МИНОБРНАУКИ РОССИИ

**ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ**

**ВЫСШЕГО ОБРАЗОВАНИЯ**

**“ВОРОНЕЖСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ”**

Факультет компьютерных наук

Кафедра программирования и информационных технологий

Система для управления и отслеживания выполнения домашних заданий и проектов «ProjectPal»

Курсовой проект

по дисциплине

Технологии программирования

*09.03.02 Информационные системы и технологии*

*Программная инженерия в информационных системах*

*6 семестр 2023/2024 учебного года*

Зав. кафедрой \_\_\_\_\_\_\_\_\_\_\_\_\_\_ к. т. н., доцент С. Д. Махортов

Обучающийся \_\_\_\_\_\_\_\_\_\_\_\_\_\_ ст. 3 курса оч. отд. Д. А. Шестопалов

Обучающийся \_\_\_\_\_\_\_\_\_\_\_\_\_\_ ст. 3 курса оч. отд. Г. В. Иванов

Обучающийся \_\_\_\_\_\_\_\_\_\_\_\_\_\_ ст. 3 курса оч. отд. Д. С. Третьяков

Руководитель \_\_\_\_\_\_\_\_\_\_\_\_\_\_ В. С. Тарасов, ст. преподаватель \_\_.\_\_.20\_\_

Воронеж 2024

**Содержание**

[Введение 4](#_Toc169819356)

[1 Постановка задачи 5](#_Toc169819357)

[1.1 Цели создания системы 5](#_Toc169819358)

[1.2 Требования к разрабатываемой системе 5](#_Toc169819359)

[1.3 Задачи проекта 5](#_Toc169819360)

[2 Анализ предметной области 6](#_Toc169819361)

[2.1 Терминология 6](#_Toc169819362)

[2.2 Обзор аналогов 8](#_Toc169819363)

[2.2.1 Moodle 8](#_Toc169819364)

[2.2.2 IStudiez Pro 9](#_Toc169819365)

[2.2.3 MyStudyLive 10](#_Toc169819366)

[2.2.4 Power Planner 11](#_Toc169819367)

[2.3 Моделирование системы 12](#_Toc169819368)

[2.3.1 Диаграмма прецедентов 12](#_Toc169819369)

[2.3.2 Диаграммы последовательности 14](#_Toc169819370)

[2.3.3 Диаграмма развертывания 21](#_Toc169819371)

[2.3.4 Диаграммы состояния 21](#_Toc169819372)

[2.3.5 ER-диаграмма 22](#_Toc169819373)

[2.3.6 Диаграммы активности 23](#_Toc169819374)

[3 Реализация 29](#_Toc169819375)

[3.1 Средства реализации 29](#_Toc169819376)

[3.2 Реализация базы данных 29](#_Toc169819377)

[3.3 Серверная часть 30](#_Toc169819378)

[3.3.1 Архитектура серверной части 30](#_Toc169819379)

[3.3.2 Диаграмма классов моделей 30](#_Toc169819380)

[3.3.3 Диаграмма классов DTO 31](#_Toc169819381)

[3.3.4 Диаграмма классов репозиториев 33](#_Toc169819382)

[3.3.5 Диаграмма классов мапперов 34](#_Toc169819383)

[3.3.6 Диаграмма классов JWT 35](#_Toc169819384)

[3.3.7 Диаграмма классов конфигураций 35](#_Toc169819385)

[3.3.8 Диаграмма классов сервисов 36](#_Toc169819386)

[3.3.9 Диаграмма классов контроллеров 38](#_Toc169819387)

[4 Клиентская часть 40](#_Toc169819388)

[4.1 Реализация 40](#_Toc169819389)

[4.2 Основные экраны 41](#_Toc169819390)

[4.2.1 Экран авторизации 41](#_Toc169819391)

[4.2.2 Экран регистрации 42](#_Toc169819392)

[4.2.3 Экран главной страницы 43](#_Toc169819393)

[4.2.4 Экран календаря 44](#_Toc169819394)

[4.2.5 Экран профиля 45](#_Toc169819395)

[4.2.6 Экран главной страницы 46](#_Toc169819396)

[4.2.7 Экран списка групп 47](#_Toc169819397)

[5 Тестирование 48](#_Toc169819398)

[5.1 Интеграционное тестирование 48](#_Toc169819399)

[5.2 Unit тестирование 49](#_Toc169819400)

[Заключение 50](#_Toc169819401)

[Список используемых источников 51](#_Toc169819402)

Введение

В наше время мобильные приложения стали неотъемлемой частью нашей повседневной жизни, предоставляя доступ к различным сервисам и удобствам всего лишь в паре кликов. Одной из самых востребованных категорий мобильных приложений являются системы управления домашними заданиями и проектами. Они успешно объединяют функции организации задач, контроля сроков и обмена информацией.

В данной курсовой работе рассматривается процесс разработки собственной системы управления домашними заданиями и проектами. Будут рассмотрены различные аспекты этого процесса, начиная с анализа предметной области и определения ее концепции и основных требований. Затем будет изучено проектирование пользовательского интерфейса и пользовательского опыта с учетом современных тенденций в дизайне и удобства использования.

Постановка задачи

* 1. Цели создания системы

Цель данной работы заключается в создании системы, которая поможет пользователям эффективно управлять и отслеживать учебные задания и проекты.

* 1. Требования к разрабатываемой системе
* обеспечение авторизации и аутентификации пользователей;
* использование механизмов защиты от SQL-инъекций;
* использование протокола передачи данных HTTPS;
* приложение должно быть построено на трехуровневой архитектуре.
  1. Задачи проекта
* авторизация в системе;
* регистрация в системе;
* просмотр ознакомительной информации о приложении;
* выход из системы;
* просмотр страницы календаря;
* просмотр определённого дня;
* просмотр страницы домашних заданий/проектов;
* просмотр информации о задании/проекте;
* просмотр страницы профиля;
* настройка профиля (изменить аватар);
* создать/удалить/изменить задание/проект;
* приписать/отписать учащегося к проекту;
* оценить работу учащегося;
* оставить комментарий к работе учащегося;
* проверить общий прогресс выполнения задания/проекта группы;
* создавать/удалять/изменять профиль пользователя;
* просматривать жалобы и выносить решения по ним.

Анализ предметной области

* 1. Терминология

1. Глоссарий

|  |  |
| --- | --- |
| Термин | Определение термина |
| Аватар | Изображение, используемое в учетной записи для персонализации пользователя |
| Авторизация | Предоставление определённому лицу прав на выполнение определённых действий; а также процесс проверки (подтверждения) данных прав при попытке выполнения этих действий |
| Авторизованный пользователь | Пользователь прошедший процесс авторизации |
| Администратор | Человек, имеющий доступ к расширенному функционалу веб-сервиса, имеющий знания о формате приема статей |
| Боковое меню (сайд меню) | Меню, которое представляет собой панель, которая находится (или открывается, при помощи, каких-либо кнопок или жестов) снизу, слева или справа от области основного контента приложения, содержащая вертикальную, независимую от основного контента приложения прокрутку, и служит основным инструментом навигации в приложении |
| Виджет (Widget) | Программный агент, который может выполнять задачи для пользователя на основе информации, введенной пользователем, данных о его местонахождении, а также информации, полученной из различных интернет-ресурсов |
| Комьюнити (Community) | Cообщество людей, объединённых общими интересами, целями, идеями |
| Неавторизованный пользователь | Пользователь, не прошедший процесс аутентификации |
| Ошибка (Bug) | Общий термин, используемый для обозначения непредвиденной ошибки или дефекта в аппаратном или программном обеспечении, что приводит к его неисправности |
| Пользователь | Лицо, которое использует действующую систему для выполнения конкретной функции |
| Профиль  (в веб-приложении) | Учетная запись пользователя в веб-приложении, вход в которую осуществляется с помощью логина / номера телефона / e-mail и пароля. В учетной записи содержится информация о пользователе |
| СУБД | Система управления базами данных. Комплекс программ, позволяющих создать базу данных (БД) и манипулировать данными |
| Фреймворк | Программные продукты, которые упрощают создание и поддержку технически сложных или нагруженных проектов. Фреймворк, как правило, содержит только базовые программные модули |
| Чат | Средство обмена сообщениями по компьютерной сети в режиме реального времени |
| Android | Операционная система для мобильных устройств |
| API | Программный интерфейс приложения. Описание способов, которыми одна компьютерная программа может взаимодействовать с другой программой |

* 1. Обзор аналогов

В современном мире цифровые технологии активно внедряются в образовательный процесс. Это обусловлено желанием учебных заведений повысить эффективность обучения, сделать его более интерактивным и доступным. Рынок образовательных технологий предлагает множество решений для управления учебным процессом, отслеживания успеваемости и взаимодействия между учащимися и преподавателями. В этом многообразии хочется выделить два проекта, которые занимают значительное место среди образовательных платформ – Moodle и IStudiez Pro. Оба проекта имеют свои уникальные функции и ориентированы на разные аспекты образовательного процесса.

* + 1. Moodle

Moodle – это популярная платформа для создания образовательных курсов и управления учебным процессом. Подходит как для дистанционного, так и для очного обучения. Платформа предлагает широкие возможности для создания курсов, проведения тестирований и организации обратной связи.

Преимущества:

* открытый исходный код, позволяющий настраивать платформу под специфические нужды учебного заведения;
* широкий спектр инструментов для создания учебного контента;
* возможность интеграции с множеством внешних сервисов и платформ;
* большое и активное сообщество пользователей.

Недостатки:

* требует определённых технических знаний для настройки и администрирования;
* интерфейс может показаться перегруженным новым пользователям.
  + 1. IStudiez Pro

IStudiez Pro – это приложение для управления личным учебным расписанием, заданиями и оценками. Оно предназначено в основном для студентов и школьников, которые хотят организовать свой учебный процесс.

Преимущества:

* интуитивно понятный интерфейс и лёгкость в использовании;
* поддержка мультиплатформенности, доступность на iOS, Android, macOS и Windows;
* возможность синхронизации данных между устройствами;
* визуализация учебного расписания и заданий.

Недостатки:

* ограниченные возможности для взаимодействия с преподавателями и другими студентами;
* отсутствие версии с открытым исходным кодом, что ограничивает кастомизацию;
* Наличие платной подписки для расширения функционала.  
  + 1. MyStudyLive

MyStudyLive – это образовательная платформа, предназначенная для организации и управления учебным процессом как в дистанционном, так и в очном формате. Платформа позволяет преподавателям и студентам взаимодействовать через функционал виртуальных классов, планирование занятий и обмен учебными материалами.

Преимущества:

* Интуитивно понятный интерфейс;
* Гибкость в планировании;
* Интерактивные элементы;
* Наличие мобильного приложения;
* Поддержка нескольких языков.

Недостатки:

* Ограниченная функциональность в бесплатной версии;
* Технические ограничения;
* Ограниченная интеграция с другими сервисами.
  + 1. Power Planner

Power Planner – это планировщик для студентов, который помогает управлять учебным расписанием, заданиями и экзаменами. Платформа предлагает удобные инструменты для организации учебного процесса и контроля за выполнением задач.

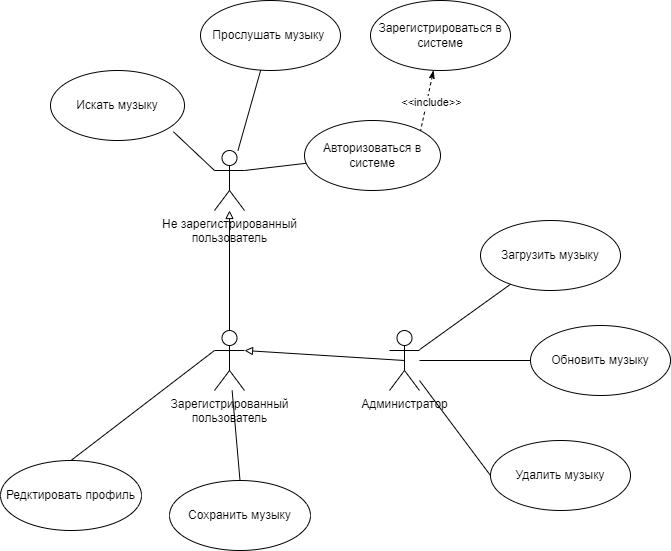
Преимущества:

* Простой и удобный интерфейс;
* Интеграция с календарем;
* Напоминания и уведомления;
* Кроссплатформенность;
* Отслеживание успеваемости;

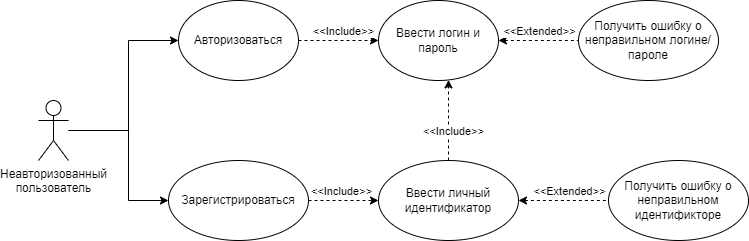
Недостатки:

* Ограниченная функциональность для образовательных учреждений;
* Платные функции;
* Отсутствие интеграции с системами управления обучением;
  1. Моделирование системы
     1. Диаграмма прецедентов

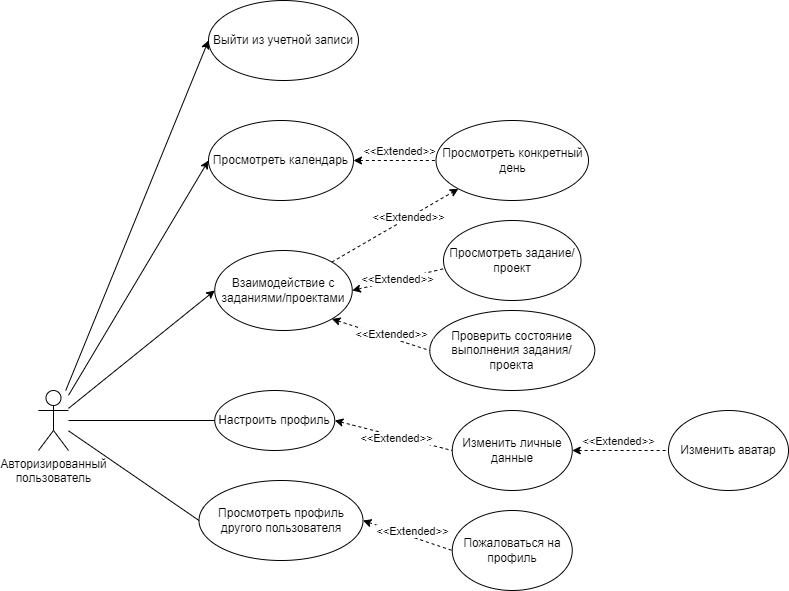
Рассмотрим полную диаграмму использования приложения различными типами пользователей (Рисунок 1-5). В данном контексте составление диаграммы прецедентов обусловлено необходимостью моделирования системы и понимания её функциональности и потребностей пользователей. Эти диаграммы помогают определить основные действия, которые пользователь должен совершить в системе для достижения определенных целей. Они также помогают выявить возможные риски и проблемы, которые могут возникнуть в процессе использования системы.



