Министерство образования Республики Беларусь

Учреждение образования

«Белорусский государственный технологический университет»

Кафедра информационных систем и технологий

**Отчет по лабораторной работе на тему**:

**«Объектно-ориентированное моделирование. Физические диаграммы UML»**

Выполнил:

студент 4 курса 7 группы ФИТ

Бобрович Г.С.

Цель:

Изучение методологии объектно-ориентированного моделирования средствами UML. Ознакомление с основными принципами объектно-ориентированного проектирования программного обеспечения, получение навыков проектирования архитектуры информационной системы с применением методологии UML.

Минск 2023

# 1. Теоретические вопросы

## **1.1 Укажите назначение физических диаграмм: компонентов и развертывания.**

Диаграмма компонентов позволяет создать физическое отражение текущей модели, показывает организацию и взаимосвязи программы компонентов, представленных в исходном коде двоичных или выполняемых файлов. Связи в данном типе диаграммы представляют зависимости одного компонента от другого и имею специальное отражение через «значок» зависимости. Так же данный тип диаграмм позволяет получить представление о поведении компонентов по предоставляемому им интерфейсу.

Компоненты представляют собой модуль ПО, такой как исходный код, двоичный код, выполняемый файл, библиотеки и т.д. Компоненты так же могут использоваться для показа взаимосвязей модулей на этапе компиляции или выполнения программы, а также показывают какие классы используются для создания конкретных компонентов. В связи с тем, что система может состоять из модулей различного типа, пользователь может использовать стереотипы для определения этих различий, что часто ведет к изменению графического отображения компонента на диаграмме.

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. Они обычно используются для визуализации физического аппаратного и программного обеспечения системы. Используя его, вы можете понять, как система будет физически развернута на аппаратном обеспечении.

## **1.2 Дайте определение понятиям: узел, артефакт, интерфейс.**

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

Артефакт – это конкретный элемент, который вызван процессом разработки. Примерами артефактов являются библиотеки, архивы, конфигурационные файлы, исполняемые файлы и т.д.

Интерфейс – набор правил, обязательный для реализации в каком-либо элементе.

## **1.3 Опишите нотации, которые используются для представления компонентов (их вариации).**

Нотации для диаграммы компонентов:

1) Компонент:

Существует три способа использования символа компонента.

- Прямоугольник со стереотипом компонента (текст <<компонент>>). Стереотип компонента обычно используется над именем компонента, чтобы не перепутать форму со значком класса.

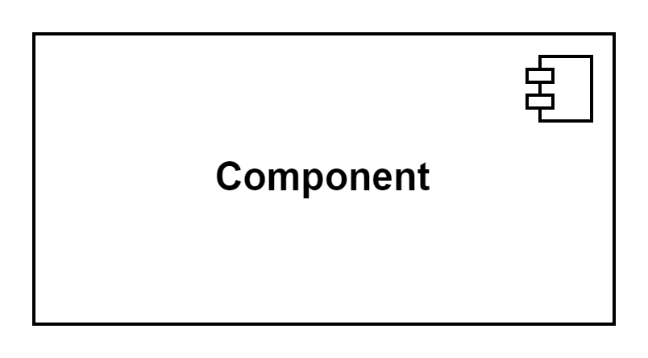


Рисунок .1 – Компонент

- Прямоугольник со значком компонента в правом верхнем углу и названием компонента.

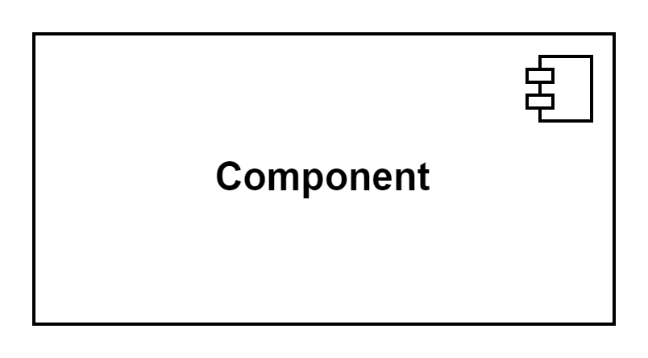


Рисунок 1. – Значок компонента

- Прямоугольник со значком компонента и стереотипом компонента.

