Московский автомобильно-дорожный государственный технический университет (МАДИ)
Кафедра «Высшая математика»

Отчет

по дисциплине

«Проектирование информационных систем»

«Мобильное приложение для университета МАДИ»

Выполнил: студент группы ЗБПМ
Осада В.В.
Акилин Я.А.

Принял: Кутейников И.А.

1 №1Техническое задание на разработку мобильного приложения «Расписание МАДИ»

Введение

Целью настоящего технического задания является определение основных требований и особенностей разработки мобильного приложения «Расписание МАДИ», предназначенного для использования студентами и преподавателями Московского автомобильно-дорожного государственного технического университета (МАДИ).

1.1 Основания для разработки

1.1.1 Исходные данные

Разработка приложения осуществляется на основании потребностей университета в улучшении информационного взаимодействия между студентами и преподавателями, а также необходимости обеспечения удобного доступа к актуальному расписанию занятий и экзаменов.

1.2 Требования к программе

1.2.1 Функциональные требования

Программа должна обеспечивать следующие функции:

- Просмотр расписания занятий и экзаменов.
- Уведомления о предстоящих парах и изменениях в расписании.
- Возможность преподавателям загружать и обновлять материалы курсов и лекций.
- Доступ к электронной библиотеке учебных материалов.

1.2.2 Нефункциональные требования

- Кроссплатформенность: приложение должно быть доступно для операционной системы iOS.
- Интуитивно понятный интерфейс, адаптированный под устройства с различными размерами экранов.
- Высокая производительность и стабильность работы приложения.
- Обеспечение безопасности персональных данных пользователей.

1.3 Требования к документации

1.4 Предварительный состав технической документации

В процессе разработки должны быть подготовлены следующие документы в соответствии с ГОСТ 19.101-77:

1. Техническое задание (ГОСТ 19.201-78)
2. Рабочий проект (ГОСТ 19.301-78)
3. Программа и методика испытаний (ГОСТ 19.301-78)

4. Руководство оператора (ГОСТ 19.505-79)
5. Текст программы (ГОСТ 19.401-78)
6. Описание программы (ГОСТ 19.402-78)

1.5 Технико-экономические показатели

1.6 Предполагаемая экономическая эффективность

Разработка приложения должна способствовать повышению удобства доступа к учебным материалам, снижению времени на поиск необходимой информации и улучшению качества образовательного процесса.

1.7 Стадии и этапы разработки

1.7.1 Техническое задание

1. Разработка концепции приложения
2. Согласование и утверждение технического задания

1.7.2 Рабочий проект

1. Разработка архитектуры приложения
2. Проектирование пользовательского интерфейса
3. Программирование и отладка приложения
4. Тестирование приложения

1.7.3 Внедрение

1. Подготовка и проведение испытаний
2. Корректировка программы и документации по результатам испытаний
3. Сдача приложения заказчику
4. Обучение персонала

1.8 Порядок контроля и приемки

1.8.1 Общие положения

Контроль качества и приемка разработанного программного продукта осуществляются в соответствии с программой и методикой испытаний, утвержденными вместе с техническим заданием.

1.8.2 Порядок проведения приемочных испытаний

Приемочные испытания проводятся заказчиком в присутствии исполнителя на основании утвержденной программы и методики испытаний.

1.8.3 Передача программного продукта заказчику

Передача программного продукта заказчику осуществляется после подписания акта о приемке программного продукта, составленного по результатам приемочных испытаний.

2 №2 Разработка календарного плана проекта

Календарный план проекта разработки мобильного приложения для университета МАДИ включает в себя несколько основных этапов, каждый из которых имеет чётко определённые сроки и распределение ответственности между участниками команды.

2.1 Этапы работы и распределение задач

1. **Исследование (Осада.В.В)**: 12.09.2023 - 21.09.2023. На этом этапе проводится анализ требований пользователей, исследование существующих аналогов и сбор идей для реализации функционала приложения.

2. **Проектирование (Акилин.Я.А)**: 22.09.2023 - 01.10.2023. Проектирование архитектуры приложения, разработка дизайн-макетов интерфейса и определение технологического стека.

3. **Разработка (Грицук.М.В)**: 02.10.2023 - 21.10.2023. Написание кода приложения, реализация функционала согласно техническому заданию и проектированию.

4. **Тестирование (Таскаев.В.И)**: 22.10.2023 - 05.11.2023. Тестирование разработанного приложения, выявление и устранение ошибок, проверка соответствия требованиям.

5. **Развертывание (Осада.В.В)**: 06.11.2023 - 15.11.2023. Подготовка к запуску, развертывание приложения на серверах университета и публикация в App Store.

2.2 Визуализация плана

Ниже представлена диаграмма Ганта, иллюстрирующая распределение и последовательность этапов работы над проектом:

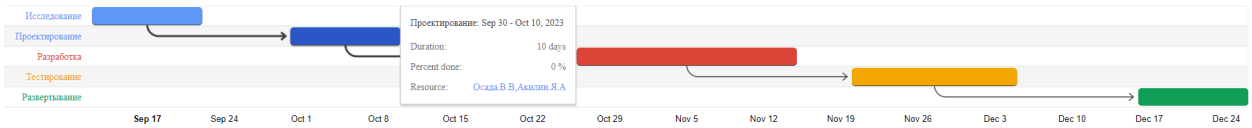

Рис. 1: Диаграмма Ганта проекта

3 №3 Построение модели данных

Модель данных представляет собой структуру информационной системы университета, включающую в себя сущности и связи между ними. Каждая сущность описывается набором атрибутов и может быть связана с другими сущностями с помощью отношений.

Сущности и их связи:

- **Студенты**: У каждого студента есть уникальный идентификатор (ID) и личные данные (ФИО). Студенты связаны с группами по отношению "многие к одному".

- **Группы**: Группы имеют уникальный ID и название. Каждая группа может иметь множество студентов и связана с расписанием по отношению "одна к многим".
- **Преподаватели**: Преподаватели идентифицируются по ID и ФИО. Они связаны с предметами, которые преподают, по отношению "многие к многим и с расписанием по отношению "один к многим".
- **Расписание**: В расписании указывается, когда и какие предметы будут преподаваться. Оно связывает преподавателей, группы и предметы.
- **Предметы**: Каждый предмет имеет ID, название и описание. Предметы связаны с экзаменами и расписанием по отношению "один к многим".
- **Экзамены**: Экзамены содержат информацию о ID, дате проведения и связанном предмете. Они связаны с предметами и группами по отношению "многие к одному".
- **Материалы курса**: Включают ID, название и тип материала. Они ассоциированы с предметами, по которым предоставляются, по отношению "многие к одному".

