Учреждение образования

«БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

Проектирование программного обеспечения

Лабораторная работа №6

Тема: «Интерфейс» системы видеоконтроля за «объектом»

Студент: Лемешевский В.О.

ФИТ 3 курс 2 группа

Преподаватель: Курилец А.В.

Минск 2024

1. **Постановка задачи (описание функциональных требований):**

**Оператор**

Настройка объектов наблюдения: оператор должен иметь возможность добавлять и редактировать объекты наблюдения, указывая их название, местоположение и описание.

Управление камерами: оператор должен иметь возможность добавлять, редактировать и удалять камеры, а также настраивать их параметры.

Просмотр видеозаписей: оператор должен иметь доступ к видеозаписям по дате, времени и объекту наблюдения.

**Руководитель системы**

Анализ данных: руководитель должен иметь возможность генерировать отчёты о событиях и активности объектов, используя доступные данные.

Настройка уведомлений: руководитель должен иметь возможность настраивать уведомления о событиях (например, движение или тревога) для определённых объектов.

Мониторинг и аналитика: администратор должен иметь инструменты для мониторинга производительности системы и анализа данных о событиях и активности.

**Администратор системы**

Управление пользователями: администратор должен иметь возможность создавать и управлять учетными записями пользователей, назначать роли и права доступа.

Безопасность и аутентификация: обеспечивать безопасность системы, включая аутентификацию пользователей и управление доступом к данным и функциональности.

Управление базой данных: добавление, удаление обьектов.

В данной лабораторной работе необходимо составить диаграмму развёртывания и компонентов, которая должна содержать: список компонент, из которых будет состоять подсистема, существующие связи между компонентами подсистемы, список физических устройств, на которых будет работать подсистема и определить соединения.

**2. Описание программных средств:**

Draw.io – это бесплатное онлайн-приложение для создания диаграмм и схем. Оно позволяет пользователям создавать диаграммы благодаря широкому набору инструментов и функций.

Название: Draw.io

Версия: Веб-приложение

Разработчик: JGraph Ltd.

Адрес : https://www.draw.io/

Режим использования: Онлайн

Доступность на платформах: любой веб-браузер.

Draw.io позволяет создавать различные типы диаграмм, такие как блок-схемы, организационные диаграммы, UML-диаграммы, сетевые диаграммы и многое другое. Он также поддерживает импорт и экспорт файлов в различных форматах, включая PNG, JPEG, PDF и SVG.

1. **Описание практического задания**

В соответствии с условием была спроектирована следующая диаграмма – рисунок 1.

