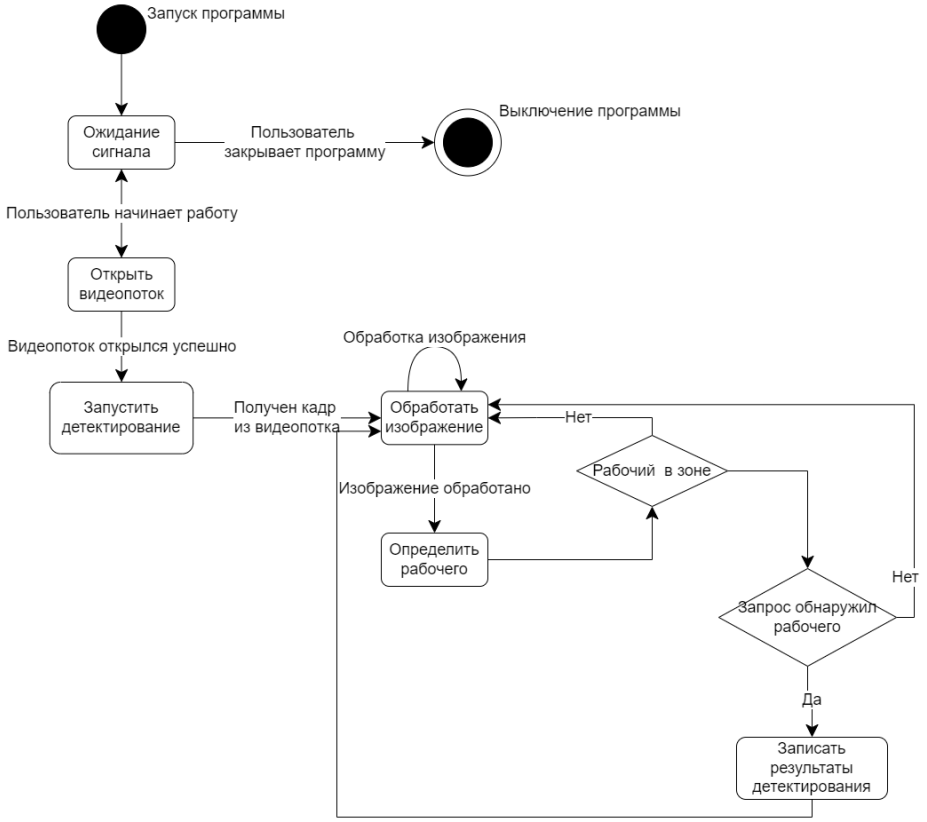


Рис. 18. STD – диаграмма

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.

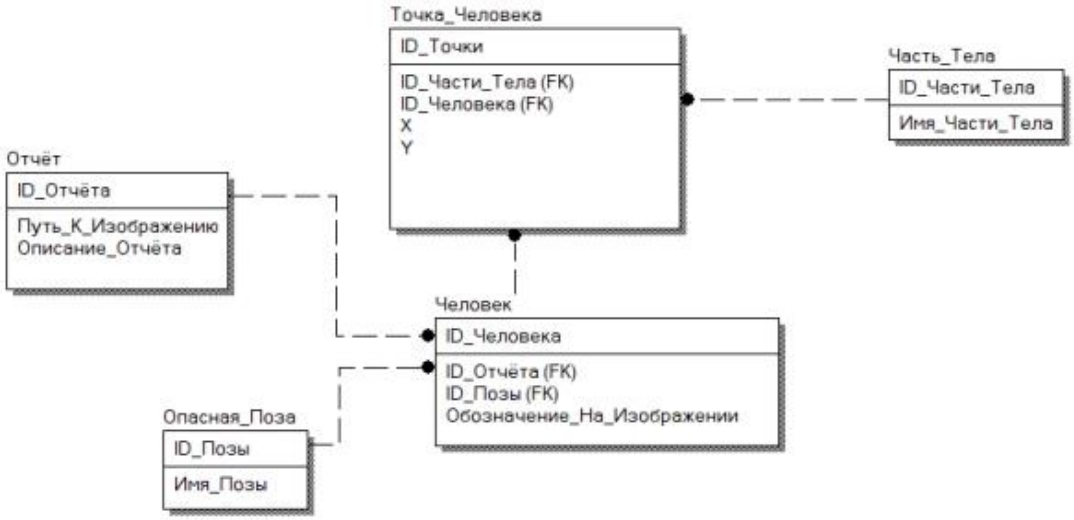


Рис. 19. ER-диаграмма

Таблица 41

Описание сущностей

Сущность	Атрибут	Описание
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
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
Отчёт	ID_Отчёта	Long Integer
	Путь_К_Изображению	Text(40)
	Описание_Отчёта	Text(80)
Опасная поза	ID_Позы	Long Integer
	Имя_Позы	Text(20)
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.

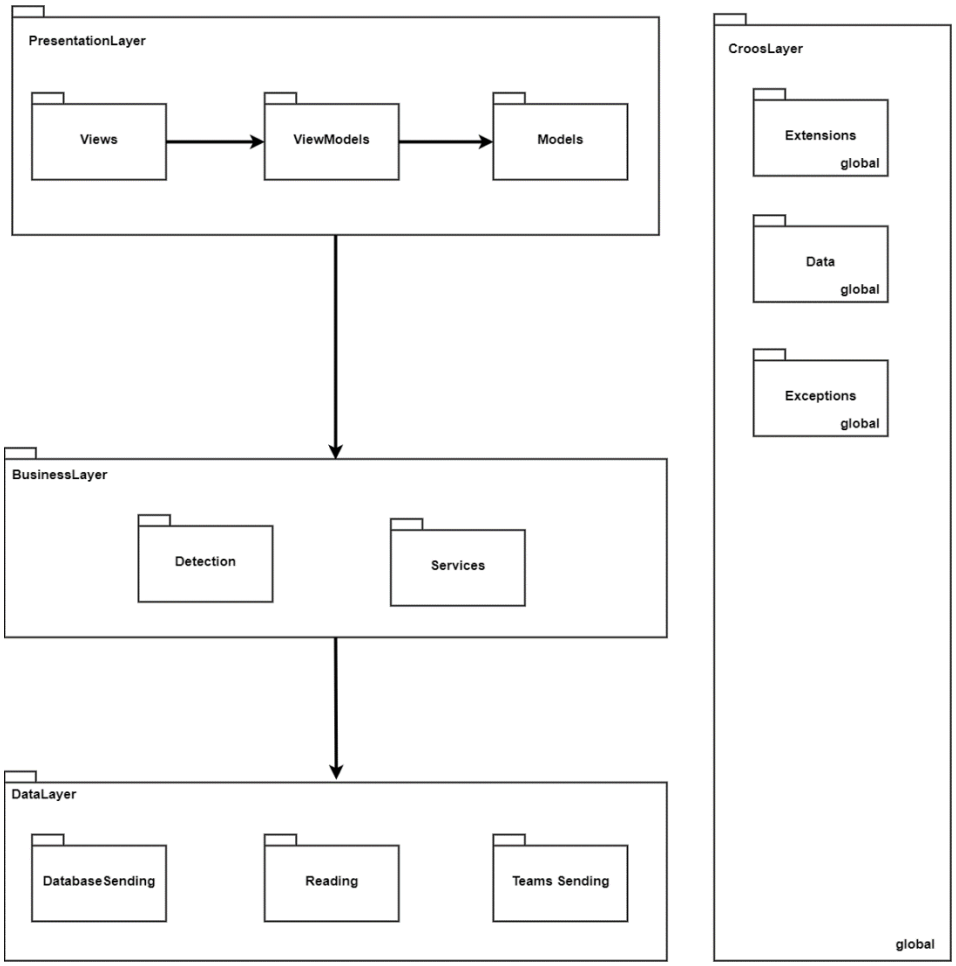


Рис. 20. Диаграмма пакетов

Таблица 43

Описание пакетов

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

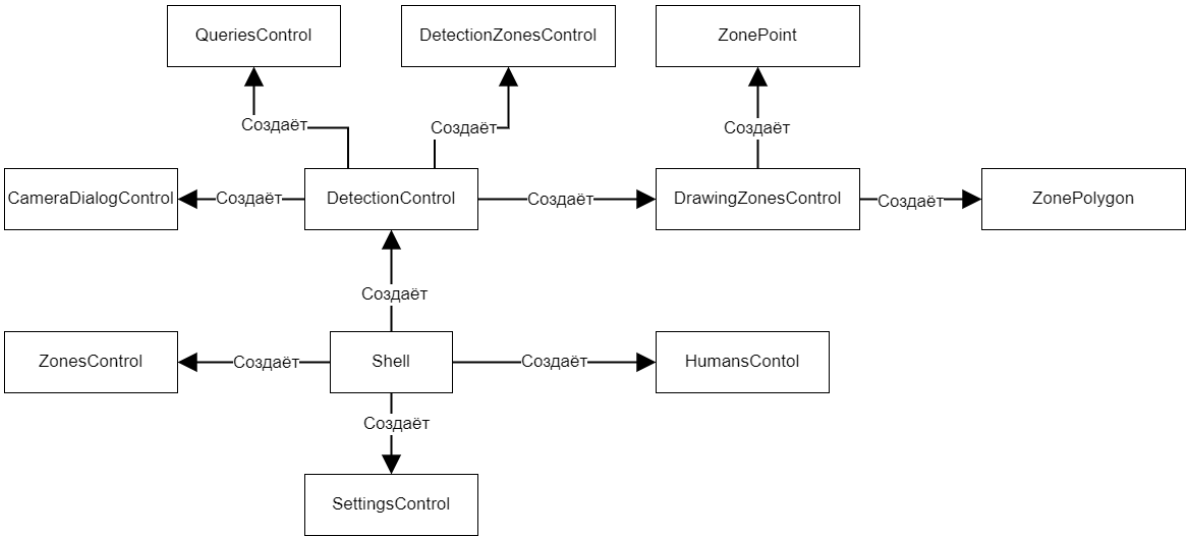


Рис. 21. Исходная диаграмма классов пакета «Views»

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.

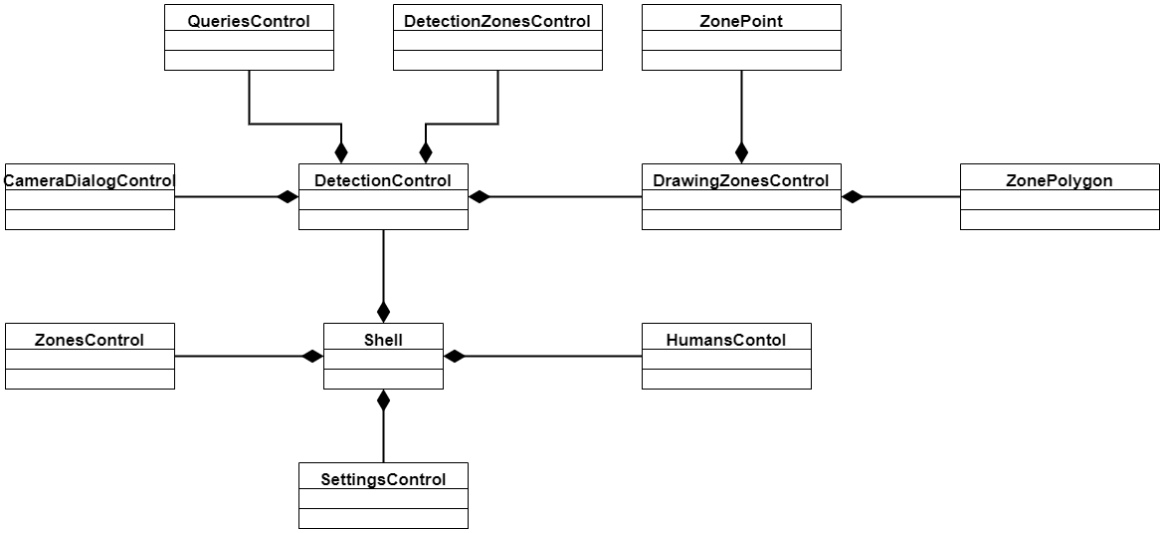


Рис. 22. Уточнённая диаграмма классов пакета «Views»

1.5.2.1.3 Детальная диаграмма классов

Детальная диаграмма классов пакета «Views» представлена на рис. 23.

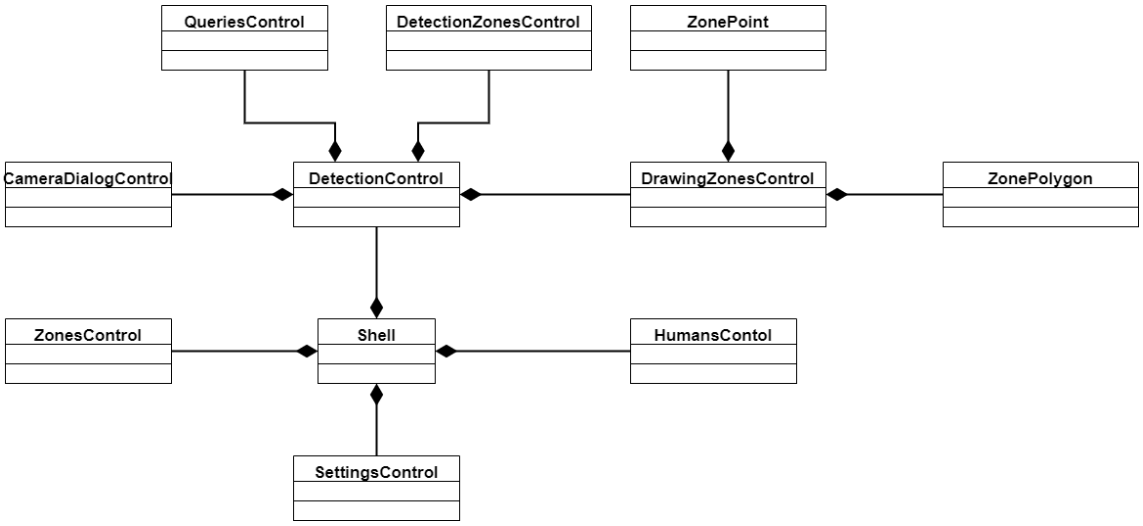


Рис. 23. Детальная диаграмма классов пакета «Views»

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. 45.

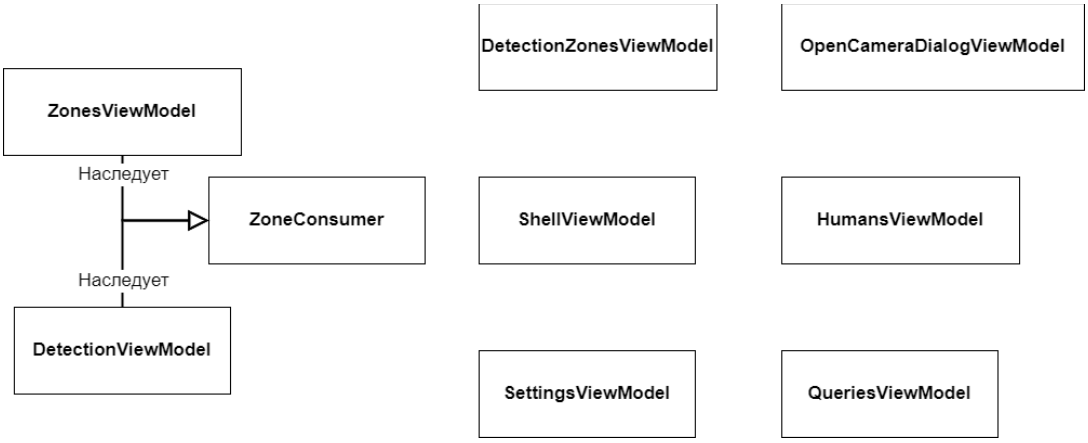


Рис. 24. Исходная диаграмма классов пакета «ViewModels»

Таблица 45

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».

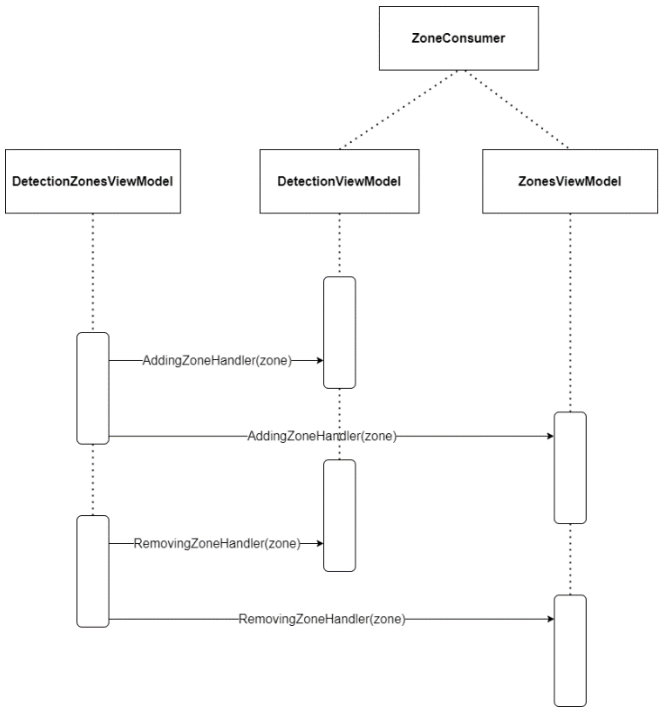


Рис. 25. Диаграмма последовательности действий классов пакета «ViewModels». Нормальный ход событий

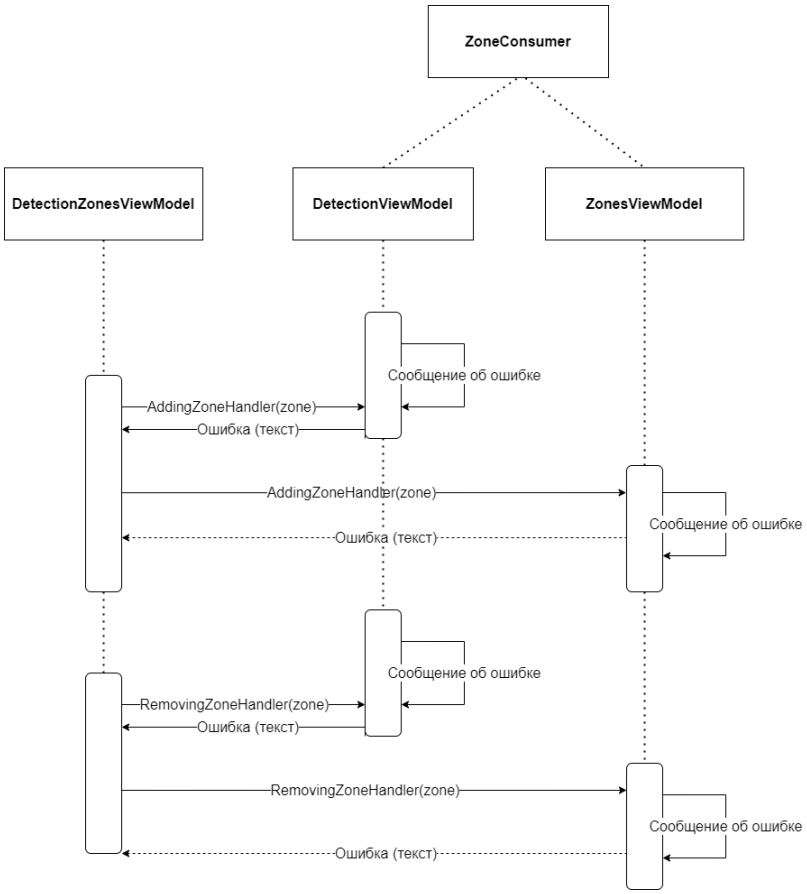


Рис. 26. Диаграмма последовательности действий классов пакета «ViewModels». Прерывание процесса системой

1.5.2.2.3 Уточнённая диаграмма классов

Уточнённая диаграмма классов пакета «ViewModels» представлена на рис. 27.