Диаграмма ниже иллюстрирует связи между сущностями:

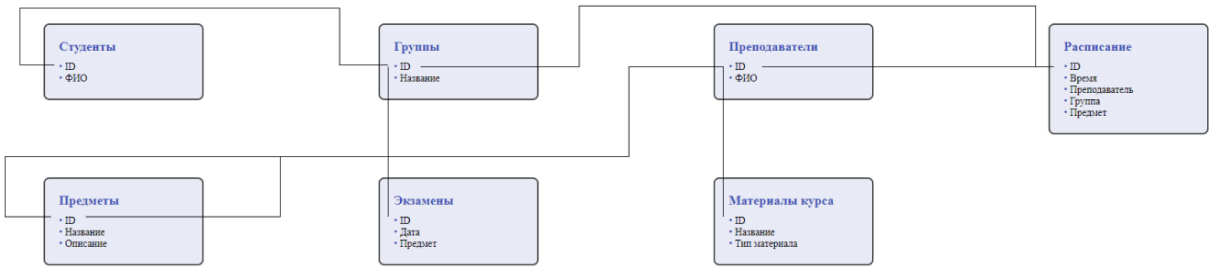

Рис. 2: ER-диаграмма информационной системы университета

№4 Выгрузка модели данных в СУБД и реверс инжиниринг БД

В рамках данного этапа работы была выполнена выгрузка разработанной модели данных в систему управления базами данных (СУБД). Для демонстрации структуры базы данных и отношений между таблицами использовался процесс реверс-инжиниринга, который позволил визуализировать схему данных в виде ER-диаграммы.

Ниже представлена ER-диаграмма, отображающая связи между таблицами в базе данных:

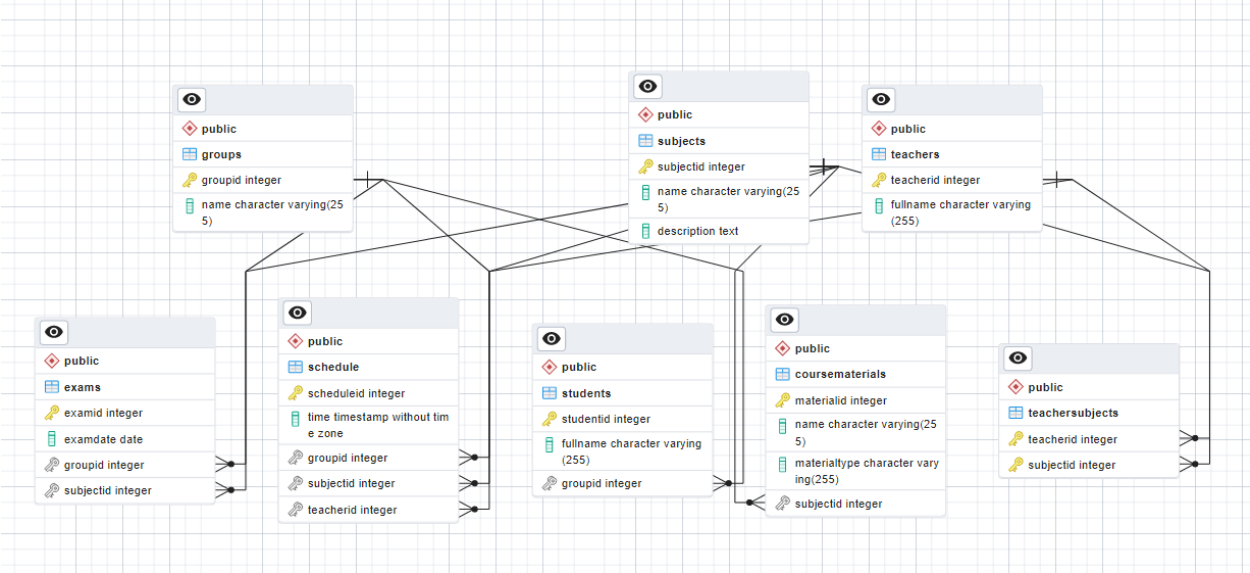

Рис. 3: ER-диаграмма базы данных после реверс-инжиниринга

№5 Построение функциональной модели

3.1 Цель работы

Целью данной работы является разработка функциональной модели мобильного приложения университета, которая позволяет наглядно представить структуру и взаимодействие основных компонентов системы.

3.2 Описание функциональной модели

Представленная функциональная модель иллюстрирует основные компоненты мобильного приложения для университета и их связи. Компоненты включают:

- **App**: Главный модуль, который служит входной точкой для пользователей и координирует взаимодействие между подсистемами.
- **User**: Основной компонент, представляющий акторов системы - студентов и преподавателей.
- **Students и Teachers**: Подсистемы, наследующие функции от компонента User и расширяющие их специфическими возможностями для каждой группы пользователей.
- **ScheduleItem**: Компонент, отвечающий за управление расписанием занятий.

- **Subject**: Модуль, который содержит информацию о предметах, доступных в приложении.
- **Exam**: Компонент, управляющий экзаменационной деятельностью в приложении.
- **CourseMaterial**: Раздел, в котором преподаватели могут размещать учебные материалы для студентов.
- **Group**: Элемент, представляющий учебные группы в системе.

3.3 Связи между компонентами

Каждый компонент взаимодействует с App напрямую или через другие компоненты. Пользователи (Students и Teachers) могут получать и изменять информацию о расписаниях, предметах, экзаменах и учебных материалах, в то время как компонент Group связан с ScheduleItem для предоставления информации о расписании конкретной группы.