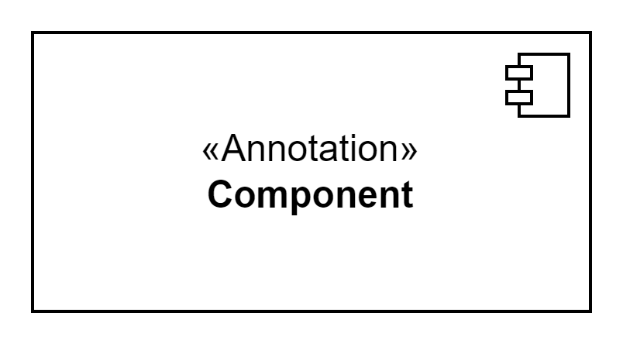


Рисунок 1. – Значок компонента и стереотип компонента

2) Предоставляемый интерфейс и требуемый интерфейс

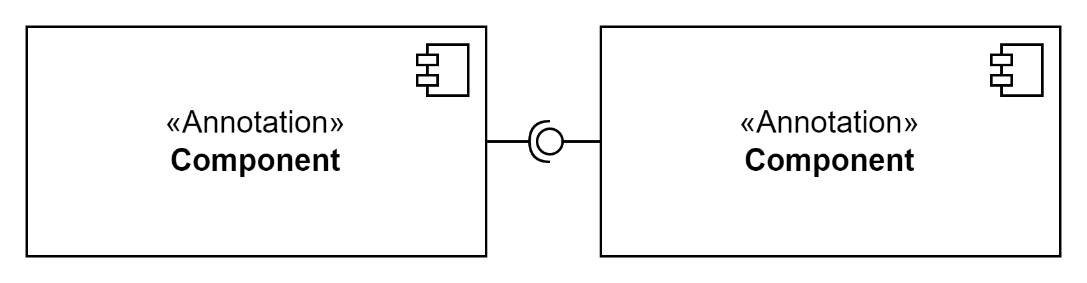


Рисунок 1. – Предоставляемый и требуемый интерфейс

Интерфейсы на компонентных схемах показывают, как компоненты соединены друг с другом и взаимодействуют друг с другом. Соединитель сборки позволяет соединить требуемый интерфейс компонента (представленный полукругом и сплошной линией) с предусмотренным интерфейсом (представленный окружностью и сплошной линией) другого компонента. Это показывает, что один компонент предоставляет услугу, которая требуется другому.

3) Порт

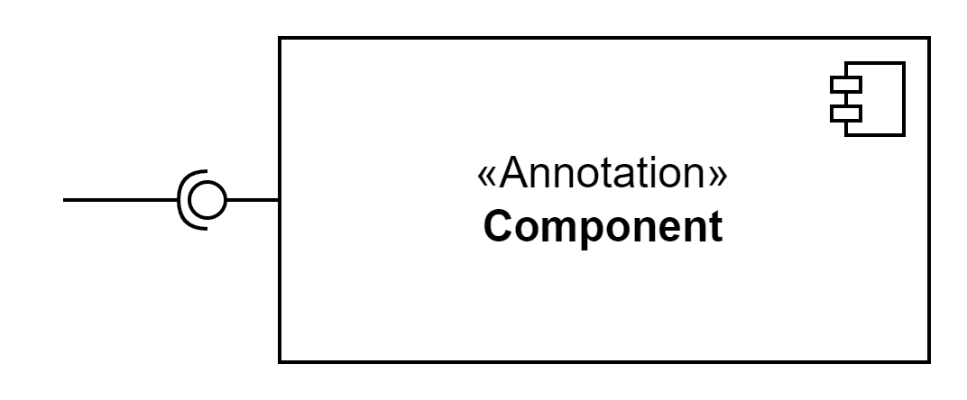


Рисунок 1. – Порт

Порт (представленный маленьким квадратом в конце требуемого интерфейса или предоставляемого интерфейса) используется, когда компонент делегирует интерфейсы внутреннему классу.