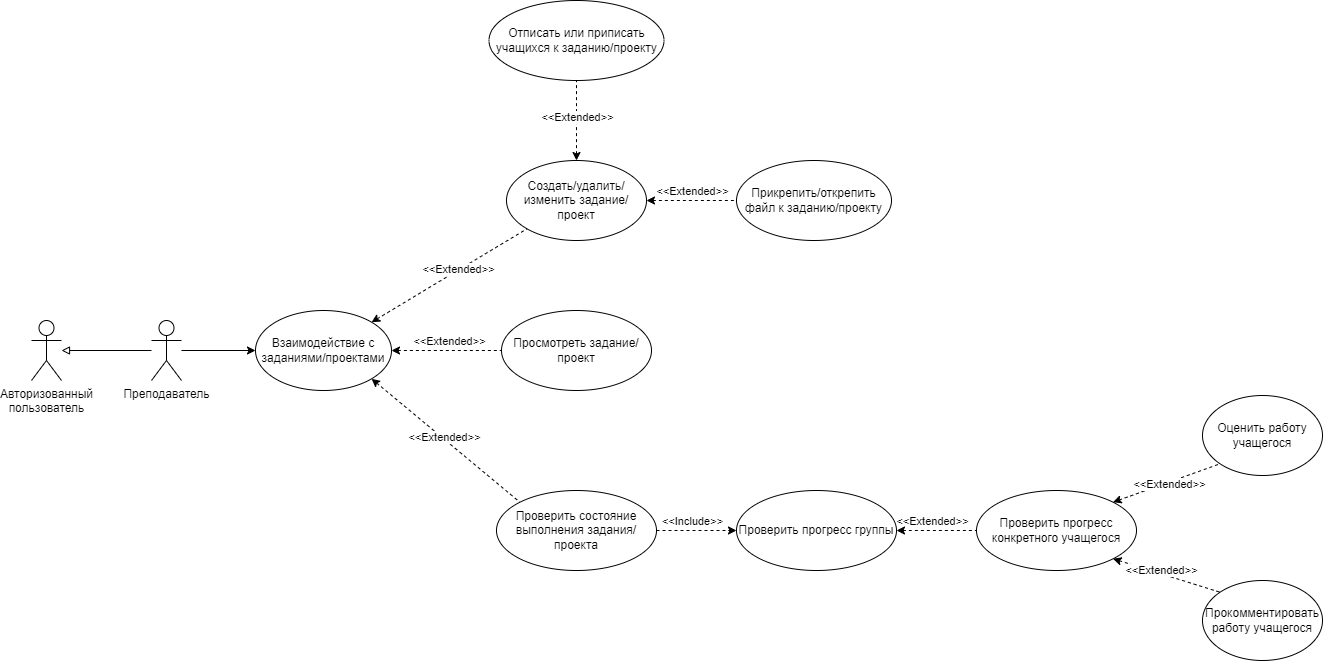
1. Диаграмма прецедентов (admin)



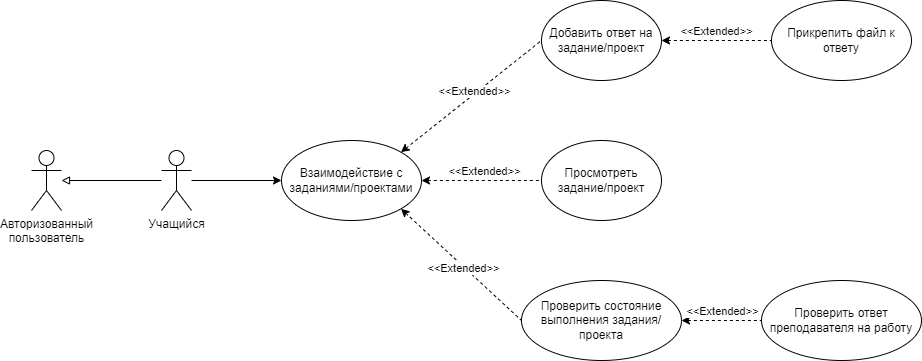
1. Диаграмма прецедентов (unauthorized user)



1. Диаграмма прецедентов (authorized user)



1. Диаграмма прецедентов (lecturer)

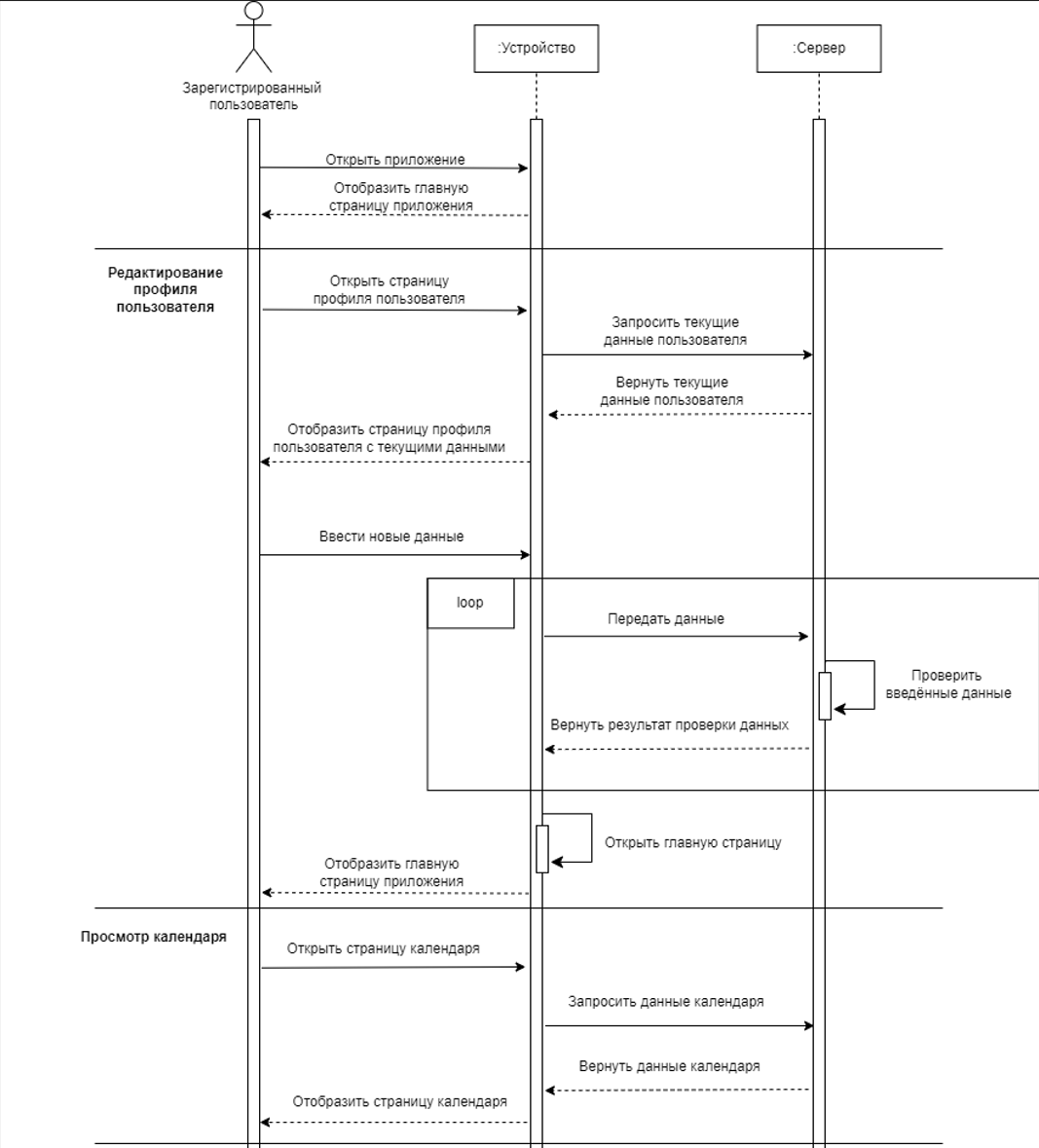


1. Диаграмма прецедентов (student)
   * 1. Диаграммы последовательности

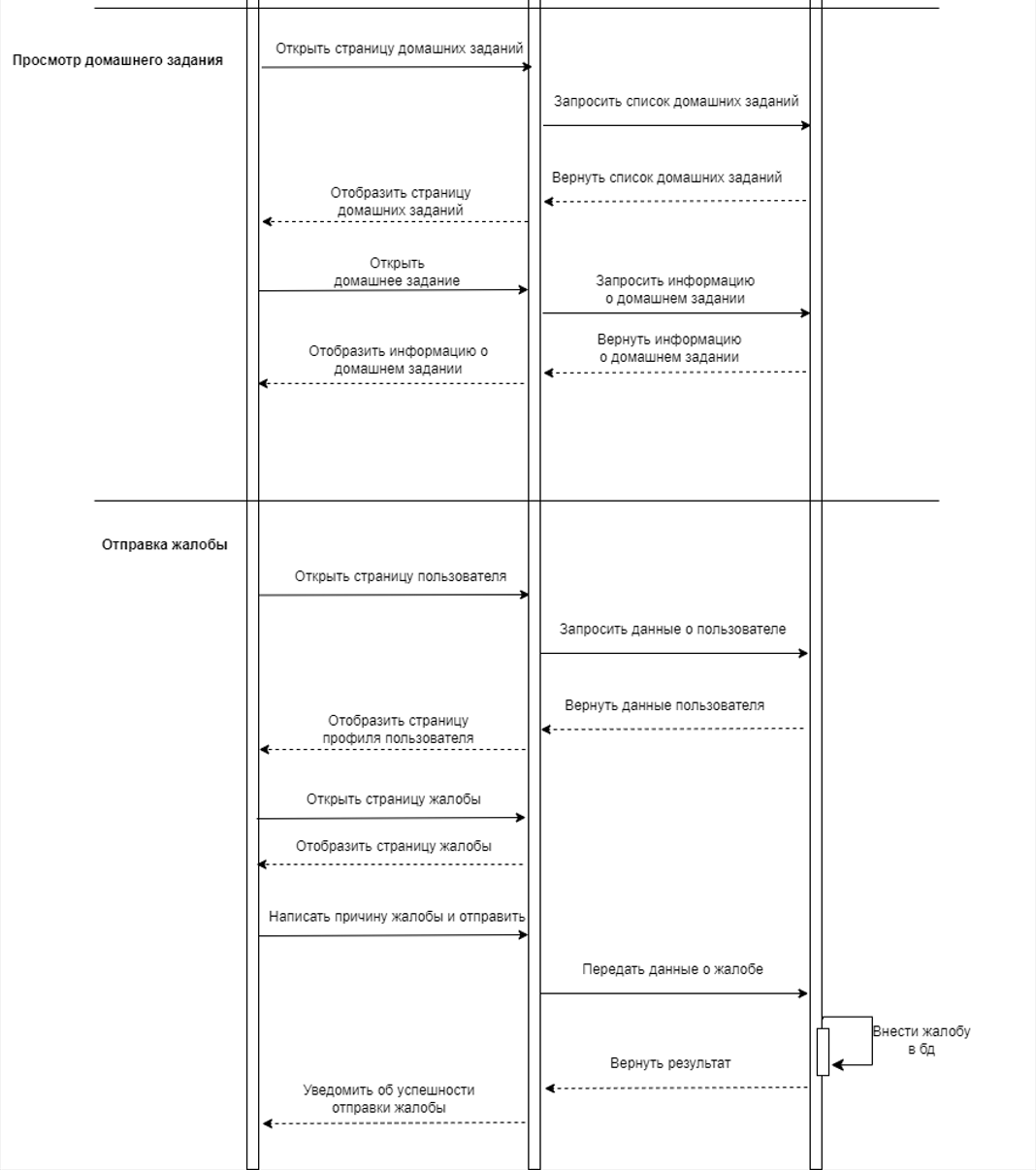
Диаграмма последовательности (Рисунок 6-11) играет важную роль в проекте, помогая более глубоко понять процесс, повысить его эффективность и упростить взаимодействие.



