Учреждение образования

«БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

Лабораторная работа №6

По дисциплине “Программирование программного обеспечения”

По теме “ Объектно-ориентированное моделирование. Физические диаграммы UML.”

Цель работы: “Изучение методологии объектно-ориентированного моделирования средствами UML”

Студент: Глухова Д. В.

ФИТ 3 курс 2 группа

Преподаватель: Якубенко К.Д.

1. **Теоретические вопросы подготовки к лабораторной работе**
2. Укажите назначение физических диаграмм: компонентов и развертывания.

Диаграмма компонентов позволяет определить состав программных компонентов, в роли которых может выступать исходный, бинарный и исполняемый код.

Диаграмма развёртывания же позволяет определить распределение компонентов системы по её физическим узлам; показать физические связи между всеми узлами реализации системы на этапе ее выполнения; выявить узкие места системы и реконфигурировать её топологию для достижения требуемой производительности.

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

Артефакт – некая физическая сущность, программный компонент, который используется или создаётся во время работы программного обеспечения.

Узел – то, что может содержать программное обеспечение (некоторый физически существующий элемент системы, обладающий вычислительным ресурсом).

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

1. Опишите нотации, которые используются для представления компонентов.

Для представления компонентов используются две нотации – рисунок 1.1, отличаются они визуально.

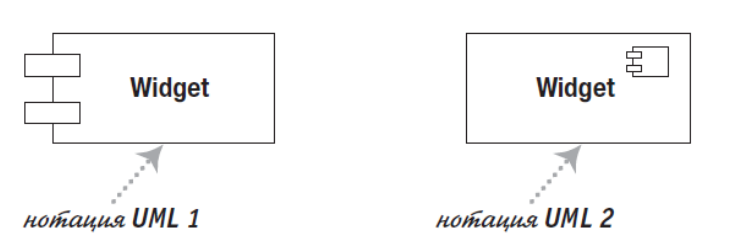


Рисунок 1.1 – Нотации для представления компонентов

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

Узлы (Nodes): представляют физические устройства или вычислительные ресурсы, такие как серверы, ПК, мобильные устройства и т.д.

Узлы изображаются в виде прямоугольников с именем устройства.

Артефакты (Artifacts): представляют программные компоненты или данные, которые размещены на узлах.

Они изображаются в виде прямоугольников или эллипсов внутри узлов.

Связи (Connections): показывают взаимодействие между узлами.

Они могут быть направленными или двунаправленными линиями, обозначающими поток данных, коммуникации или зависимостей между узлами.

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

Сетевые связи (Network Links):

Эти связи показывают физические сетевые соединения между различными узлами системы. Они могут включать в себя Ethernet, Wi-Fi, VPN и другие типы физических сетевых подключений. Эти связи определяют, как компоненты системы физически соединены друг с другом через сетевую инфраструктуру.

Коммуникационные связи (Communication Links):

Коммуникационные связи представляют логические протоколы и способы обмена данными между компонентами системы. Примерами могут быть HTTP, TCP/IP, MQTT, AMQP и другие протоколы прикладного уровня. Эти связи описывают, как компоненты взаимодействуют и обмениваются информацией в рамках логической архитектуры системы.

Зависимости (Dependencies):

Связи зависимости показывают, какие компоненты или узлы полагаются на другие компоненты или узлы. Например, если компонент приложения зависит от базы данных, это может быть изображено стрелкой, указывающей на зависимость от базы данных. Зависимости помогают понять, как компоненты системы связаны друг с другом и что произойдет, если один из компонентов выйдет из строя.

Репликация и балансировка нагрузки (Replication and Load Balancing):

Эти связи отображают, когда компоненты или данные реплицируются на несколько узлов. Они также показывают, когда происходит балансировка нагрузки между узлами, чтобы повысить производительность и отказоустойчивость системы. Эти связи помогают понять, как организована масштабируемость и высокая доступность в архитектуре системы.

Прокси и шлюзы (Proxies and Gateways):

Эти связи представляют промежуточные узлы, используемые для обеспечения безопасности, маршрутизации или преобразования данных между компонентами. Примерами могут быть прокси-серверы, шлюзы API, брандмауэры и другие промежуточные компоненты. Эти связи помогают понять, как организована безопасность, доступ и интеграция между различными частями системы.