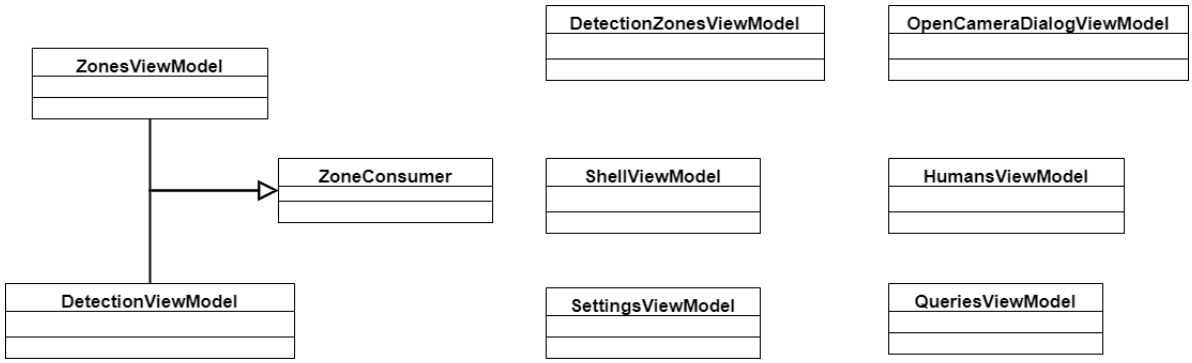


Рис. 27. Уточнённая диаграмма классов пакета «ViewModels»

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. 46-53).

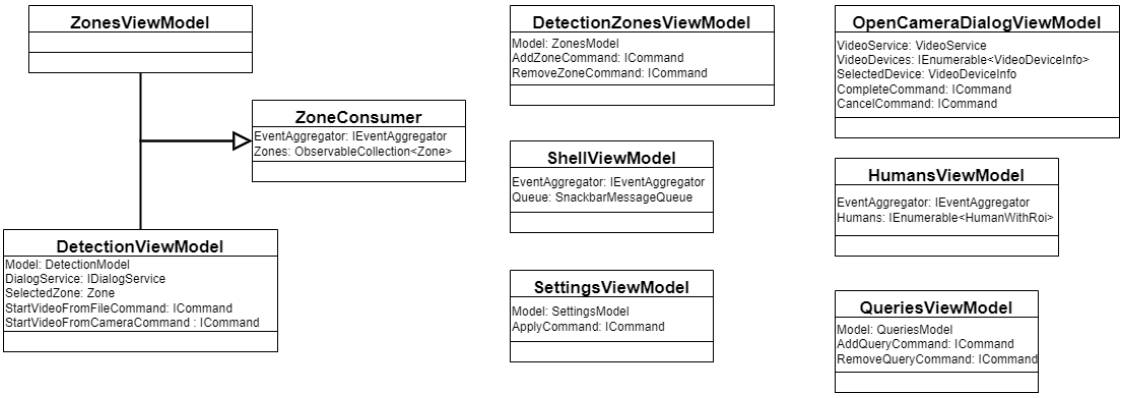


Рис. 28. Детальная диаграмма классов пакета «ViewModels»

Таблица 46

Описание полей класса «ZoneConsumer»

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

Таблица 47

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 48

Описание полей класса «DetectionZonesViewModel»

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

Таблица 49

Описание полей класса «ShellViewModel»

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

Таблица 50

Описание полей класса «SettingsViewModel»

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

Таблица 51

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 52

Описание полей класса «HumansViewModel»

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

Таблица 53

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. 54.

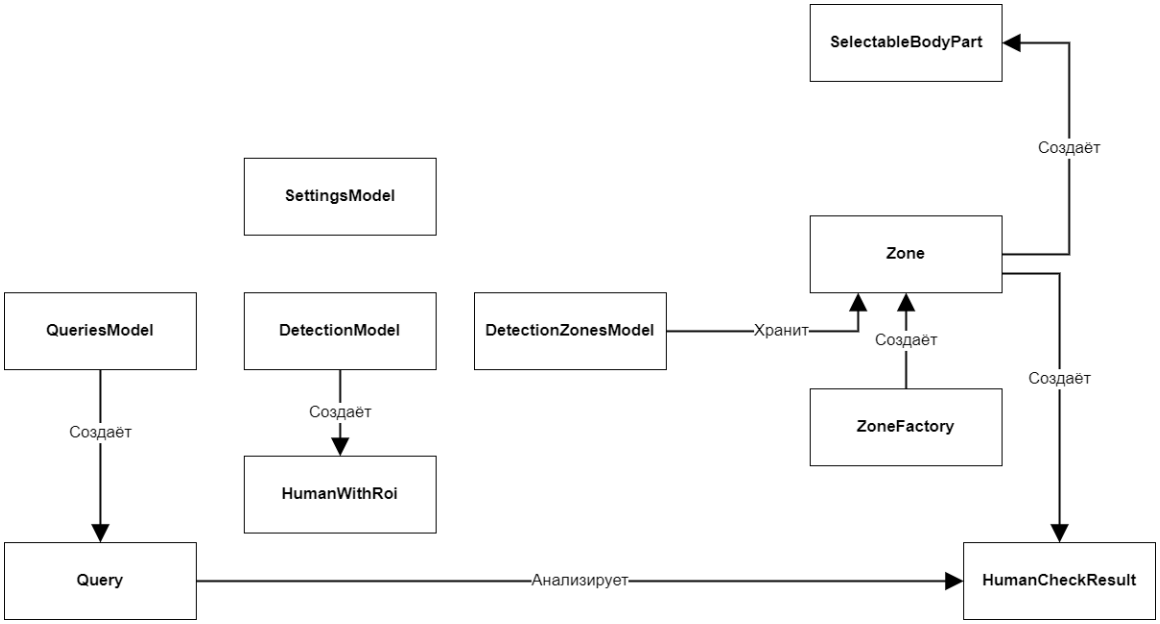


Рис. 29. Исходная диаграмма классов пакета «Models»

Таблица 54

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».

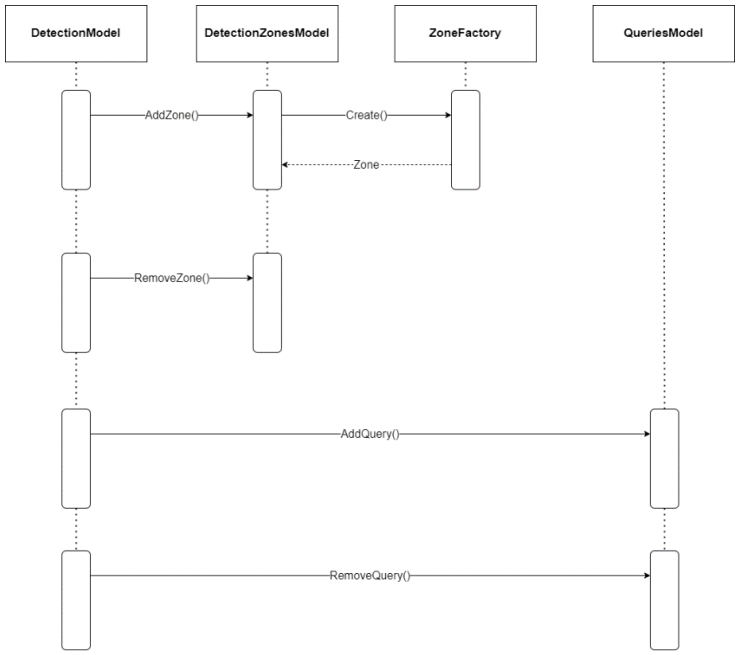


Рис. 30. Диаграмма последовательности действий классов пакета «Models». Нормальный ход событий

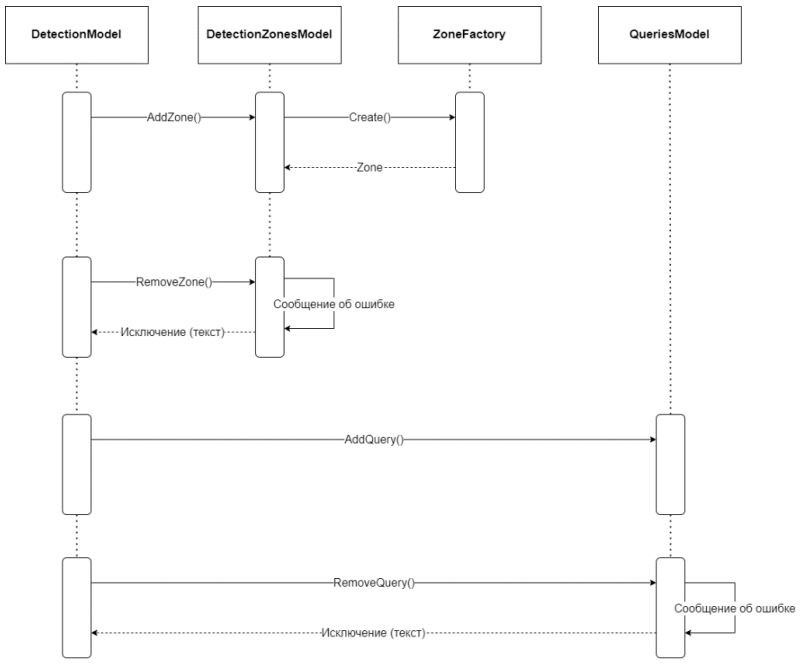


Рис. 31. Диаграмма последовательности действий классов пакета «Models». Прерывание процесса системой

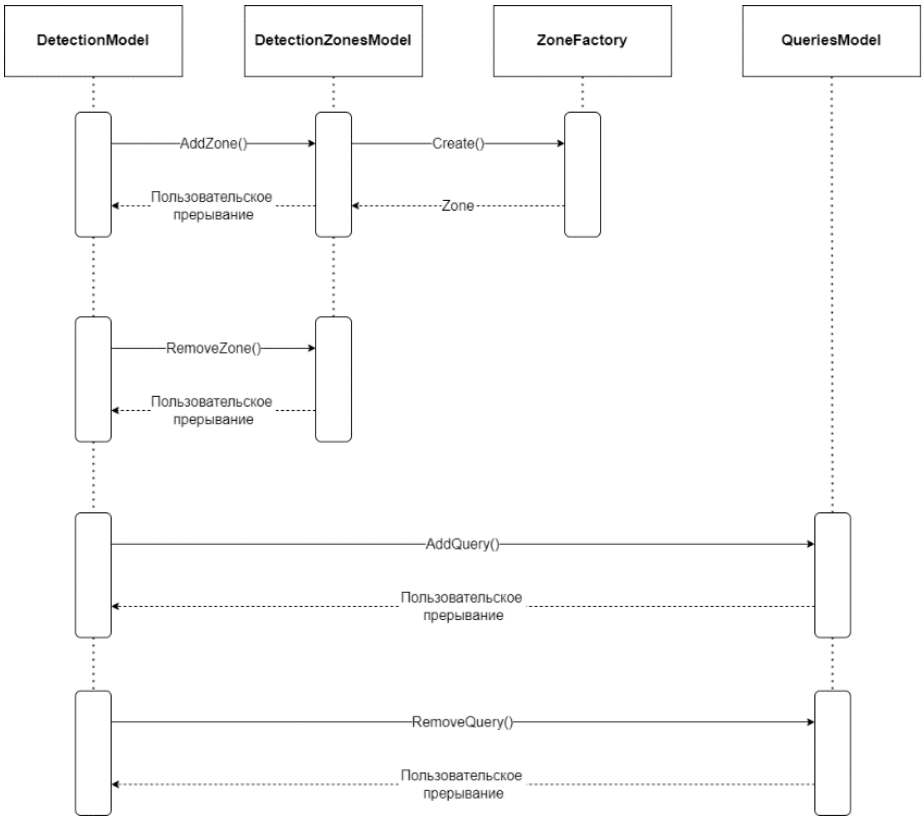


Рис. 32. Диаграмма последовательности действий классов пакета «Models». Прерывание процесса пользователем

1.5.2.3.3 Уточнённая диаграмма классов

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

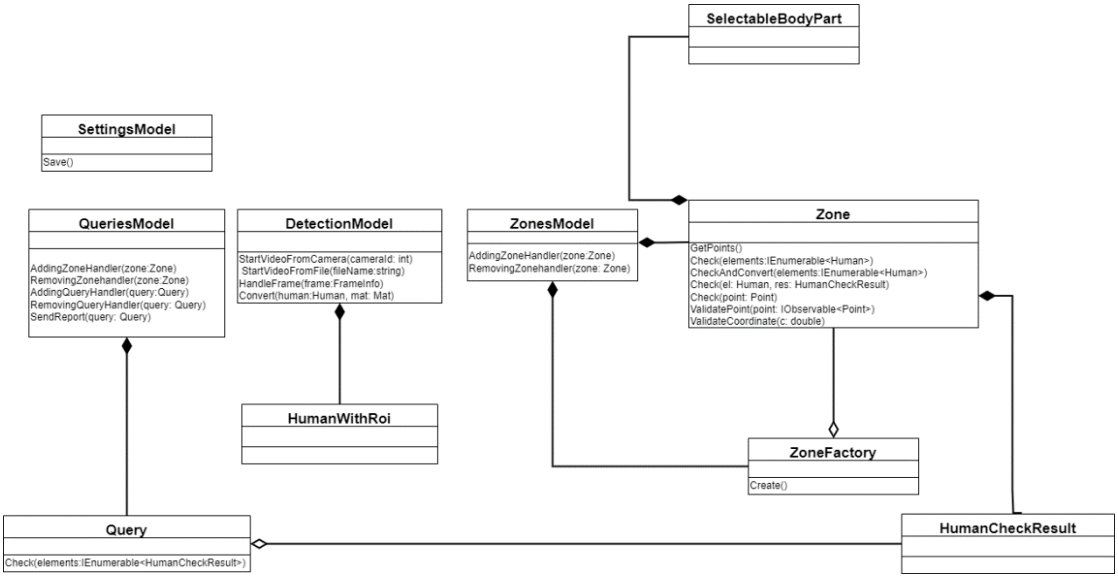


Рис. 33. Уточнённая диаграмма классов пакета «Models»

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. 55-71).

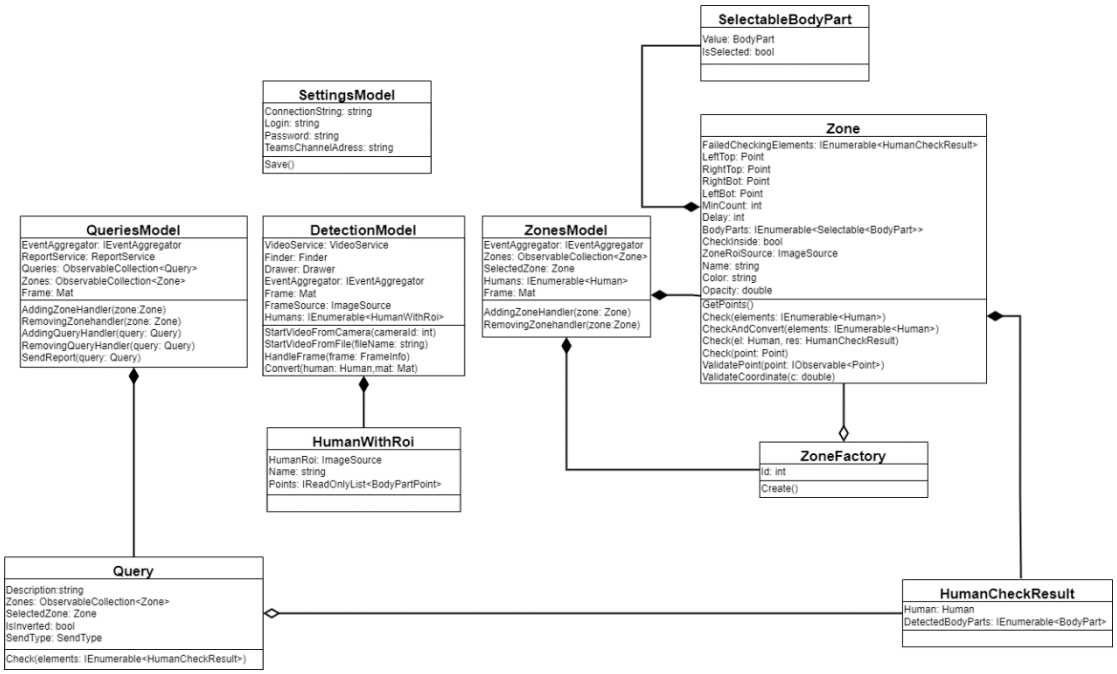


Рис. 34. Детальная диаграмма классов пакета «Models»

Таблица 55

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 56

Описание методов класса «SettingsModel»

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

Таблица 57

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 58

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 59

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 60

Описание методов класса «Query»

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

Таблица 61

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 62

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 63

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 64

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 65

Описание методов класса «ZonesModel»

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

Таблица 66

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 67

Описание методов класса «Zone»

Название	Параметры	Возвращаемое значение	Описание
GetPoints	-	IEnumerable<Point>	Метод получения точек зоны в виде списка
Check	elements: IEnumerable<Human>	-	Метод проверки людей
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 68

Описание полей класса «SelectableBodyPart»

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

Таблица 69

Описание полей класса «ZoneFactory»

Название	Тип	Описание
Id	int	Идентификатор зоны

Таблица 70

Описание методов класса «ZoneFactory»

Название	Параметры	Возвращаемое значение	Описание
Create	-	Zone	

Таблица 71

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. 72.

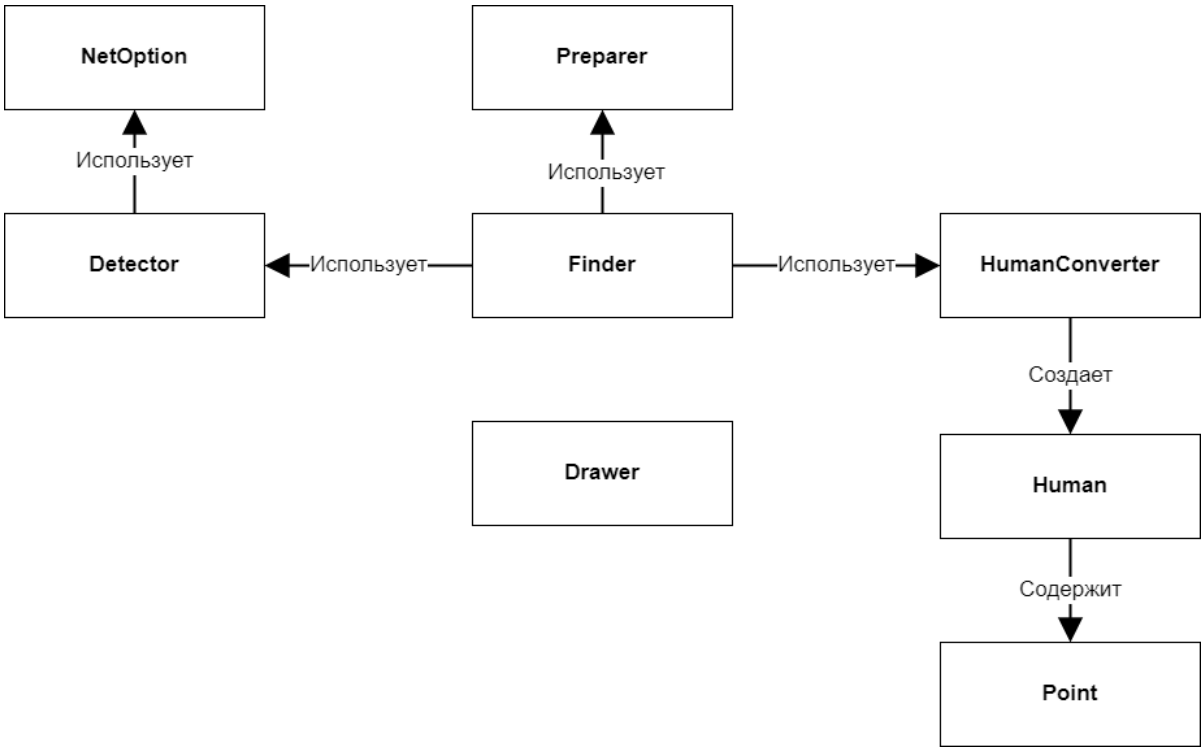


Рис. 35. Исходная диаграмма классов пакета «Detection»

Таблица 72

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».

