

|  |  |
| --- | --- |
| Вариант использования | Автоматическая детекиция посторонних объектов |
| Краткое описание | вариант использования, при котором сотрудник производит наблюдение в реальном времени за работоспособностью системы |
| Цель | Распознавание посторонних объектов |
| Тип | Основной |
| Ссылки на другие варианты | * Информирование машиниста световым сигналом * Сохранение результатов |
| Исключения | * Нет подключения к БД * Нет подключения к IP-камере * Нет данных для загрузки нейронной сети |

Данный вариант использования начинает выполняться, когда пользователь запускает приложение.

Типичный ход событий

Таблица 1

Типичный ход событий

|  |  |
| --- | --- |
| Действие исполнителя | Отклик системы |
| 1.  Пользователь запускает приложение. | 2. Система производит загрузку пользовательских настроек, нейронной сети, формирует пользовательский интерфейс. |
| 3. Пользователь выбирает видеопоток, если он не задан по умолчанию в настройках. | 4. Система воспроизводит выбранный видеопоток, производя детекцию посторонних предметов. При обнаружении недопустимых объектов система подает сигнал диспетчеру в виде сообщения, а машинисту – посредством подсистемы информирования.  Полученные результаты система сохраняет в базу данных, выбранную в настройках при конфигурировании системы. |

|  |  |
| --- | --- |
| Вариант использования | Информирование машиниста световым сигналом |
| Краткое описание | Это вариант использования, при котором система, найдя посторонний предмет в сыпучем материале, подаёт сигнал на уличный проблесковый маячок, тем самым предупреждаю машиниста бульдозера о наличии постороннего предмета, который необходимо извлечь. |
| Цель | Информирование машиниста |
| Тип | Основной |
| Ссылки на другие варианты | - |
| Исключения | - |

|  |  |
| --- | --- |
| Действие исполнителя | Отклик системы |
| 1. Запуск обработки видеопотока | 2. Система находит посторонний предмет и уведомляет диспетчера об этом соответствующим сообщением. |

|  |  |
| --- | --- |
| Вариант использования | Сохранение результатов |
| Краткое описание | это вариант использования, при котором система, найдя посторонний предмет в сыпучем материале, сохраняет полученные данные в базу данных, тем самым формируя архив. Для дальнейшего обучения системы, данный вариант использования еще включает в себя сохранение промежуточных кадром (каждые N). |
| Цель | Сохранять информацию для дальнейшего анализа и принятия решений |
| Тип | Основной |
| Ссылки на другие варианты | - |
| Исключения | * Нет подключение к БД |

|  |  |
| --- | --- |
| Действие исполнителя | Отклик системы |
| 1. Запуск обработки видеопотока | 1. При обнаружении постороннего предмета система сохраняет информацию о нем в БД. |

|  |  |
| --- | --- |
| Вариант использования | Просмотр архива полученных материалов работы системы |
| Краткое описание | вариант использования, при котором диспетчер может просмотреть архив данных за необходимы промежуток времени, делать выгрузку в виде отчетов, для дальнейшего анализа и принятия решений |
| Цель | Просмотр материала для анализа и принятия решения |
| Тип | Основной |
| Ссылки на другие варианты | Ручная корректировка ложных срабатываний |
| Исключения | * Нет подключения к БД |