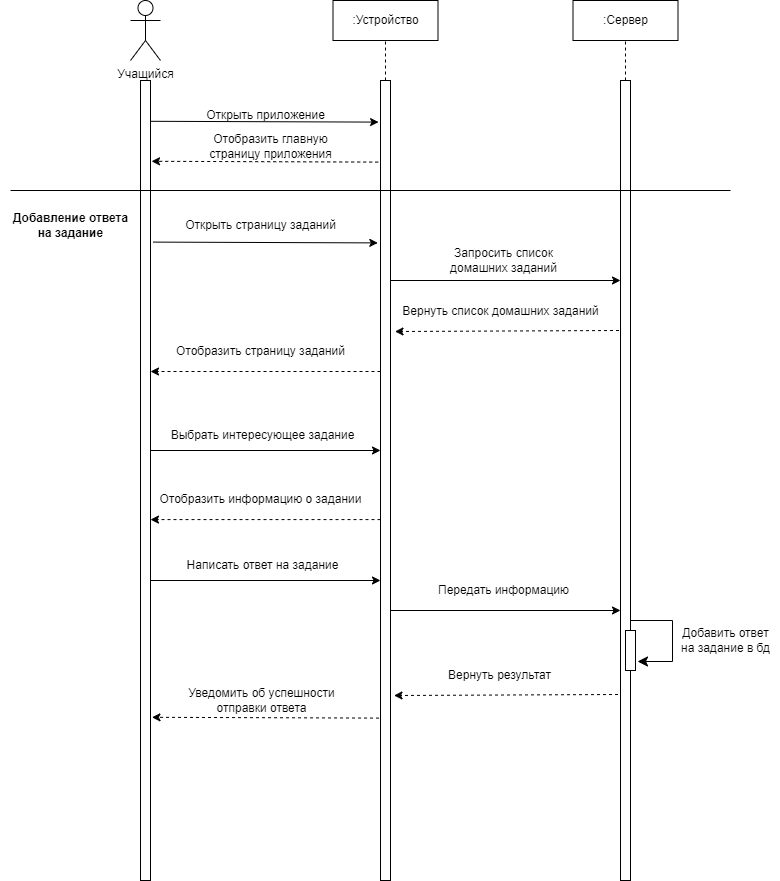
1. Диаграмма последовательности (Регистрация и авторизация)



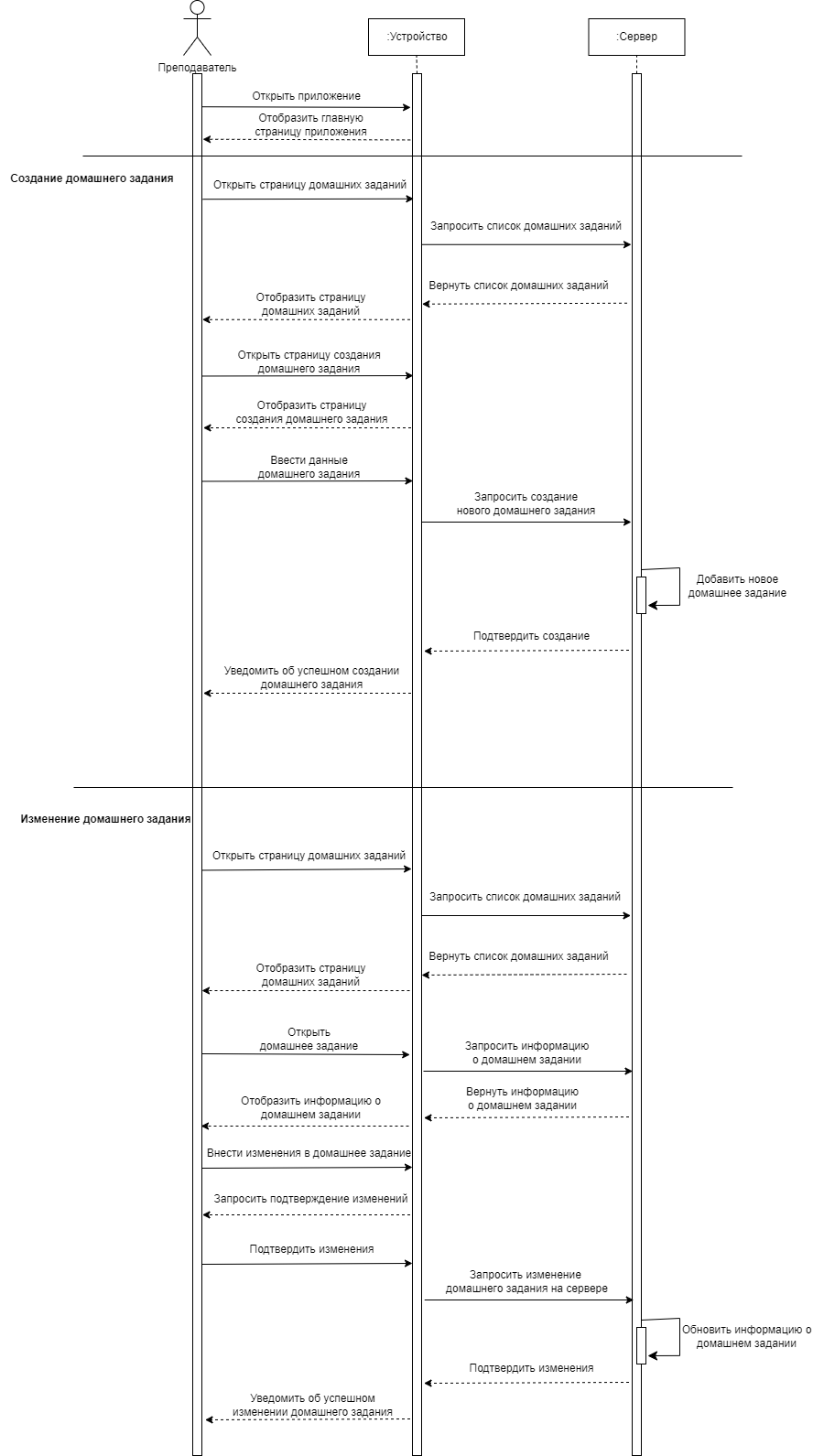
1. Диаграмма последовательности (Зарегистрированный пользователь)



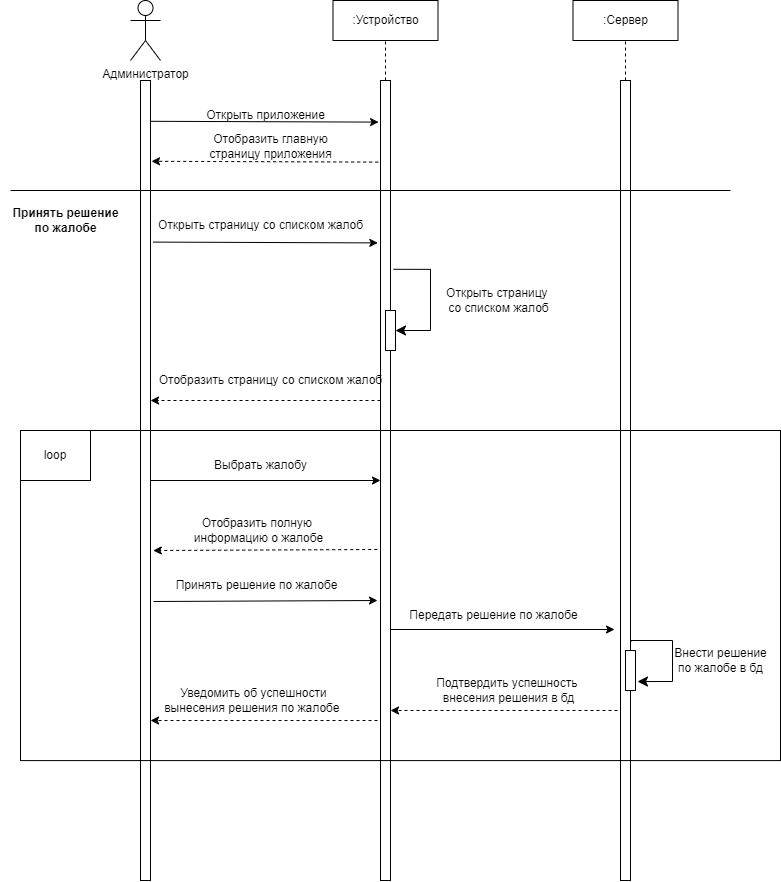
1. Диаграмма последовательности (Зарегистрированный пользователь) - продолжение



1. Диаграмма последовательности (Учащийся)

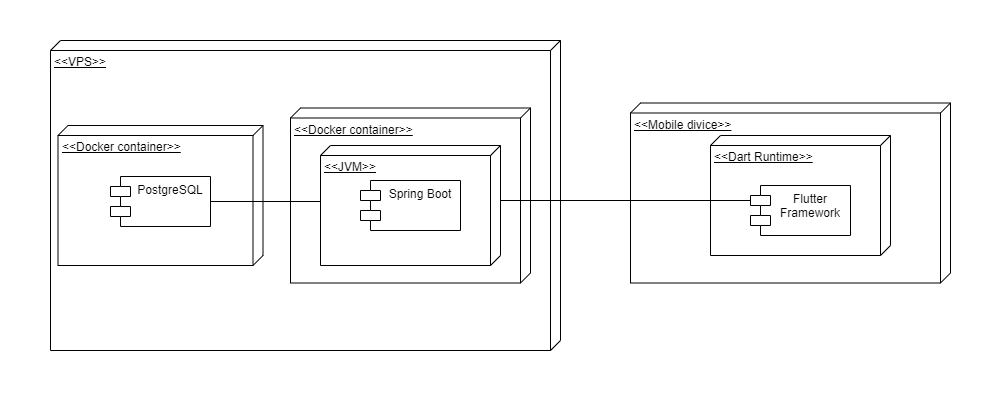


1. Диаграмма последовательности (Преподаватель)



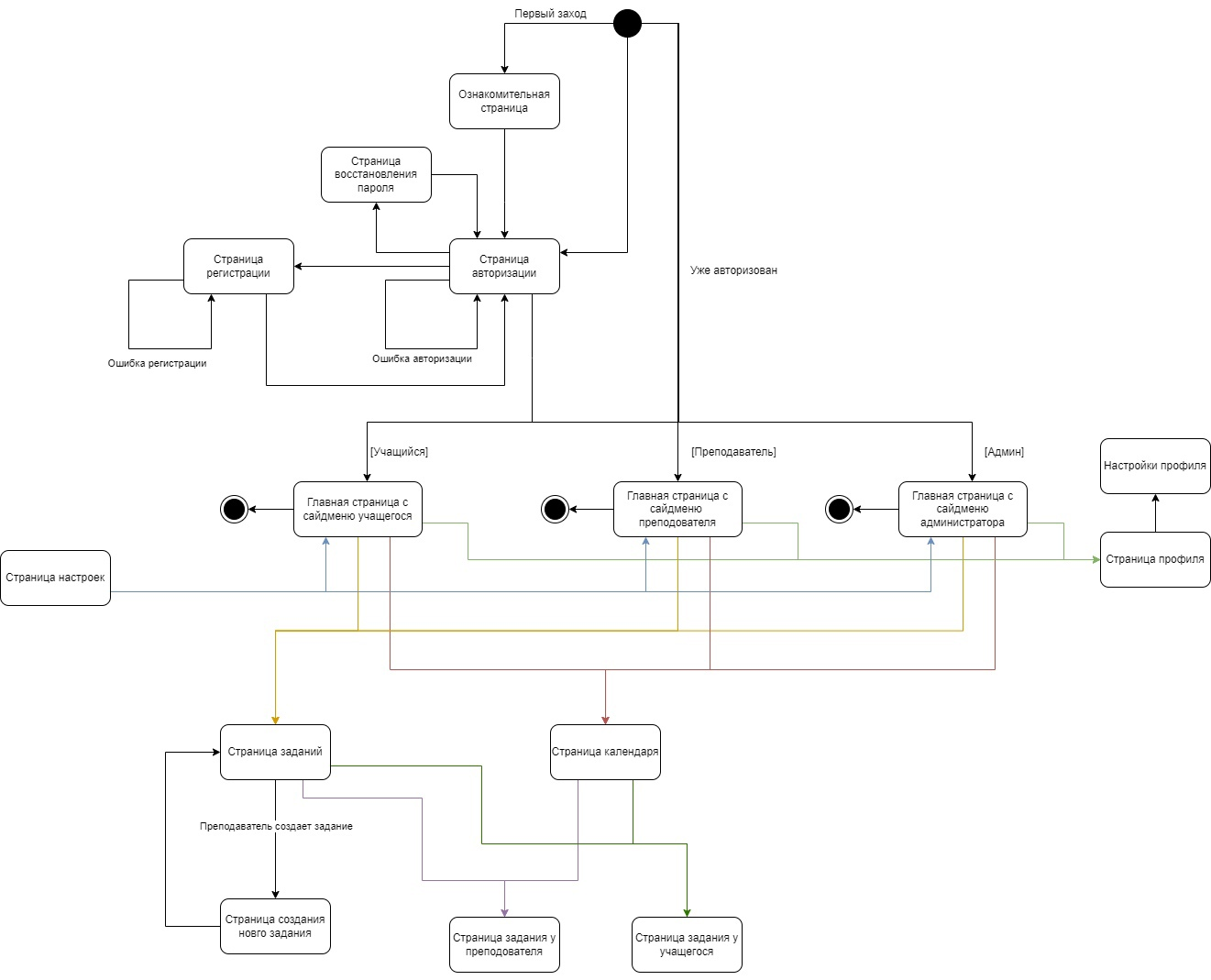
1. Диаграмма последовательности (Администратор)
   * 1. Диаграмма развертывания

Диаграмма развертывания (Рисунок 12) помогает определить необходимости в аппаратном обеспечении, спланировать установку и настройку компонентов системы, а также оценить её производительность и масштабируемость.



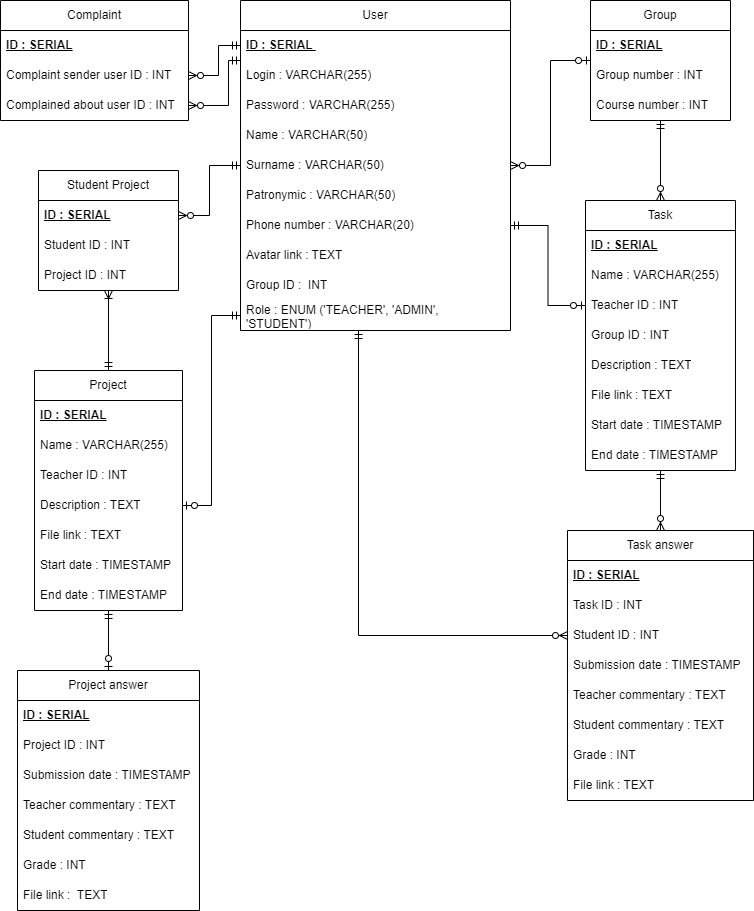
1. Диаграмма развертывания приложения
   * 1. Диаграммы состояния

Диаграмма состояния (Рисунок 13) позволяет анализировать возможные сценарии поведения системы, выделять ключевые состояния и переходы между ними, а также оценивать её надежность и устойчивость к ошибкам. В рамках нашего проекта были разработаны три диаграммы состояний для администратора, зарегистрированного пользователя и незарегистрированного пользователя.