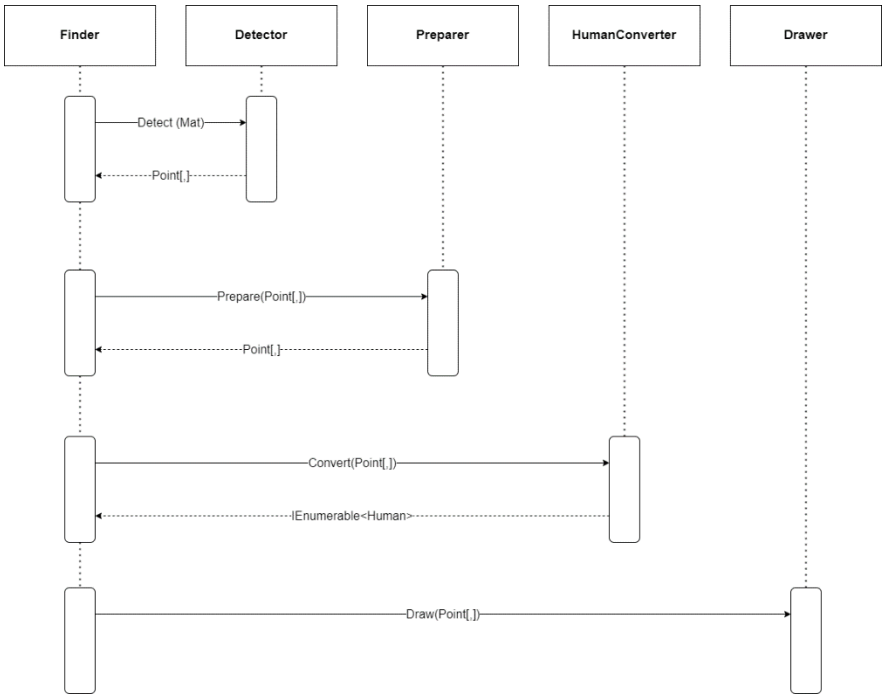


Рис. 36. Диаграмма последовательности действий классов пакета «Detection». Нормальный ход событий

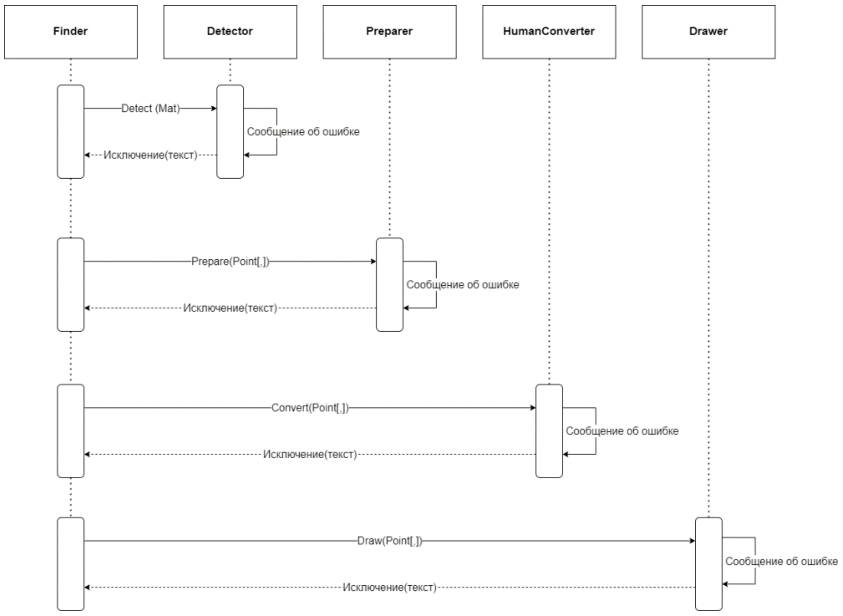


Рис. 37. Диаграмма последовательности действий классов пакета «Detection». Прерывание процесса системой

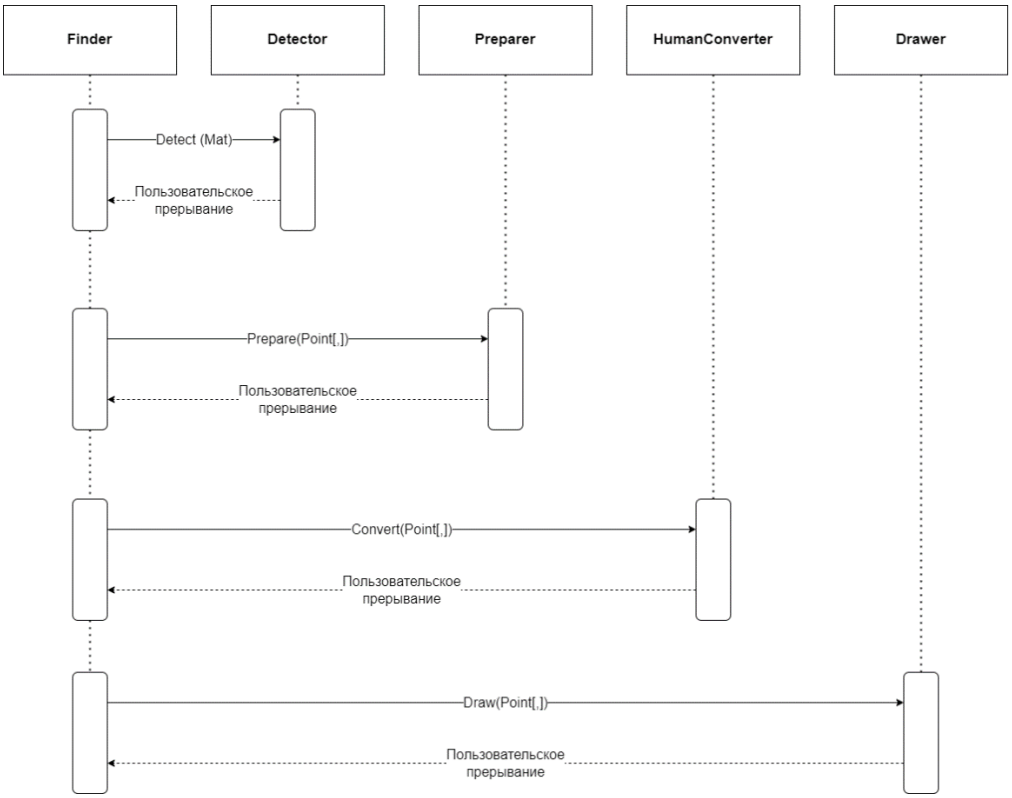


Рис. 38. Диаграмма последовательности действий классов пакета «Detection». Прерывание процесса пользователем

1.5.2.4.3 Уточнённая диаграмма классов

Уточнённая диаграмма классов пакета «Detection» представлена на рис. 39.

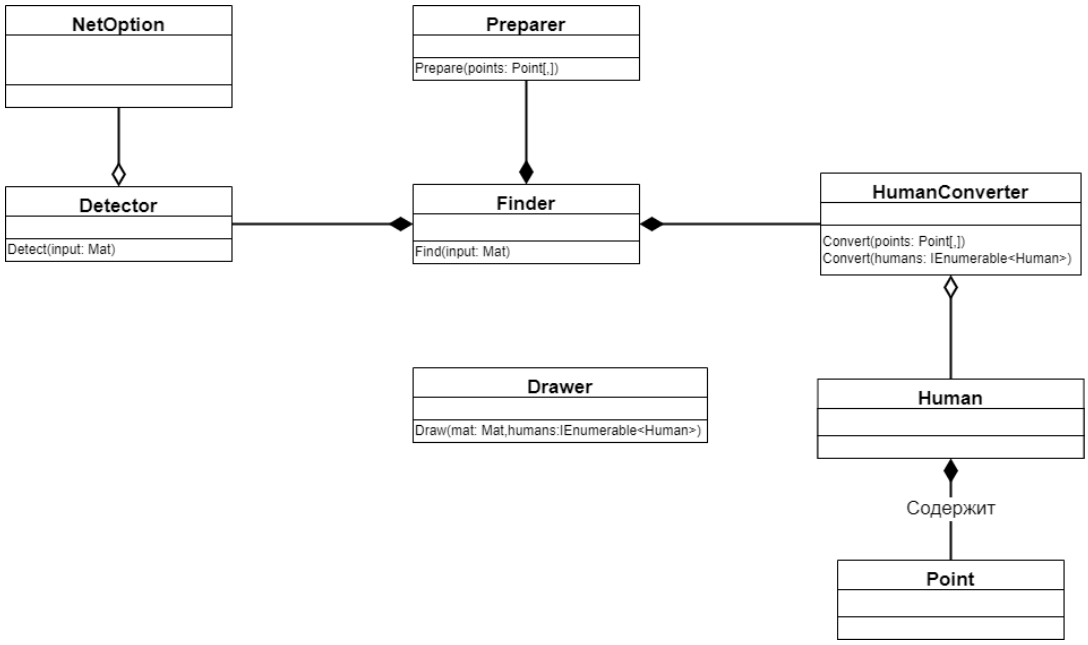


Рис. 39. Уточнённая диаграмма классов пакета «Detection»

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. 73-82).

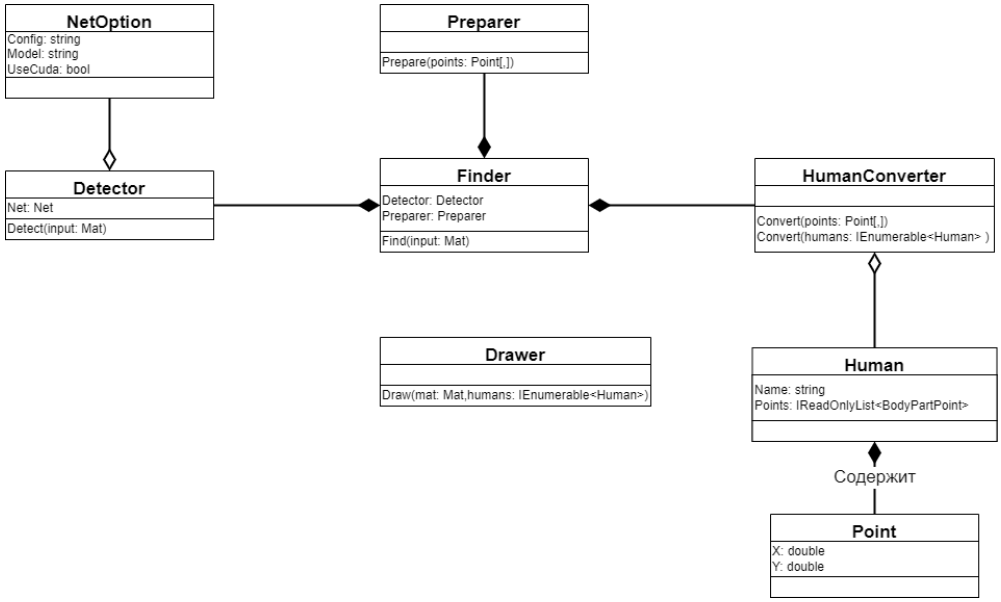


Рис. 40. Детальная диаграмма классов пакета «Detection»

Таблица 73

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 74

Описание методов класса «Finder»

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

Таблица 75

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 76

Описание полей класса «Detector»

Название	Тип	Описание
Net	Net	Объект, характеризующий нейронную сеть

Таблица 77

Описание методов класса «Detector»

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

Таблица 78

Описание методов класса «Preparer»

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

Таблица 79

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 80

Описание методов класса «Drawer»

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

Таблица 81

Описание полей класса «Human»

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

Таблица 82

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. 83.

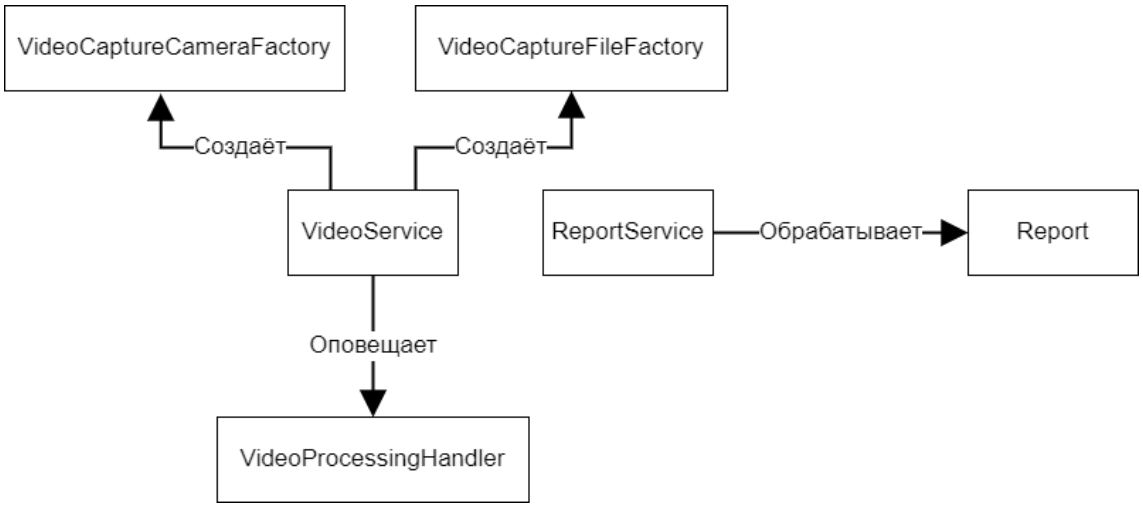


Рис. 41. Исходная диаграмма классов пакета «Services»

Таблица 83

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».

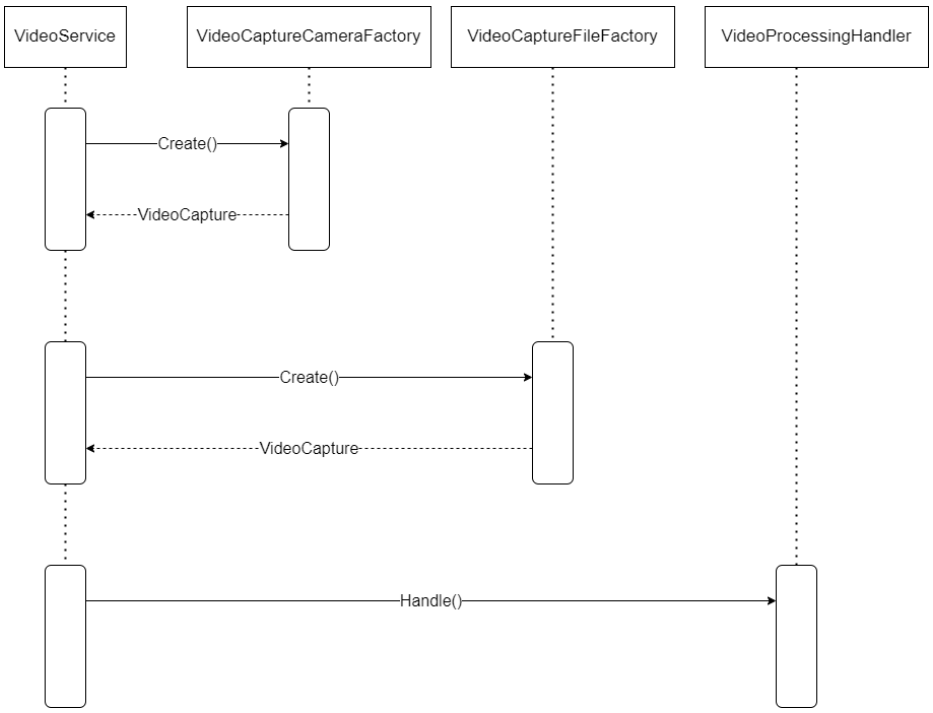


Рис. 42. Диаграмма последовательности действий классов пакета «Services». Нормальный ход событий

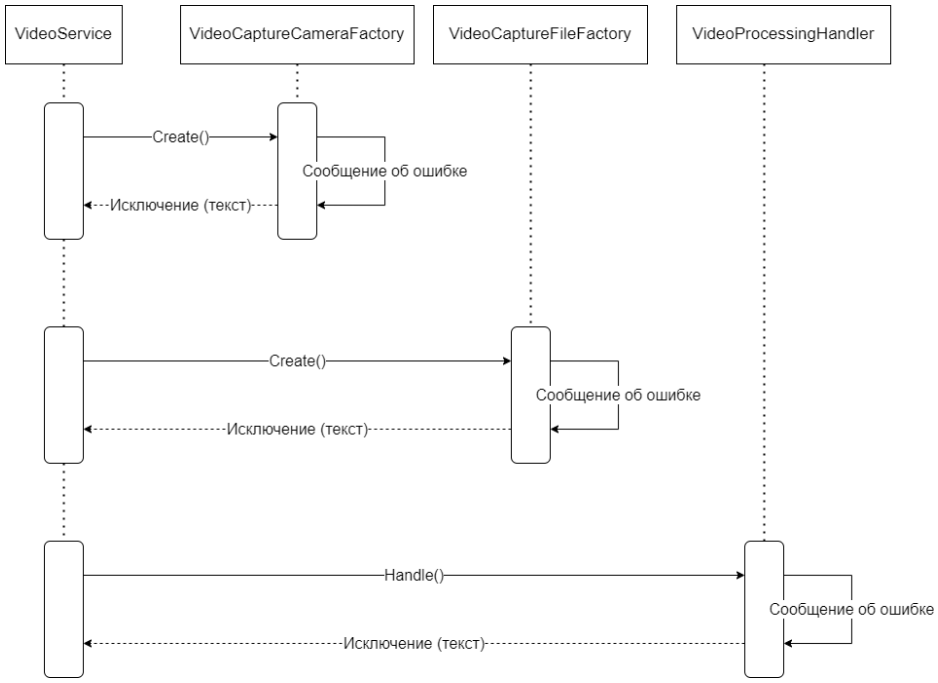


Рис. 43. Диаграмма последовательности действий классов пакета «Services». Прерывание процесса системой