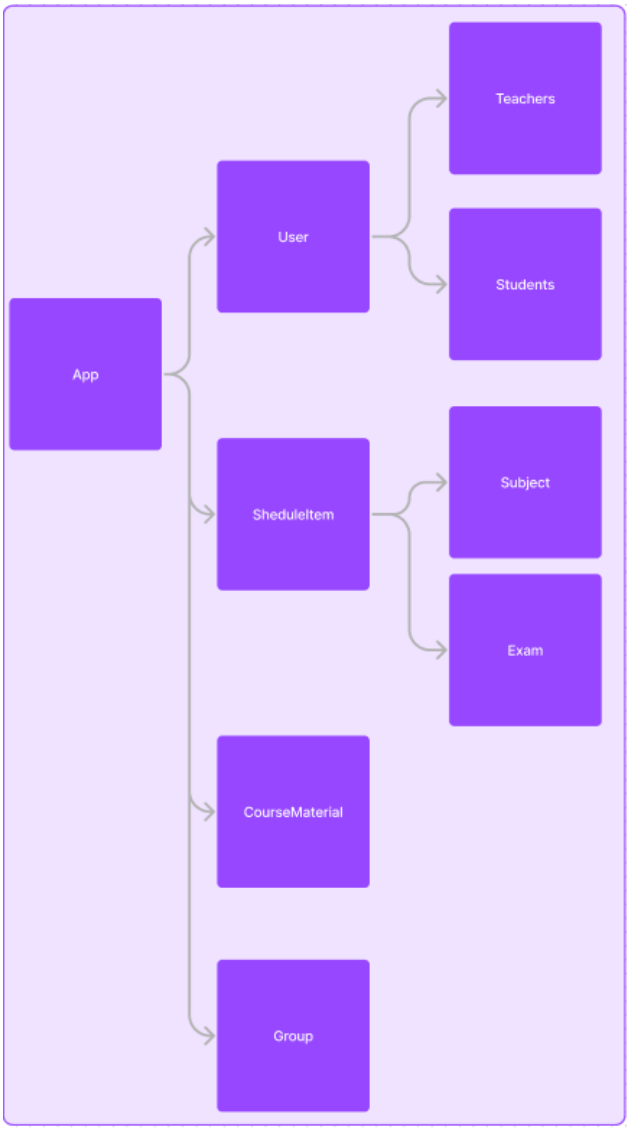

Рис. 4: Функциональная модель

4 №6 Построение процессной модели

4.1 Цель работы

Разработка процессной модели мобильного приложения для университета МАДИ, позволяющей детализировать и визуализировать потоки взаимодействия между различными видами представлений (views) в приложении.

4.2 Описание процессной модели

Процессная модель мобильного приложения демонстрирует потоки переходов между различными представлениями, доступными пользователям. Ключевые представления включают:

- **LoginView**: Экран входа, через который пользователи получают доступ к функциям приложения после аутентификации.
- **RegistrationView**: Экран регистрации для создания нового пользовательского аккаунта.
- **ScheduleView**: Представление расписания, где пользователи могут просматривать актуальное расписание занятий.
- **GroupView**: Представление, которое позволяет пользователям просматривать информацию о группах и их расписаниях.
- **ExamView**: Экран, на котором студенты могут узнать информацию о предстоящих экзаменах.
- **ProfileView**: Представление профиля пользователя, где можно обновить личные данные.
- **MaterialView и MaterialListView**: Экраны для просмотра учебных материалов. Если пользователь является преподавателем, он может перейти к MaterialUploadView для загрузки новых материалов.

4.3 Логика переходов

Пользователи могут переходить между различными экранами в зависимости от своих потребностей и статуса (студент или преподаватель). Например, после успешного входа в систему через **LoginView**, студенты направляются на **ScheduleView**, а преподаватели могут перейти к загрузке материалов через **MaterialUploadView**.

4.4 Анализ модели

Процессная модель позволяет определить и оптимизировать потоки пользовательского опыта в приложении. Она подчеркивает удобство навигации и доступность функций приложения для различных пользовательских ролей.

5 №7 Построение модели потоков данных

5.1 Цель работы

Разработка модели потоков данных для мобильного приложения университета МАДИ, позволяющей визуализировать и анализировать передачу информации между различными акторами и системами.

5.2 Описание модели потоков данных

Модель потоков данных показывает взаимодействие между участниками образовательного процесса и информационной системой. Включает в себя следующие элементы:

- **Teacher**: Преподаватель загружает учебные материалы и взаимодействует с базой данных для управления курсами и оценками студентов.
- **Student**: Студент имеет доступ к расписанию, учебным материалам и своему профилю через мобильное приложение.
- **PostgreSQL**: Центральная база данных, которая хранит всю информацию, связанную с учебным процессом, включая учебные материалы, расписание и профили пользователей.
- **DevOps**: Инструменты и процессы, используемые для непрерывной интеграции и развертывания приложения, что обеспечивает актуальность данных и доступность сервиса.

5.3 Потоки данных

Потоки данных включают запросы и обновления, производимые преподавателями и студентами:

- **Material Upload**: Преподаватели загружают материалы, которые затем хранятся в базе данных PostgreSQL.
- **Request**: Студенты отправляют запросы на получение данных, таких как расписание и учебные материалы, из базы данных.

