Министерство образования Республики Беларусь Учреждение образования

БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

КАФЕДРА ИНФОРМАТИКИ

Отчёт по лабораторной работе №4

По теме “Проектирование и разработка ПО на языке UML”

Выполнила: студентка гр. 053501 Шурко Т.А.

Проверил: ассистент кафедры информатики Гриценко Н. Ю.

Минск 2023

**СОДЕРЖАНИЕ**

[1 Цель работы 3](#_Toc147702999)

[2 Диаграмма компонентов 4](#_Toc147703000)

[2.1 Теоретические сведения 4](#_Toc147703001)

[2.2 Построение диаграммы 4](#_Toc147703002)

[3 Диаграмма развретывания 6](#_Toc147703003)

[3.1 Теоретические сведения 6](#_Toc147703004)

[3.2 Построение диаграммы 6](#_Toc147703005)

[4 Диаграмма объектов 8](#_Toc147703006)

[4.1 Теоретические сведения 8](#_Toc147703007)

[4.2 Построение диаграммы 8](#_Toc147703008)

[5 Диаграмма связей 9](#_Toc147703009)

[5.1 Теоретические сведения 9](#_Toc147703010)

[5.2 Построение диаграммы 9](#_Toc147703011)

[6 Диаграмма пакетов 10](#_Toc147703012)

[6.1 Теоретические сведения 10](#_Toc147703013)

[6.2 Построение диаграммы 10](#_Toc147703014)

[7 Диаграмма времени 11](#_Toc147703015)

[7.1 Теоретические сведения 11](#_Toc147703016)

[7.2 Построение диаграммы 11](#_Toc147703017)

[8 Диаграмма профиля 13](#_Toc147703018)

[8.1 Теоретические сведения 13](#_Toc147703019)

[8.2 Построение диаграммы 13](#_Toc147703020)

[Заключение 14](#_Toc147703021)

[Список использованных источников 15](#_Toc147703022)

# **1 ЦЕЛЬ РАБОТЫ**

Целью лабораторной работы №4 является разработать диаграмму компонентов (Component Diagram), диаграмму развертывания (Deployment Diagram), диаграмму объектов (Object Diagram), диаграмму связей (Relationship Diagram), диаграмму пакетов (Package Diagram), диаграмму времени (Timing Diagram), диаграмму профиля (Profile Diagram) и подготовить отчет о проделанной работе.

# **2 ДИАГРАММА КОМПОНЕНТОВ**

## **2.1 Теоретические сведения**

Диаграмма компонентов (Component Diagram) — элемент языка моделирования [UML](https://ru.wikipedia.org/wiki/UML), статическая структурная [диаграмма](https://ru.wikipedia.org/wiki/%D0%94%D0%B8%D0%B0%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B0), которая показывает разбиение программной системы на структурные [компоненты](https://ru.wikipedia.org/w/index.php?title=%D0%9A%D0%BE%D0%BC%D0%BF%D0%BE%D0%BD%D0%B5%D0%BD%D1%82_(UML)&action=edit&redlink=1) и связи (зависимости) между компонентами. В качестве физических компонентов могут выступать [файлы](https://ru.wikipedia.org/wiki/%D0%A4%D0%B0%D0%B9%D0%BB), библиотеки, модули, исполняемые файлы, пакеты.

С помощью диаграммы компонентов представляются инкапсулированные классы вместе с их интерфейсными оболочками, портами и внутренними структурами (которые тоже могут состоять из компонентов и коннекторов).

Компоненты связываются через зависимости, когда соединяется требуемый [интерфейс](https://ru.wikipedia.org/wiki/%D0%98%D0%BD%D1%82%D0%B5%D1%80%D1%84%D0%B5%D0%B9%D1%81) одного компонента с имеющимся интерфейсом другого компонента. Таким образом иллюстрируются отношения клиент-источник между двумя компонентами.

Зависимость показывает, что один компонент предоставляет сервис, необходимый другому компоненту. Зависимость изображается стрелкой от интерфейса или порта клиента к импортируемому интерфейсу.

Когда диаграмма компонентов используется, чтобы показать внутреннюю структуру компонентов, предоставляемый и требуемый интерфейсы составного компонента могут делегироваться в соответствующие интерфейсы внутренних компонентов.

Делегация показывает связь внешнего контракта компонента с внутренней реализацией этого поведения внутренними компонентами [1].

## **2.2 Построение диаграммы**

На рисунке 2.1 изображена диаграмма компонентов. Каждый компонент, по сути, является типом класса.