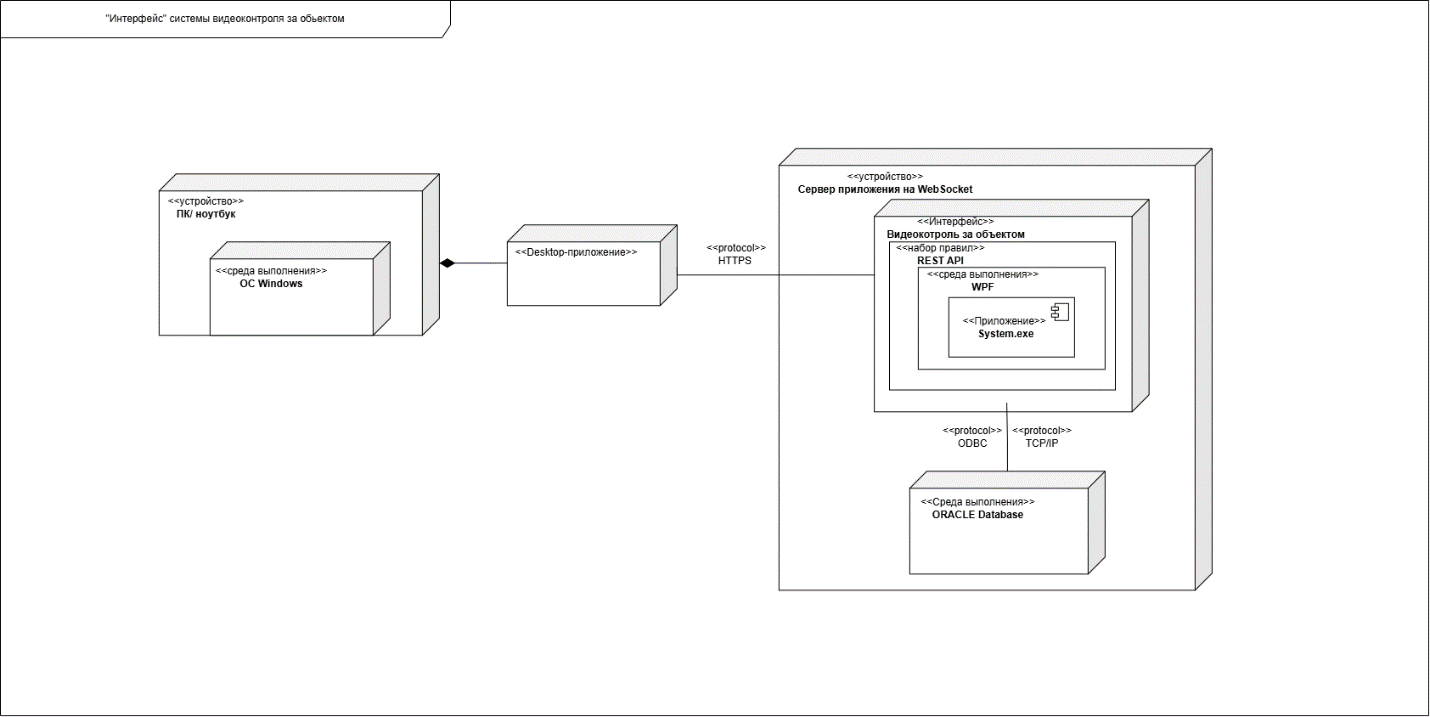


Рисунок 1 – Диаграмма развёртывания и компонентов

**Список компонентов подсистемы:**

1. Компонент «ПК/ноутбук»:
   * Технология реализации: Общепринятые технологии для ПК/ноутбуков.
2. Компонент «Операционная система (Windows)
   * Технология реализации: Microsoft Windows.
3. Компонент «Приложение System.exe»:
   * Технология реализации: C#.
4. Компонент «Среда выполнения WPF»:
   * Технология реализации: WPF.
5. Компонент «ORACLE Database»:
   * Технология реализации: ORACLE Database.
6. Компонент «Набор правил Rest API/Restful»
   * Технология реализации: Rest API/Restful.

**Список физических устройств:**

1. Устройство «ПК/ ноутбук»:

Основные характеристики: ПК или ноутбук, поддерживающий работу с приложениями.

1. Устройство «Сервер приложения»:

Основные характеристики: компьютер-сервер на WebSocket предоставляет постоянное двустороннее соединение между клиентом и сервером, что идеально подходит для приложений с быстром потоком данных в обе стороны.

**Соединения - протоколы связи между устройствами:**

1. Соединение между «ПК/ноутбуком»» и «Сервером приложения»: Протокол связи: HTTPS.
2. Соединение между «Сервером приложения» и «Средой выполнения Базы данных»: Протокол связи: TCP/IP.

**Среды выполнения:**

1. Среда выполнения «WPF»: это технология, разработанная для создания настольных приложений только для операционной системы Windows

Размещение компонентов: «Приложения System.exe» размещается в среде выполнения «WPF».

1. Среда выполнения «Операционная система (Windows) Данные ОС являются средой выполнения для приложений, созданных с помощью технологии WPF.

Размещение компонентов: «Операционная система» является обязательной частью «Устройства».