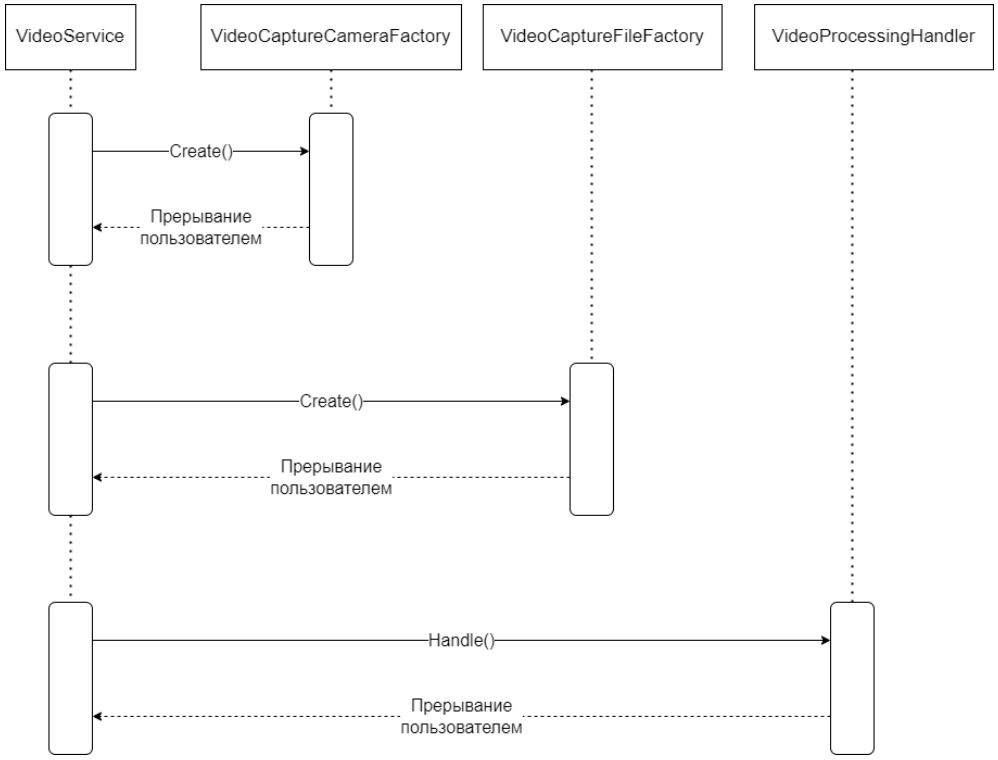


Рис. 44. Диаграмма последовательности действий классов пакета «Services». Прерывание процесса пользователем

1.5.2.5.3 Уточнённая диаграмма классов

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

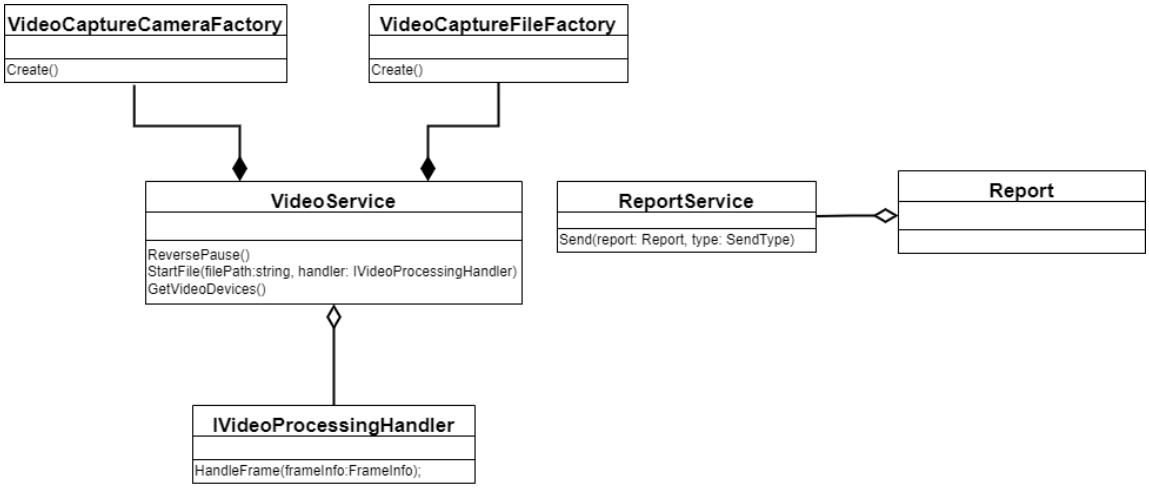


Рис. 45. Уточнённая диаграмма классов пакета «Services»

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. 84-93).

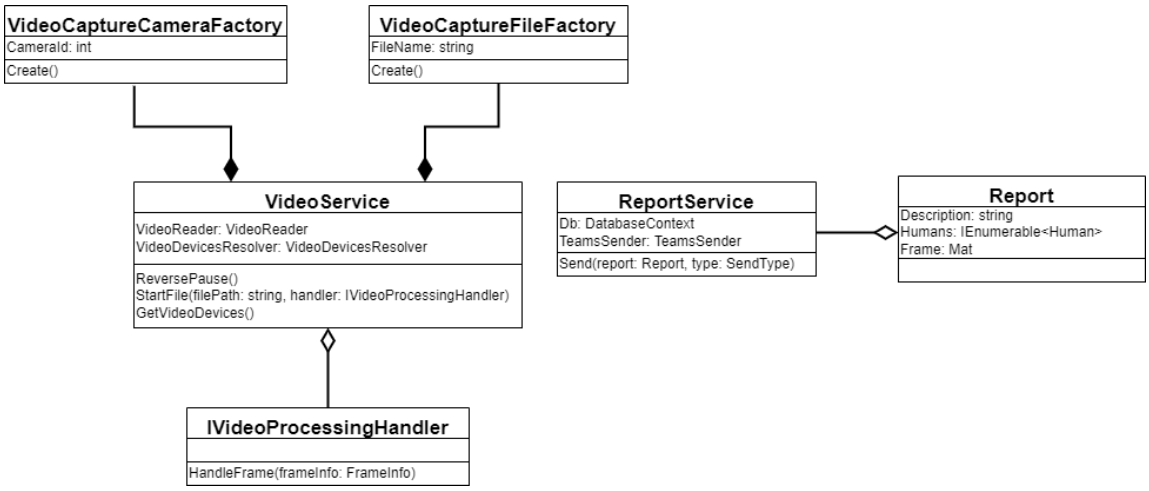


Рис. 46. Детальная диаграмма классов пакета «Services»

Таблица 84

Описание полей класса «VideoCaptureCameraFactory»

Название	Тип	Описание
CameraId	int	Идентификатор камеры

Таблица 85

Описание методов класса «VideoCaptureCameraFactory»

Название	Параметры	Возвращаемое значение	Описание
Create	-	VideoCapture	Метод создания видеопотока

Таблица 86

Описание полей класса «VideoCaptureFileFactory»

Название	Тип	Описание
FileName	string	Путь для файла

Таблица 87

Описание методов класса «VideoCaptureFileFactory»

Название	Параметры	Возвращаемое значение	Описание
Create	-	VideoCapture	Метод создания видеопотока

Таблица 88

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 89

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 90

Описание методов класса «IVideoProcessingHandler»

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

Таблица 91

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 92

Описание методов класса «ReportService»

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

Таблица 93

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. 94.

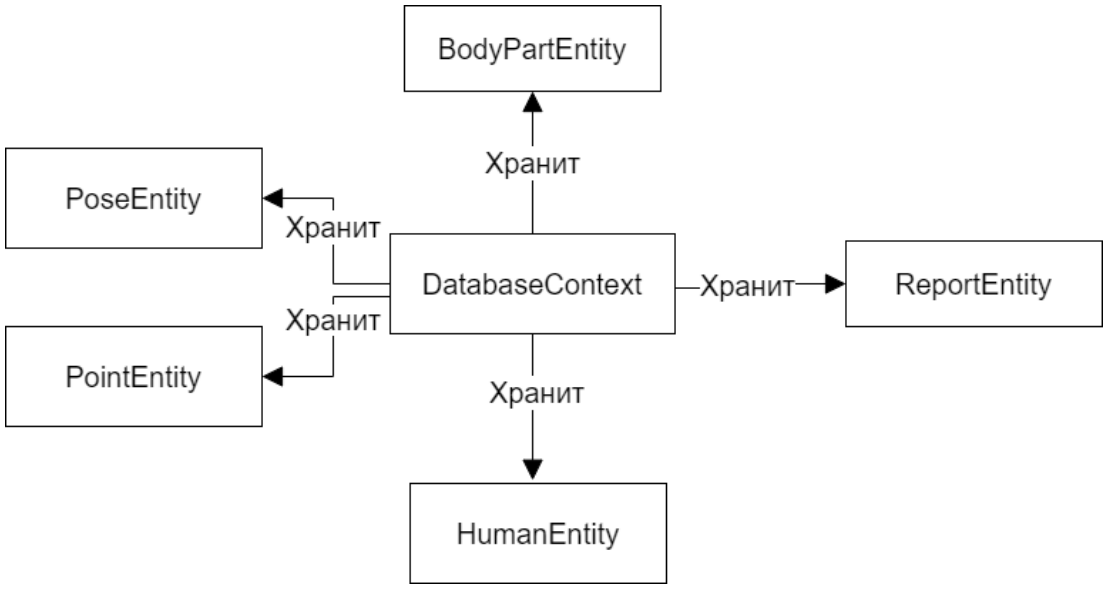


Рис. 47. Исходная диаграмма классов пакета «Database Sending»

Таблица 94

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.

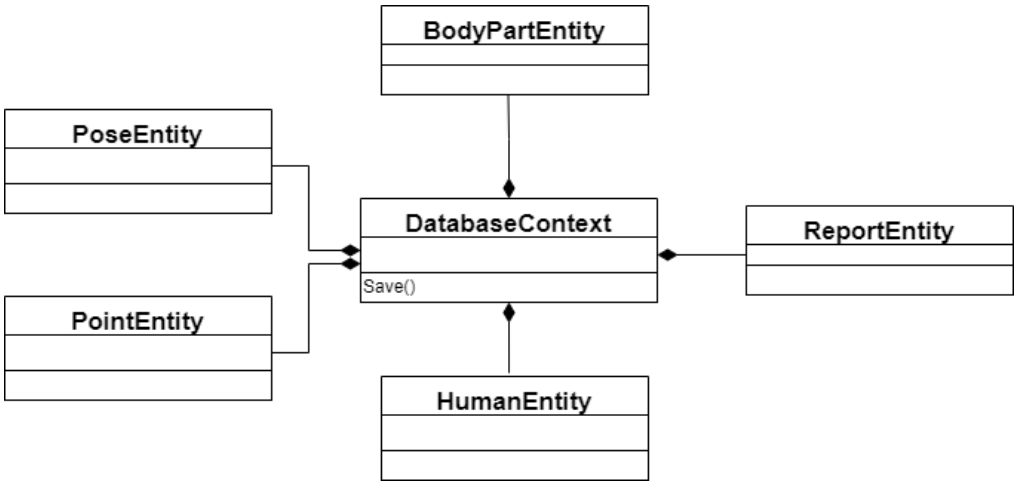


Рис. 48. Уточнённая диаграмма классов пакета «Database Sending»

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. 95-101).

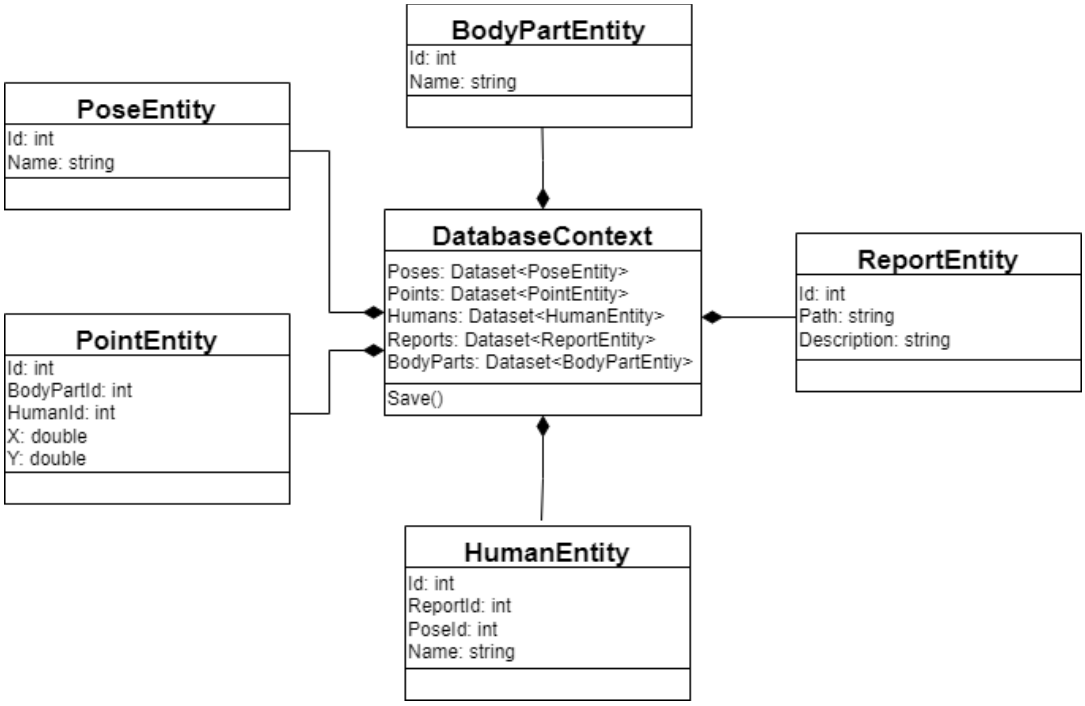


Рис. 49. Детальная диаграмма классов пакета «Database Sending»

Таблица 95

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 96

Описание методов класса «DatabaseContext»

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

Таблица 97

Описание полей класса «PoseEntity»

Название	Тип	Описание
Id	int	Идентификатор позы
Name	string	Имя позы

Таблица 98

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 99

Описание полей класса «HumanEntity»

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

Таблица 100

Описание полей класса «ReportEntity»

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

Таблица 101

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. 102.

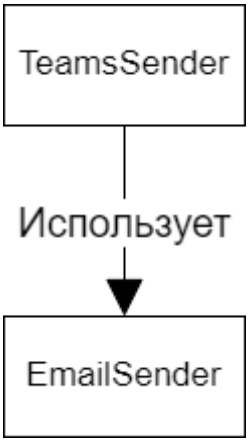


Рис. 50. Исходная диаграмма классов пакета «Teams Sending»

Таблица 102

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».

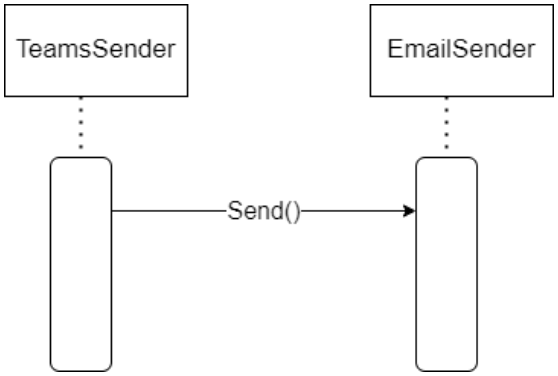


Рис. 51. Диаграмма последовательности действий классов пакета «Teams Sending». Нормальный ход событий

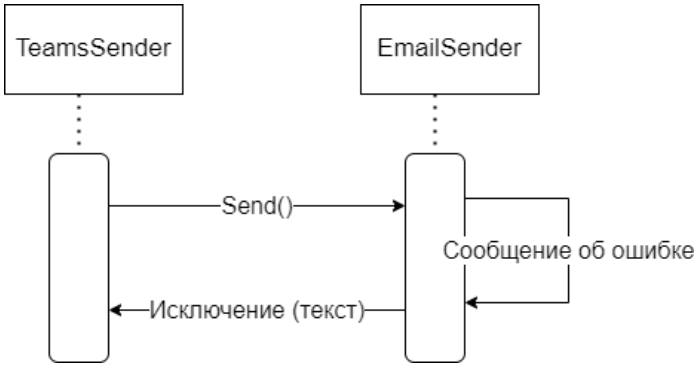


Рис. 52. Диаграмма последовательности действий классов пакета «Teams Sending». Прерывание процесса системой

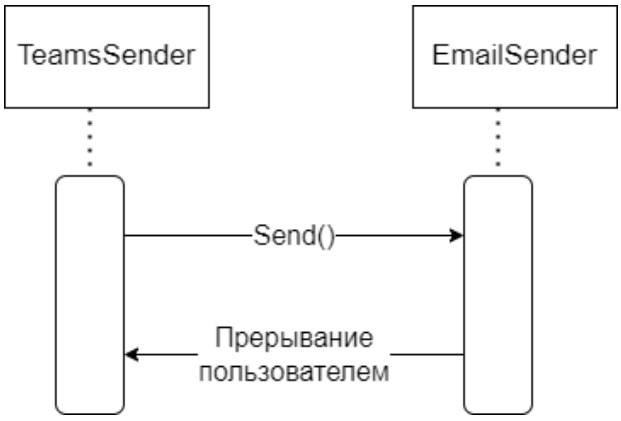


Рис. 53. Диаграмма последовательности действий классов пакета «Teams Sending». Прерывание процесса пользователем

1.5.2.7.3 Уточнённая диаграмма классов

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

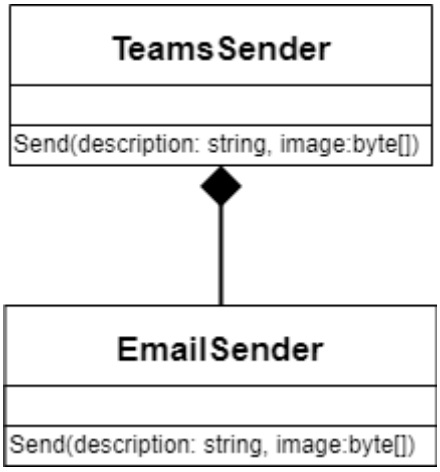


Рис. 54. Уточнённая диаграмма классов пакета «Teams Sending»

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. 103-105).

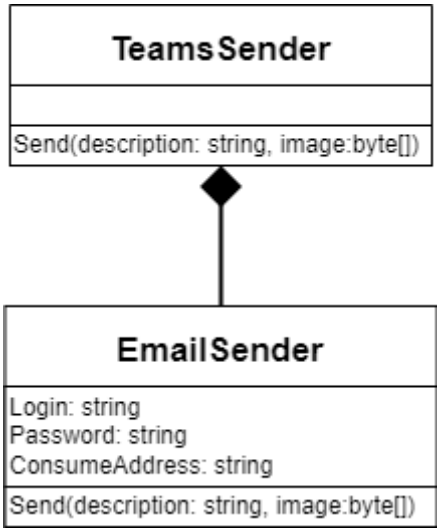


Рис. 55. Детальная диаграмма классов пакета «Teams Sending»

Таблица 103

Описание полей класса «EmailSender»

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

Таблица 104

Описание методов класса «EmailSender»

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

Таблица 105

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. 106.

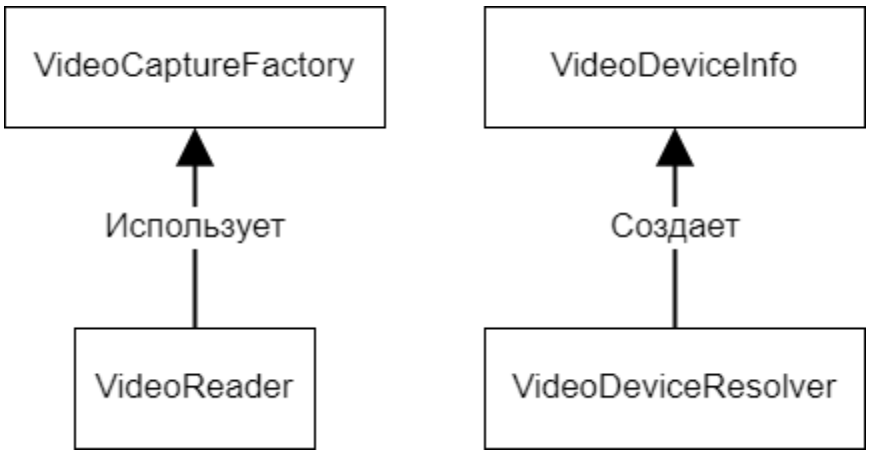


Рис. 56. Исходная диаграмма классов пакета «Reading»

Таблица 106

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.