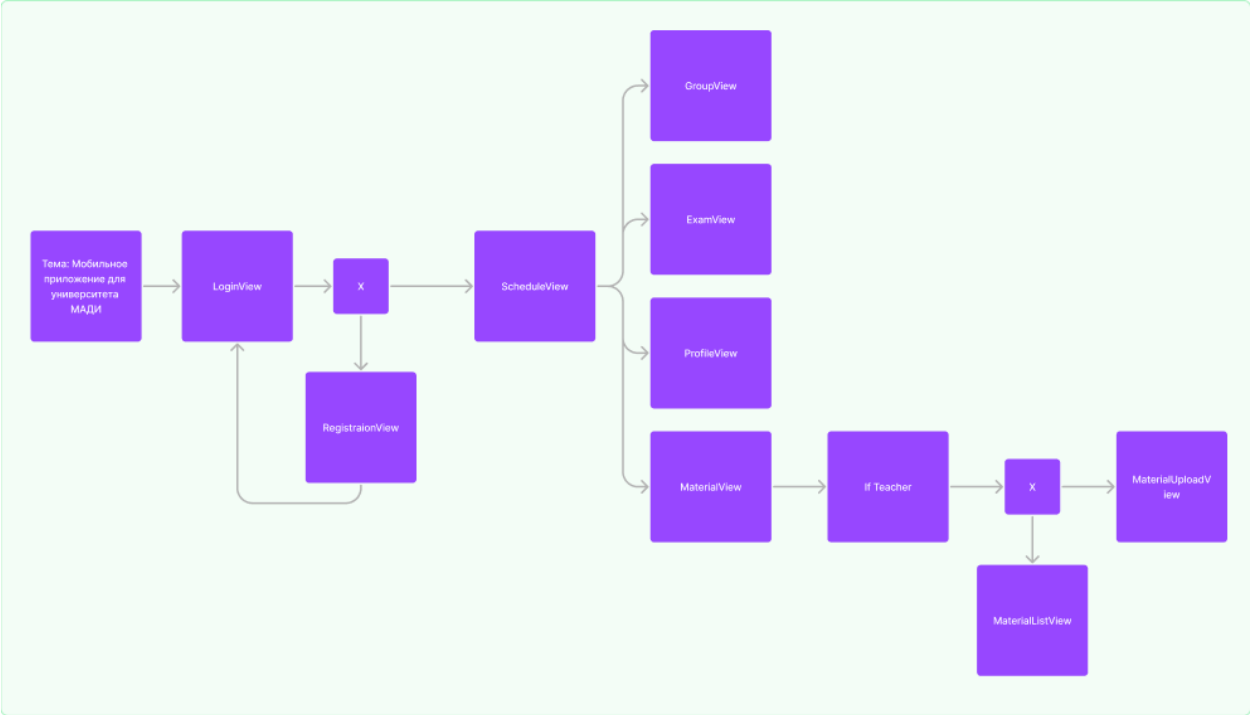

Рис. 5: Процессная модель

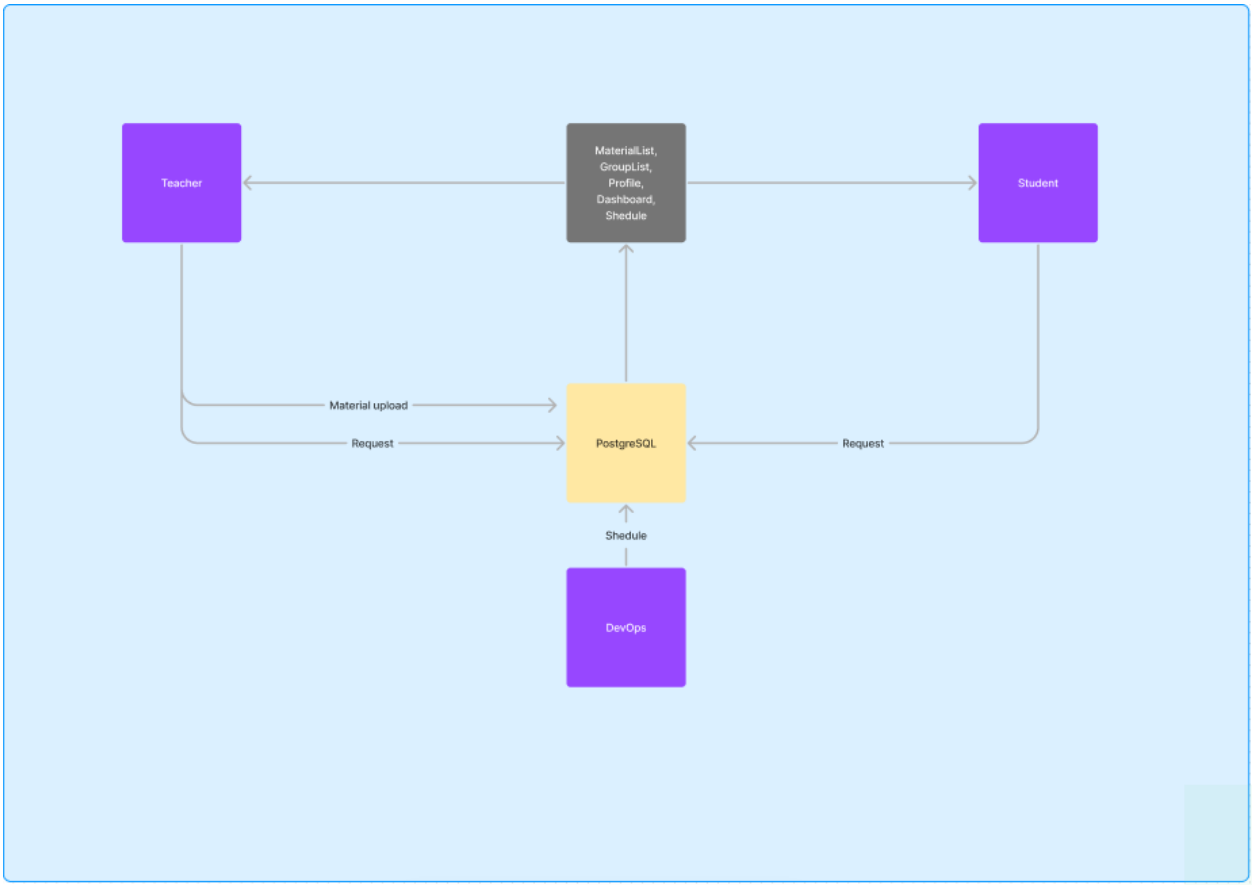

Рис. 6: Модель потоков данных

№9 UML. Диаграмма классов

5.4 Цель работы

Применение диаграммы классов для описания структуры выбранного решения и ознакомление с инструментами, позволяющими строить диаграмму классов.

5.5 Задание

В соответствии с заданием лабораторной работы №3 разработать диаграмму классов для описания структуры выбранного решения. Диаграмма классов должна чётко отражать все основные компоненты системы, их атрибуты, методы и взаимосвязи.

5.6 Методология

Для разработки диаграммы классов были использованы следующие принципы объектно-ориентированного проектирования:

- Инкапсуляция данных и поведения в классах.
- Абстракция для определения ключевых характеристик классов.
- Наследование для создания иерархии классов и переиспользования кода.
- Полиморфизм для использования объектов производных классов через интерфейс базового класса.

5.7 Результаты

В результате были идентифицированы и описаны ключевые классы системы, включая *User*, *Student*, *Teacher*, и другие. Были определены основные атрибуты и методы каждого класса, а также отношения между классами, такие как ассоциации, зависимости и наследование.

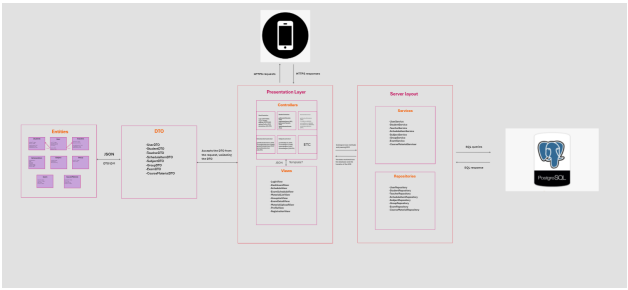

Рис. 7: UML диаграмма классов

6 №10 UML. Диаграммы объектов

6.1 Цель работы

Целью данной работы является применение диаграммы объектов UML для наглядного представления структуры выбранного программного решения, показывающей объекты системы, их атрибуты и взаимосвязи в определённый момент времени.

6.2 Задание

На основе результатов лабораторной работы №9 по созданию диаграммы классов разработать диаграмму объектов, которая детализирует взаимодействие между экземплярами классов в рамках выбранной системы. Диаграмма объектов должна включать экземпляры таких классов, как *User*, *Student*, *Teacher*, *ScheduleItem*, *Subject*, *Group*, *Exam* и *CourseMaterial*.

