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

Фурсик А.А

Цель:

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

Минск 2021

# 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.

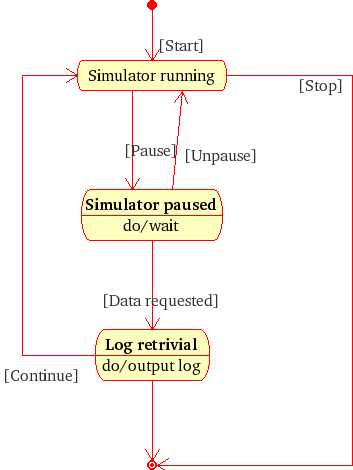


Рисунок 1.1. - Диаграмма последовательности

## 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

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 draw.io – 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, 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.

Для создания диаграммы состояний и диаграммы последовательности использовалось приложение draw.io (https://drawio-app.com/).

# 4. Описание практического задания

На рисунке 4.1 обозначена диаграмма последовательности. На ней мы обозначаем жизненный цикл нашего приложения в определённой ситуации. На рисунке рассмотрен случай прохождения теста нашим клиентом, который пользуется мобильным приложением. Штриховая линия обозначает жизненный цикл, прямоугольник, расположенный поверх этой линии, обозначает деятельность объекта.

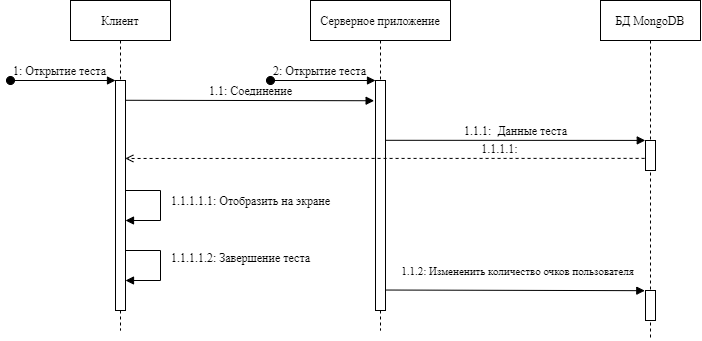


Рисунок 4.1 – Диаграмма последовательности

Чёрные закрашенные точки обозначают начало жизненного цикла в данном моменте времени. Как видно, при открытии теста мы соединяемся с серверным приложением с целью получения данных. Данная операция происходит синхронно и обозначается стрелкой с закрашенным острием. После того, как мы получили данные из базы через серверное приложение (штриховая линия), мы отображаем их на экране. По завершению теста мы изменяем количество очков у нашего пользователя.

На рисунке 4.2 показана диаграмма состояний. Она предназначена для того, чтобы выделить особые состояния приложения. На данном рисунке рассмотрены состояния, возникающие при выполнении теста. 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. Как видно, при начале теста запускается таймер, в ходе которого мы должны успеть ответить на все вопросы, предоставленные в тесте. Независимо от того, дадим ли мы правильный ответ или нет, приложение будет ожидать либо завершения по таймеру, либо завершения по нажатию на соответствующую кнопку интерфейса.

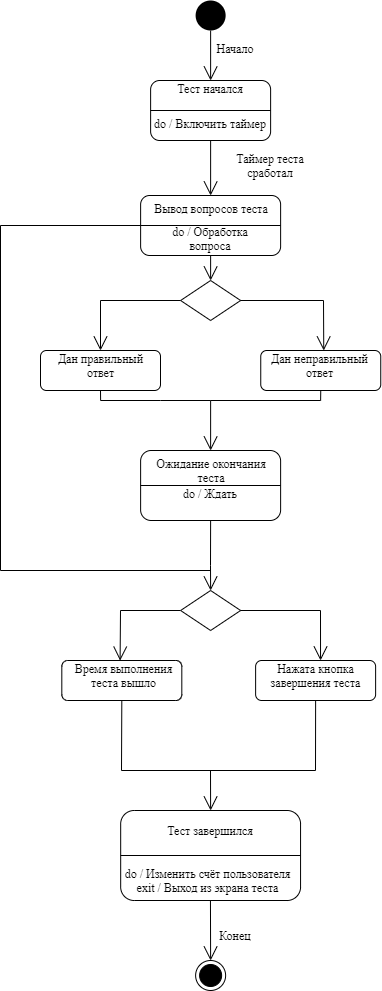


Рисунок 4.2 – Диаграмма состояний

После того, как тест завершён, мы изменяем счёт пользователя и выходим из экрана теста.