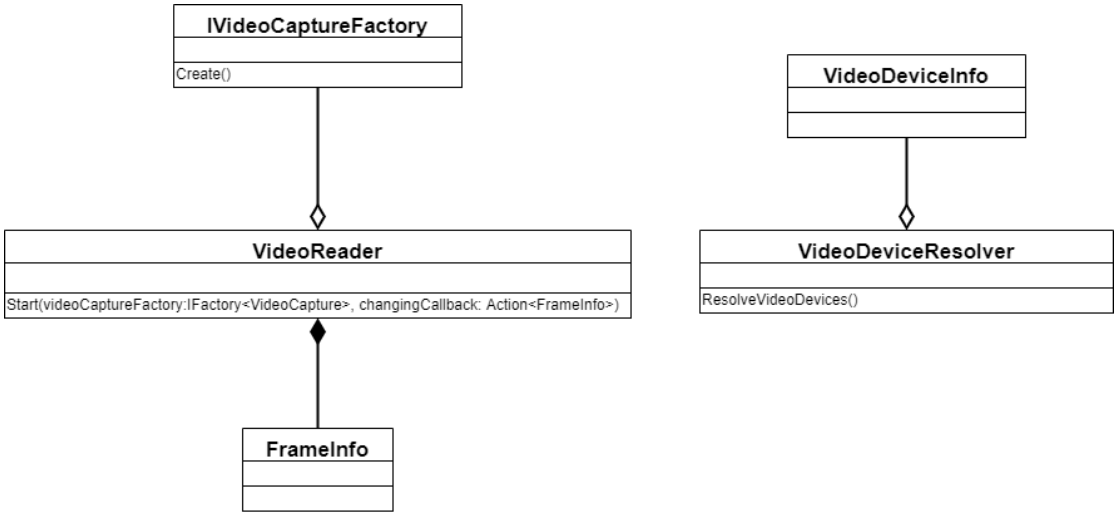


Рис. 57. Уточнённая диаграмма классов пакета «Reading»

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. 107-113).

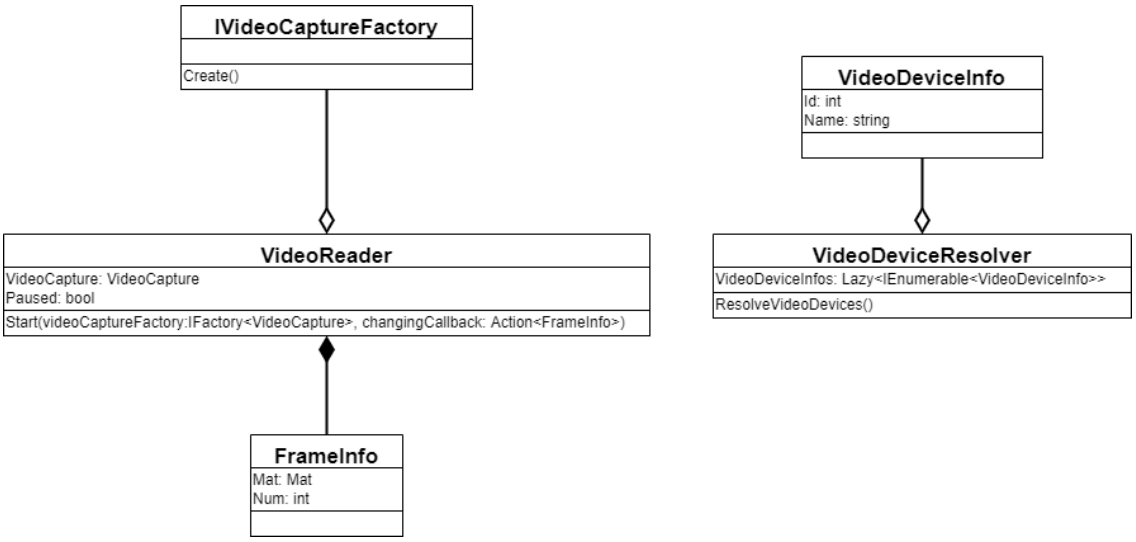


Рис. 58. Детальная диаграмма классов пакета «Reading»

Таблица 107

Описание полей класса «VideoReader»

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

Таблица 108

Описание методов класса «VideoReader»

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

Таблица 109

Описание полей класса «VideoDeviceResolver»

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

Таблица 110

Описание методов класса «VideoDeviceResolver»

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

Таблица 111

Описание полей класса «VideoDeviceInfo»

Название	Тип	Описание
Id	int	Идентификатор камеры
Name	string	Имя камеры

Таблица 112

Описание полей класса «FrameInfo»

Название	Тип	Описание
Mat	Mat	Кадр
Num	int	Номер кадра

Таблица 113

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. 114.

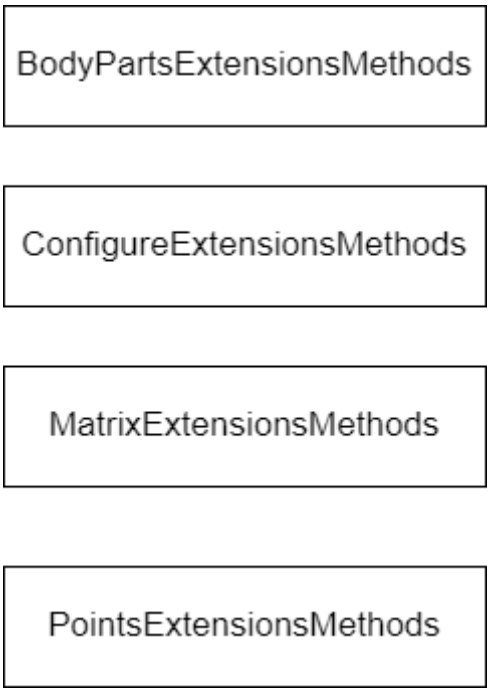


Рис. 59. Исходная диаграмма классов пакета «Extensions»

Таблица 114

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.

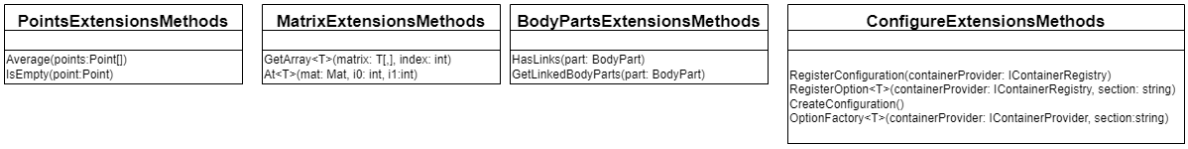


Рис. 60. Уточнённая диаграмма классов пакета «Extensions»

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. 115-118).

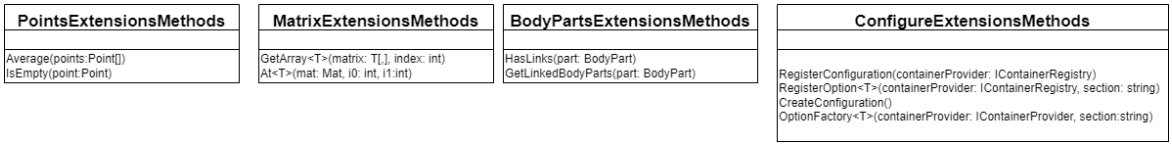


Рис. 61. Детальная диаграмма классов пакета «Extensions»

Таблица 115

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 116

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 117

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 118

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. 119.

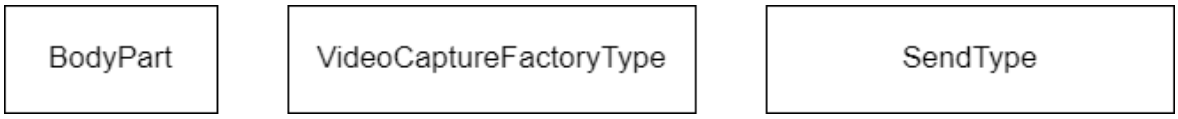


Рис. 62. Исходная диаграмма классов пакета «Data»

Таблица 119

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.

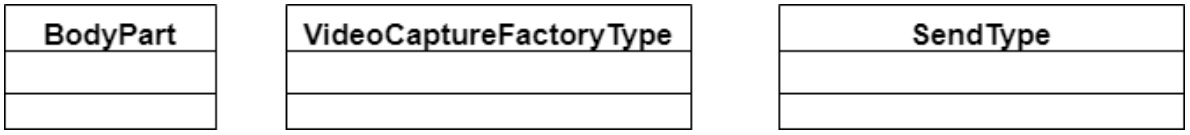


Рис. 63. Уточнённая диаграмма классов пакета «Data»

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. 120-122).

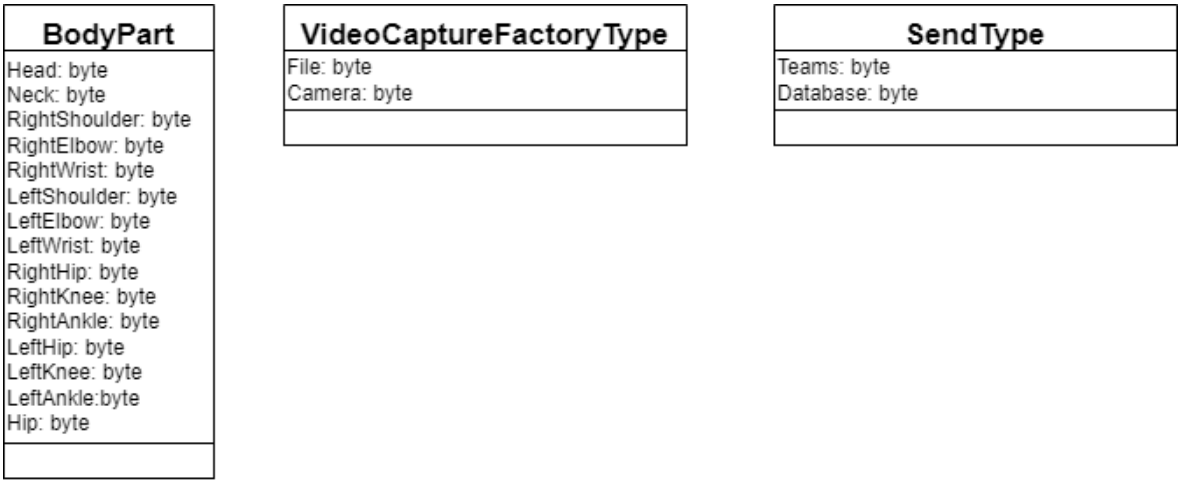


Рис. 64. Детальная диаграмма классов пакета «Data»

Таблица 120

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 121

Описание полей класса «VideoCaptureFactoryType»

Название	Описание
File	Файл
Camera	Камера

Таблица 122

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. 123.

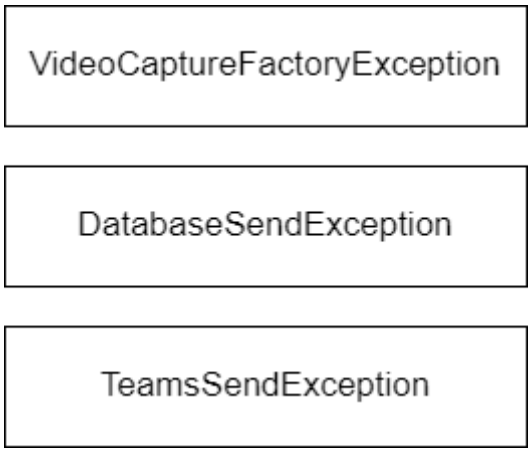


Рис. 65. Исходная диаграмма классов пакета «Exceptions»

Таблица 123

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.

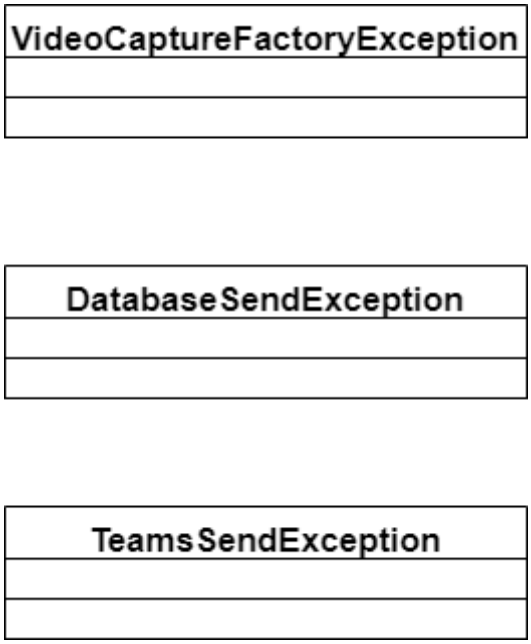


Рис. 66. Уточнённая диаграмма классов пакета «Exceptions»

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. 124-126).

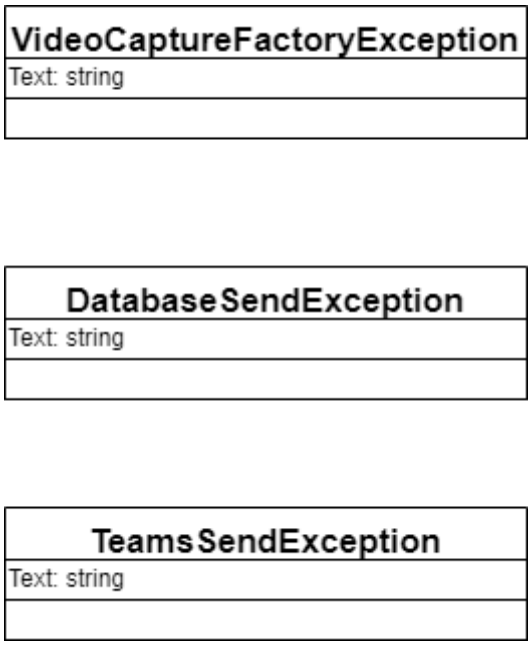


Рис. 67. Детальная диаграмма классов пакета «Exceptions»

Таблица 124

Описание полей класса «VideoCaptureFactoryException»

Название	Тип	Описание
Text	string	Сообщение исключения

Таблица 125

Описание полей класса «DatabaseSendException»

Название	Тип	Описание
Text	string	Сообщение исключения

Таблица 126

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. 127.

Таблица 127

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	Видеопоток	Записи в БД и в канале Teams (журнал) и отчет о нахождении рабочего в зоне

	● Создавать отчет.		
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.

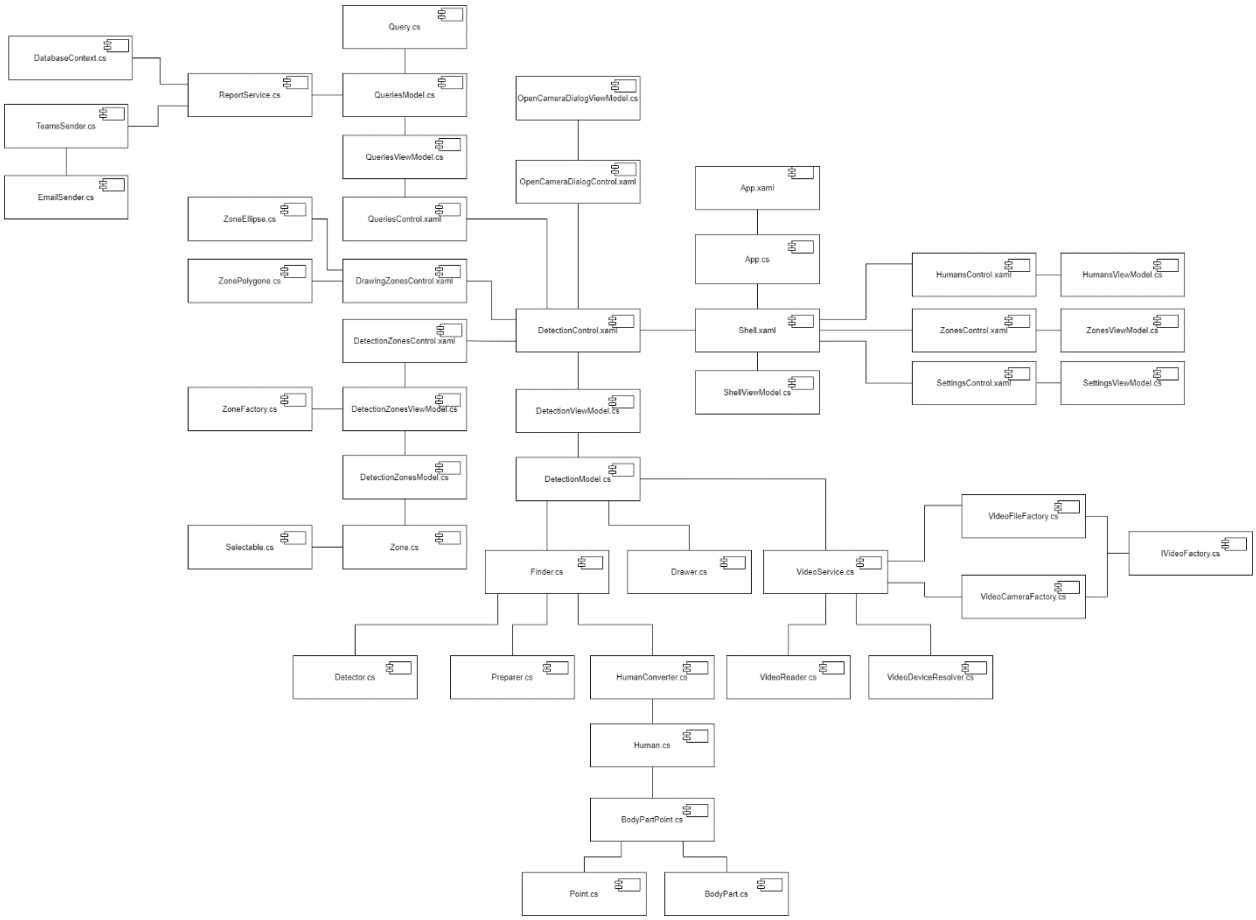


Рис. 69. Модульная структура программы

Сцепление модулей. Сцепление является мерой взаимозависимости модулей, которая определяет, насколько хорошо модули отделены друг от друга. Модули независимы, если каждый из них не содержит о другом никакой информации. Чем больше информации о других модулях хранит модуль, тем больше он с ними сцеплен [7]. Тип сцепления модулей – по данным.

Связность модулей. Связность - мера прочности соединения функциональных и информационных объектов внутри одного модуля. Если сцепление характеризует качество отделения модулей, то связность характеризует степень взаимосвязи элементов, реализуемых одним модулем. Размещение сильно связанных элементов в одном модуле уменьшает межмодульные связи и, соответственно, взаимовлияние модулей. В то же время помещение сильно связанных элементов в разные модули не только усиливает межмодульные связи, но и усложняет понимание их взаимодействия. Объединение слабо связанных элементов также уменьшает

технологичность модулей, так как такими элементами сложнее мысленно манипулировать [7]. Тип связности модулей – последовательный.

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.