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

Функциональные требования к системе можно разделить на требования для различных ролей приложения – учителя, ученика, родителя и администратора образовательного учреждения.

Функционал для учителя:

* Регистрация и авторизация: возможность создания учетной записи и входа в систему.
* Управление расписанием: создание, редактирование и просмотр расписания уроков.
* Ведение успеваемости: выставление оценок и комментариев к ним для каждого ученика.
* Создание и редактирование заданий: возможность добавления домашних заданий и материалов для уроков.
* Общение с учениками и родителями: инструменты для отправки уведомлений и сообщений.
* Просмотр статистики успеваемости: анализ успеваемости и активности учеников через отчеты.
* Отмена или изменение заданий: возможность редактирования или отмены ранее установленных заданий.

Функционал для ученика:

* Регистрация и авторизация: создание личного кабинета и вход в систему.
* Просмотр расписания: доступ к расписанию уроков и информации о предстоящих занятиях.
* Просмотр заданий: доступ к домашним заданиям, срокам их выполнения и комментариям учителей.
* Обратная связь: возможность оставлять комментарии и задавать вопросы учителям.
* Просмотр успеваемости: возможность отслеживания собственных оценок и прогресса.

Функционал для администратора:

* Просмотр списка всех пользователей: доступ к информации о всех учителях, учениках и родителях.
* Управление учетными записями: добавление, изменение и удаление пользователей.
* Просмотр и управление расписанием: возможность редактирования расписаний и учебных планов.
* Подтверждение регистрации новых пользователей: проверка и подтверждение заявок на регистрацию учителей и родителей.

1. **Описание программных средств**

Для построения диаграмм IDEF0 в рамках нашего проекта использовался веб-ресурс Draw.io, разработанный компанией JGraph Ltd. Этот инструмент предназначен для создания разнообразных диаграмм и визуальных представлений информации. Адрес веб-ресурса – https://www.drawio.com. Draw.io доступен на всех платформах, которые имеют веб-браузер и подключение к Интернету, что делает его универсальным и удобным для пользователей.

Draw.io предлагает широкий функционал, который позволяет пользователям создавать профессиональные графические диаграммы с минимальными затратами времени и усилий. В числе доступных возможностей — построение графиков, смысловых карт, UML-диаграмм, диаграмм Венна, а также Agile и Kanban досок. Эти инструменты особенно полезны в рамках проектного управления и разработки программного обеспечения, так как они помогают визуализировать процессы, повышая их наглядность и понятность.

Кроме того, Draw.io поддерживает создание диаграмм мозговых штурмов, архитектурных диаграмм технических систем и других типов визуальных представлений, необходимых для анализа и планирования. Интуитивно понятный интерфейс и наличие различных шаблонов делают процесс создания диаграмм доступным даже для новичков. Пользователи могут легко добавлять, редактировать и настраивать элементы диаграмм, что способствует более эффективному обмену идеями и информацией.

Одной из ключевых особенностей Draw.io является возможность совместной работы в реальном времени. Пользователи могут одновременно редактировать диаграммы, что особенно полезно для команд, работающих над общими проектами. Интеграция с популярными облачными сервисами, такими как Google Drive, SharePoint и Dropbox, позволяет удобно сохранять и делиться созданными диаграммами.

Важно отметить, что Draw.io придерживается принципов конфиденциальности и безопасности. Пользователи имеют возможность хранить свои данные в выбранном облачном хранилище или на локальном устройстве, что обеспечивает дополнительный уровень контроля над информацией. Платформа не имеет доступа к пользовательским данным, что делает её надежным инструментом для работы с чувствительной информацией.

В заключение, Draw.io является мощным и многофункциональным инструментом, который отвечает современным требованиям к визуализации данных и совместной работе. Его доступность на различных платформах и богатый функционал делают его идеальным выбором для построения диаграмм IDEF0 и других визуальных представлений в рамках образовательных и профессиональных проектов.