На диаграмме отображено взаимодействие с хранилищем данных и с базой данных, в целом. Компонент тренировка взаимодействует с компонентом сегмент.

При взаимодействии с данными, также, происходит их шифрование, с целью обезопасить данные пользователя от третьих лиц.

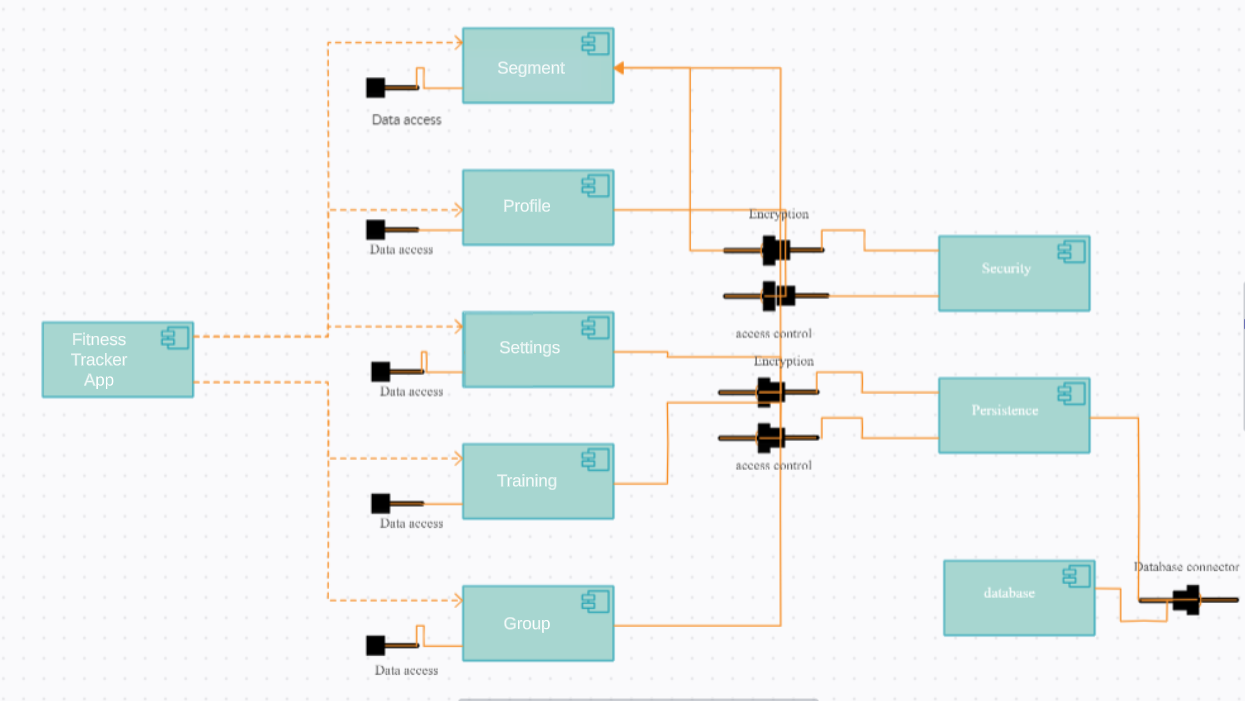


Рисунок 2.1 — Диаграмма компонентов (Component diagram)

# **3 ДИАГРАММА РАЗВЕРТЫВАНИЯ**

## **3.1 Теоретические сведения**

Диаграмма развёртывания (Deployment Diagram) в [UML](https://ru.wikipedia.org/wiki/UML) моделирует физическое развертывание артефактов на [узлах](https://ru.wikipedia.org/w/index.php?title=Node_(UML)&action=edit&redlink=1). Например, чтобы описать веб-сайт, диаграмма развертывания должна показывать, какие аппаратные компоненты («узлы») существуют (например, веб-сервер, сервер базы данных, сервер приложения), какие программные компоненты («артефакты») работают на каждом узле.

Узлы представляются как прямоугольные параллелепипеды с артефактами, расположенными в них, изображёнными в виде прямоугольников. Узлы могут иметь подузлы, которые представляются как вложенные прямоугольные параллелепипеды. Один узел диаграммы развертывания может концептуально представлять множество физических узлов, таких как кластер серверов баз данных.

Существует два типа узлов: узел устройства и узел среды выполнения

Узлы устройств — это физические вычислительные ресурсы со своей памятью и сервисами для выполнения программного обеспечения, такие как обычные ПК, мобильные телефоны.

Узел среды выполнения — это программный вычислительный ресурс, который работает внутри внешнего узла и который представляет собой сервис, выполняющий другие исполняемые программные элементы [2].