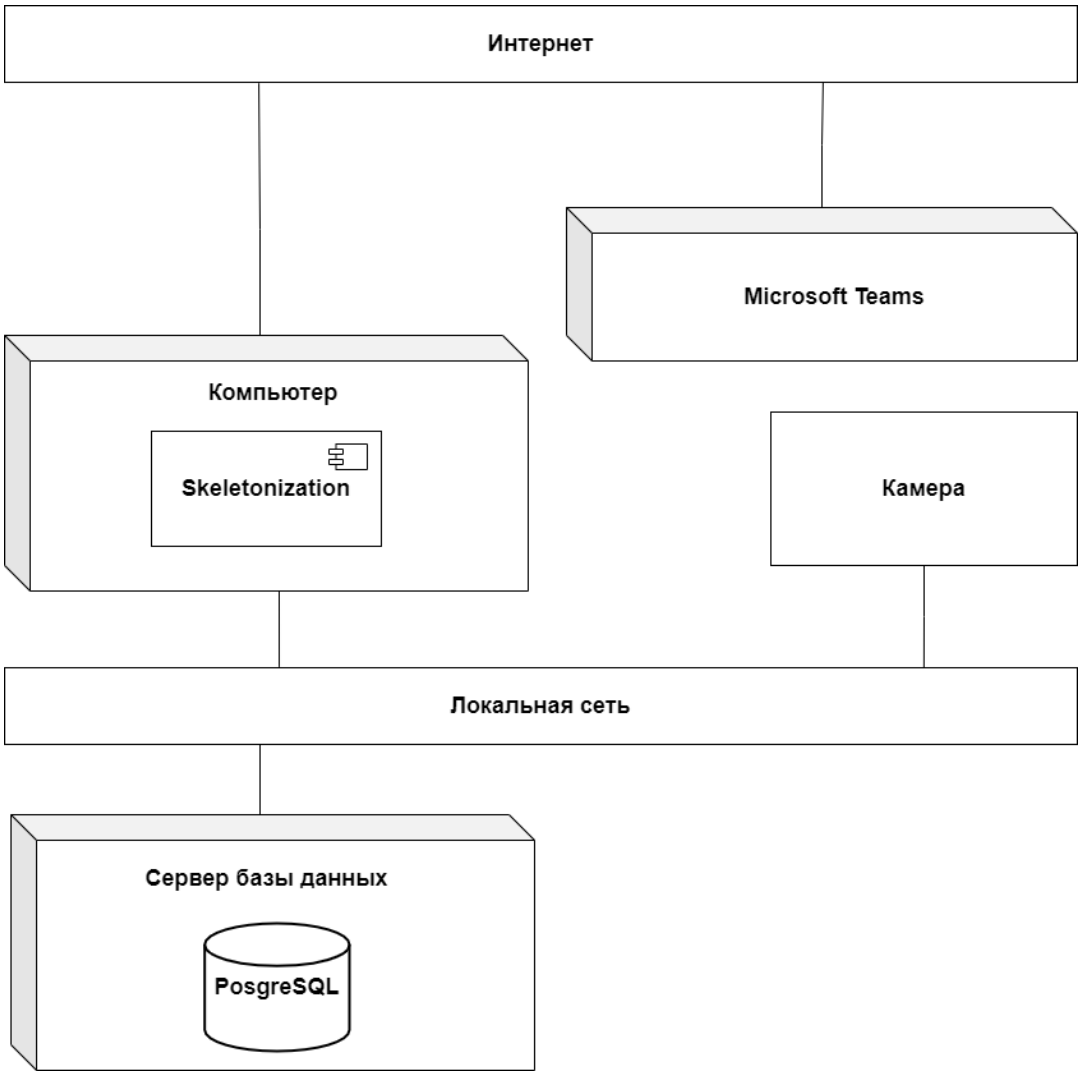


Рис. 70. Диаграмма размещения

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.

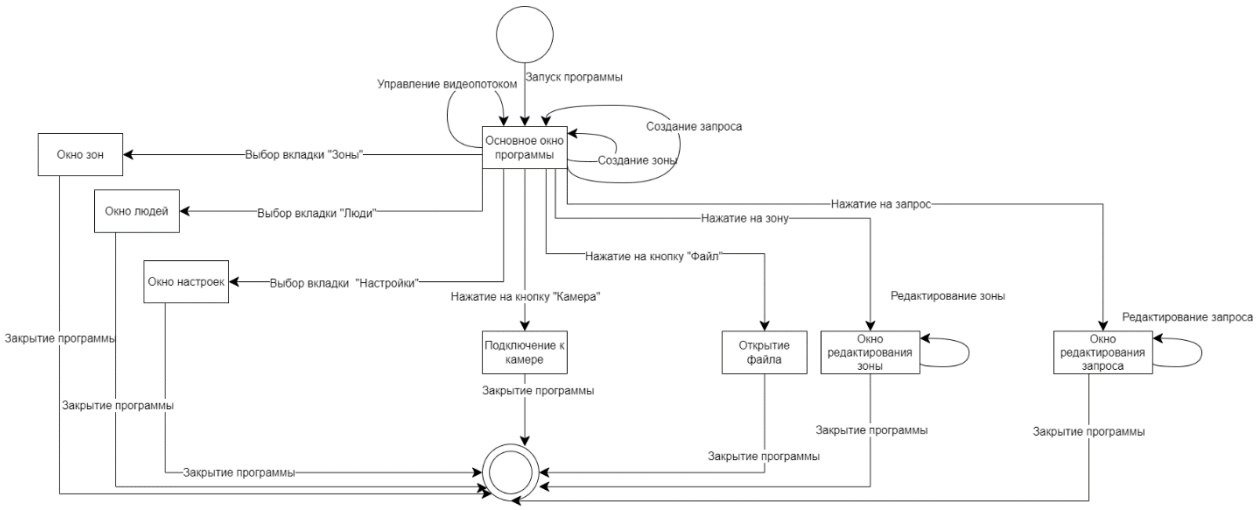


Рис. 71. Граф диалога

1.6.2 Разработка форм ввода-вывода информации

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

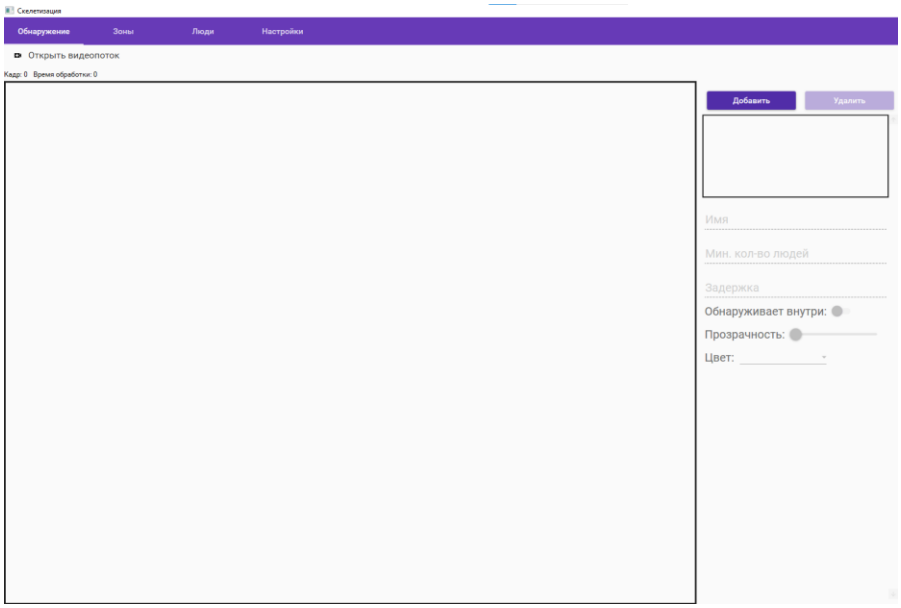


Рис. 72. Главное окно приложения

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

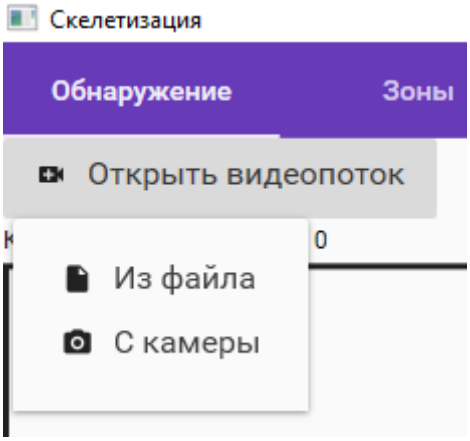


Рис. 73. Меню открытия видеопотка

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

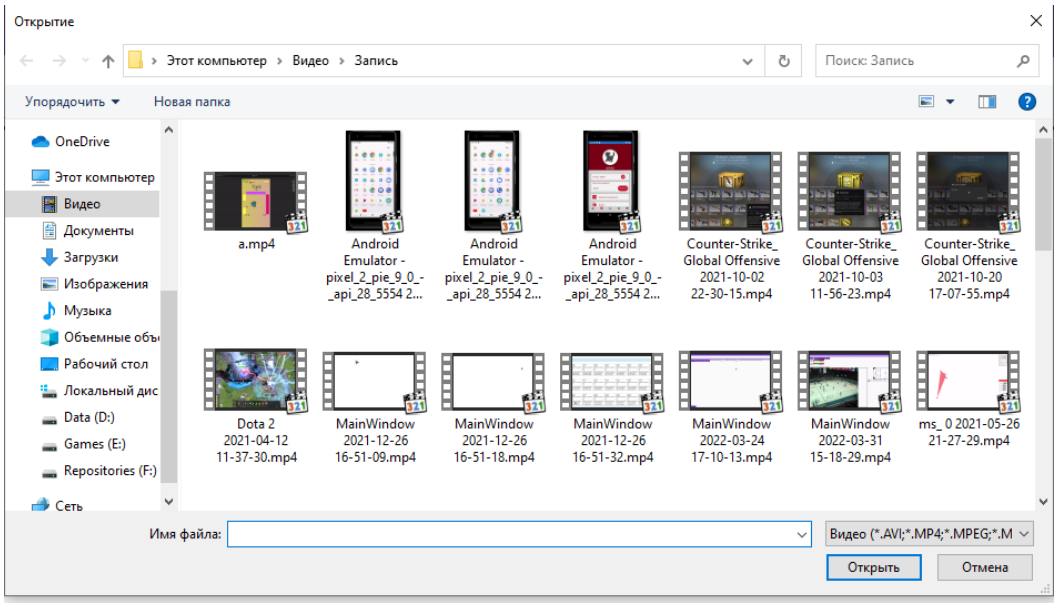


Рис. 74. Открытие файла

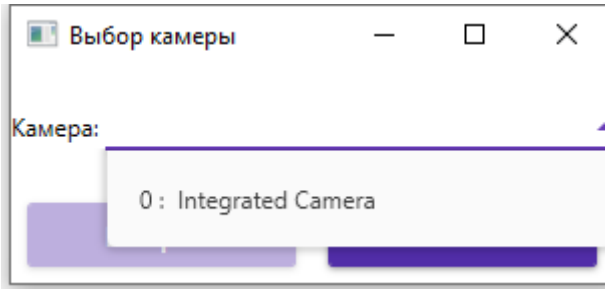


Рис. 75. Выбор камеры

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

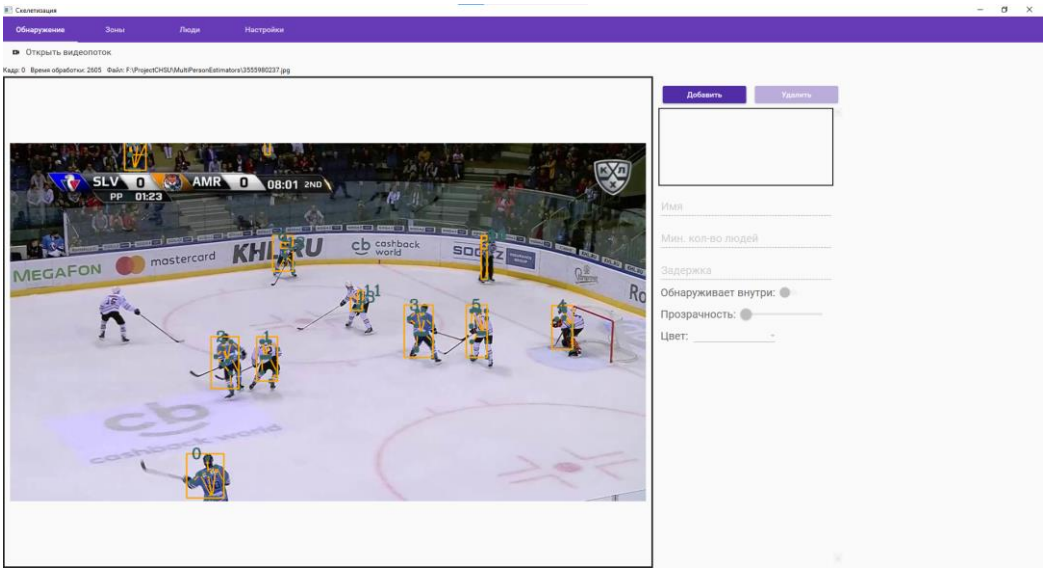


Рис. 76. Детектирование

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

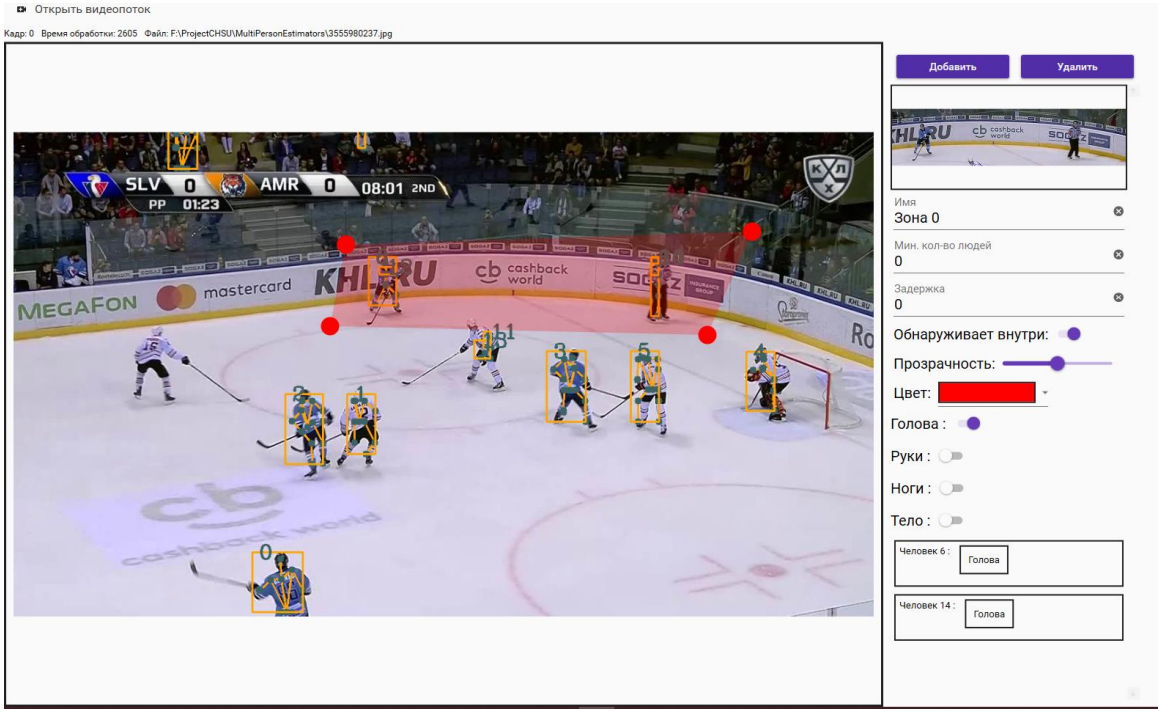


Рис. 77. Работа с зонами

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

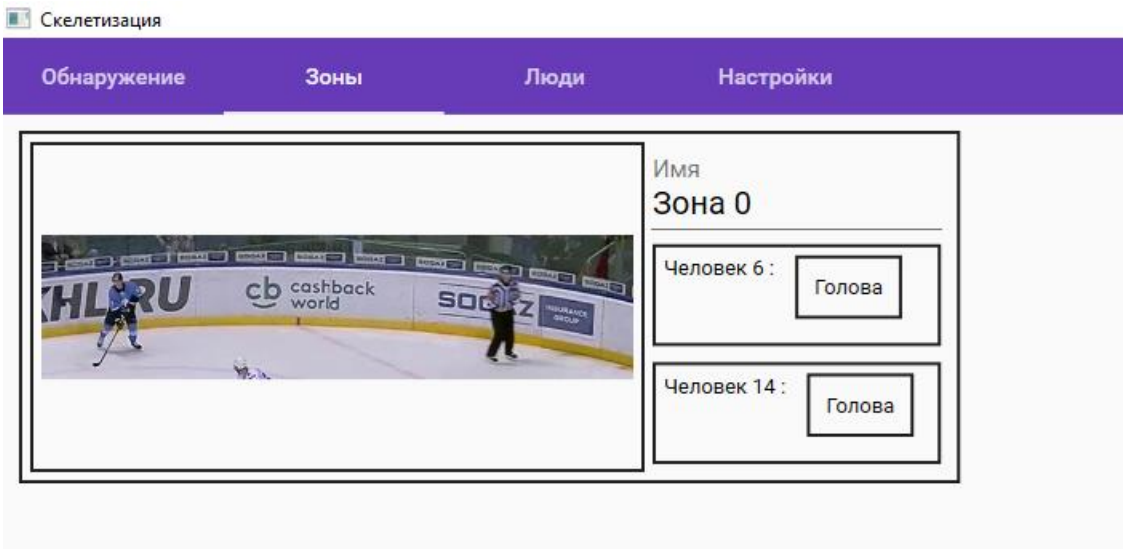


Рис. 78. Окно зон

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

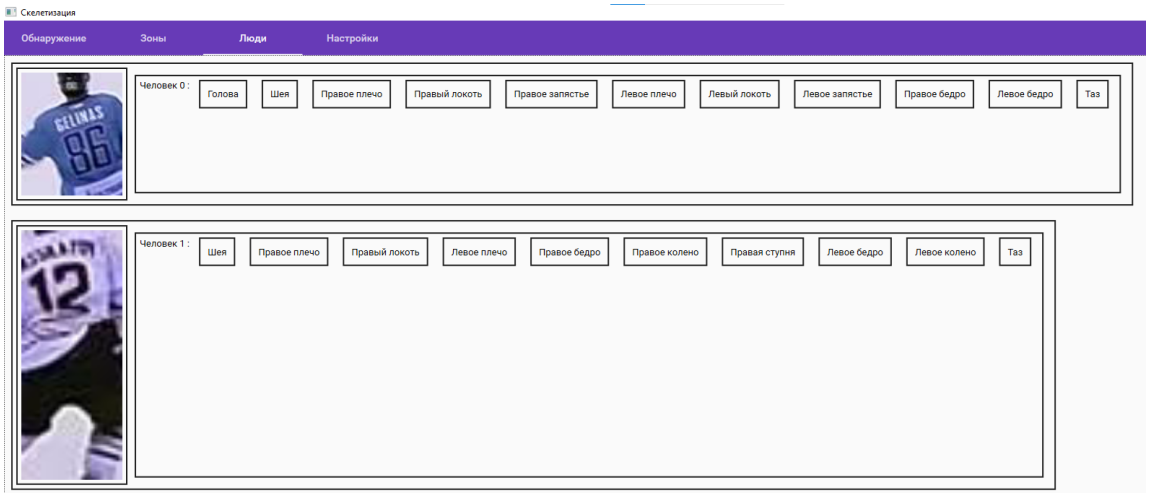


Рис. 79. Окно людей

1.7 Тестирование

Тестирование - очень важный и трудоемкий этап процесса разработки программного обеспечения, так как правильное тестирование позволяет выявить подавляющее большинство ошибок, допущенных при составлении программ [11].

Выделяют следующие виды тестирования:

- модульное. Оно заключаются в тестировании отдельных методов и функций классов, компонентов или модулей, используемых в ПО;
- интеграционное. Проверяется, хорошо ли работают вместе различные модули и сервисы, используемые приложением;
- функциональное. Проверяются только результат некоторого действия и не проверяют промежуточные состояния системы при выполнении этого действия;
- сквозное. Копирует поведение пользователя при работе с ПО в контексте всего приложения;
- и др.

1.7.1 Тестирование модуля формирования опасной ситуации

Результаты тестирования и характер тестов представлены в табл. 128.