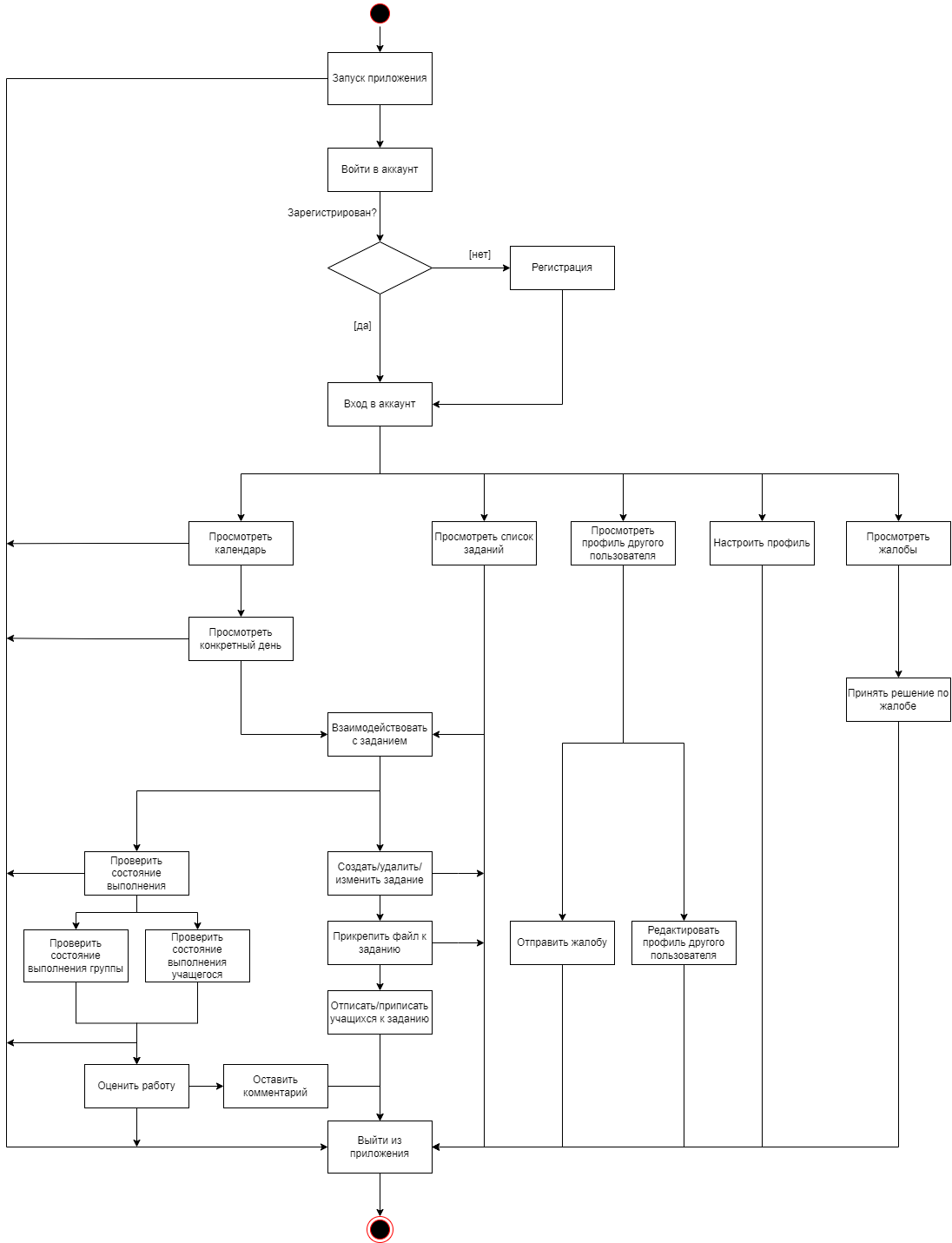
1. Диаграмма состояния
   * 1. ER-диаграмма

ER-диаграмма (Рисунок 14) предоставляет структурное представление данных, иллюстрируя сущности (объекты) в системе и их взаимосвязи. Она помогает определить основные сущности, их атрибуты и отношения между ними, что облегчает проектирование базы данных и анализ требований к системе.

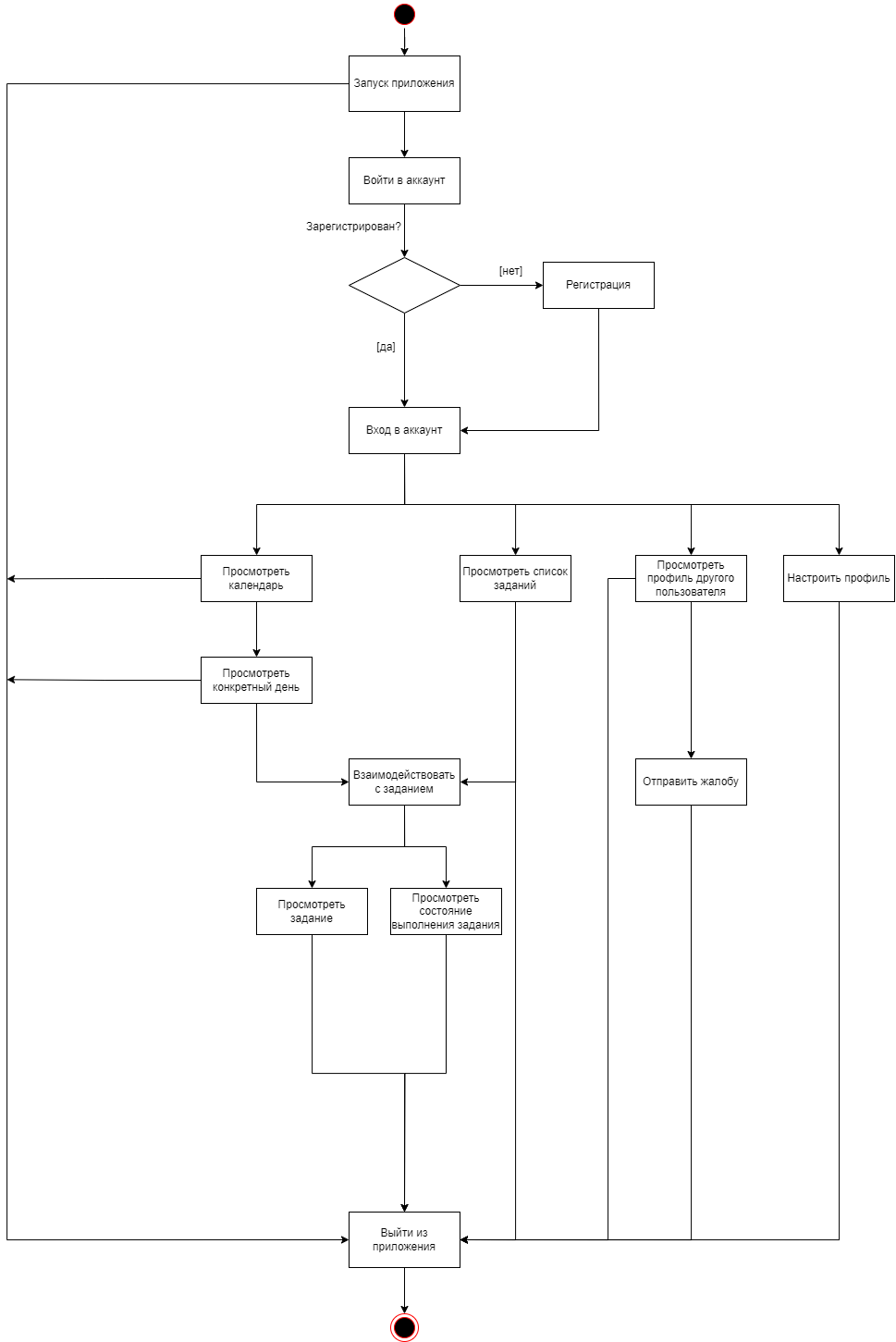


1. ER-диаграмма
   * 1. Диаграммы активности

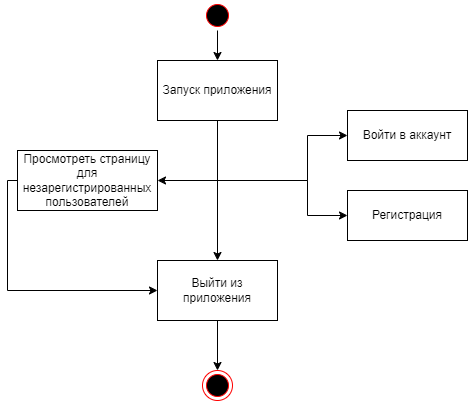
Диаграмма активности (Рисунок 15-19) помогает улучшить понимание системных процессов, выявить и оптимизировать узкие места. Кроме того, она применяется для описания бизнес-процессов и управления проектами. Для данного проекта было спроектировано 5 диаграмм активности: для администратора, преподавателя, учащегося, зарегистрированного пользователя и незарегистрированного пользователя.



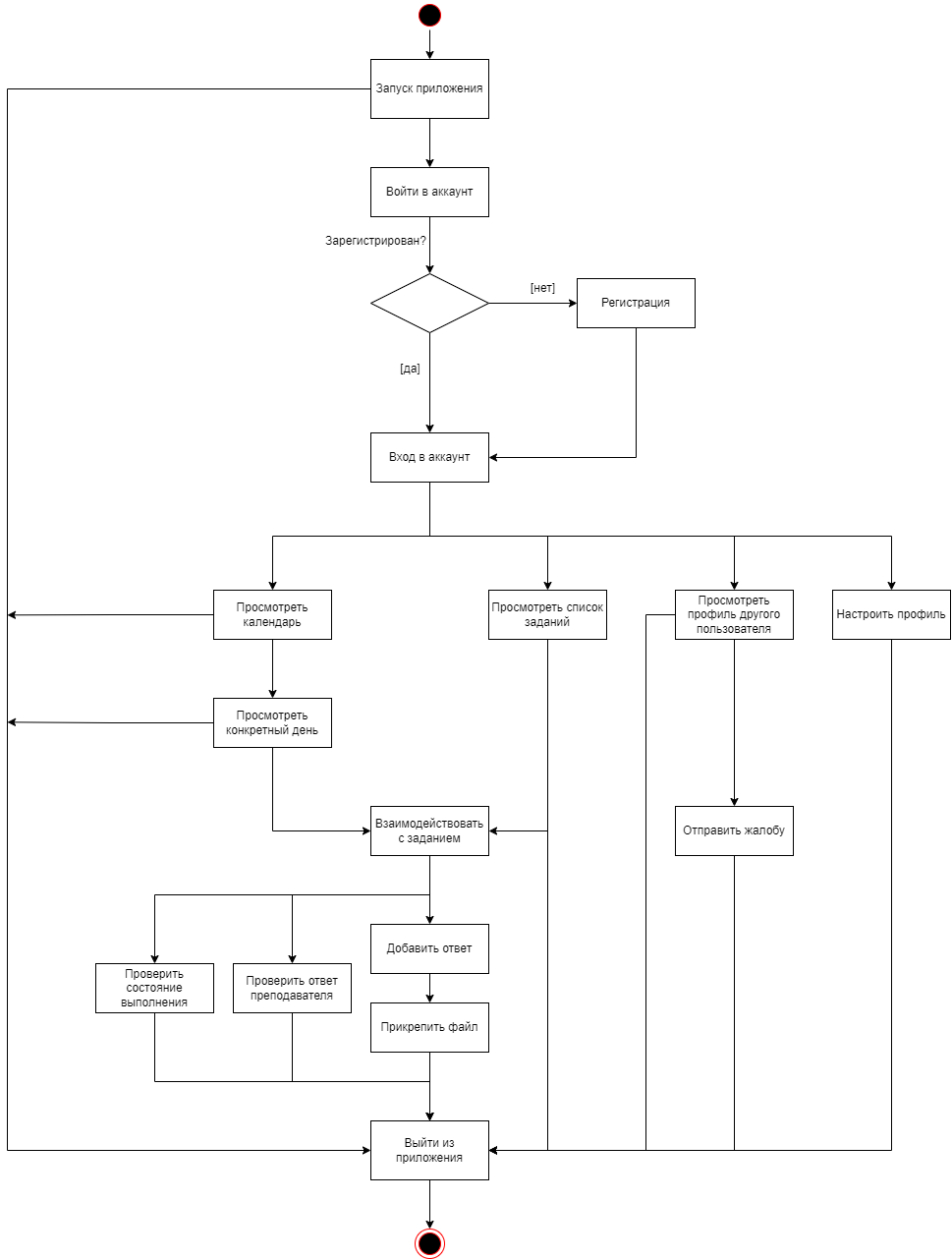
1. Диаграмма активности администратора



1. Диаграмма активности авторизованного пользователя



1. Диаграмма активности неавторизованного пользователя



1. Диаграмма активности (Учащийся)



1. Диаграмма активности (Преподаватель)

Реализация

* 1. Средства реализации

Перечень используемых технологий.

Backend:

* Java;
* SpringBootFramework;
* PostgreSQL;
* FlyWay;
* Docker.

Frontend:

* AndroidSDK;
* Dart;
* Flutter.

Инструменты для ведения документации:

* Miro;
* Swagger;
* Draw.io;
* Figma.

Дополнительный инструментарий:

* Git;
* GitHub;
* Trello.
  1. Реализация базы данных

Данные приложения хранятся в базе данных PostgreSQL. Она поддерживает множество типов данных, включая числовые, символьные, дата и время, логические, двоичные, сетевые адреса, геометрические, JSON, массивы и специальные типы, что делает её гибкой и мощной СУБД для разнообразных задач.