## **3.2 Построение диаграммы**

На рисунке 3.1 диаграмма развертывания, на которой изображены основный компоненты для запуска приложения на мобильном устройстве.

Необходимо подготовить загрузочный файл c расширением «.apk» и смартфон, c операционной системой Android. Для корректного запуска приложения, необходимо также свободное пространство.

Устройство будет взаимодействовать с внешним хранилищем, а также с помощью сервиса Firebase будет возможность посредством google-аккаунта, для корректной работы которой, необходимо подключение к сети.

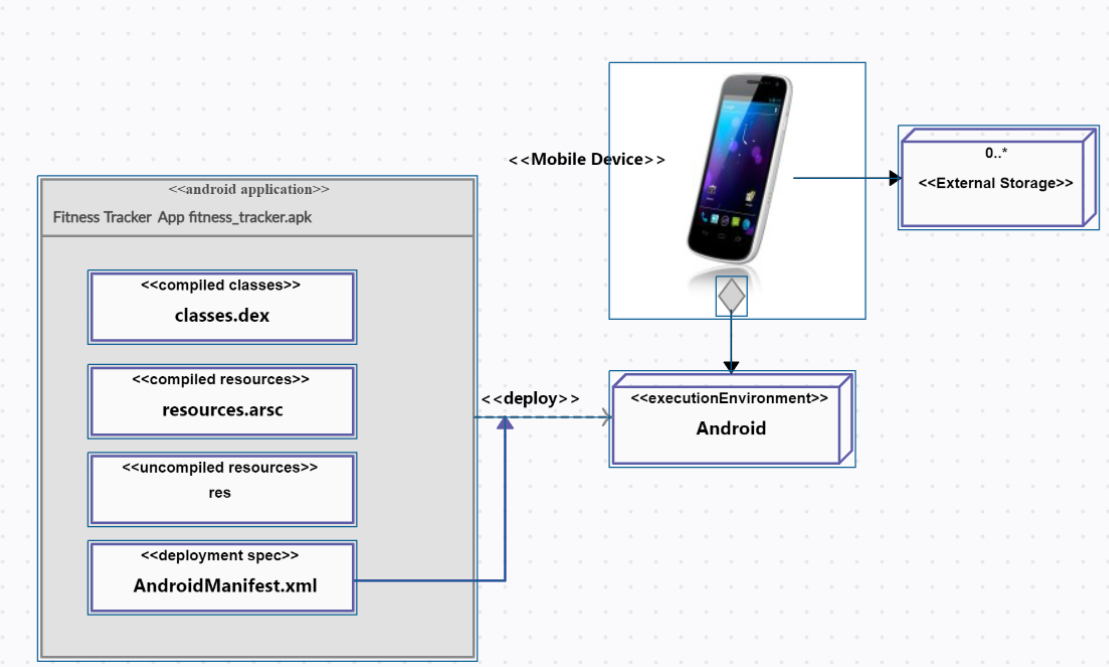


Рисунок 3.1 — Диаграмма развёртывания (Deployment diagram)

# **4 ДИАГРАММА ОБЪЕКТОВ**

## **4.1 Теоретические сведения**

Диаграмма объектов (Object Diagram) в языке моделирования [UML](https://ru.wikipedia.org/wiki/UML) предназначена для демонстрации совокупности моделируемых объектов и связей между ними в фиксированный момент времени.

Диаграмма объектов описывает конкретные экземпляры объектов и напрямую соотносится с [диаграммой классов](https://ru.wikipedia.org/wiki/%D0%94%D0%B8%D0%B0%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B0_%D0%BA%D0%BB%D0%B0%D1%81%D1%81%D0%BE%D0%B2), которая даёт общее представление о конфигурации системы. Она используется для документирования структур данных и создания статических снимков состояний объектов принимая во внимание реальные экземпляры или прототипы. Динамику поведения объектов обычно изображают в виде последовательности таких диаграмм [3].

## **4.2 Построение диаграммы**

На рисунке 4.1 изображена диаграмма объектов в конкретный момент времени: создано 2 объекта пользователей, каждый из которых прошел авторизацию; у пользователя 1 создано 2 тренировки, а у пользователя 2 они отсутствуют; также создан объект группа между ними.

У каждой тренировки есть сегменты, каждый из которых имеют свои идентификаторы, время и расстояние.