Таблица 128

Тестирование

Дата	Тестируемый модуль или подпрограмма	Кто проводил тестирование	Способ тестирования	Результат тестирования
25.12.2021	Query.cs CheckPerson	Разработчик	Функциональное	Неудача, люди, не подходящие под запрос, были помечены как нарушители
25.12.2021	Query.cs	Разработчик	Функциональное	Успех

	CheckPerson			
25.12.2021	Query.cs AddZone	Разработчик	Функциональное	Успех
25.12.2021	Query.cs RemoveZone	Разработчик	Функциональное	Неудача, удалена зона не была удалена из запроса
25.12.2021	Query.cs RemoveZone	Разработчик	Функциональное	Успех
25.12.2021	Query.cs CreateReport	Разработчик	Функциональное	Неудача, в отчёте неправильно сформировалось описание ситуации
25.12.2021	Query.cs CreateReport	Разработчик	Функциональное	Успех
25.12.2021	Zone.cs CheckPoint	Разработчик	Функциональное	Неудача, зона обнаруживает все точки, находящиеся не внутри, а снаружи
25.12.2021	Zone.cs CheckPoint	Разработчик	Функциональное	Успех
25.12.2021	Zone.cs AddPart	Разработчик	Функциональное	Успех
25.12.2021	Zone.cs RemovePart	Разработчик	Функциональное	Успех
25.12.2021	Zone.cs SetMaxHumanCount	Разработчик	Функциональное	Неудача, удалось задать отрицательное количество
25.12.2021	Zone.cs	Разработчик	Функциональное	Успех

	SetMaxHumanCount			
25.12.2021	Zone.cs SetDetectionInside	Разработчик	Функциональное	Неудача, изменение параметра работает инвертировано
25.12.2021	Zone.cs SetDetectionInside	Разработчик	Функциональное	Успех
25.12.2021	Report.cs Send	Разработчик	Функциональное	Неудача, не удалось отправить отчёт в Teams
25.12.2021	Zone.cs Send	Разработчик	Функциональное	Успех

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

Рис. 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 назначим данные 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

Рис. 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.

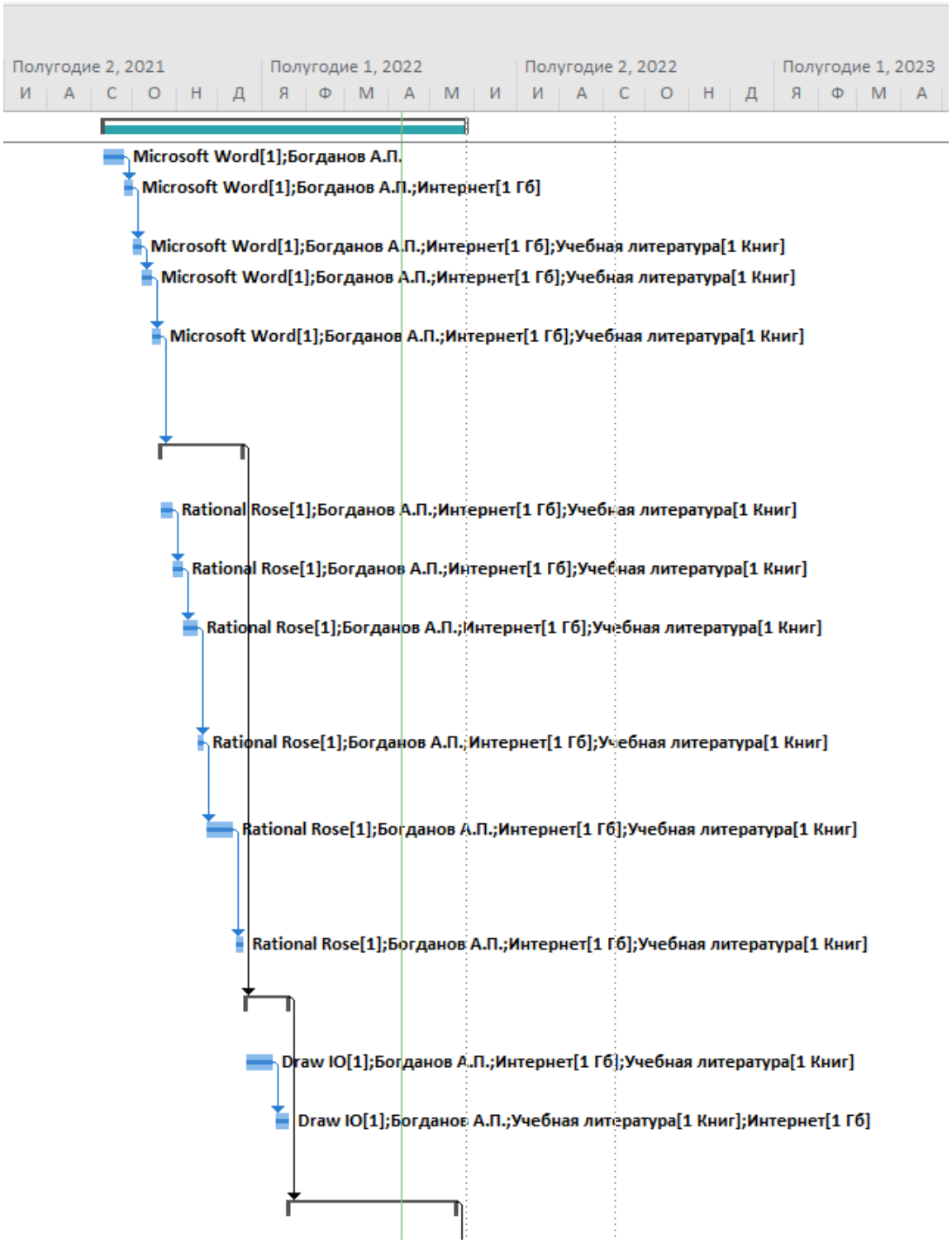
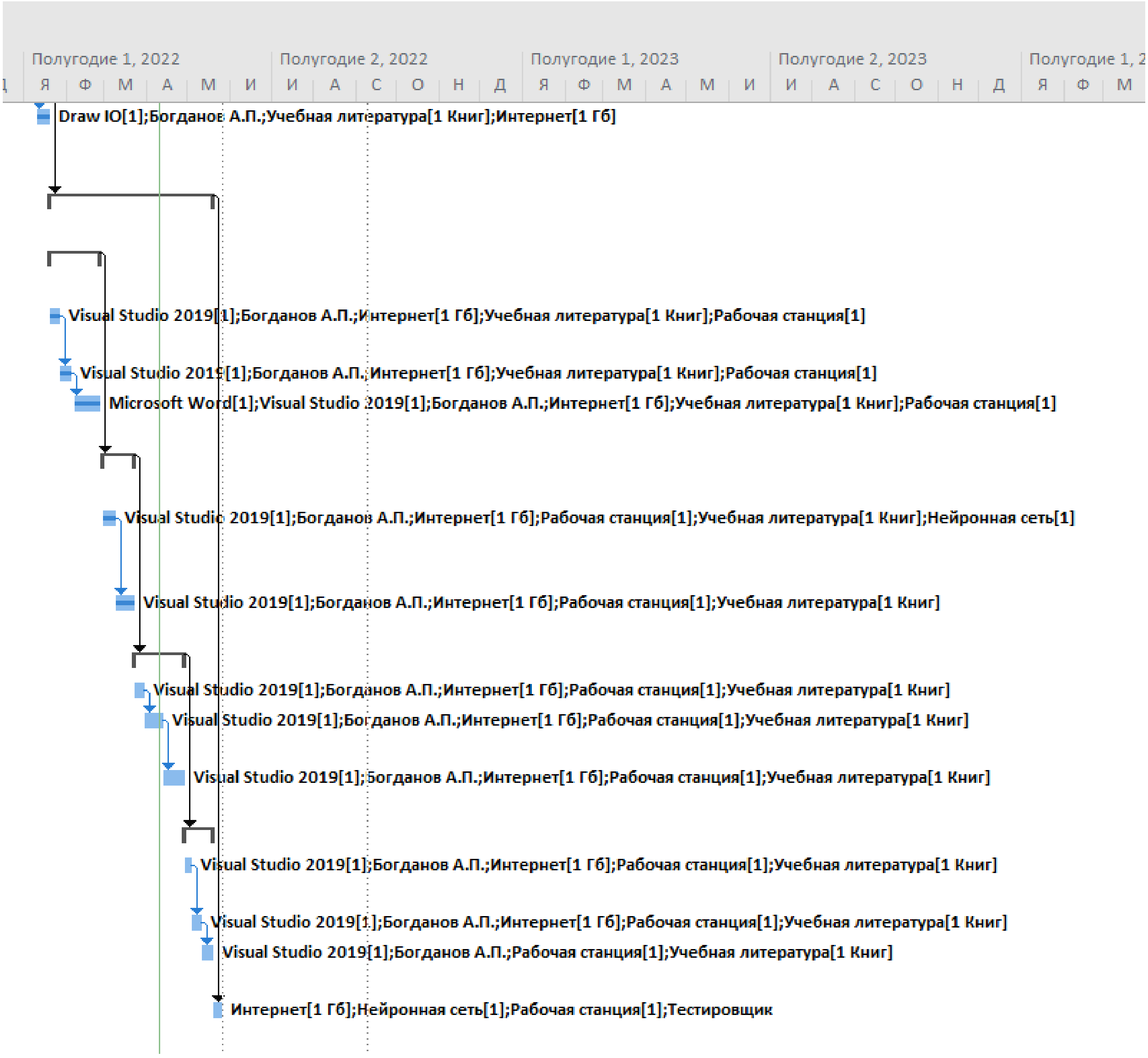


Рис. 83. Критический путь

Рис. 83. Продолжение



2.4 Расчёт стоимости

Используя Microsoft Project, была посчитана общая стоимость разработки, которая оказалась равна 450 300 руб.

Заключение

В данной курсовой работе была спроектировано и реализовано программное обеспечение системы детекции частей тела для контроля опасных действий работников. Проведено логическое проектирование классов пакета «Формирование опасной ситуации». Результатом является набор моделей и диаграмм для создания целевого ПО.

Созданное решение позволит усилить контроль над безопасностью работников на различных предприятиях. Благодаря конструктору запросов количество возможных решенных задач в области безопасности людей ограничивается лишь фантазией. К примеру, данное программное обеспечение способно решать, как и обычные задачи (нельзя чтобы руки находились в данной области, необходимо, чтобы у агрегата работало одновременно не более 3х людей), так и сложные (контроль действий работников по передвижению по лестницам – необходимо, чтобы каждый, чья нога находилась в области ступеней, держался одной рукой за любые перила).

Список литературы

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

2. 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.

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

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

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.

Проектирование программного обеспечения при объектном подходе

Техническое задание на курсовой проект

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 ПАО «Северсталь». Дата утверждения: 10.02.2021.

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

К функциональным характеристикам разрабатываемого ПО предъявлены следующие требования:

- отслеживать положение работников;
- отслеживать отдельные части тела работников;
- определять позу работников;
- добавлять нескольких зон опасности;
- у зоны должна быть возможность обнаружения конкретных частей тела;
- необходимо создать конструктор для формирования опасной ситуации.

3.2 Требования к надёжности

К разрабатываемому программному обеспечению предъявляются следующие требования к надежности:

- система должна иметь защиту от некорректных действий оператора;
- система не должна во время работы модифицировать свой код;
- корректный вывод данных на экран;
- проверка вводимых в настройках пользователем данных;
- система должна обеспечивать контроль целостности структур баз данных, нарушение которой возможно после аппаратных сбоев.

3.3 Условия эксплуатации

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

- температура окружающего воздуха от +10°C до +35°C;
- 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

Минимальная конфигурация:

- процессор с тактовой частотой 1,6 ГГц и более;
- 2 Гб ОЗУ;
- наличие устройств ввода (клавиатура, мышь);
- наличие устройств вывода (монитор);
- доступ в Интернет на скорости 1 Мбит/с и более;
- современный Интернет-браузер.

3.5 Требования к информационной и программной совместимости

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

Для стабильного функционирования программного обеспечения необходимо наличие операционной системы Windows, Linux или MacOS и современного Интернет-браузер.

4. Требования к программной документации

Программная документация должна содержать расчётно-пояснительную записку (РПЗ) с содержанием: текст программы (прил. 2).

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	10.09.2021	Готовое техническое задание	
Изучение предметной области	15.09.2021	Предметная область изучена	
Проведение сравнительного анализа аналогов проектируемого ПО	18.09.2021	Выявлены преимущества и недостатки аналогов	
Выбор технологии, среды и языка программирования	23.09.2021	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	26.10.2021	Разработаны спецификации проектируемого ПО	
Проектирование ПО	31.10.2021	Спроектировано ПО	
Организация работ	19.10.2021	Выполнена организация работ	
Разработка первой версии ПО	15.11.2021	Разработана первая версия ПО	
Выбор методики тестирования и тестирование первой версии ПО	17.11.2021	Протестированная первая версия ПО	
Разработка итоговой версии ПО	2.12.2021	Разработанное ПО	
Выбор методики тестирования и тестирование ПО	27.12.2021	Протестированное ПО	
Оформление документации	12.01.2022	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 курсовой работы	Сроки контроля	Результат выполнения	Отметка о приемке результата контрольного этапа
Проверка технического задания	10.09.2021	Техническое задание утверждено	
Демонстрация спроектированного ПО	11.10.2021	Спроектированное ПО согласованно	
Демонстрация финальной версии ПО	8.12.2021	Финальная версия ПО утверждена	
Демонстрация стратегии тестирования и проведённых тестов	8.12.2021	Стратегии тестирования утверждены, ПО работает исправно	
Подготовка документации	12.12.2021	Расчётно-пояснительная записка прошла норм контроль и утверждена	
Защита курсовой работы	12.01.2022	Курсовая работа защищена	

Приложение 2

Текст программы

Текст класса MainViewModel представлен на рис. П2.1.

```
using Emgu.CV;
using Emgu.CV.CvEnum;
using Emgu.CV.Structure;
using MoreLinq;
using MultiPersonEstimators.AwaitCounter;
using MultiPersonEstimators.ExtensionsMethods;
using MultiPersonEstimators.MatExtensions;
using MultiPersonEstimators.Model;
using MultiPersonEstimators.Model.Humans;
using MultiPersonEstimators.Model.Preparing;
using MultiPersonEstimators.Subscribe;
using MultiPersonEstimators.ViewModel.Reports;
using ReactiveUI;
using System;
using System.Collections.Generic;
using System.Collections.ObjectModel;
using System.Collections.Specialized;
using System.Drawing;
using System.IO;
using System.Linq;
using System.Net.Mail;
using System.Reactive.Linq;
using System.Windows.Input;

namespace MultiPersonEstimators.ViewModel
{
   public class MainViewModel : ReactiveObject
   {
      private const int ZONE_ROING_DELAY = 100;
      private const int HUMAN_DISTRIBUTION_DELAY = 100;

      public IModel Model { get; }
      public event Action Started;

      public string FileName
      {
         get => fileName;
         set => this.RaiseAndSetIfChanged(ref fileName, value);
      }
      private string fileName;

      public bool CanRead
      {
         get => canRead;
         set => this.RaiseAndSetIfChanged(ref canRead, value);
      }
      private bool canRead;
      public ICommand Start { get; }
      public ICommand OpenCamera { get; }

      public byte[] FrameAsBytes
      {
         get => frameAsBytes;
         set => this.RaiseAndSetIfChanged(ref frameAsBytes, value);
      }
      private byte[] frameAsBytes;

      #region Zones
      public ICommand Pause { get; }

      public ICommand AddZone { get; }
      public ICommand RemoveZone { get; }

      public Zone CurrentZone
      {
         get => currentZone;
         set => this.RaiseAndSetIfChanged(ref currentZone, value);
      }
      private Zone currentZone;
      public ObservableCollection<ZoneGroup> ZoneGroups { get; } = new ObservableCollection<ZoneGroup>();
      public ObservableCollection<Zone> Zones { get; } = new ObservableCollection<Zone>();

      public bool ShowCurrentZonePanel
      {
         get => showCurrentZonePanel;
         set => this.RaiseAndSetIfChanged(ref showCurrentZonePanel, value);
      }
      private bool showCurrentZonePanel;

      public IEnumerable<System.Windows.Media.Brush> Colors { get; }
      #endregion

      #region ZonesQueries
      public EmailSender EmailSender { get; } = EmailSender.DefaultSender;
      public ICommand AddZonesQuery { get; }
      public ICommand RemoveZonesQuery { get; }

      public ZonesQuery CurrentZonesQuery
      {
         get => currentZonesQuery;
         set => this.RaiseAndSetIfChanged(ref currentZonesQuery, value);
      }
      public ObservableCollection<ZonesQuery> ZonesQueries { get; } = new ObservableCollection<ZonesQuery>();
      private ZonesQuery currentZonesQuery;
      #endregion

      public IEnumerable<Human> Humans
      {
         get => humans;
         set => this.RaiseAndSetIfChanged(ref humans, value);
      }
      private IEnumerable<Human> humans = Enumerable.Empty<Human>();

      public MainViewModel()
      {
         Model = new MainModel();

         Colors = typeof(System.Windows.Media.Brushes)
            .GetProperties()
            .Where(x => x.Name.Count(char.IsUpper) == 1)
            .Select(x => x.GetValue(null) as System.Windows.Media.Brush);

         Start = ReactiveCommand.Create(StartMethod);
         OpenCamera = ReactiveCommand.Create(OpenCameraMethod);

         AddZone = ReactiveCommand.Create(() => Zones.Add(CreateNewZone()));

         Pause = ReactiveCommand.Create(() => { });

         RemoveZone = ReactiveCommand.Create(() => Zones.Remove(CurrentZone),
            this.ObservableForProperty(x => x.CurrentZone, z => z != null));
```