* 1. Серверная часть

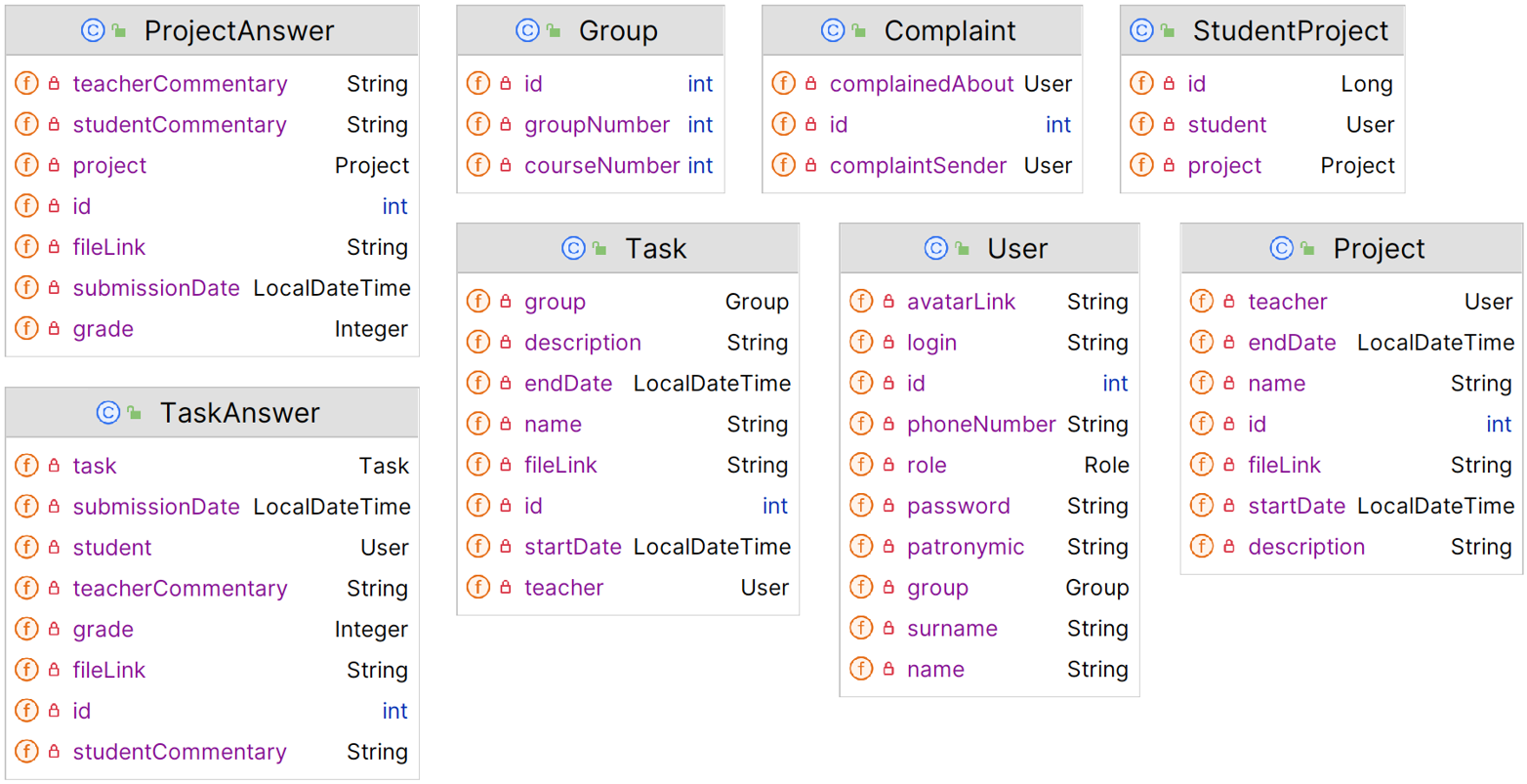
Серверная часть была реализована на строго типизированном объектно-ориентированном языке программирования Java. Он был выбран из-за своей надежности, безопасности и большого количества библиотек и фреймворков, которые позваляют создавать маштабируемые и высокопроизводительные приложения.

* + 1. Архитектура серверной части

Сервеная часть приложения была реализована с использованием трехслойной архитектуры веб-приложения с REST API. Для создания бэкенда взят фреймворк Spring boot, который позволяет работать с базой данных, добавлять зависимости, а также делает все приложение более адаптируемым и расширяемым.

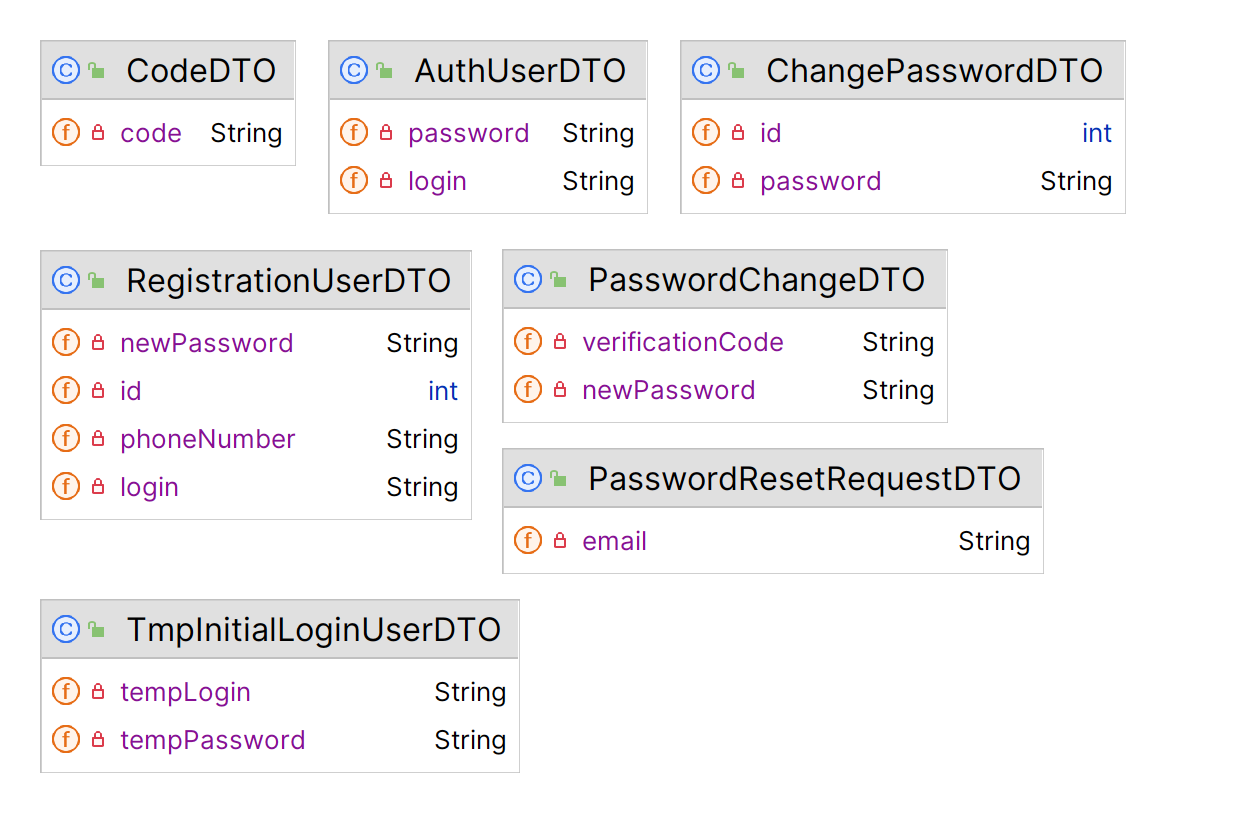
* + 1. Диаграмма классов моделей

Прежде всего в приложении были реализованы модели для взаимодействия с данными из PostgreSQL и для дольнейшего их использования в логике программы. Они содержат поля соответсвующие полям из таблиц в базе данных. Их классы представлены ниже (Рисунок 20).

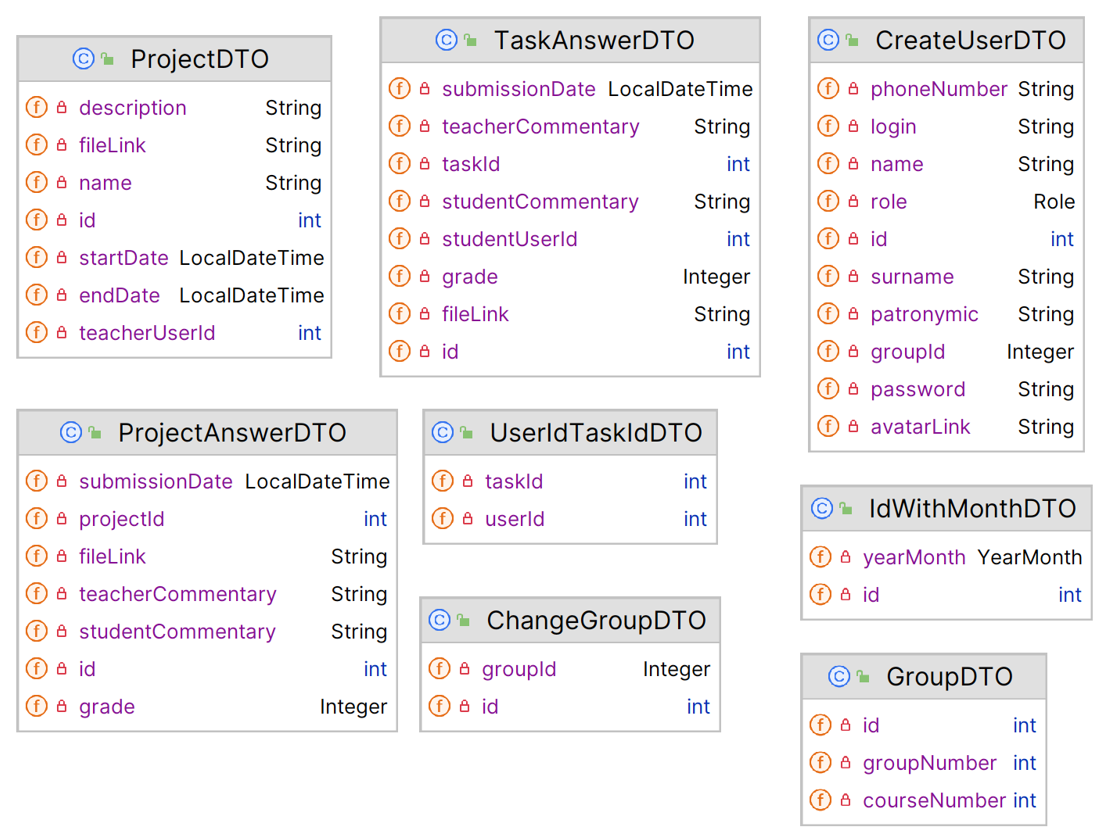


1. Диаграмма классов моделей
   * 1. Диаграмма классов DTO

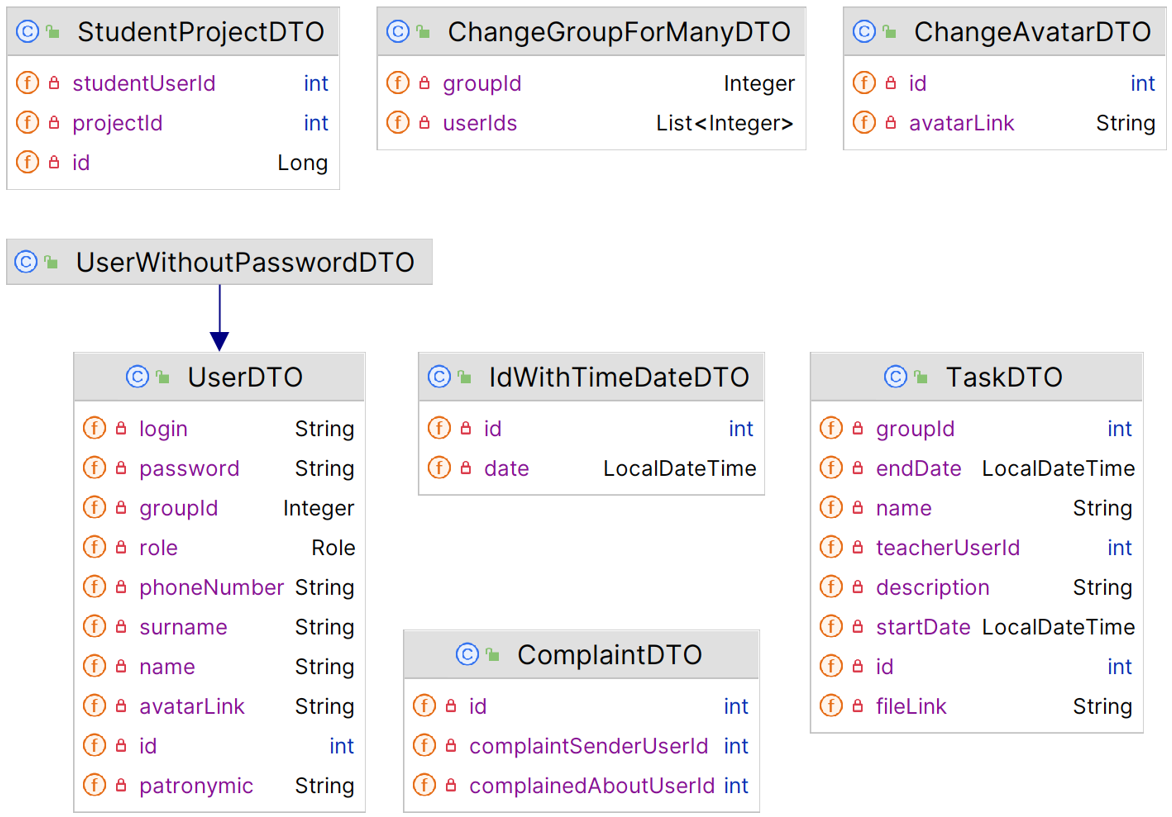
Для передачи данных между клиентом и сервером нужны объекты, которые не содеражат бизнесс логики, а лишь необходимые для работы данные. Таковыми являются DTO, которые помогают организовть взаимодействие через REST API. Диограмма ниже (Рисунок 21-23).