4) Зависимости

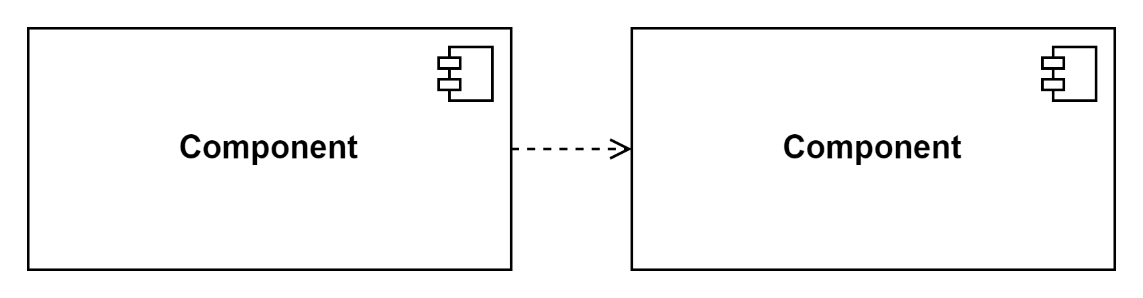


Рисунок 1. – Зависимости

Хотя вы можете показать более подробную информацию об отношениях между двумя компонентами с помощью нотации ball-and-socket (предусмотренный интерфейс и требуемый интерфейс), вы можете также использовать стрелку зависимостей, чтобы показать отношения между двумя компонентами.

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

Основные нотации в диаграмме развёртывания:

1) Узлы

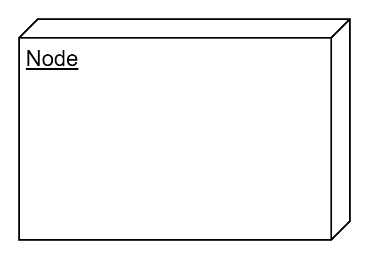


Рисунок 1. – Узел

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

2) Артефакты

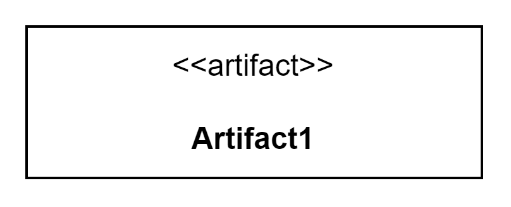


Рисунок 1. – Артефакт

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

3) Коммуникационная ассоциация

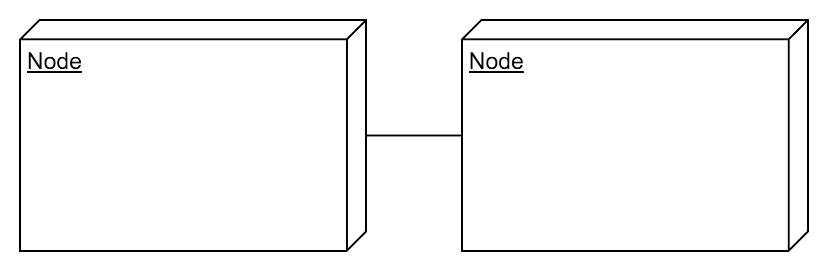


Рисунок 1. – Коммуникационная ассоциация

Это представлено сплошной линией между двумя узлами. Он показывает путь связи между узлами.

4) Устройства

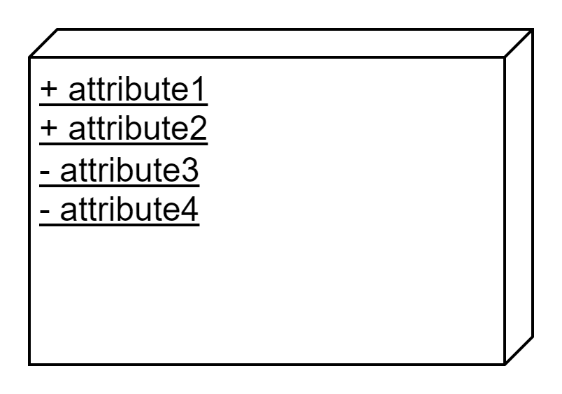


Рисунок 1. – Устройство

Устройство – это узел, который используется для представления физического вычислительного ресурса в системе. Примером устройства является сервер приложений.

5) Спецификации развертывания

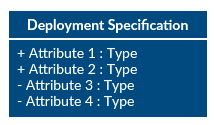


Рисунок 1. – Спецификация развёртывания

Спецификации развертывания – это файл конфигурации, например текстовый файл или XML-документ. В нем описывается, как артефакт развертывается на узле.