6.3 Методика выполнения

Для каждого класса были созданы уникальные объекты с их атрибутами и отношениями. В качестве примера, объект класса *Student* может быть представлен следующим образом:

- student1: *Student*
  - studentID: 1001
  - groupID: 501
  - username: "OsadaV"
  - password: "jd_secure123"
  - role: "student"
  - ...

Аналогичные объекты созданы для других классов, отражая их реальные экземпляры в рамках системы.

6.4 Результаты

Была разработана диаграмма объектов, позволяющая визуализировать статическую структуру системы на примере конкретных объектов. Она демонстрирует, как объекты могут быть созданы, какие данные они хранят и как между ними происходит взаимодействие.

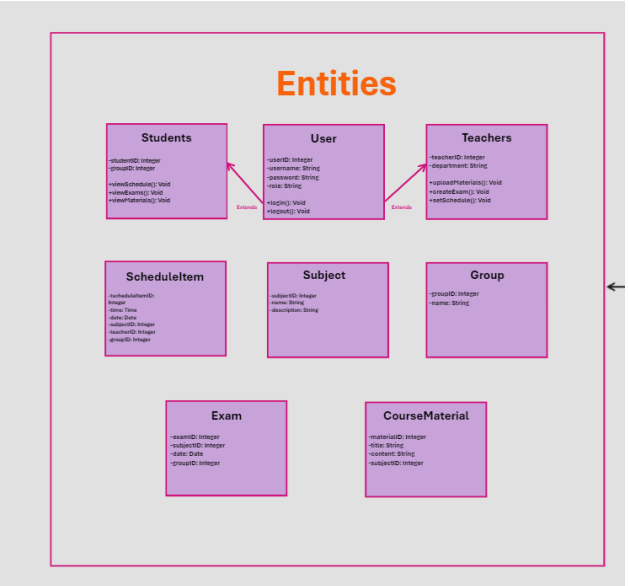

Рис. 8: UML диаграмма обьектов

7 №11 UML. Диаграмма вариантов использования

7.1 Цель работы

Целью данной работы является использование диаграммы вариантов использования для наглядного представления функциональных требований к разрабатываемой системе, а также ознакомление с инструментарием для создания UML диаграмм.

7.2 Описание диаграммы вариантов использования

Диаграмма вариантов использования для системы мобильного приложения университета позволяет наглядно представить ключевые функции, доступные двум основным актерам системы: студентам и преподавателям.

- **Студент** может выполнять следующие операции:
  - Регистрация/вход в систему
  - Обновление профиля
  - Общение в чате
  - Просмотр оценок
  - Просмотр расписаний
  - Просмотр учебных материалов
  - Просмотр экзаменов
- **Преподаватель** имеет возможность:
  - Регистрация/вход в систему
  - Обновление профиля
  - Общение в чате
  - Создание экзаменов
  - Загрузка учебных материалов
  - Оценка студентов
  - Управление расписанием занятий

7.3 Сценарии использования

7.3.1 Регистрация/вход в систему

Основной поток событий для сценария регистрации/входа в систему начинается, когда студент или преподаватель выбирает опцию регистрации или входа в приложении. Пользователь вводит необходимые данные, и система обрабатывает их для аутентификации или регистрации нового профиля.

7.3.2 Просмотр расписаний

Студент выбирает опцию просмотра расписаний. Приложение отображает актуальное расписание занятий. В случае отсутствия интернет-соединения пользователю предлагается последнее сохранённое офлайн расписание.

7.3.3 Создание экзаменов

Преподаватель выбирает опцию создания экзамена, вводит необходимые данные и публикует экзамен для группы студентов.

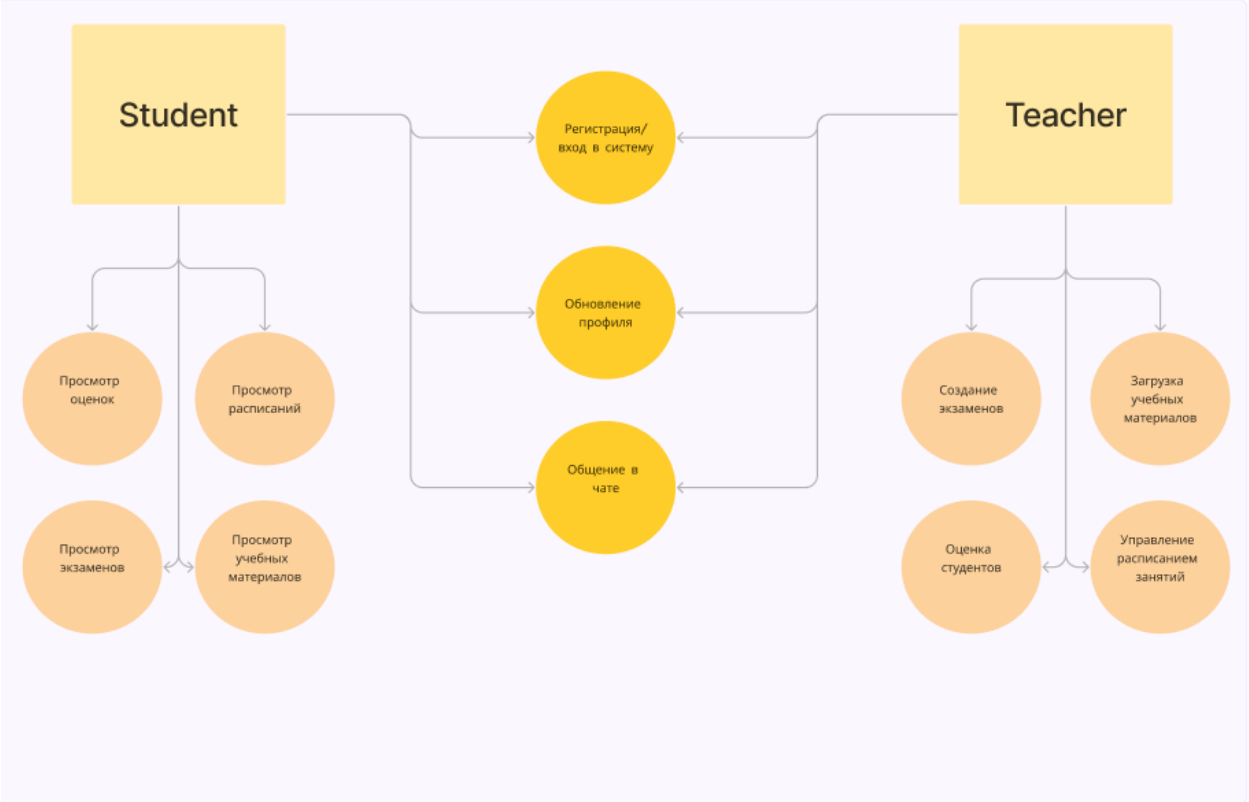

Рис. 9: UML.Диаграмма вариантов использования

8 №12 UML. Диаграмма последовательности

8.1 Цель работы

Применение диаграммы последовательности для детального описания процессов внутри разрабатываемой системы, позволяющее визуализировать порядок взаимодействия между объектами системы.