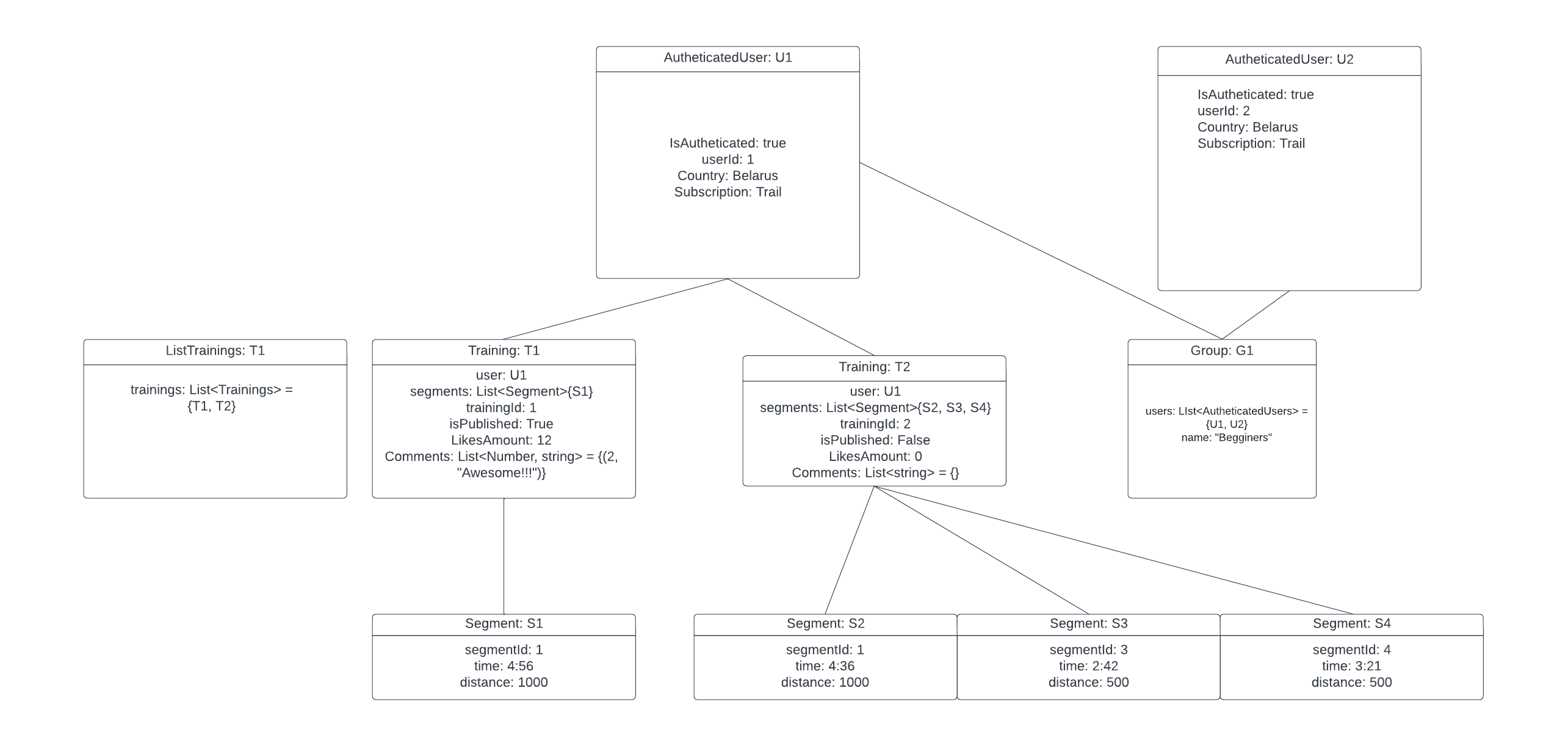


Рисунок 4.1 — Диаграмма объектов (Object Diagram)

# **5 ДИАГРАММА СВЯЗЕЙ**

## **5.1 Теоретические сведения**

Диаграмма связей — инструмент, позволяющий выявить логические связи между основной идеей, проблемой или различными данными.

Задачей этого инструмента управления является установление соответствия основных причин нарушения процесса, выявленных с помощью диаграммы сродства или диаграммы шести слов, тех проблем, которые требуют решения. Между диаграммой связей и диаграммой «рыбьи кости» есть некоторое сходство. Классификация причин нарушения процесса по важности осуществляется с учетом используемых в компании ресурсов, а также числовых данных, характеризующих причины [4].

## **5.2 Построение диаграммы**

На рисунке 5.1 изображена диаграмма связей (Relationship Diagram), которая отображает основную проблему. Для решения конкретной проблемы и создается программный продукт.

