Министерство образования Республики Беларусь

Учреждение образования

«Белорусский государственный технологический университет»

Кафедра информационных систем и технологий

**Отчет по лабораторной работе на тему**:

**«Объектно-ориентированное моделирование. Диаграммы поведения UML»**

Выполнил:

студент 4 курса 7 группы ФИТ

Бобрович Г.С.

Цель:

Изучение методологии объектно-ориентированного моделирования средствами UML. Получение дополнительных навыков проектирования моделей информационной системы с применением возможностей UML диаграмм.

Минск 2023

# 1. Теоретические вопросы

## **1.1 Укажите виды диаграмм поведения. Какая между ними связь?**

Виды диаграмм поведения:

1) Диаграмма деятельности;

2) Диаграмма вариантов использования;

3) Обзорная диаграмма взаимодействия;

4) Временная диаграмма;

5) Диаграмма конечного автомата;

6) Диаграмма последовательности;

7) Диаграмма связи.

Основное внимание в диаграммах поведения уделяется динамическим аспектам системы программного обеспечения или процесса. Они показывают функциональные возможности системы и демонстрируют, что должно происходить в моделируемой системе.

## **1.2 Опишите назначение диаграммы деятельности.**

Этот тип изображает пошаговый процесс с четким началом и концом. Это набор операций, которые должны быть выполнены, чтобы достичь цели. Она показывает, как каждое действие ведет к следующему, и как все они связаны. Помимо разработки программного обеспечения, они могут использоваться практически в любой бизнес-среде. Их также называют картированием или моделированием бизнес-процессов.

## **1.3 Опишите основные нотации, которые используются на диаграмме состояний.**

Основные нотации диаграммы состояний:

1) Круг, обозначающий начальное состояние.

2) Окружность с маленьким кругом внутри, обозначающая конечное состояние (если есть).

3) Скруглённый прямоугольник, обозначающий состояние. Верхушка прямоугольника содержит название состояния. В середине может быть горизонтальная линия, под которой записываются активности, происходящие в данном состоянии.

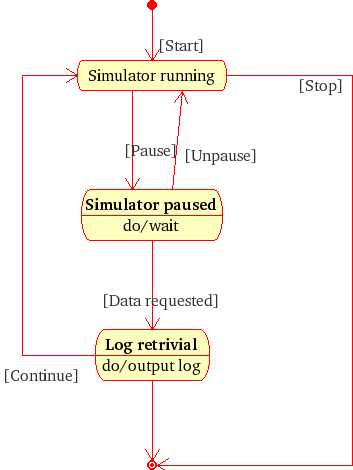


Рисунок 1.1. - Диаграмма последовательности

## **1.4 Укажите виды связей между объектами на диаграмме последовательностей.**

Виды связей:

1) Стрелка, обозначающая переход. Название события (если есть), вызывающего переход, отмечается рядом со стрелкой. Охраняющее выражение может быть добавлено перед «/» и заключено в квадратные скобки (название\_события[охраняющее\_выражение]), что значит, что это выражение должно быть истинным, чтобы переход имел место. Если при переходе производится какое-то действие, то оно добавляется после «/» (название\_события[охраняющее\_выражение]/действие).

2) Толстая горизонтальная линия с либо множеством входящих линий и одной выходящей, либо одной входящей линией и множеством выходящих. Это обозначает объединение и разветвление соответственно.

## **1.5 Какая диаграмма позволяет моделировать параллельные вычисления?**

Диаграмма деятельности UML отображает разложение определенной деятельности на несколько составных частей. В данном случае понятием «деятельность» называется спецификация определенного исполняемого поведения в виде параллельного, а также координированного последовательного выполнения различных подчиненных элементов – вложенных типов деятельности и различных действий, объединенных потоками, идущими от выходов определенного узла к входам другого.

Именно она используются для того, чтобы моделировать различные бизнес-процессы, параллельные и последовательные вычисления.

# 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**

Приложение состоит из 3 основных компонентов:

* Сервер, необходимый для хранения информации в базе данных и для взаимодействия с пользователями интерфейса;
* Мобильное приложение;
* Веб приложение.

База данных сервера хранит забронированное время пользователей, отзывы, отмененное время. Основана на СУБД MySQL. Так же, присутствует локальная база данных SQLite, необходимая для работы приложения без подключения к сети.

Взаимодействие пользователей с интерфейсом происходит с помощью REST API. Используется технология Node.js, а передача данных выполняется в формате JSON.