Рис. П2.1. Текст класса MainViewModel

SchemaCoolingSystem.js

```
        AddZonesQuery = ReactiveCommand.Create(AddZonesQueryMethod);
        RemoveZonesQuery = ReactiveCommand.Create(RemoveZonesQueryMethod,
            this.ObservableForProperty(x => x.CurrentZonesQuery, z => z != null));

        Model.ObservableForProperty(mod => mod.DrawedFrame)
            .Subscribe(arg => FrameAsBytes = arg.Value.ToBytes());

        this.ObservableForProperty(x => x.CurrentZone, x => x != null)
            .Subscribe(arg => ShowCurrentZonePanel = arg);

        this.ObservableForProperty(x => x.Humans)
            .Where(x => x.Value != null)
            .Subscribe(_ => Zones.ForEach(AddHumansToZone));


Zones.ObservableFor(NotifyCollectionChangedAction.Remove)
             .Subscribe(zone =>
             {
                 ZoneGroups.Remove(zone.RootZoneGroup);

                 zone.Dispose();
                 CurrentZone = null;
             });

        Zones.ObservableFor(NotifyCollectionChangedAction.Add)
            .Subscribe(zone =>
            {
                foreach (var point in zone.Points)
                {
                    point.WhenAnyValue(p => p.X, p => p.Y)
                        .Where(_ => Model.Frame != null)

.Throttle(TimeSpan.FromMilliseconds(ZONE_ROING_DELAY))
                        .Subscribe(_ => RoiZone(zone));


                    point.WhenAnyValue(p => p.X, p => p.Y)
                        .Where(_ => Humans != null)

.Throttle(TimeSpan.FromMilliseconds(HUMAN_DISTRIBUTION_DELAY))
                        .Subscribe(arg => AddHumansToZone(zone));
                }

                foreach (var selectedHumanPart in zone.SelectedBodyParts)
                {
                    selectedHumanPart.WhenAnyValue(x => x.Value, x => x.IsSelected, x => x.Location)
                                .Where(_ => Humans != null)
                                .Subscribe(_ => AddHumansToZone(zone));
                }

                CurrentZone = zone;
            });

        Model.ObservableForProperty(x => x.DrawedFrame)
            .Where(arg => arg.Value != null)

.Throttle(TimeSpan.FromMilliseconds(ZONE_ROING_DELAY))
            .Subscribe(_ => Zones.ForEach(RoiZone));

        Model.ObservableForProperty(x => x.HumansPoints)
            .Where(arg => arg.Value != null)
            .Subscribe(arg => Humans = PrepareHumans(arg.Value));

        Model.ObservableForProperty(x => x.MillisecondsForFrame)
             .Subscribe(arg => Zones.ForEach(z => z.CurrentTicks += arg.Value));
 }
private void AddHumansToZone(Zone zone)
    {
        var humansBodyParts = new List<HumanBodyPartsInZone>();

        foreach (var human in humans)
        {
            var humanPartsInZone = new List<BodyPart>();

            foreach (var bodyPartPoint in human.Points.Where(p => p != null))
            {
                if (zone.CheckBodyPartPoint(bodyPartPoint))
                {
                    humanPartsInZone.Add(bodyPartPoint.BodyPart);
                }
            }

            if (humanPartsInZone.Count > 0)
            {
                humansBodyParts.Add(new HumanBodyPartsInZone { Human = human, BodyParts = humanPartsInZone });
            }

        }

        zone.HumanBodyParts = humansBodyParts;
    }

    private void RoiZone(Zone zone)
    {
        var rect = GetRect(zone.Points);

        if (rect.Width > 0 && rect.Height > 0)
        {
            zone.Roi = new Mat(Model.Frame, rect).ToBytes();
        }
    }

    private int zonesId = 0;
    private Zone CreateNewZone()
    {
        var zone = new Zone(0.1, 0.1, 0.2, 0.2)
        {
            Name = $"Зона {zonesId++}"
        };

        zone.RootZoneGroup = new ZoneGroup { Group = $"{zone.Name} группа" };
        zone.ZoneGroup = zone.RootZoneGroup;
        zone.ZoneGroups = ZoneGroups;

        ZoneGroups.Add(zone.RootZoneGroup);

        return zone;
    }

    private IEnumerable<Human> PrepareHumans(Point[,] points)
    {
        var findedHumans = new List<Human>();

        for (int i = 0; i < points.GetLength(0); i++)
        {
            var humanPoints = points.GetArray(i);
            var human = new Human
            {
                Name = $"Человек {i}",
                Points = humanPoints.Select
                (
                    (p, i) => p.IsEmpty() ? null : new BodyPartPoint
                    {
                        Point = new Model.Points.Point((double)p.X / Model.Frame.Width, (double)p.Y / Model.Frame.Height),
                        BodyPart = (BodyPart)i
                    }
                ).ToList(),
```

П2.1. Продолжение

```
    Poses = Model.PoseDetector.GetPoses(humanPoints)
            };

            var humanRect = GetRect(human.Points.Where(x => x != null).Select(x => x.Point));

            if (humanRect.Width > 0 && humanRect.Height > 0)
            {
                human.Roi = new Mat(Model.Frame, humanRect).ToBytes();
            }

            findedHumans.Add(human);
        }

        return findedHumans;
    }

    private void AddZonesQueryMethod()
    {
        var zoneQuery = new ZonesQuery { AllZones = Zones, AllHumans = Humans };

        this.ObservableForProperty(x => x.Humans)
            .Subscribe(arg => zoneQuery.AllHumans = arg.Value)
            .AddSubscribeBy(zoneQuery);

        zoneQuery.ObservableForProperty(x => x.WrongHumans)
            .Where(h => false)
            .Subscribe
            (
                async arg =>
                {
                    var frame = new Mat();
                    string text = $"Люди, не удовлетворяющие зонам {(zoneQuery.Inverted ? "инвертируемого (зоны не обнаружили - нарушение)" : "обычного (зоны обнаружили - нарушение)")} запроса:\n";

                    lock (Model.DrawedFrame)
                    {
                        Model.DrawedFrame.CopyTo(frame);
                        text += string.Concat(arg.Value.Select(x => $"{x.Name} "));

                        text += "\n\nЛюди, проверяемые запросом:\n";
                        text += string.Concat(zoneQuery.AllHumans.Select(x => $"{x.Name} "));

                        text += "\n\nЗоны запроса:\n";
                        foreach (var zoneGroup in zoneQuery.QueryZones.GroupBy(x => x.ZoneGroup))
                        {
                            text += $"{zoneGroup.Key.Group}: {string.Concat(zoneGroup.Select(x => $"{x} "))}\n";
                        }

                        foreach (var zone in zoneQuery.QueryZones)
                        {
                            (double x, double y) cent = (zone.Points.Min(x => x.X), zone.Points.Average(x => x.Y));

                            CvInvoke.PutText(frame, zone.Name, new Point((int)(cent.x * Model.DrawedFrame.Width), (int)(cent.y * Model.DrawedFrame.Height)),
                                FontFace.HersheyComplex, 1.5, new MCvScalar(0, 0, 255), 2);
CvInvoke.Polylines(frame, zone.Points
                                .Select(p => new Point((int)(p.X * Model.DrawedFrame.Width), (int)(p.Y * Model.DrawedFrame.Height)))
                                .ToArray(),
                            true,
                            new MCvScalar(50, 0, 225),
                            2);
                        }
                    }

                    using var byteStream = new MemoryStream(frame.ToBytes());
                    await EmailSender.SendMessageAsync("НАРУШЕНИЕ!", text, new Attachment(byteStream, "warning.png"));
                }
            );

        ZonesQueries.Add(zoneQuery);
    }

    private void RemoveZonesQueryMethod()
    {
        CurrentZonesQuery.Unsubscribe();
        ZonesQueries.Remove(CurrentZonesQuery);
    }

    private void StartMethod()
    {
        CanRead = true;
        Started?.Invoke();

        if (CanRead)
        {
            Model.VideoCapture = new VideoCapture(FileName);
        }
    }

    private void OpenCameraMethod()
    {
        CanRead = true;

        if (CanRead)
        {
            Model.VideoCapture = new VideoCapture(0);
        }
    }

    private Rectangle GetRect(IEnumerable<Model.Points.Point> points)
    {
        var (startX, startY, endX, endY) = (points.Min(p => p.X), points.Min(p => p.Y), points.Max(p => p.X), points.Max(p => p.Y));
        int x = (int)(Model.Frame.Width * startX);
        int y = (int)(Model.Frame.Height * startY);

        int width = (int)(Model.Frame.Width * endX) - x;
        int height = (int)(Model.Frame.Height * endY) - y;

        return new Rectangle(x, y, width, height);
    }
}

public class Class1
{
}
```

Текст класса Zone представлен на рисунке П2.2

```
using MultiPersonEstimators.ExtensionsMethods;
using MultiPersonEstimators.Model.Humans;
using MultiPersonEstimators.Model.Points;
using MultiPersonEstimators.Model.Preparing;
using ReactiveUI;
using System;
using System.Collections.Generic;
using System.Collections.ObjectModel;
using System.Diagnostics;
using System.Linq;
using System.Text.Json;
using System.Windows.Media;

namespace MultiPersonEstimators.ViewModel
{
   public class Zone : ReactiveObject, IDisposable
   {
      public event EventHandler Disposed;

      public string Name
      {
         get => name;
         set => this.RaiseAndSetIfChanged(ref name, value);
      }
      private string name;
      public byte[] Roi
      {
         get => roi;
         set => this.RaiseAndSetIfChanged(ref roi, value);
      }
      private byte[] roi;
      public ZoneGroup RootZoneGroup { get; set; }
      public ZoneGroup ZoneGroup
      {
         get => zoneGroup;
         set => this.RaiseAndSetIfChanged(ref zoneGroup, value);
      }
      private ZoneGroup zoneGroup;
      public IEnumerable<ZoneGroup> ZoneGroups { get; set; }

      public IEnumerable<Human> Humans =>
HumanBodyParts.Select(x => x.Human).Distinct();
      public IEnumerable<Point> Points { get; }

      public IEnumerable<HumanBodyPartsInZone>
HumanBodyParts
      {
         get => humanBodyParts;
         set
         {
            casheHumanBodyParts = value;
            int humansCount = value.Select(x =>
x.Human).Distinct().Count();
            if (humansCount == 0)
            {
               CurrentTicks = 0;
            }

            this.RaiseAndSetIfChanged(ref humanBodyParts,
               CurrentTicks >= MaxTicks && humansCount >
MaxHumansCount ?
                  value :
Enumerable.Empty<HumanBodyPartsInZone>());
         }
      }
      private IEnumerable<HumanBodyPartsInZone>
humanBodyParts;
      private IEnumerable<HumanBodyPartsInZone>
casheHumanBodyParts =
Enumerable.Empty<HumanBodyPartsInZone>();

      public IEnumerable<Selected<BodyPart>> SelectedBodyParts {
get; }
      public IEnumerable<Selected<string>> SelectedTexts { get; }


      private IReadOnlyDictionary<string, IEnumerable<BodyPart>>
textToHumanParts = new Dictionary<string,
IEnumerable<BodyPart>>
      {
         { "Голова", new [] { BodyPart.Head } },
         {  "Руки",
            new []
            {
               BodyPart.RightShoulder, BodyPart.RightElbow,
BodyPart.RightWrist,
               BodyPart.LeftShoulder, BodyPart.LeftElbow,
BodyPart.LeftWrist
            }
         },
         {  "Ноги",
            new []
            {
               BodyPart.RightHip, BodyPart.RightKnee,
BodyPart.RightAnkle,
               BodyPart.LeftHip, BodyPart.LeftKnee,
BodyPart.LeftAnkle
            }
         },  {
            "Тело",
            new []
            {
               BodyPart.Neck, BodyPart.LeftShoulder,
BodyPart.RightShoulder,
               BodyPart.Hip, BodyPart.LeftHip, BodyPart.RightHip
            }
         }
      };

      public Brush Color
      {
         get => color;
         set => this.RaiseAndSetIfChanged(ref color, value);
      }
      private Brush color = Brushes.Red;

      public double Opacity
      {
         get => opacity;
         set => this.RaiseAndSetIfChanged(ref opacity, value);
      }
      private double opacity = 0.5;

      public long MaxTicks
      {
         get => maxTicks;
         set => this.RaiseAndSetIfChanged(ref maxTicks, value);
      }
      private long maxTicks = 0;

      public long CurrentTicks
      {
         get => currentTicks;
         set => this.RaiseAndSetIfChanged(ref currentTicks, value);
      }
      private long currentTicks = 0;

      public bool CanRenameGroup
      {
         get => canRenameGroup;
         set => this.RaiseAndSetIfChanged(ref canRenameGroup,
value);
      }     private bool canRenameGroup;
      public int MaxHumansCount
      {
```

Рис. П2.2. Текст класса Zone

П2.2. Продолжение

```
 get => maxHumansCount;
        set => this.RaiseAndSetIfChanged(ref maxHumansCount, value);
    }
    private int maxHumansCount = 0;

    public Zone(double left, double top, double right, double bot)
    {
        Points = new ObservableCollection<Point>
        {
            new Point(left, top),
            new Point(right, top),
            new Point(right, bot),
            new Point(left, bot)
        };

        this.WhenAnyValue(x => x.RootZoneGroup, x => x.ZoneGroup)
            .Subscribe(_ => CanRenameGroup = RootZoneGroup == ZoneGroup);

        this.ObservableForProperty(x => x.MaxTicks)
            .Subscribe(_ => CurrentTicks = 0);

        this.WhenAnyValue(x => x.MaxTicks, x => x.MaxHumansCount)
            .Subscribe(arg => HumanBodyParts = casheHumanBodyParts);

        foreach (var point in Points)
        {
            point.WhenAnyValue(p => p.X, p => p.Y)
                .Subscribe(_ => CurrentTicks = 0);
        }

        SelectedBodyParts = Enum.GetValues(typeof(BodyPart))
                        .Cast<BodyPart>()
                        .Select(x => new Selected<BodyPart> { Value = x })
                        .ToList();

        SelectedTexts = textToHumanParts.Keys
                            .Select(x => new Selected<string> { Value = x })
                            .ToList();

        foreach (var selectedText in SelectedTexts)
        {
            selectedText.ObservableForProperty(x => x.IsSelected)
                .Subscribe(arg =>
                {
                    foreach (var selectedHumanPart in SelectedBodyParts
                                        .Where(x => textToHumanParts[selectedText.Value]
                                        .Contains(x.Value)))
                    {
                        selectedHumanPart.IsSelected = arg.Value;
                    }
                });

            selectedText.ObservableForProperty(x => x.Location)
                .Subscribe(arg =>
                {
                    foreach (var selectedHumanPart in SelectedBodyParts
                                        .Where(x => textToHumanParts[selectedText.Value]
                                        .Contains(x.Value)))
                    {
                        selectedHumanPart.Location = arg.Value;
                    }
                });
        }

        this.ObservableForProperty(x => x.Opacity)
            .Subscribe(_ => Opacity = Math.Round(Opacity, 2));
    }

    public bool ContainsBodyPart(BodyPart bodyPart)
    {
        return SelectedBodyParts
            .Where(p => p.IsSelected)
            .Select(p => p.Value)
            .Contains(bodyPart);
    }

    public bool ContainsPoint(Point point)
    {
        var poly = Points.ToArray();

        bool inside = false;
        for (int i = 0, j = poly.Length - 1; i < poly.Length; j = i++)
        {
            if (poly[i].Y > point.Y != poly[j].Y > point.Y
                && point.X < (poly[j].X - poly[i].X) * (point.Y - poly[i].Y) / (poly[j].Y - poly[i].Y) + poly[i].X)
            {
                inside = !inside;
            }
        }

        return inside;
    }

    public bool CheckBodyPartPoint(BodyPartPoint bodyPartPoint)
    {
        bool containsPoint = ContainsPoint(bodyPartPoint.Point);

        foreach (var part in SelectedBodyParts.Where(p => p.IsSelected))
        {
            if (part.Value == bodyPartPoint.BodyPart)
            {
                if (part.Location == Location.Inside && containsPoint)
                {
                    return true;
                }

                if (part.Location == Location.OutSide && !containsPoint)
                {
                    return true;
                }
            }

        }
        return false;
    }


    public void Dispose()
    {
        Disposed?.Invoke(this, EventArgs.Empty);
        Disposed = null;
    }

    public override string ToString()
    {
        return $"{Name} [{string.Concat(SelectedTexts.Where(x => x.IsSelected).Select(x => $"({x.Value} {x.Location.ToDescriptionOrString()}) "))}]";
    }
  }
}
```

Текст класса Query представлен на рисунке П2.3

```
using MultiPersonEstimators.Model;
using MultiPersonEstimators.Model.Humans;
using MultiPersonEstimators.Subscribe;
using ReactiveUI;
using System;
using System.Collections.Generic;
using System.Collections.ObjectModel;
using System.Linq;
using System.Windows.Input;

namespace MultiPersonEstimators.ViewModel
{
   public class ZonesQuery : ReactiveObject
   {
      public ObservableCollection<Zone> QueryZones { get; } = new
ObservableCollection<Zone>();

      public bool Inverted
      {
         get => inverted;
         set => this.RaiseAndSetIfChanged(ref inverted, value);
      }
      private bool inverted = false;

      public Zone AddableZone
      {
         get => addableZone;
         set => this.RaiseAndSetIfChanged(ref addableZone, value);
      }
      private Zone addableZone;

      public Zone RemovableZone
      {
         get => removableZone;
         set => this.RaiseAndSetIfChanged(ref removableZone, value);
      }
      private Zone removableZone;

      public IEnumerable<Zone> AllZones { get; set; }

      public IEnumerable<QueryHumans> QuerysHumans { get; }

      public IEnumerable<Human> WrongHumans
      {
         get => wrongHumans;
         set => this.RaiseAndSetIfChanged(ref wrongHumans, value);
      }
      private IEnumerable<Human> wrongHumans;

      public IEnumerable<Human> AllHumans
      {
         get => allHumans;
         set => this.RaiseAndSetIfChanged(ref allHumans, value);
      }
      private IEnumerable<Human> allHumans;

      public QueryHumans CurrentQueryHumans
      {
         get => currentQueryHumans;
         set => this.RaiseAndSetIfChanged(ref currentQueryHumans,
value);
      }
      private QueryHumans currentQueryHumans =
QueryHumans.All;

      public bool CurrentQueryHumansIsFromZone
      {
         get => currentQueryHumansIsFromZone;
         set => this.RaiseAndSetIfChanged(ref
currentQueryHumansIsFromZone, value);
      }
      private bool currentQueryHumansIsFromZone = false;

      public Zone CurrentQueryHumansZone
      {
         get => currentQueryHumansZone;
         set => this.RaiseAndSetIfChanged(ref
currentQueryHumansZone, value);
      }
      private Zone currentQueryHumansZone;

      public ICommand AddZone { get; }
      public ICommand RemoveZone { get; }

      public ZonesQuery()
      {
         QuerysHumans =
Enum.GetValues(typeof(QueryHumans)).Cast<QueryHumans>().To
List();

         AddZone = ReactiveCommand.Create(AddZoneMethod,
            this.ObservableForProperty(x => x.AddableZone, z => z !=
null && !QueryZones.Contains(AddableZone)));

         RemoveZone =
ReactiveCommand.Create(RemoveZoneMethod,
            this.ObservableForProperty(x => x.RemovableZone, z => z
!= null));

         this.ObservableForProperty(x => x.CurrentQueryHumans, x
=> x == QueryHumans.FromZone)
            .Subscribe(arg =>
            {
               CurrentQueryHumansIsFromZone = arg;

               if (!arg)
               {
                  CurrentQueryHumansZone = null;
               }
            });

         this.WhenAnyValue(x => x.AllHumans, x =>
x.CurrentQueryHumans, x => x.CurrentQueryHumansZone, x =>
x.Inverted)
            .Subscribe(_ => Calculate());
      }

      public void Calculate()
      {
         var wrongHumans = new List<Human>();

         var queryHumans = CurrentQueryHumans switch
         {
            QueryHumans.All => AllHumans,
            QueryHumans.FromZone =>
CurrentQueryHumansZone?.Humans ??
Enumerable.Empty<Human>(),
            _ => Enumerable.Empty<Human>(),
         };

         foreach (var queryZone in QueryZones.GroupBy(x =>
x.ZoneGroup))
         {
            var zoneHumans = queryZone.SelectMany(x =>
x.Humans).ToList();

            foreach (var human in queryHumans)
            {
               bool zoneHumansContains =
zoneHumans?.Contains(human) ?? false;

               if (zoneHumansContains ^ Inverted)
               {
                  wrongHumans.Add(human);
               }
            }
         }
```

Рис. П2.3. Текст класса Query

```
            WrongHumans = wrongHumans.Distinct().OrderBy(x => x.Name).ToList();
        }

        private void AddZoneMethod()
        {
            AddableZone.WhenAnyValue(x => x.HumanBodyParts, x => x.ZoneGroup)
                .Subscribe(_ => Calculate())
                .AddSubscribeBy((this, AddableZone));

            QueryZones.Add(AddableZone);
            var disposingZone = AddableZone;

            AddableZone.Disposed += ZoneDisposed;

            void ZoneDisposed(object sender, EventArgs e)
            {
                if (QueryZones.Contains(disposingZone))
                {
                    RemovableZone = disposingZone;
                    RemoveZoneMethod();
                }
            }

            AddableZone = null;

            Calculate();
        }

        private void RemoveZoneMethod()
        {
            (this, RemovableZone).Unsubscribe();
            QueryZones.Remove(RemovableZone);

            Calculate();
        }
    }
}
```