Центральной проблемой является неудобство ведения и отслеживания статистики во время и после тренировок. Если рассмотреть проблему более подробно, то основными неудобствами является необходимость дополнительных материалов для записи, отслеживание временных интервалов и ведения статистических данных, что требует довольно больших трудозатрат.

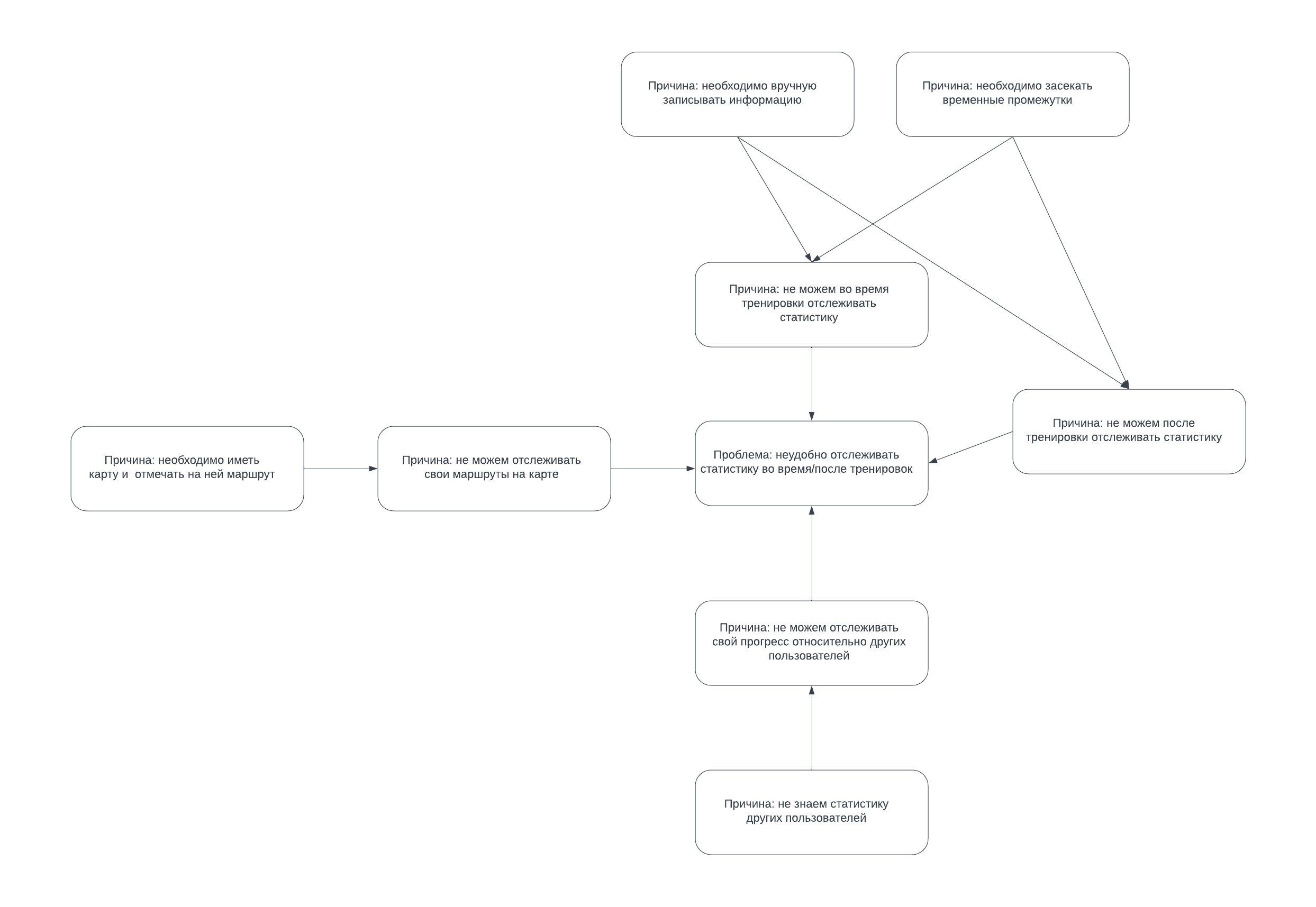


Рисунок 5.1 — Диаграмма связей (Relationship Diagram)

# **6 ДИАГРАММА ПАКЕТОВ**

## **6.1 Теоретические сведения**

Диаграммы пакетов унифицированного языка моделирования([UML](https://ru.wikipedia.org/wiki/UML)) отображают зависимости между [пакетами](https://ru.wikipedia.org/wiki/%D0%9F%D0%B0%D0%BA%D0%B5%D1%82_(UML)), составляющими модель. В дополнение к стандартным отношениям зависимостей в UML есть два специальных вида зависимостей, определенных между пакетами: импортирование пакета и слияние пакета.