Мобильное и веб приложение написано на языке Dart. Реализованы следующие экраны приложения:

* Регистрация и авторизация;
* Главная (информация о студии);
* Расписание;
* Профиль:
  + Обновление профиля;
  + История бронирования;
* Отзывы;
* Админ-панель;

## **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

На рисунке 4.1 обозначена диаграмма последовательности. Тут показан жизненный цикл приложения в определённой ситуации. В данной диаграмме показана ситуация, когда пользователь бронирует время в студии. Штриховая линия обозначает жизненный цикл, а прямоугольник, расположенный поверх этой линии, обозначает деятельность объекта.

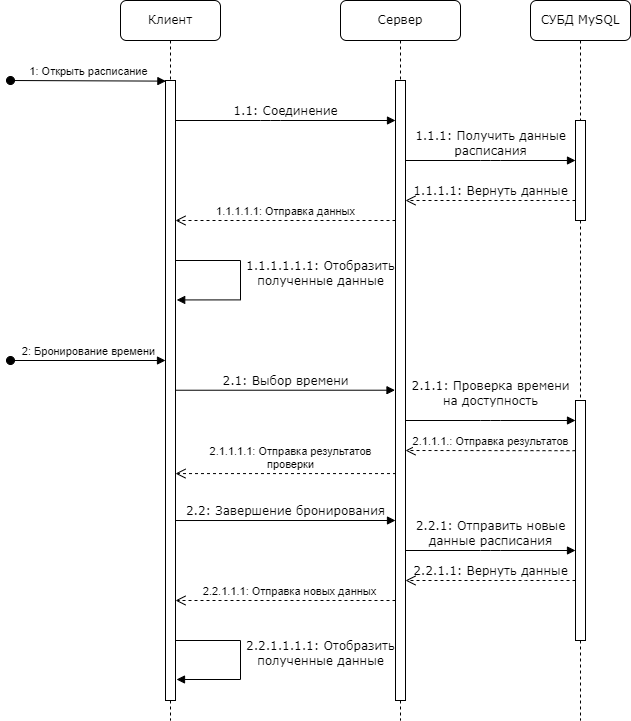


Рисунок 4.1 – Диаграмма последовательности

Чёрные закрашенные точки обозначают начало жизненного цикла в данном моменте времени. В начале, идет открытие страницы расписания. Клиент делает запрос к серверу для получения текущих данных в расписании, сервер получает их с базы данных и отправляет клиенту. Затем, клиент бронирует время, сервер проверяет выбранное время на доступность в базе данных, возвращает информацию клиенту и клиент завершает бронирование. Сервер отправляет новые данные в базу данных, и возвращает обновленную информацию о расписании клиенту.

На рисунке 4.2 показана диаграмма состояний. Она предназначена для того, чтобы выделить особые состояния приложения. В данном контексте, указаны состояния при бронировании времени. Чёрным кругом обозначено начало, чёрным кругом с обводкой – конец. В скруглённых прямоугольниках указывается состояние и процесс, который в нём происходит. Процесс чаще всего запускает новое состояние.

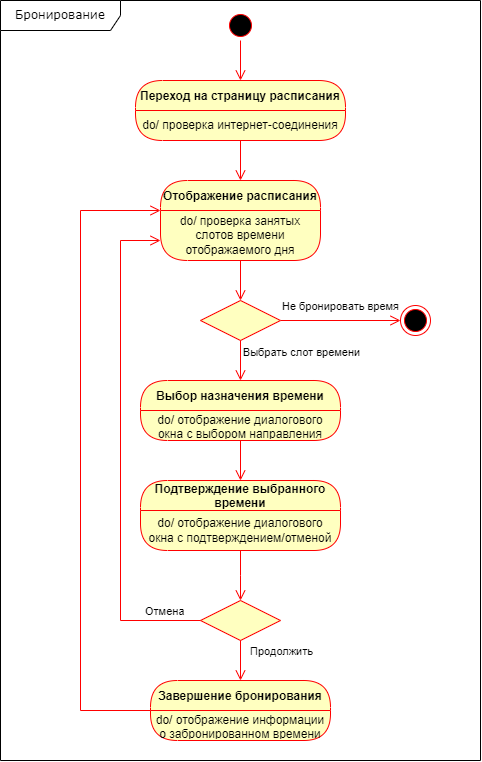


Рисунок 4.2 – Диаграмма состояний

Как видно, в начале мы переходим на страницу расписания, где выполняется проверка интернет-соединения. Далее, отображается расписание, и идет проверка занятых слотов. Затем, у пользователя есть выбор, не бронировать время и уйти со страницы, либо же продолжить и выбрать слот времени. При выборе слота предлагается выбрать назначение времени и подтвердить его. Если же пользователь отказался подтверждать время, его кидает обратно на страницу расписания, если продолжает, то отображается информация о забронированном времени, после которого он переходит на изначальную страницу.