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

Фурсик А.А

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 2021

# 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.

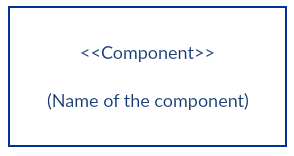


Рисунок 1.1 – Компонент

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

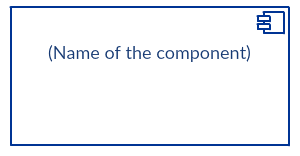


Рисунок 1.2 – Значок компонента

- Прямоугольник со значком компонента и стереотипом компонента.

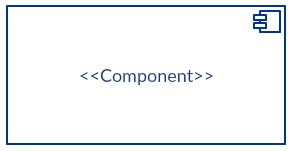


Рисунок 1.3 – Значок компонента и стереотип компонента

2) Предоставляемый интерфейс и требуемый интерфейс

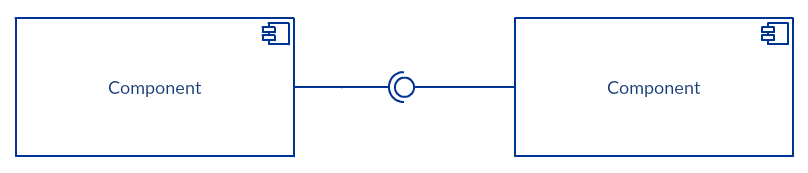


Рисунок 1.4 – Предоставляемый и требуемый интерфейс

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

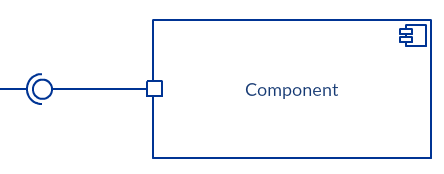


Рисунок 1.5 – Порт

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

4) Зависимости



Рисунок 1.6 – 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

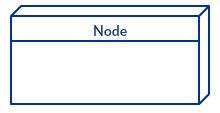


Рисунок 1.7 – Узел

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 1.8 – Артефакт

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

3) Коммуникационная ассоциация

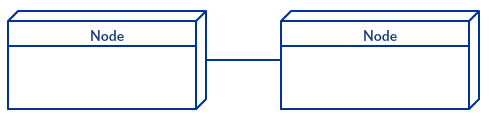


Рисунок 1.9 – Коммуникационная ассоциация

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

4) Устройства

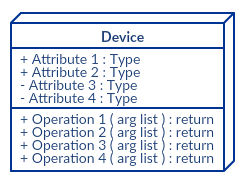


Рисунок 1.10 – Устройство

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

5) Спецификации развертывания

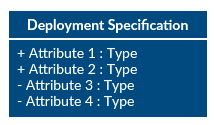


Рисунок 1.11 – Спецификация развёртывания

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

Возможности приложения:

- Администратор:

* Добавление новой информации;
* Изменение информации;
* Удаление информации;
* Просмотр статистики пользователей;
* Вход в аккаунт.

- Пользователь:

* Просмотр интересующей информации;
* Пометка просмотренной информации;
* Получение достижений;
* Получение звания;
* Прохождение опросов;
* Прохождение тестов;
* Выбор стороны конфликта (только при регистрации);
* Вход в аккаунт.

## 2.2 Основные системные требования

Приложение состоит из 3 основных компонентов:

* Сервер, хранящий информацию в базе данных, и взаимодействующий с пользователями;
* Мобильное приложение, позволяющее выполнять функции роли пользователя;
* Веб-приложение администратора, позволяющее выполнять функции роли администратора.

База данных сервера хранит информацию о событиях первой мировой и пользователях. Основана на СУБД MongoDB.

Сервер предоставляет REST API для взаимодействия клиентов с системой. Использует технологию Node.js, передача данных осуществляется в формате JSON.