Импортирование пакета — это отношение между импортирующим пространством имен и пакетом, указывающим на то, что импортирующее пространство имен добавляет имена членов пакета в их собственное пространство имен. По умолчанию, непомеченная зависимость между двумя пакетами интерпретируется как отношение «импорт пакета».

Слияние пакета — направленное отношение между двумя пакетами, которое указывает, что содержимое двух пакетов должно быть объединено. Это очень похоже на «Обобщение» в том смысле, что исходный элемент как бы добавляет характеристики целевого элемента к своим собственным характеристикам, в результате чего получается элемент, который сочетает в себе характеристики обоих элементов [5].

## **6.2 Построение диаграммы**

На рисунке 6.1 изображена диаграмма сторонних пакетов. При разработке программного продукта будут использоваться вспомогательные пакеты для взаимодействия с Firebase, для отслеживания местоположения и использования GPS.

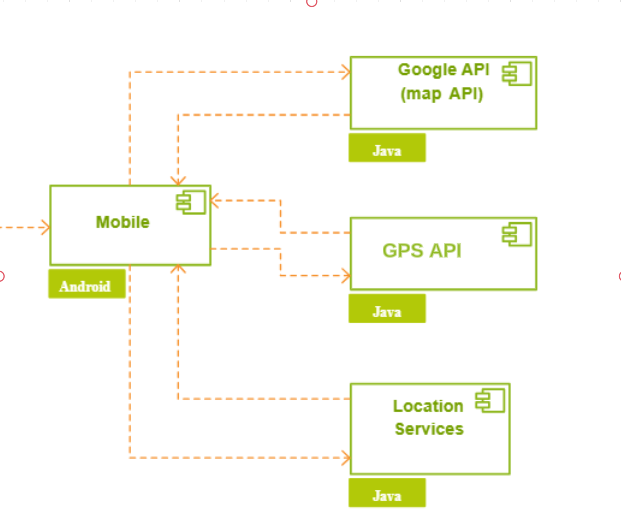


Рисунок 6.1 — Диаграмма пакетов (Package Diagram)

# **7 ДИАГРАММА ВРЕМЕНИ**

## **7.1 Теоретические сведения**

Переходы от одного состояния к другому представлены изменением уровня линии жизни. В течение периода времени, когда объект находится в заданном состоянии, временная шкала движется параллельно этому состоянию. Изменение состояния проявляется как вертикальное изменение с одного уровня на другой. Причиной изменения, как и в случае с диаграммой состояний или последовательностей, является получение сообщения, событие, вызывающее изменение, состояние внутри системы или даже просто течение времени.

Основные элементы временной диаграммы UML — линия жизни, временная шкала, состояние или состояние, сообщение, ограничение продолжительности, линейка времени [6].

## **7.2 Построение диаграммы**

На рисунках 7.1 и 7.2 изображена диаграмма времени для обработки данных при регистрации или входе в систему пользователя. Как можно увидеть, при отсылке данных происходит небольшая задержка, далее данные отправляются на валидацию, а после выполняется запрос в базу данных для проверки наличия пользователя с таким email и паролем или же создается новый пользователь.