1. Среда выполнения «ORACLE Database»: Данная среда выполнения содержит в себе экземпляр базы данных ORACLE.

Размещение компонентов: «ORACLE Database» размещается в среде выполнения «Сервер приложения».

**Размещение компонентов на устройствах:**

1. Серверное устройство:
   * Все необходимые среды выполнения и компоненты расположены исключительно на серверном устройстве.
2. Клиентское устройство:
   * Клиентское устройство посылает все необходимые ответы, и принимает запросы.

**Ответы на теоретические вопросы**

1. **Укажите назначение физических диаграмм: компонентов и развертывания**

Диаграмма развертывания – это тип UML-диаграммы, которая показывает архитектуру исполнения системы, включая такие узлы, как аппаратные или программные среды исполнения, а также промежуточное программное обеспечение, соединяющее их. Диаграммы развертывания обычно используются для визуализации физического аппаратного и программного обеспечения системы. Используя его, вы можете понять, как система будет физически развернута на аппаратном обеспечении. Диаграммы развертывания помогают моделировать аппаратную топологию системы по сравнению с другими типами UML-диаграмм, которые в основном описывают логические компоненты системы.

Диаграммы компонентов используются для визуализации организации компонентов системы и зависимостей между ними. Они позволяют получить высокоуровневое представление о компонентах системы.Компонентами могут быть программные компоненты, такие как база данных или пользовательский интерфейс; или аппаратные компоненты, такие как схема, микросхема или устройство; или бизнес-подразделение, такое как поставщик, платежная ведомость или доставка.

Компонентные диаграммы

* Используются в компонентно-ориентированных разработках для описания систем с сервис-ориентированной архитектурой
* Показать структуру самого кода
* Может использоваться для фокусировки на отношениях между компонентами, скрывая при этом детализацию спецификации
* Помощь в информировании и разъяснении функций создаваемой системы заинтересованным сторонам

1. **Дайте определение понятиям: узел, артефакт, интерфейс**

Артефакты – это конкретные элементы, которые вызваны процессом разработки. Примерами артефактов являются библиотеки, архивы, конфигурационные файлы, исполняемые файлы и т.д.

Узел - представляет собой физическую сущность, которая выполняет одну или несколько компонентов, подсистем или исполняемых файлов. Узел может быть аппаратным или программным элементом.

Интерфейс – элемент, который служит для спецификации параметров модели, которые видимы извне без указания их внутренней структуры.

1. **Опишите нотации, которые используются для представления компонентов**

|  |  |
| --- | --- |
| **Тип графического элемента** | **Нотация** |
| 1.Компонент с текстовым стереотипом |  |
| 2.Компонент с пиктограммой стереотипа |  |
| 3.Компонент с предоставляемым интерфейсом |  |
| 4.Компонент имеет порт с предоставляемым интерфейсом |  |
| 5.Компонент с требуемым интерфейсом |  |
| 6.Компонент имеет порт с требуемым интерфейсом |  |
| 7.Компонент имеет несколько портов с предоставляемыми и требуемыми интерфейсами |  |

1. **Опишите основные нотации, которые используются для представления архитектуры системы в виде диаграммы развертывания**

Узлы (Nodes): Они представляют собой материальные элементы или вычислительные активы, такие как серверы, персональные компьютеры, мобильные устройства и т.д. Узлы отображаются в виде прямоугольников с названием устройства.

Артефакты (Artifacts): Они представляют собой программные элементы или данные, которые размещены на узлах. Они отображаются в виде прямоугольников или эллипсов внутри узлов.

Связи (Connections): Они демонстрируют взаимодействие между узлами. Они могут быть однонаправленными или двунаправленными линиями, обозначающими поток данных, коммуникации или зависимости между узлами.

1. **Укажите основные виды связей между компонентами и между узлами**

Прямая связь (Direct Link): Между компонентами: Компоненты взаимодействуют напрямую друг с другом без участия посредника. Между узлами: Прямая связь между узлами подразумевает, что они могут обмениваться данными или командами напрямую.