Мобильное приложение используется для выполнения функций пользователя, написано на языке Java, а также имеет локальную базу данных SQLite, содержащую в себе данные пользователя и информацию, необходимую для работы приложения (без подключения к сети интернет) и последующей синхронизацией с сервером. Имеет следующие экраны для роли пользователя:

* Вход;
* Регистрация;
* Года войны;
  + События года войны;
* Боевая техника войны;
  + Наземная техника;
  + Авиация;
  + Флот;
* Оружие;
* Достижения;
* Тесты;
* Профиль.

Достижения выдаются пользователю после совершения определённых действий, указанных в описании достижений. По умолчанию в каждой статье присутствует опрос различной степени сложности, служащий для получения достижения. За их выполнение, а также за прохождение тестов пользователь получает очки. При определённом количестве очков пользователь получает соответствующее звание.

Веб-приложение используется для выполнения функций администратора и представляет из себя SPA, работающее в браузере. Использует фреймворк Node.js Express. Интерфейс состоит из экранов:

* Вход;
* Регистрация;
* Список пользователей с их статистикой;
* Записи к году войны;
* Записи о технике;
* Записи о оружии;
* Записи об авиации;
* Список вопросов для тестов;
* Список вопросов для опросов;
* Список достижений.

## 2.3 Аппаратные требования

* ОЗУ 512Mb;
* OS Android 6.0 и выше (API 23+);
* Сетевая пропускная способность выше 200Kb/s;
* Внутренняя память не менее 128Mb.

# 3. Описание программных средств

Для создания схем используется Microsoft Visio – программа, с помощью которой появляется возможность для составления графиков, чертежей, диаграмм, блок-схем. Приложение помогает представить графическую информацию в простом и доступном виде.

Это отличный графический редактор, позволяющий работать с диаграммами и схемами. Программа содержит мощный набор инструментов, который будет полезен для работы. Приложение может применяться в разных сферах. Его используют it-специалисты, менеджеры, аналитики. Программа входит в пакет Office, а также может использоваться в виде отдельного компонента.

Программа, имеющая множество полезных инструментов, которые отлично подходят для управления каким-либо проектом. Приложение располагает мощным арсеналом средств. Причем от пользователя не требуются какие-либо профессиональные знания в технической или изобразительной области, а также связанные с этим навыки. Вы можете использовать готовые шаблоны, фигуры и элементы, с помощью которых добьетесь нужного результата.

Какие-либо навыки работы в Microsoft Visio для этого не нужны. Разобраться в программе сможет даже неопытный и начинающий пользователь. Приложение имеет удобный и понятный интерфейс. Вы сможете легко создавать нужные диаграммы и схемы, необходимые для решения конкретных задач.

Возможности приложения зависят от версии. Она может быть обычной или полнофункциональной. В первом случае вы можете просматривать диаграммы и выполнять их печать. Во втором – появляется возможность для создания и редактирования диаграмм. Версия с полным функционалом является более привлекательной.

Для создания диаграммы развёртывания и диаграммы компонентов использовалось приложение Microsoft Visio Pro 2021 (https://www.microsoft.com/ru-RU/download/details.aspx?id=51188).

# 4. Описание практического задания

Данная диаграмма описывает физические компоненты системы. На ней мы видим, как именно у нас будет представлен тот или иной компонент в конечном виде (файл, набор библиотек и т.д.). Здесь мы обозначаем устройства, которые могут взаимодействовать со средами. В нашем случае 3 компонента (мобильный клиент, веб-клиент и база данных) взаимодействуют с сервером по стеку протоколов TCP/IP. Сервер же в свою очередь обрабатывает приходящие к нему запросы при помощи тех компонент, которые мы в нём обозначили (DbUtils.java, AuthUtils.java и т.д.).

Конкретная реализация наших устройств и сред описывается при помощи артефактов (прямоугольник с иконкой файла). Спецификацию устройства или среды мы описываем в правом нижнем углу.



Рисунок 4.1 – Диаграмма компонентов и развёртывания