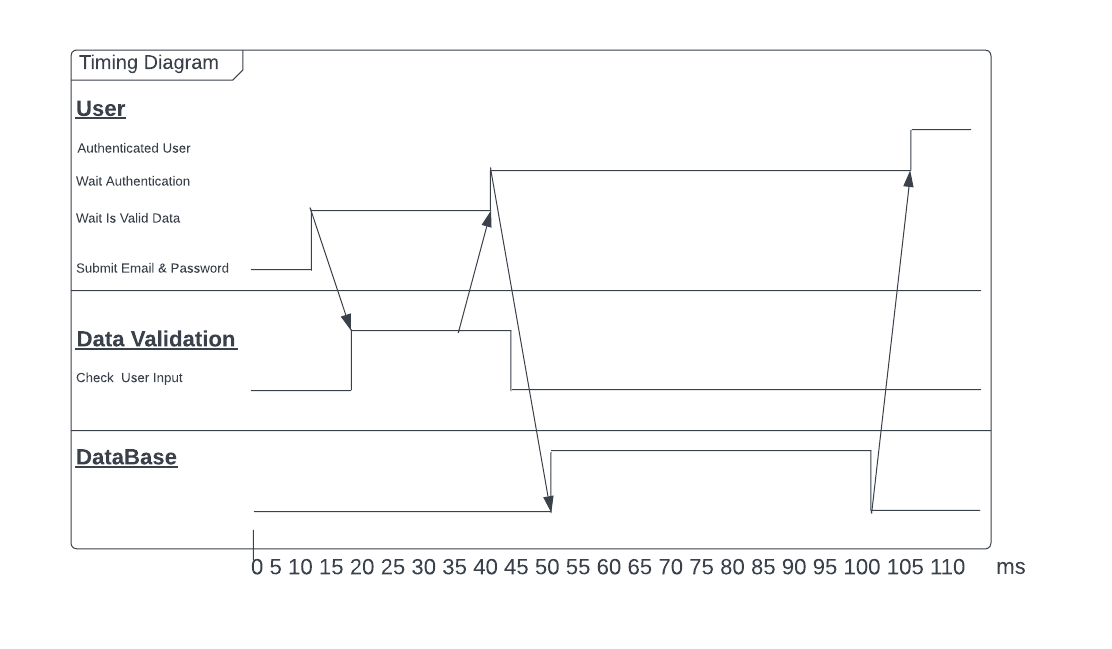


Рисунок 7.1 — Диаграмма времени входа/регистрации (Timing Diagram)

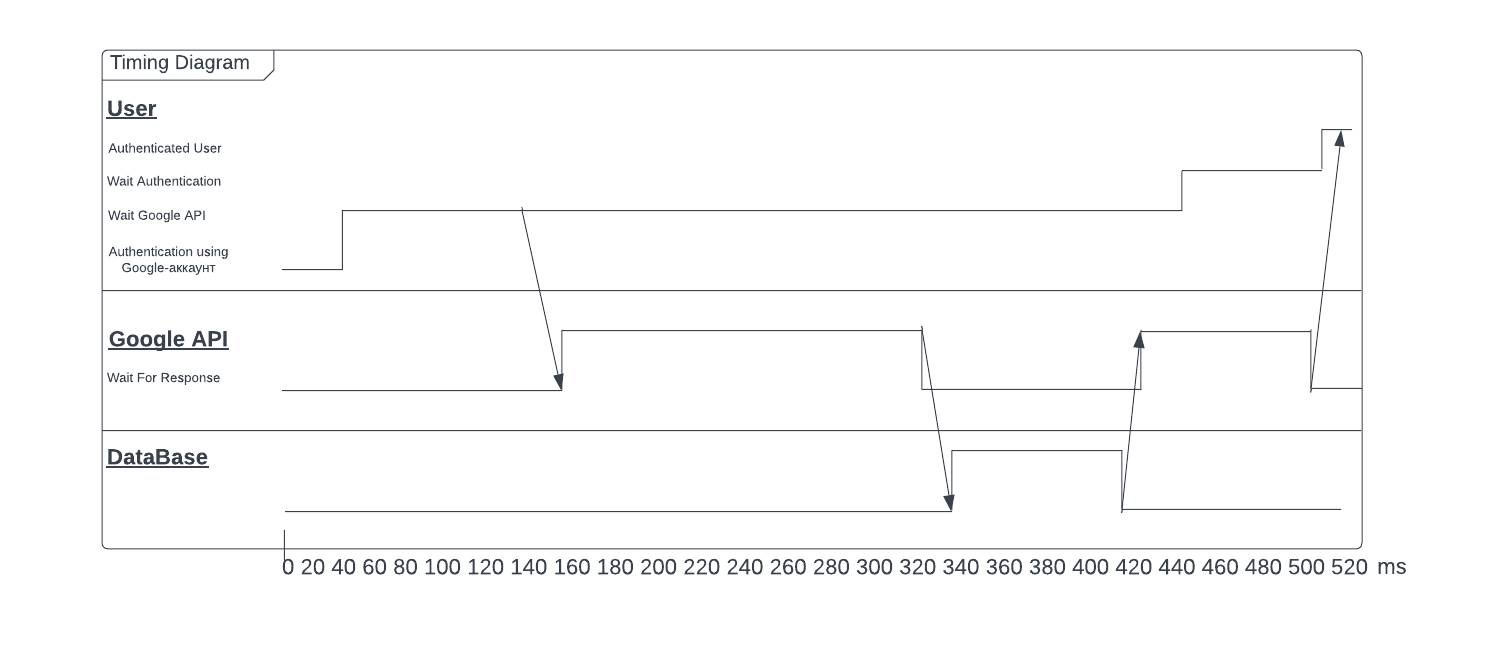


Рисунок 7.2 — Диаграмма времени входа/регистрации с помощью google-аккаунта (Timing Diagram)

# **8 ДИАГРАММА ПРОФИЛЯ**

## **8.1 Теоретические сведения**

Диаграмма профиля — это, по сути, механизм расширяемости, который позволяет расширять и настраивать UML, добавляя новые строительные блоки, создавая новые свойства и определяя новую семантику, чтобы сделать язык подходящим для вашей конкретной предметной области [7].