1. Диаграмма DTO для авторизации

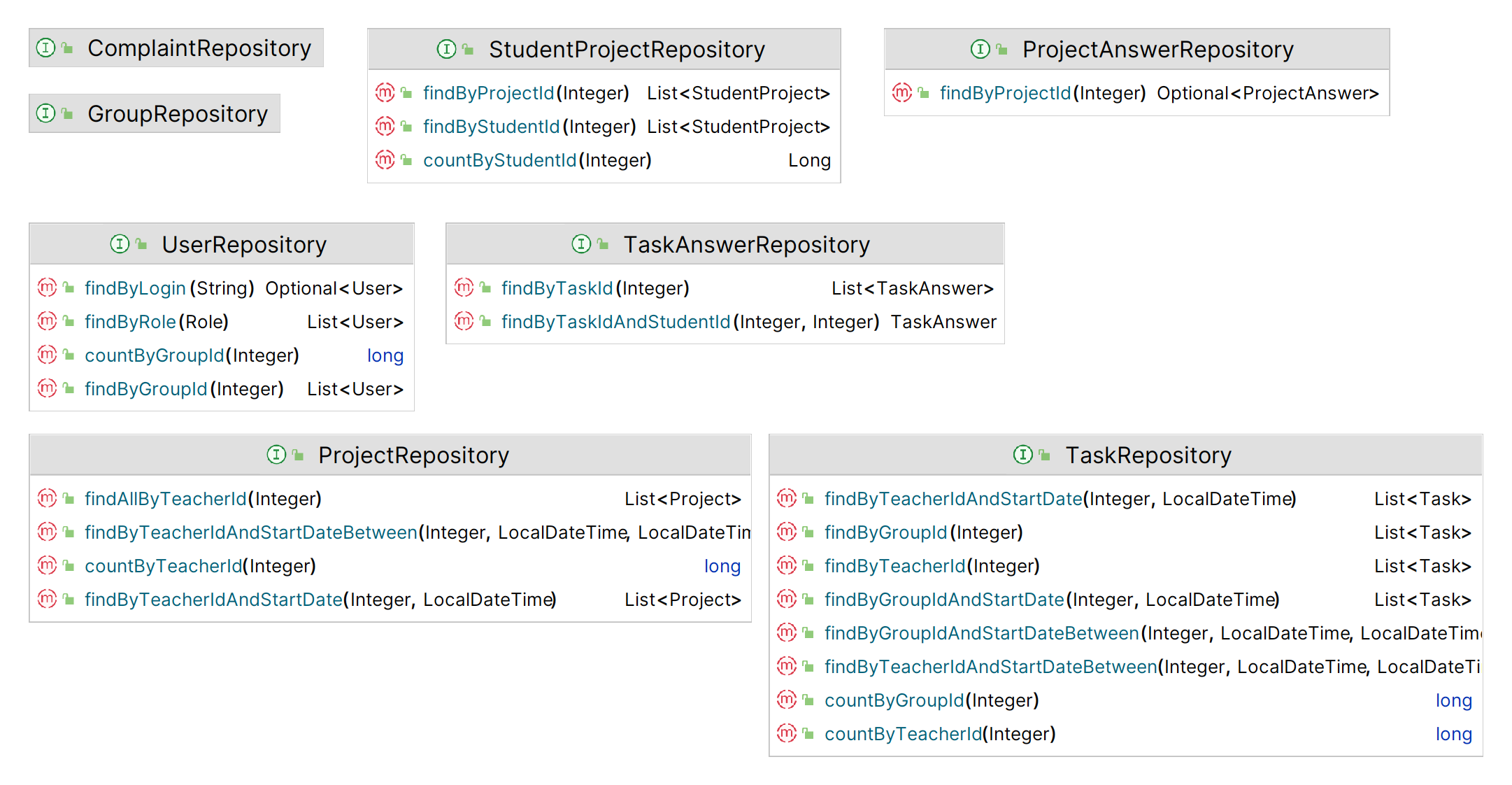


1. Диаграмма DTO часть 1



1. Диаграмма DTO часть 2
   * 1. Диаграмма классов репозиториев

Репазитории обеспечивают доступ к данным из баззы данных. Они сипользуются для выполнения операция, связанных с чтением и записью информации. Их классы представлены ниже (Рисунок 24).

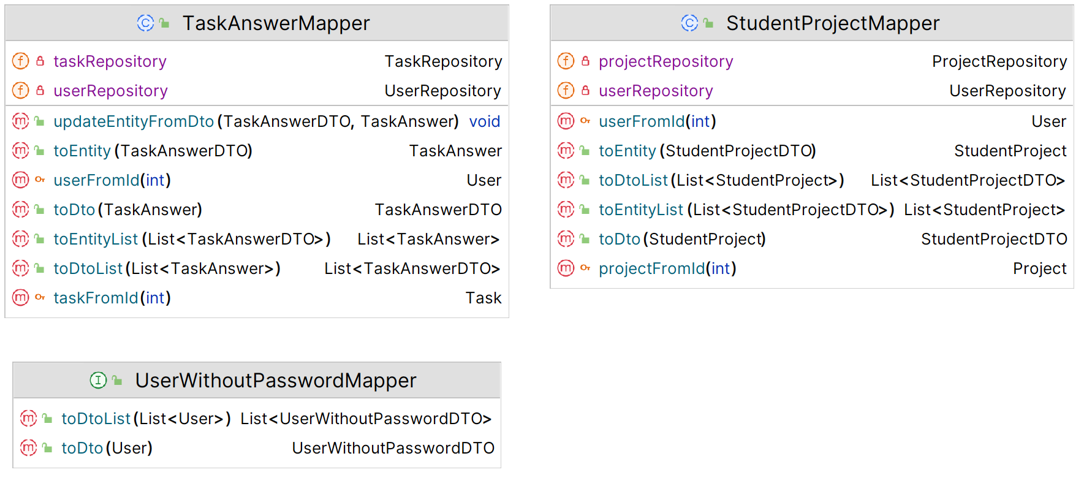


1. Диаграмма классов репозиториев
   * 1. Диаграмма классов мапперов

Классы мапперы преобразуют данные мжду различными типами или моделями, здесь же они используються для превода модлей в DTO и обратно. Для их реализации в проекте используется библиотека MapStruct, которая предоставляет автоматическуб генерацию классов для пробразований, используя прописанные интерфейсы или абстракные классы. Ниже представленны эти самые интерфейсы и абстракные классы(Рисунок 25-26).

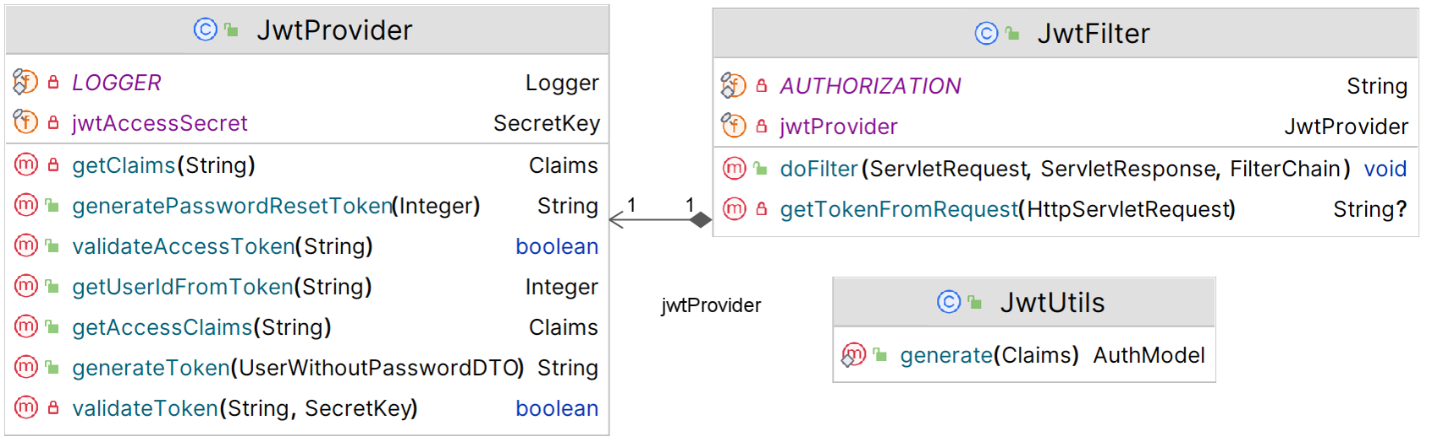


1. Диаграмма классов маперов часть 1



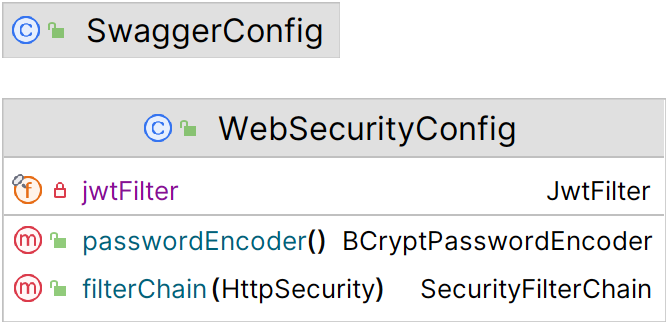
1. Диаграмма классов мапперов часть 2
   * 1. Диаграмма классов JWT

В проекте для авторизации и смены пароля используються JWT токены, а именно их реализация из библиотеки JJWT. В токен можно закодировать любую информацию о пользователю, а главное его ID и роль. Когда приходит запрос на сервер, из него берется на проверку разрешений для пользователя этот самый токен. Классы, реализующие данный функционал представлены на рисунке (Рисунок 27).



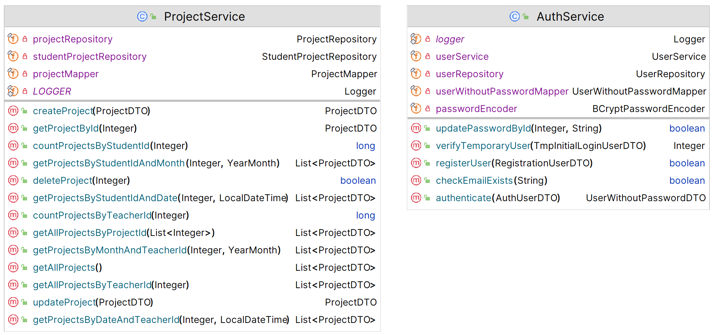
1. Диаграмма классов JWT
   * 1. Диаграмма классов конфигураций

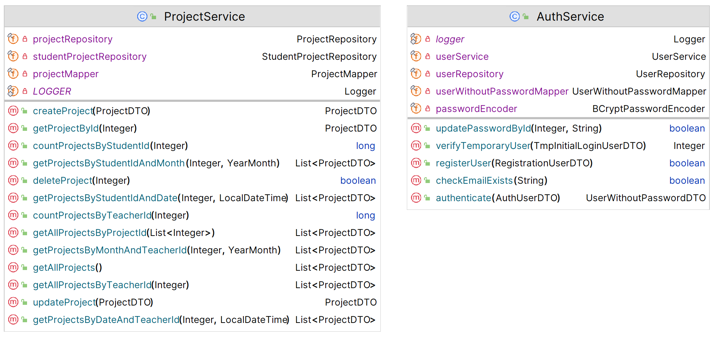
Spring boot позволяет при помощи анотаций bean, value, autowired и других настраивать поведение различных аспектов проекта. Здесь же они используются для настройки документации Swagger и ограничения доступа к различным запросам при помощи WebSecurityConfig. Их можно увидеть на рисунке (Рисунок 28).



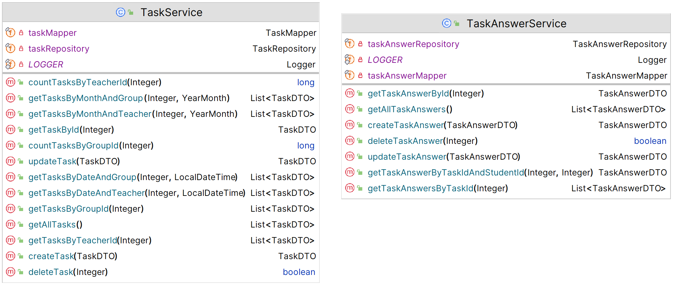
1. Диаграмма классов конфигураций
   * 1. Диаграмма классов сервисов

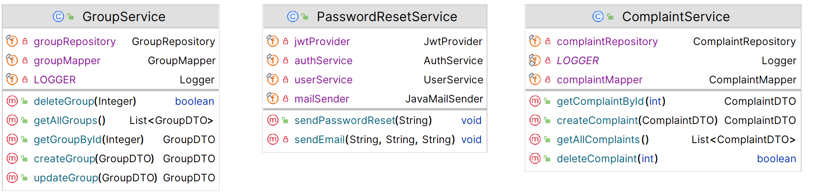
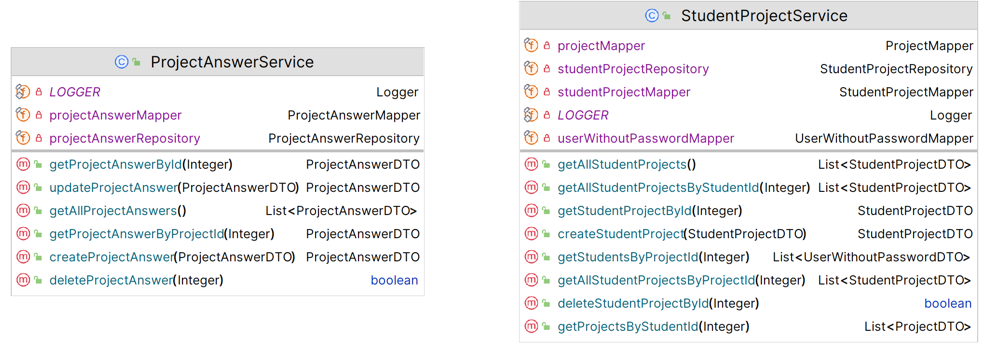
Сервисы в Spring Boot содеражать в себе практически всю бизнес-логику, а также выполняют операции, которые требует конроллеры. Именно сервисы используют в себе ранее упомянутые репозитории, модели и DTO в полной мере. Их реализация представленна ниже(Рисунок 29-30).