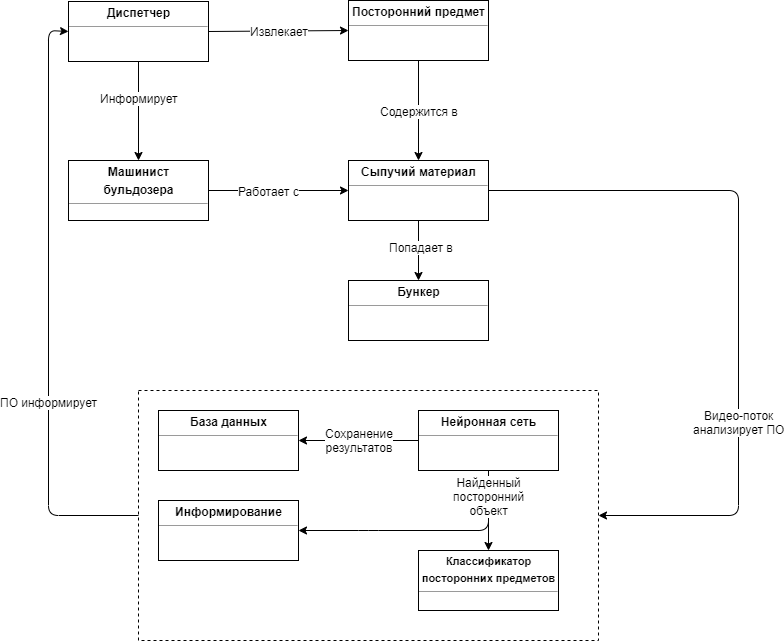
|  |  |
| --- | --- |
| Действие исполнителя | Отклик системы |
| 1. Запуск архива | 1. Система отображает всю сохраненную в БД ранее информацию о найденных посторонних предметах |

|  |  |
| --- | --- |
| Вариант использования | Ручная корректировка ложных срабатываний |
| Краткое описание | вариант использования, при котором диспетчер может на промежуточных кадрах исправить решение, принятое системой в ходе ее работы, с дальнейшим сохранением. Это необходимо для до обучения системы. |
| Цель | Подготовить материал для обучения системы |
| Тип | Основной |
| Ссылки на другие варианты | Сохранение результатов |
| Исключения | * Нет подключения к БД |

|  |  |
| --- | --- |
| Действие исполнителя | Отклик системы |
| 1. Запуск архива | 1. Система отображает всю сохраненную в БД ранее информацию о найденных посторонних предметах |
| 1. Разметка постороннего объекта, который не был распознан | 1. Сохранение полученной информации |

|  |  |
| --- | --- |
| Вариант использования | Настройка системы детекции посторонних предметов |
| Краткое описание | вариант использования, при котором диспетчер настраивает систему: подключение к базе данных, выбор видеопотока с ip-камеры и т.п. |
| Цель | Подготовка программы к работе |
| Тип | Основной |
| Ссылки на другие варианты | - |
| Исключения | * Неверные данные для подключения к БД * Неверные данные для подключения к ip-камерам |

|  |  |
| --- | --- |
| Действие исполнителя | Отклик системы |
| 1.  Пользователь выбирает пункт меню «Настройки» | 2. Система загружает пользовательские настройки и формирует интерфейс. |
| 3. Пользователь вносит изменения в конфигурацию системы. | 4. Система «изменяется» согласно выбранным изменениям и сохраняет на локальном диске набор пользовательских настроек. |



В предметной области можно выделить такие объекты, которые являются отдельными сущностями, как:

* диспетчер – человек ….;
* машинист бульдозера;
* сыпучий материал;
* посторонний предмет;
* бункер.

В связи с тем, что предметная область декомпозируется на объекты, целесообразным подходом к проектированию ПО будет объектно-ориентированный подход (ООП).