1. **Описание практического задания**

Компоненты:

* Device Personal Computer: клиентское приложение, реализованное с помощью технологии <component>.
* Device Phone: клиентское мобильное приложение, реализованное с помощью технологии <component>.

Школьная среда: включает в себя компоненты:

* Keyboards: физические устройства ввода.
* Config.js: конфигурационный компонент.
* main.js: основной компонент приложения.

Handlers: компоненты, отвечающие за обработку запросов и данных.

Database server: серверный компонент, отвечающий за хранение и управление данными, включает в себя функциональность для учета студентов, учителей и т.д.

Server app: серверный компонент приложения, взаимодействующий с базой данных и клиентами.

Связи между компонентами:

* Клиентские приложения (Device Personal Computer, Device Phone) взаимодействуют с основным компонентом main.js в школьной среде через интерфейсы <component>.
* Keyboards взаимодействуют с main.js для ввода данных.
* main.js взаимодействует с Config.js для получения конфигурационных параметров.
* main.js взаимодействует с Handlers для обработки запросов и данных.
* Handlers взаимодействуют с Database server для получения и сохранения данных.
* Server app взаимодействует с Database server для выполнения операций с данными.
* Server app взаимодействует с клиентскими приложениями (Device Personal Computer, Device Phone) для обмена данными.

Физические устройства:

* Device Personal Computer: стационарные персональные компьютеры.
* Device Phone: мобильные устройства (смартфоны, планшеты).
* Database server: выделенный сервер для хранения данных.
* Keyboards: физические клавиатуры, подключенные к стационарным компьютерам.

Соединения между устройствами:

* Клиентские приложения (Device Personal Computer, Device Phone) взаимодействуют с Server app по сетевым протоколам (например, HTTP/HTTPS).
* Server app взаимодействует с Database server по протоколам доступа к базам данных (например, SQL).
* Keyboards подключаются к стационарным компьютерам по протоколам ввода (например, USB).

Среды выполнения:

* Клиентские приложения (Device Personal Computer, Device Phone) работают в средах выполнения, соответствующих операционным системам персональных компьютеров и мобильных устройств.
* Серверные компоненты (Server app, Database server) работают на выделенных серверных платформах.

Размещение компонентов:

* Клиентские приложения (Device Personal Computer, Device Phone) размещаются на соответствующих физических устройствах (стационарные ПК, мобильные устройства).
* Keyboards, Config.js, main.js размещаются в школьной среде.
* Handlers размещаются на сервере приложений.
* Database server размещается на выделенном сервере для хранения данных.
* Server app размещается на сервере приложений.

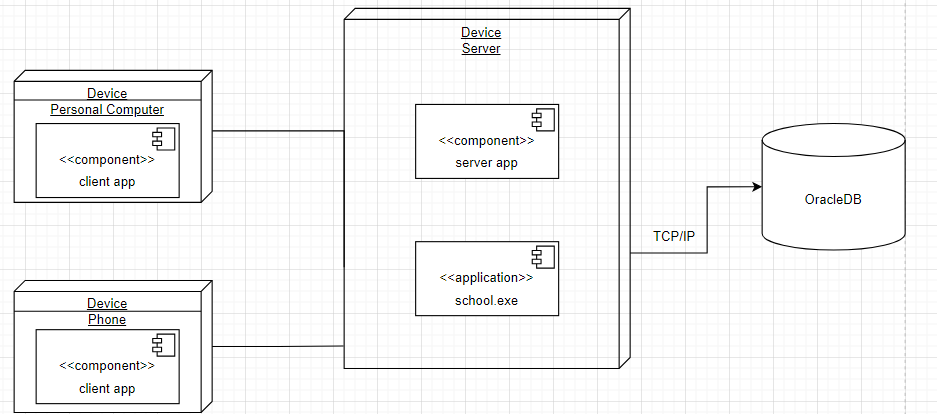


Рисунок 4.1 – Диаграмма развертывания и компонентов

Общая структура диаграммы развертывания и компонентов демонстрирует хорошо продуманную архитектуру вашего приложения, ориентированную на модульность, масштабируемость и расширяемость.