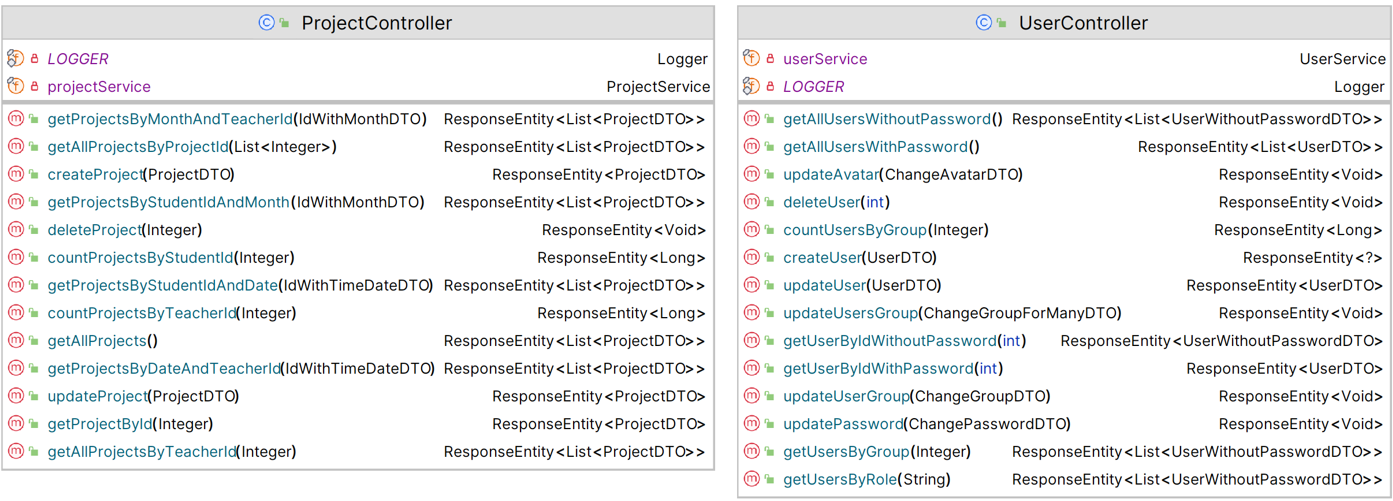
1. Диаграмма классов сервисов часть 1

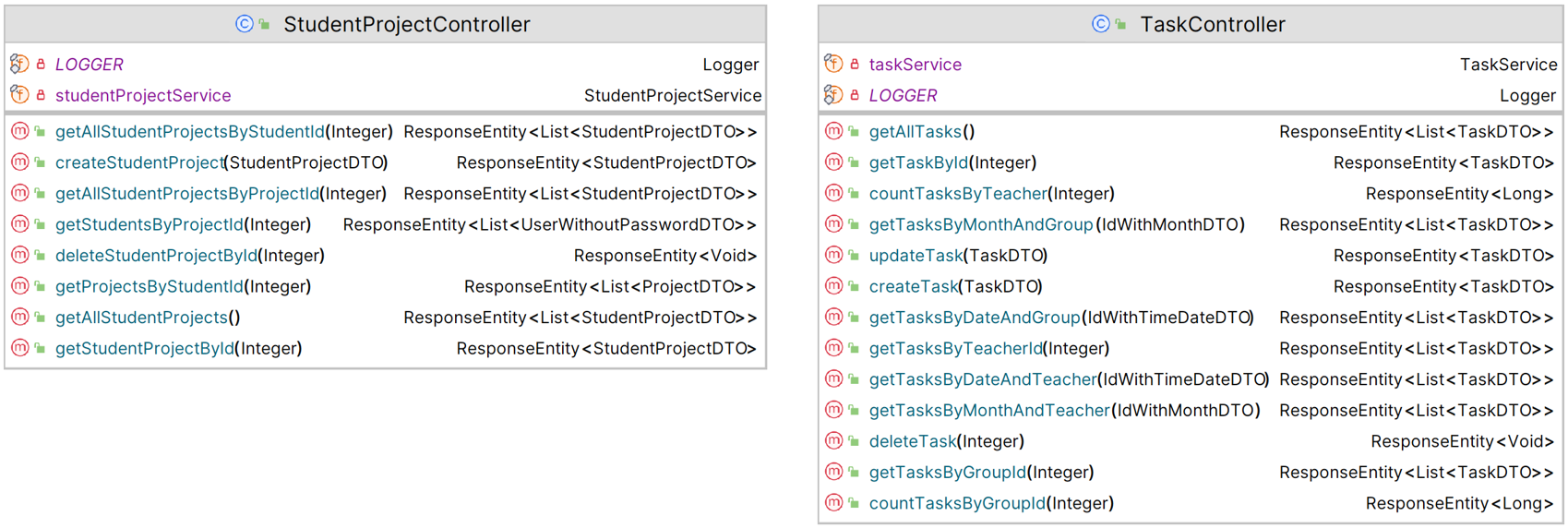


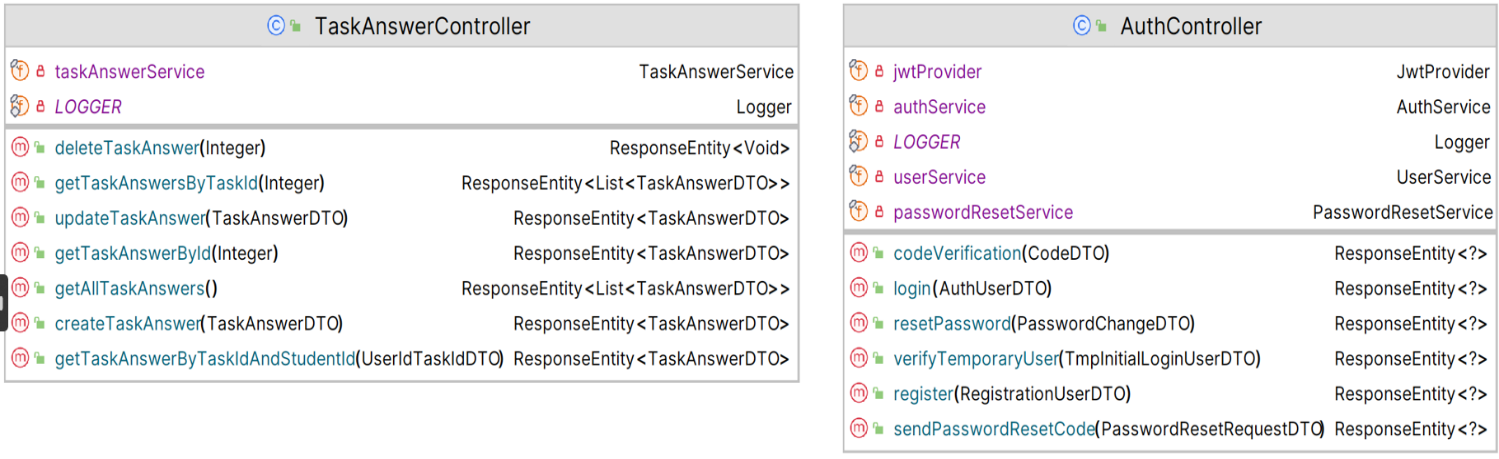


1. Диаграмма классов сервисов часть 2
   * 1. Диаграмма классов контроллеров

Контроллеры отвечают за обработку входящих HTTP-запросов, а также за управление птоком выполнения приложения. Они определяют методы, которые и отвечают за обработку запросов. Данные классы ниже (Рисунок 31-32).

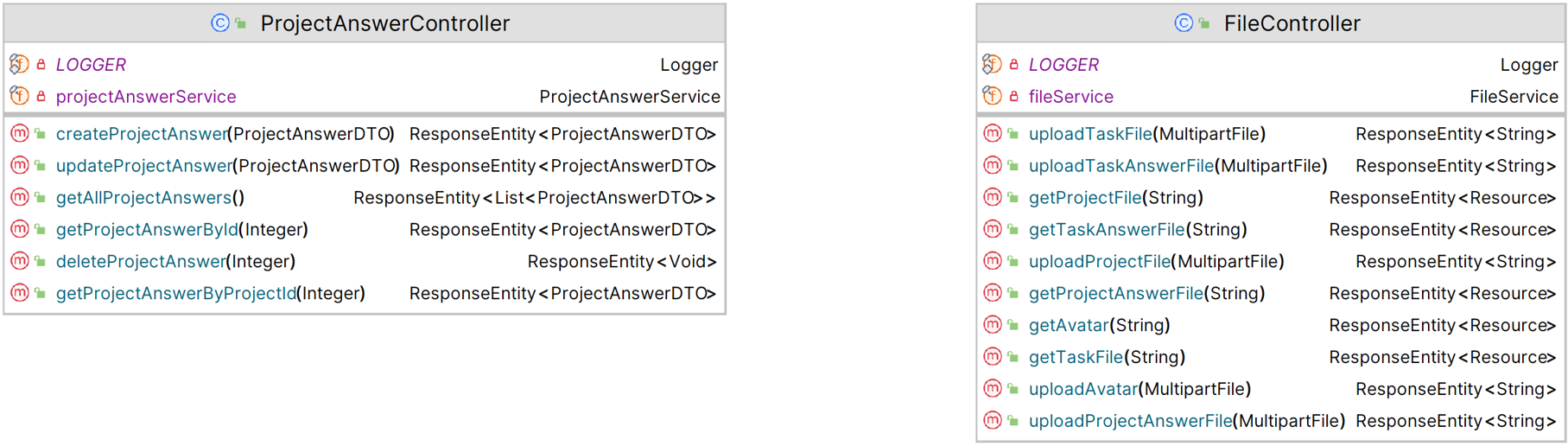






1. Диаграмма классов контроллеров часть 1



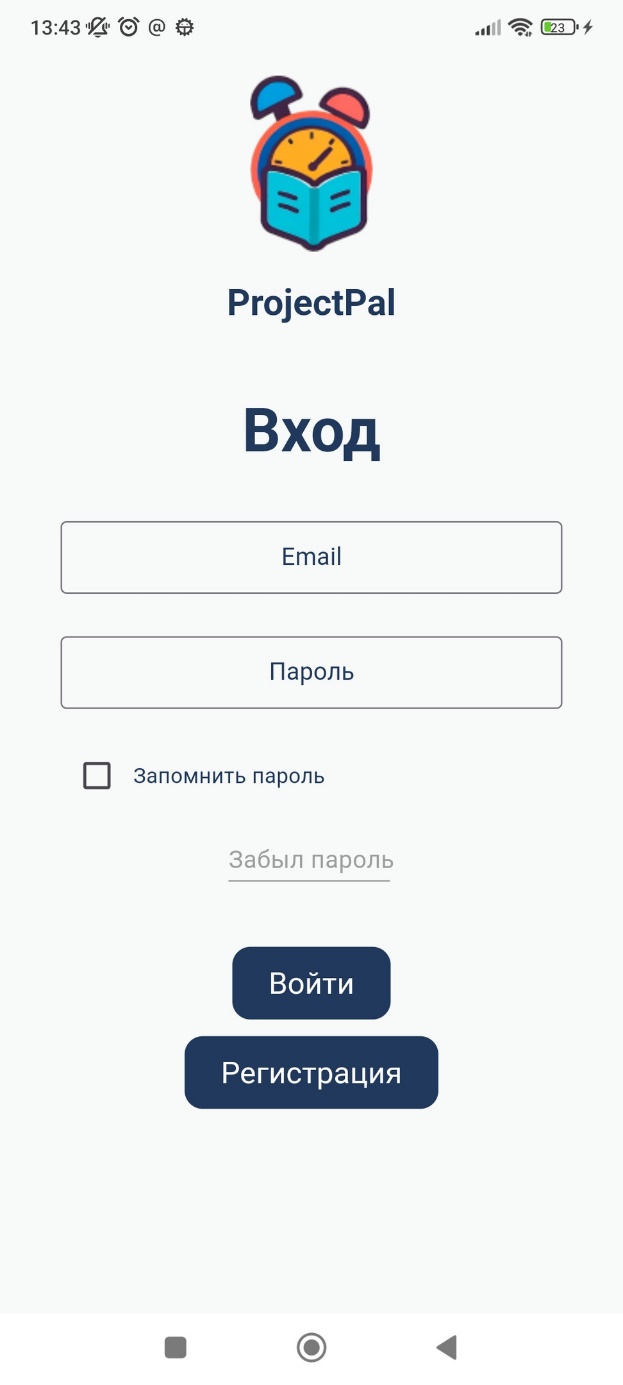


1. Диаграмма классов контроллеров часть 2
2. Клиентская часть
   1. Реализация

Для реализации клиентской части приложения было выбрано сочетание Dart и Flutter. Этот набор инструментов предоставляет разработчикам множество возможностей для создания мобильных приложений под Android и iOS. С помощью Flutter разработчики могут использовать широкий спектр виджетов для создания пользовательского интерфейса, а также получать доступ к аппаратным возможностям устройства и взаимодействовать с различными сервисами и API. Язык программирования Dart, в свою очередь, обеспечивает надежную и мощную основу для создания сложных и высокопроизводительных приложений.

Клиентская часть приложения построена на архитектуре, основанной на модели MVC (Model-View-Controller). Контроллеры обрабатывают входящие запросы и отправляют ответы пользователю, действуя как интерфейс между пользователем и бизнес-логикой приложения.