## **1.5 Укажите основные виды связей между компонентами и между узлами.**

Основные виды связей между компонентами и между узлами:

1) Отношение зависимости

Зависимости могут отражать связи отдельных файлов программной системы на этапе компиляции и генерации объектного кода. В других случаях зависимость может указывать на наличие в независимом компоненте описаний классов, которые используются в зависимом компоненте для создания соответствующих объектов. Применительно к диаграмме компонентов зависимости могут связывать компоненты и импортируемые этим компонентом интерфейсы, а также различные виды компонентов между собой.

2) Соединитель сборки

Он позволяет соединить требуемый интерфейс компонента (представленный полукругом и сплошной линией, рисунок 1.5) с предусмотренным интерфейсом (представленный окружностью и сплошной линией) другого компонента. Это показывает, что один компонент предоставляет услугу, которая требуется другому.

# 2. Описываемые функциональные требования

## **2.1 Функциональные требования**

Пользователю доступны функции, такие как:

* Регистрация/авторизация;
* Бронирование времени репетиции/записи;
* Система обратной связи (отзывы);
* Просмотр истории забронированного времени;
* Просмотр информации о студии;
* Редактирование профиля пользователя.

Модератору, помимо вышеперечисленных функций, предоставляется:

* Блокировка пользователей и ее отмена;
* Редактирование информации о студии;
* Отмена и восстановление забронированного времени;
* Просмотр списка пользователей;
* Просмотр актуального бронирования.

Администратор имеет весь описанный выше функционал, с добавлением:

* Выдача/снятие полномочий модератора;
* Удаление отзывов.

## **2.2 Основные системные требования**

Пользователю доступны функции, такие как:

* Регистрация/авторизация;
* Бронирование времени репетиции/записи;
* Система обратной связи (отзывы);
* Просмотр истории забронированного времени;
* Просмотр информации о студии;
* Редактирование профиля пользователя.

Модератору, помимо вышеперечисленных функций, предоставляется:

* Блокировка пользователей и ее отмена;
* Редактирование информации о студии;
* Отмена и восстановление забронированного времени;
* Просмотр списка пользователей;
* Просмотр актуального бронирования.

Администратор имеет весь описанный выше функционал, с добавлением:

* Выдача/снятие полномочий модератора;
* Удаление отзывов.

## **Аппаратные требования**

– ОЗУ 1 GB;

– Частота процессора не меньше 1 GHz

– Операционная система: Android 4.1 и выше, iOS 9.0 и выше;

– Интернет-соединение: 3G, 4G, Wi-Fi;

– Внутренняя память не менее 128 Mb.

# 3. Описание программных средств

Построение моделей выполнялось в <https://app.diagrams.net/>. Draw.io — инструмент для создания диаграмм, блок-схем, интеллект-карт, бизнес-макетов, отношений сущностей, программных блоков и другого. Сервис распространяется на бесплатной основе с открытым исходным кодом. Draw.io обладает богатым набором функций для визуализации большинства задач пользователя.

*Разработчик*: JGraph Ltd. Сервис распространяется на бесплатной основе с открытым исходным кодом.

# 4. Описание практического задания

Эта диаграмма представляет физические элементы системы и их окончательное представление. Она показывает, как каждый компонент будет представлен в конечной форме.

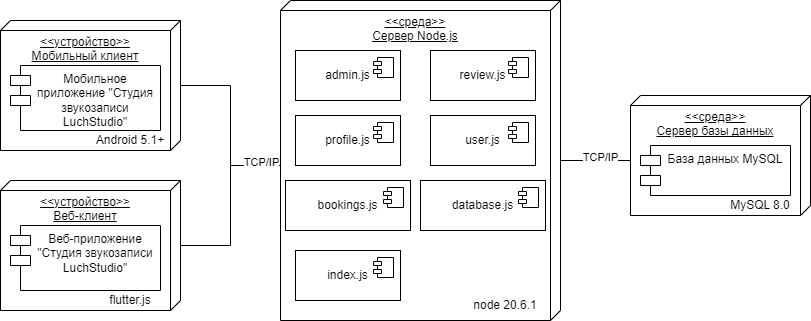


Рисунок 4.1 – Диаграмма компонентов и развёртывания

Здесь отмечены устройства, способные взаимодействовать с окружающими средами. Например, три компонента (мобильный клиент, веб-клиент и база данных) взаимодействуют с сервером через протоколы TCP/IP. Сервер обрабатывает входящие запросы при помощи обозначенных компонентов. Реальная реализация устройств и окружающих сред описывается с помощью артефактов.