8.2 Описание диаграммы последовательности

На диаграмме последовательности представлен процесс регистрации пользователя в системе. Диаграмма включает в себя следующие шаги:

1. Пользователь вводит свои данные на форме регистрации и нажимает кнопку "Зарегистрироваться".
2. UI отправляет запрос на `RegistrationController`.
3. `RegistrationController` принимает данные и обращается к `RegistrationService`.
4. `RegistrationService` запрашивает у `UserRepository` добавление нового пользователя.

5. `UserRepository` выполняет запрос к базе данных для сохранения информации.
6. База данных подтверждает сохранение и возвращает подтверждение обратно к `UserRepository`.
7. `UserRepository` возвращает результат в `RegistrationService`.
8. `RegistrationService` передает результат в `RegistrationController`.
9. `RegistrationController` отправляет ответ об успешной регистрации на UI.
10. UI отображает пользователю сообщение об успешной регистрации.

8.3 Анализ диаграммы

Диаграмма последовательности демонстрирует взаимодействие между пользовательским интерфейсом, контроллером, сервисом и репозиторием, а также их связь с базой данных. Это позволяет отследить жизненный цикл запроса пользователя от начала до конца и является основой для понимания и разработки бэкенд-логики системы.

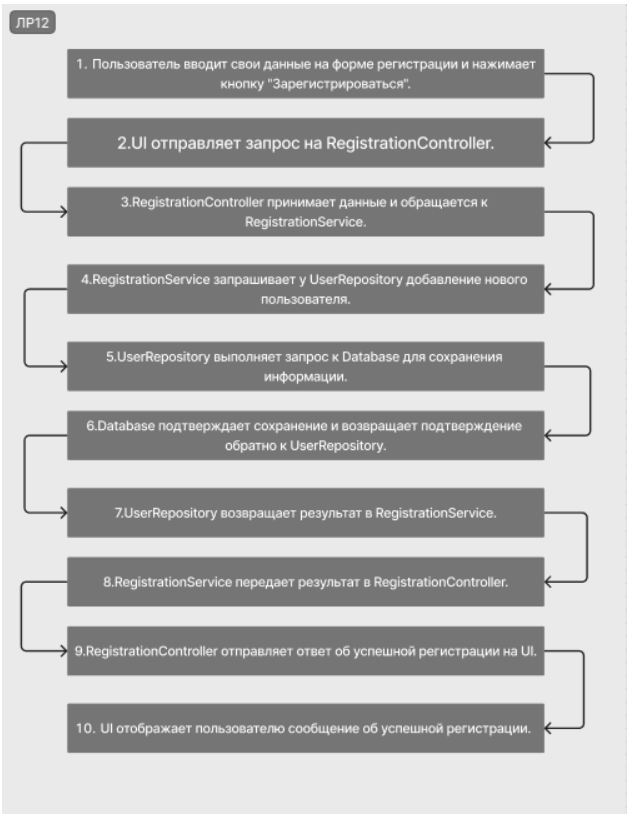

Рис. 10: UML. Диаграмма последовательности

9 №13UML. Диаграмма сотрудничества

9.1 Цель работы

Применение диаграммы сотрудничества для визуализации взаимодействий между различными компонентами системы при процессе регистрации пользователя.

9.2 Описание диаграммы

Диаграмма сотрудничества показывает процесс регистрации пользователя, включая следующие компоненты системы:

- **Пользовательский интерфейс**: Место, где пользователь вводит свои данные для регистрации.
- **Контроллер регистрации**: Получает данные от пользователя и отправляет запрос на регистрацию в сервис регистрации.
- **Сервис регистрации**: Обрабатывает запрос на регистрацию и взаимодействует с репозиторием пользователей.
- **База данных**: Хранит информацию о пользователях и обрабатывает запросы на сохранение новых данных.
- **Репозиторий пользователей**: Служит посредником между сервисом регистрации и базой данных.

Каждый компонент связан со следующим через определенные действия, такие как *Регистрация/Данные*, *Запрос на регистрацию*, *Подтверждение на сохранение данных/ошибка*, и *Взаимодействие данными*.

9.3 Взаимодействия

1. Пользователь вводит данные в пользовательском интерфейсе и инициирует процесс регистрации.
2. Контроллер регистрации принимает данные и отправляет их в сервис регистрации.
3. Сервис регистрации обрабатывает данные и делегирует задачу сохранения информации репозиторию пользователей.
4. Репозиторий пользователей взаимодействует с базой данных для сохранения информации.
5. База данных возвращает результат операции репозиторию пользователей, который, в свою очередь, информирует сервис регистрации.
6. Сервис регистрации отправляет результат контроллеру регистрации.
7. Контроллер регистрации возвращает результат операции в пользовательский интерфейс.
8. Пользовательский интерфейс отображает сообщение об успешной регистрации или об ошибке.