## **8.2 Построение диаграммы**

На рисунке 8.1 изображена диаграмма профиля. Так как область не подразумевает сложной логики, которую необходимо выносить в компоненты и пояснять, на диаграмме представлены основные ограничения на типы, которые присутствуют в атрибутах классов.

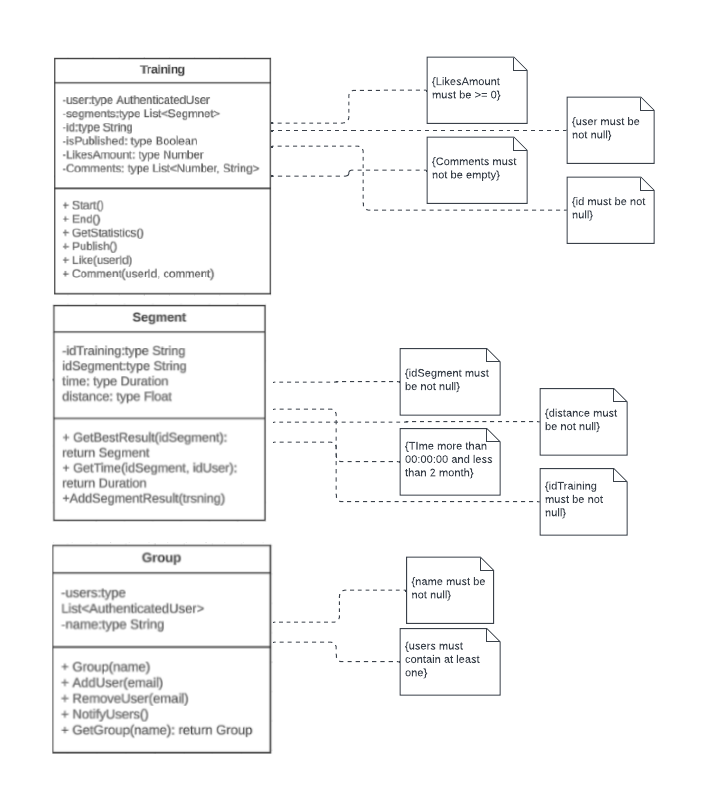


Рисунок 8.1 — Диаграмма профиля (Profile Diagram)

# **ЗАКЛЮЧЕНИЕ**

В результате выполнения лабораторной работы были изучены и разработаны: диаграмма компонентов (Component Diagram), диаграмма развертывания (Deployment Diagram), диаграмма объектов (Object Diagram), диаграмма связей (Relationship Diagram), диаграмма пакетов (Package Diagram), диаграмма времени (Timing Diagram), диаграмма профиля (Profile Diagram). А также подготовлен отчет о проделанной работе.

# **СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ**

1. Википедия [Электронный ресурс]: Википедия — Диаграмма компонентов. Режим доступа: <https://en.wikipedia.org/wiki/Диаграмма_компонентов> (дата обращения: 08.10.2023).
2. Википедия [Электронный ресурс]: Википедия — Диаграмма развертывания. Режим доступа: <https://en.wikipedia.org/wiki/Диаграмма_развертывания> (дата обращения: 08.10.2023).
3. Википедия [Электронный ресурс]: Википедия — Диаграмма активности. Режим доступа: <https://en.wikipedia.org/wiki/Диаграмма_>объектов (дата обращения: 08.10.2023).
4. Studfile [Электронный ресурс]: Studfile — Диаграмма связей. Режим доступа: <https://studfile.net/preview/8321952/page:23/> (дата обращения: 08.10.2023).
5. Википедия [Электронный ресурс]: Википедия — Диаграмма классов. Режим доступа: <https://en.wikipedia.org/wiki/Диаграмма_пакетов> (дата обращения: 08.10.2023).
6. Visual Paradigm [Электронный ресурс]: Visual Paradigm — What is Timing Diagram in UML? Режим доступа: <https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-timing-diagram/> (дата обращения: 08.10.2023).
7. Visual Paradigm [Электронный ресурс]: Visual Paradigm — What is Profile Diagram in UML? Режим доступа: <https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-profile-diagram/> (дата обращения: 08.10.2023).