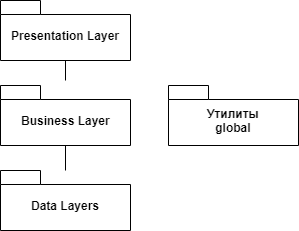


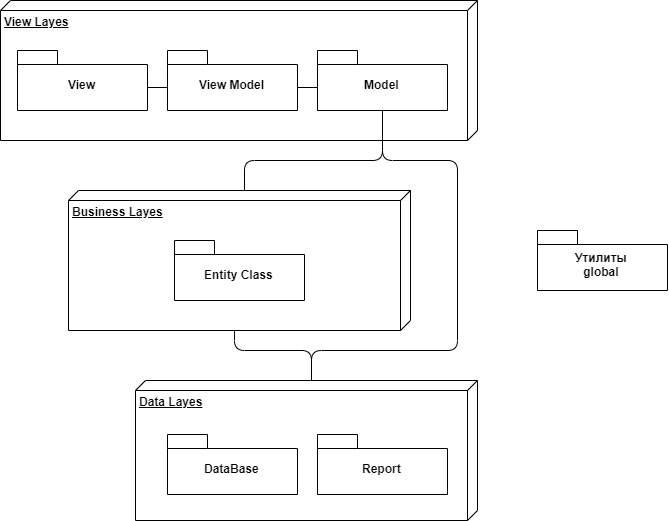
Диаграмма пакетов

Независимо от типа проектируемого приложения и того, имеется ли у него пользовательский интерфейс или оно является сервисным приложением, которое просто предоставляет сервисы (не путайте со слоем сервисов приложения), его структуру можно разложить на логические группы программных компонентов. Эти логические группы называются слоями. Слои помогают разделить разные типы задач, осуществляемые этими компонентами, что упрощает создание дизайна, поддерживающего возможность повторного использования компонентов.

Каждый логический слой включает ряд отдельных типов компонентов, сгруппированных в подслои, каждый из подслоев выполняет определенный тип задач. Разделение приложения на слои, выполняющие разные роли и функции, помогает максимально повысить удобство и простоту обслуживания кода, оптимизировать работу приложения при различных схемах развертывания и обеспечивает четкое разграничение областей применения определенной технологии или принятия определенных проектных решений.

Типовой трехслойный дизайн, представленный на рис. , включает следующие слои:

* Слой представления. Данный слой содержит ориентированную на пользователя функциональность, которая отвечает за реализацию взаимодействием пользователя с системой, и, как правило, включает компоненты, обеспечивающие общую связь с основной бизнес-логикой, инкапсулириванной в бизнес-слое
* Бизнес-слой. Этот слой реализует основную функциональность системы и инкапсулирует связанную с ней бизнес-логику. Обычно он состоит из компонентов, некоторые из которых предоставляют интерфейсы сервисов, доступные для использования другими участниками взаимодействия.
* Слой доступа к данным. Этот слой обеспечивает доступ к данным, хранящимся в рамках системы, и данным, предоставляемым другими сетевыми системами. Доступ может осуществляться через сервисы. Слой данных предоставляет универсальные интерфейсы, которые могут использоваться компонентами бизнесслоя.



Рассмотрим описание слоев, используя паттерн проектирования MVVM.

Model-View-ViewModel (MVVM) — шаблон проектирования архитектуры приложения. Представлен в 2005 году Джоном Госсманом (John Gossman) как модификация шаблона Presentation Model. Ориентирован на современные платформы разработки, такие как Windows Presentation Foundation, Silverlight от компании Microsoft.

Используется для разделения модели и её представления, что необходимо для их изменения отдельно друг от друга. Например, разработчик задаёт логику работы с данными, а дизайнер работает с пользовательским интерфейсом.

Каждый слой имеет ряд пакетов:

* View layer:
  + - View – рафический интерфейс (окна, списки, кнопки и т. п.). Выступает подписчиком на событие изменения значений свойств или команд, предоставляемых Моделью Представления. В случае, если в Модели Представления изменилось какое-либо свойство, то она оповещает всех подписчиков об этом, и Представление, в свою очередь, запрашивает обновлённое значение свойства из Модели Представления. В случае, если пользователь воздействует на какой-либо элемент интерфейса, Представление вызывает соответствующую команду, предоставленную Моделью Представления.
    - View Model – с одной стороны, абстракция Представления, а с другой — обёртка данных из Модели, подлежащие связыванию. То есть, она содержит Модель, преобразованную к Представлению, а также команды, которыми может пользоваться Представление, чтобы влиять на Модель.
    - Model – представляет собой логику работы с данными и описание фундаментальных данных, необходимых для работы приложения.
* Business layer:
  + - Entity Class – представляет собой набор классов-сущностей, которые будут описывать дополнительные объекты предметной области.
* Data layer:
  + - DataBase – представляет собой набор классов, описывающих взаимодействие системы с БД.
    - Report – представляет собой набор классов, реализующих возможность вывода информации из системы в видео отчетов (Excel, Word).