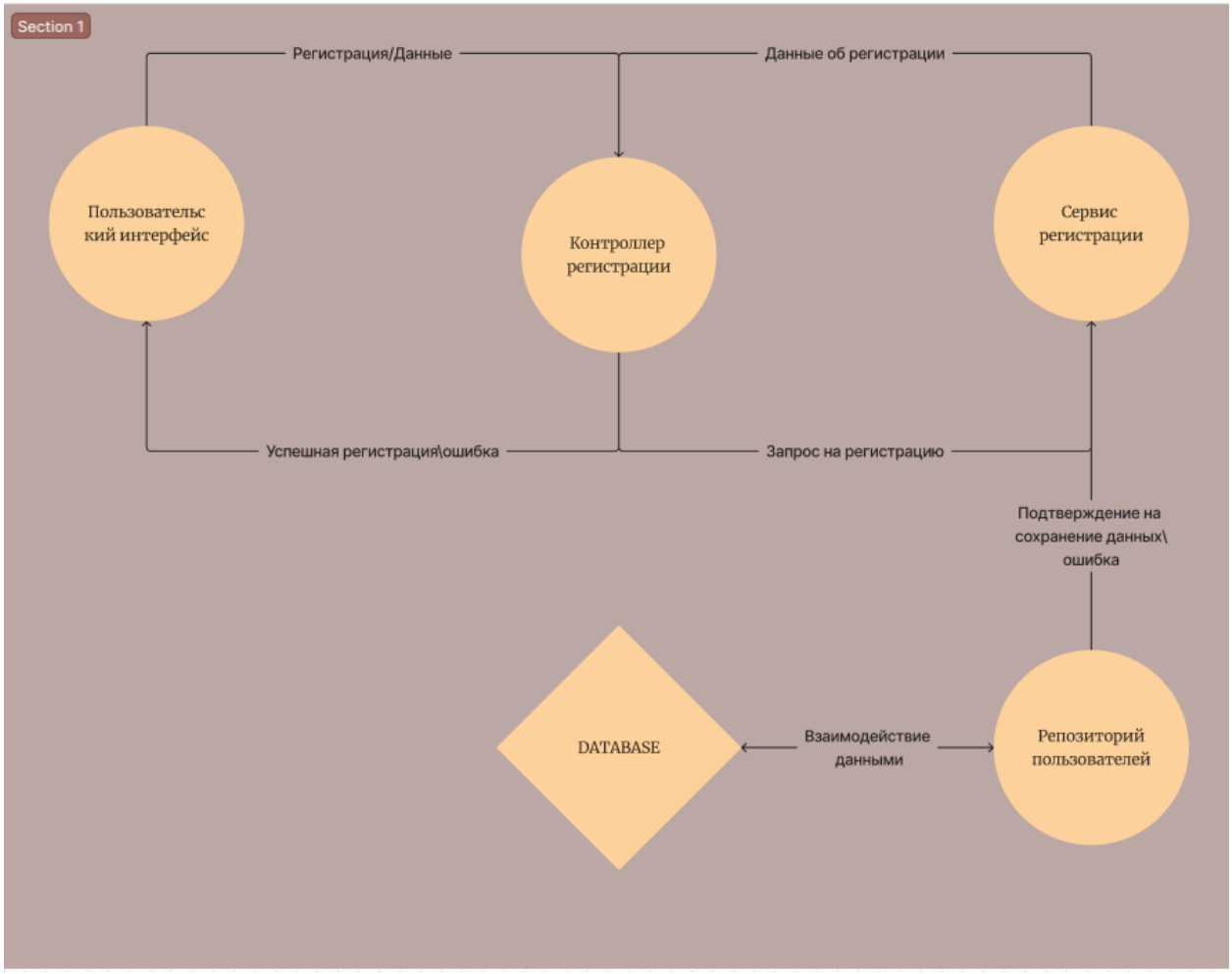

Рис. 11: UML. Диаграмма сотрудничества

10 №14 UML. Диаграмма схем состояний

10.1 Цель работы

Цель данной работы — создание UML диаграммы схемы состояний, отображающей жизненный цикл пользователя в системе мобильного приложения университета МАДИ.

10.2 Описание диаграммы схемы состояний

Диаграмма схемы состояний визуализирует различные состояния, через которые проходит пользователь во время взаимодействия с системой. На диаграмме отображены следующие состояния:

- Начальное состояние, где пользователь начинает взаимодействие с системой.
- Процесс аутентификации, который определяет, является ли пользователь авторизованным или нет.
- Авторизованный пользователь получает доступ к функциям системы, таким как просмотр расписания и учебных материалов.
- Не авторизованный пользователь направляется к процессу аутентификации или к просмотру общедоступной информации.
- Для администраторов и сотрудников БД предусмотрены дополнительные функции, связанные с управлением системой.

10.3 Переходы между состояниями

Переходы между состояниями осуществляются посредством событий, таких как успешная аутентификация или запрос на просмотр данных. Каждый переход четко определен и управляется системой безопасности.

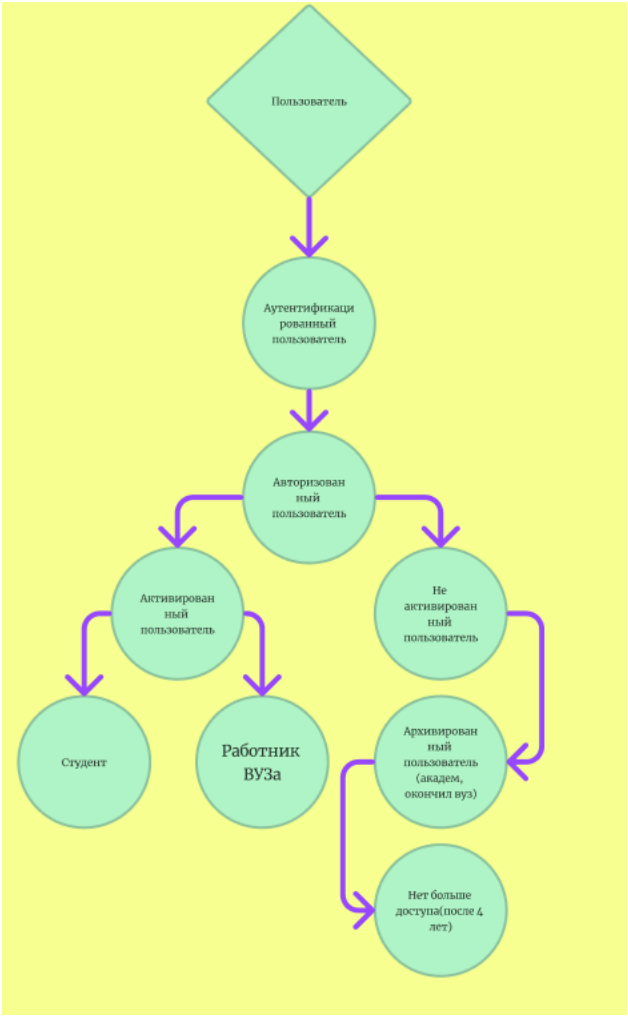

Рис. 12: Диаграмма состояний

11 №15 UML. Диаграмма деятельности

11.1 Цель работы

Целью данной работы является использование UML диаграммы деятельности для описания различных потоков взаимодействия акторов с мобильным приложением университета МАДИ.

11.2 Описание диаграммы

Диаграмма деятельности представляет потоки действий, которые выполняются различными акторами системы. Включает в себя следующие процессы и акторы:

- **Студенты**, выполняющие действия такие как просмотр расписания и материалов, а также сдача заданий.
- **Преподаватели**, занимающиеся загрузкой материалов и оценкой студентов.
- **Административный персонал**, управляющий профилем и вносящий изменения в расписание, а также добавляющий новые функционалы в систему.