Индиректная связь (Indirect Link):Между компонентами: Взаимодействие осуществляется через посредника или промежуточный слой. Между узлами: Данные передаются между узлами через промежуточные узлы или сетевые сервисы.

Однонаправленная связь (One-Way Link):Между компонентами: Информация передается только в одном направлении. Между узлами: Передача данных идет только от одного узла к другому, без возврата.

Двунаправленная связь (Two-Way Link): Между компонентами: Взаимодействие происходит в обоих направлениях. Между узлами: Обмен данными между узлами возможен в обе стороны.

Синхронная связь (Synchronous Link):Между компонентами: Компоненты взаимодействуют в режиме реального времени, ожидая ответа друг от друга. Между узлами: Коммуникация происходит с соблюдением временных рамок.

Асинхронная связь (Asynchronous Link): Между компонентами: Взаимодействие происходит независимо от времени, без ожидания мгновенного ответа. Между узлами: Данные передаются без строгого соблюдения временных рамок.

Распределенная связь (Distributed Link): Между компонентами: Компоненты распределены по разным узлам системы. Между узлами: Узлы в сети обмениваются информацией и ресурсами.

Физическая связь (Physical Link): Между компонентами: Прямое физическое соединение между компонентами, например, проводами. Между узлами: Физическое подключение узлов через сетевые кабели или беспроводные технологии.

Сетевые связи (Network Links): Показывают физические сетевые соединения между узлами. Это могут быть Ethernet, Wi-Fi, VPN и другие типы сетевых соединений.

Коммуникационные связи (Communication Links): Представляют логические связи между компонентами системы. Это могут быть протоколы обмена данными, такие как HTTP, TCP/IP, MQTT и другие, которые описывают способы обмена информацией между компонентами.

Зависимости (Dependencies): Показывают, какие компоненты или узлы зависят от других. Например, если один компонент зависит от базы данных, это может быть представлено стрелкой или связью, указывающей на эту зависимость.

Репликация и балансировка нагрузки (Replication and Load Balancing): Если компоненты или данные реплицируются на несколько узлов или происходит балансировка нагрузки между узлами для повышения производительности и надежности, это может быть показано на диаграмме развертывания.

Прокси и шлюзы (Proxies and Gateways): Представляют промежуточные узлы, используемые для обеспечения безопасности, переадресации или трансляции данных между компонентами.

**Вывод**: В ходе выполнения данной лабораторной работы было осуществлено изучение методологии объектно-ориентированного моделирования с использованием Unified Modeling Language (UML). Основной целью работы было ознакомление с основными принципами объектно-ориентированного проектирования программного обеспечения и приобретение навыков проектирования структуры информационной системы с применением UML.

В процессе выполнения лабораторной работы была выбрана конкретная система — "Интерфейс" системы видеоконтроля за объектом. С использованием UML была разработана диаграмма развёртывания данной системы, что позволило визуализировать её архитектуру и взаимодействие компонентов. Эта диаграмма предоставила наглядное представление о распределении элементов системы по физическим узлам, а также о взаимосвязях между ними.

Полученные теоретические знания о проектировании программного обеспечения в рамках объектно-ориентированных методологий и использование UML позволили глубже понять принципы построения сложных информационных систем. В частности, овладение навыками разработки диаграммы развёртывания дало возможность не только визуализировать структуру системы, но и эффективно анализировать её компоненты, их взаимосвязи и влияние на общую функциональность.

Таким образом, результаты лабораторной работы подтвердили эффективность использования UML в процессе объектно-ориентированного проектирования, предоставив студентам не только теоретические знания, но и практические навыки в области проектирования сложных информационных систем. Полученный опыт оказывается важным элементом в подготовке к разработке современных программных продуктов, учитывающих требования качественного и устойчивого проектирования.