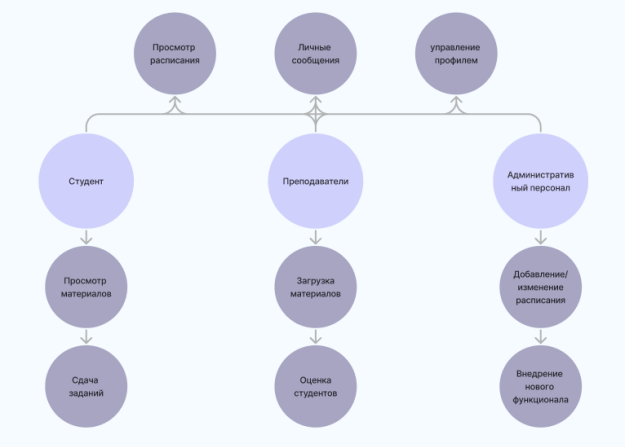

Рис. 13: Диаграмма состояний

12 №16 UML. Компонентная диаграмма

12.1 Цель работы

Целью работы является разработка компонентной диаграммы для мобильного приложения университета МАДИ, отображающей структуру и взаимосвязь между различными модулями и их интерфейсами.

12.2 Описание компонентов

На диаграмме представлены следующие основные компоненты системы:

- Пользователи, разделенные на преподавателей и студентов.
- Специализированные компоненты для каждой группы пользователей, обеспечивающие доступ к соответствующим данным, таким как лекции, бакалавриаты, специальности, аспирантуры и т.д.

- Компоненты данных, такие как расписание, литература, распределение и предметы, которые управляют соответствующей информацией в системе.
- Материалы, связанные с учебным процессом и доступные для пользователей.

12.3 Взаимодействие между компонентами

Каждый компонент системы обеспечивает определенный интерфейс, через который он взаимодействует с другими компонентами:

- Преподаватели и студенты взаимодействуют с системой через предоставленные им компоненты, которые позволяют обращаться к требуемым данным.
- Компоненты данных предоставляют интерфейсы JSON для интеграции с сервером приложений и обработки запросов от пользователей.

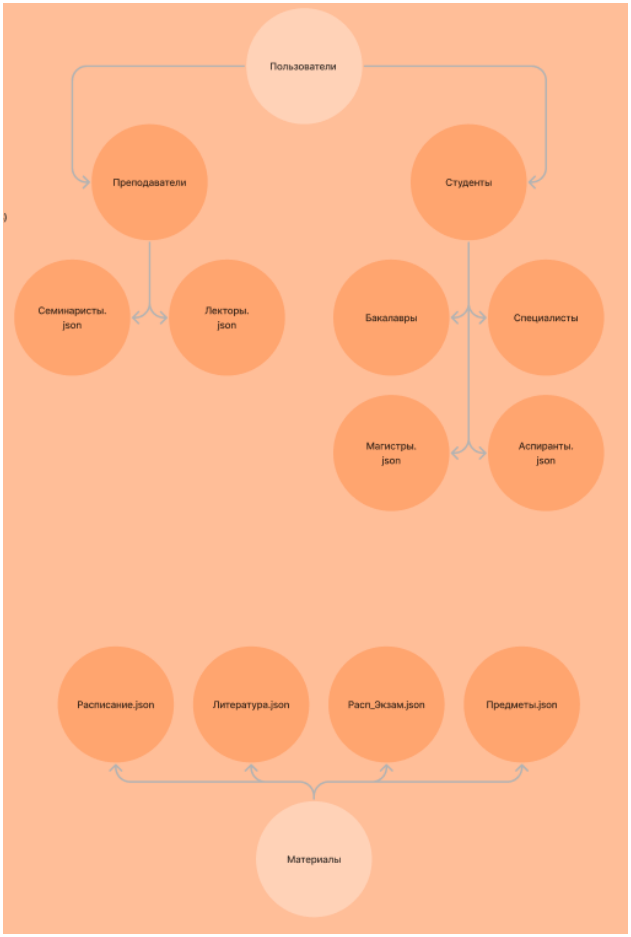

Рис. 14: Диаграмма состояний

13 №17 UML. Диаграмма размещения

13.1 Цель работы

Целью данной работы является разработка UML диаграммы размещения для иллюстрации физического развертывания компонентов системы мобильного приложения университета МАДИ в рамках аппаратной инфраструктуры.

13.2 Описание диаграммы

На представленной диаграмме размещения иллюстрируются следующие компоненты системы:

- **Пользователи** — отображают физические точки входа в систему, через которые осуществляется доступ к мобильному приложению.
- **Сервер** — центральный узел системы, который обрабатывает входящие запросы от пользователей и взаимодействует с базой данных.
- **База данных** — хранит все данные, необходимые для функционирования мобильного приложения.

13.3 Физическое развертывание

Каждый из пользователей подключается к серверу, который в свою очередь связан с базой данных. Это развертывание позволяет пользователям взаимодействовать с централизованным хранилищем данных через сервер приложений.

13.4 Взаимодействие компонентов

Пользователи отправляют запросы на сервер, который обрабатывает эти запросы и, при необходимости, обращается к базе данных для извлечения или сохранения информации. Затем сервер отправляет ответы обратно пользователям. Вся коммуникация между пользователями и сервером, а также между сервером и базой данных осуществляется через защищенные соединения.

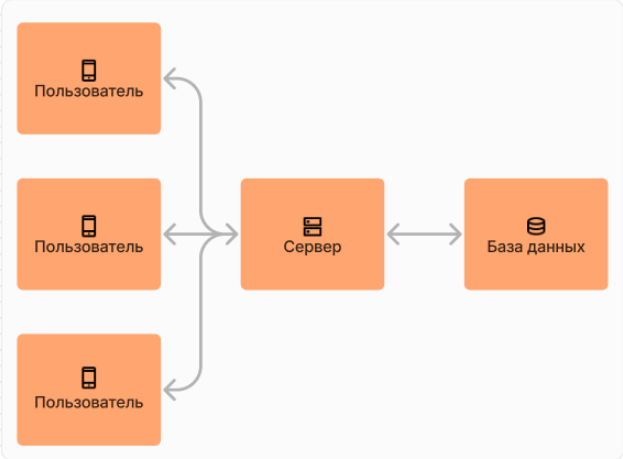

Рис. 15: Диаграмма состояний