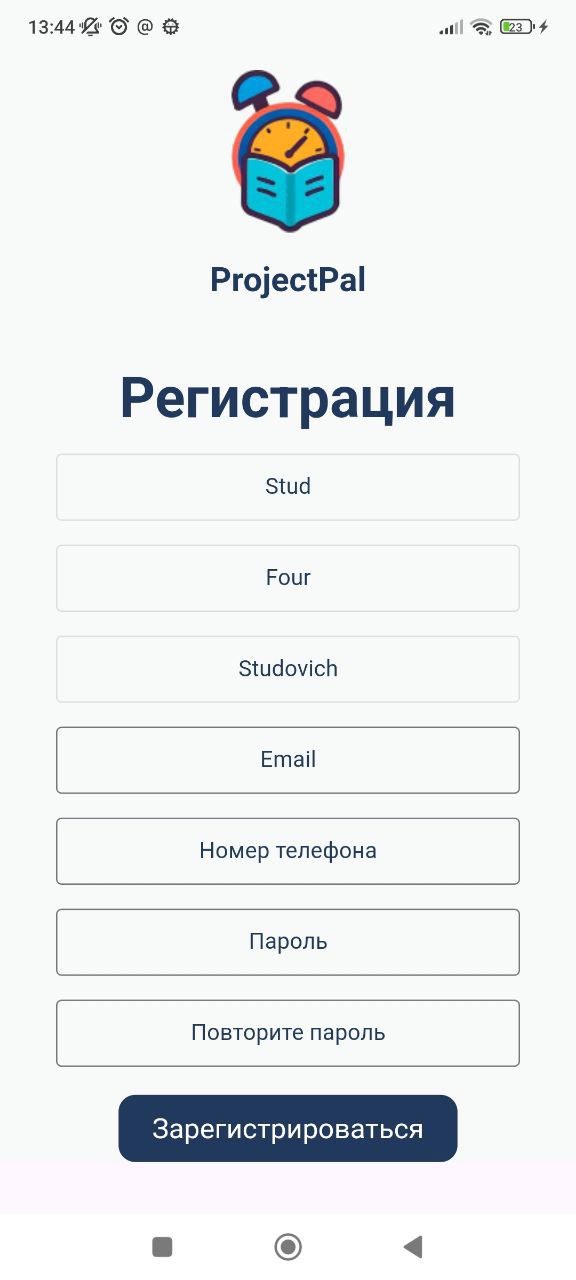
* 1. Основные экраны
     1. Экран авторизации



1. Экран авторизации

На данном экране отображены поля с вводом логина и пароля к аккаунту, кнопка для входа, кнопка для регистрации и кнопка для запоминания и восстановления пароля. При вводе несуществующих данных пользователю будет выведено сообщение об ошибке, указывающее на отсутствие профиля с такими данными.

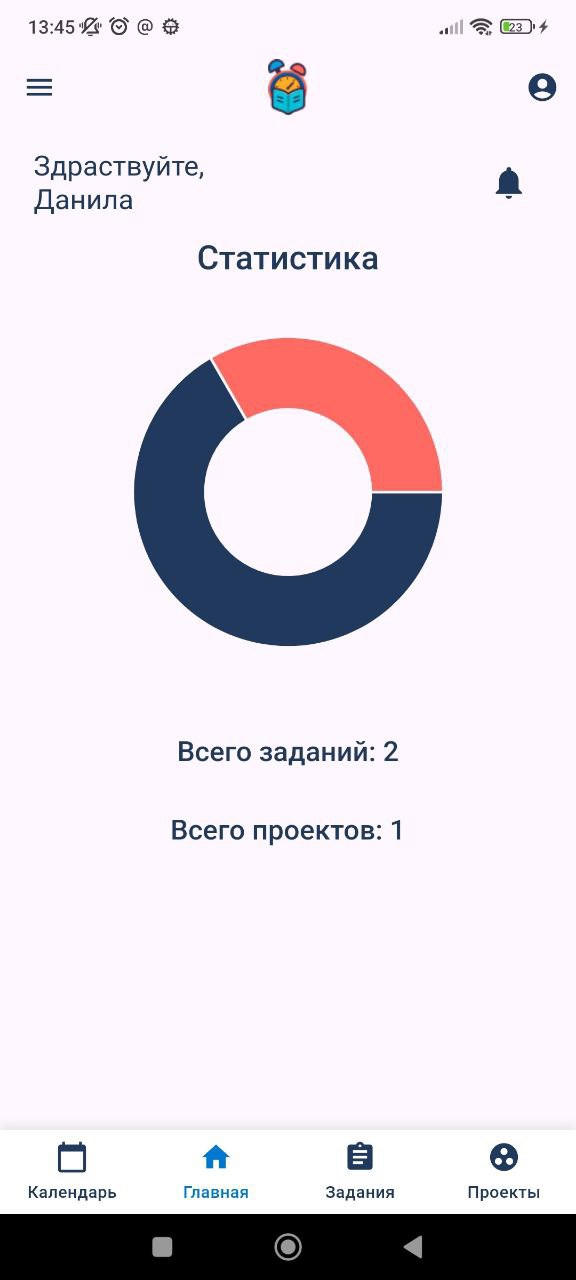
* + 1. Экран регистрации



1. Экран заполнения личных данных для регистрации

На данном экране отображены поля для ввода Email, номера телефона, пароля и его повтора, а также кнопка для возможности зарегистрироваться. Поля с ФИО пользователя недоступны для редактирования, поскольку изначально должны быть заполнены администратором в базе данных. Если пользователь совершает ошибку при вводе адреса электронной почты, система выдаст сообщение об ошибке, указывающее на то, что введенный адрес не существует.

* + 1. Экран главной страницы



1. Экран главной страницы

На этом экране отображены кнопки: для вызова сайд меню, для перехода по своему профилю, иконка колокольчика для уведомлений, а в центре экрана изображено кольцо с отображением количества задач и проектов у пользователя . В самом низу есть кнопки для перехода на другие экраны.

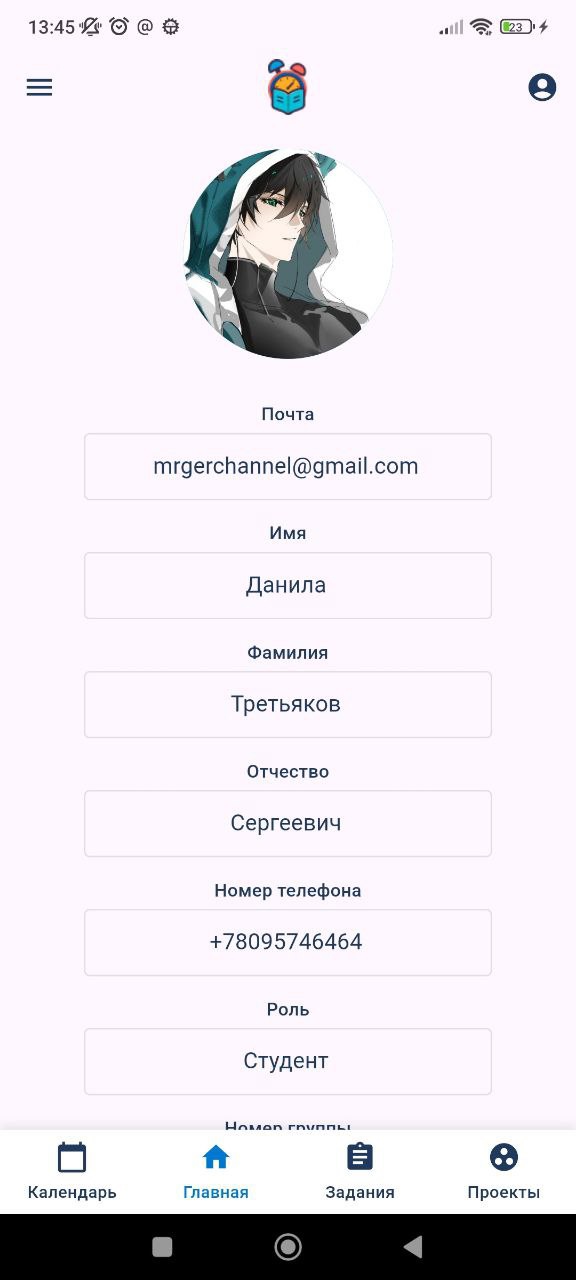
* + 1. Экран календаря



1. Экран календаря

На данном экране отображён календарь, представленный в виде подписанных кнопок, каждая из которых обозначает задания или проекты на соответствующий день. Здесь также присутствуют кнопки для пролистывания календаря на месяц вперед или назад, вызова бокового меню. Внизу экрана находятся кнопки для перехода на другие разделы.

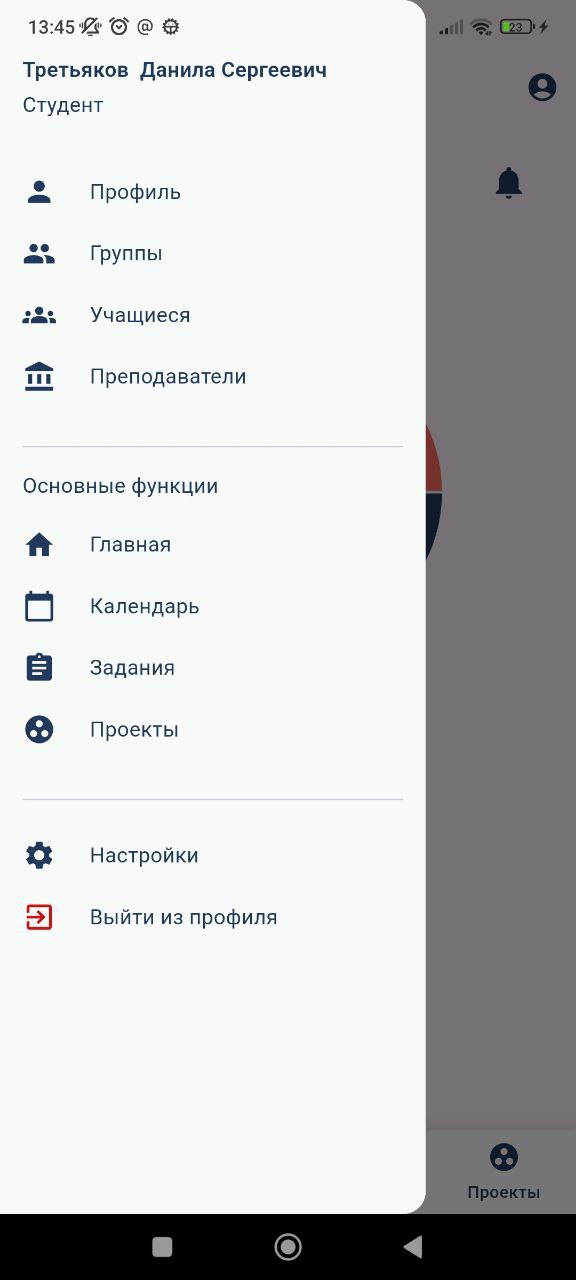
* + 1. Экран профиля



1. Экран профиля

На данном экране отображены редактируемые поля с информацией профиля такие как: ФИО, почта и номер телефона. Поля с ролью и номером группы редактируются только администраторами или преподавателями. Кнопки вызова сайд меню и подтверждения изменений данных профиля. В самом низу есть кнопки для перехода на другие экраны.

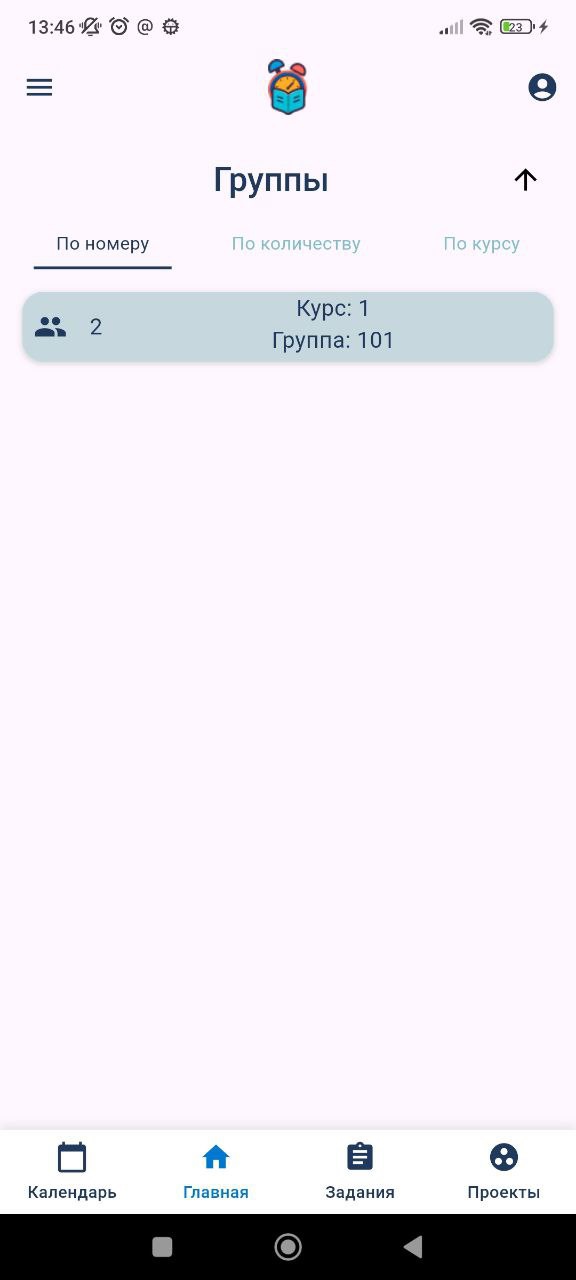
* + 1. Экран главной страницы



1. Экран главной страницы

На данном экране отображены кнопки для переходов на все доступные пользователю экраны, а также кнопка для выхода из профиля.

* + 1. Экран списка групп



1. Экран списка групп

На данном экране отображён список существующих групп, кнопки фильрации по количеству, по номеру, и по курсу, а так же кнопка для сортировки по возрастанию или убыванию. Вызова сайд меню слева сверху и так же в самом низу есть кнопки для перехода на другие экраны.

1. Тестирование
   1. Интеграционное тестирование

Интеграционное тестирование направлено на проверку взаимодействия различных компонентов или модулей системы. Цель этого этапа заключается в выявлении дефектов, возникающих при совместной работе отдельных частей системы, и в обеспечении корректного взаимодействия компонентов друг с другом. Тесты проводятся в контексте реального приложения, часто используя тестовые базы данных и другие внешние зависимости. Это позволяет проверить функциональность системы в условиях, максимально приближенных к реальным. Интеграционное тестирование помогает обнаружить проблемы на уровне взаимодействия компонентов, которые могут не проявляться при модульном тестировании отдельных частей системы.

Код для настройки окружения для интеграционных тестов:

**@Testcontainers**

**public** **abstract** **class** **IntegrationEnvironment** {

**@Container**

**public** **static** **final** PostgreSQLContainer<?> POSTGRE\_SQL\_CONTAINER = **new** PostgreSQLContainer<>("postgres:latest")

.withDatabaseName("ProjectPAL")

.withUsername("Admin")

.withPassword("VeryLongPassword");

**@DynamicPropertySource**

**static** **void** **postgresqlProperties**(DynamicPropertyRegistry registry) {

registry.add("spring.datasource.url", **POSTGRE\_SQL\_CONTAINER:**:getJdbcUrl);

registry.add("spring.datasource.username", **POSTGRE\_SQL\_CONTAINER:**:getUsername);

registry.add("spring.datasource.password", **POSTGRE\_SQL\_CONTAINER:**:getPassword);

}

}

https://cdn.discordapp.com/attachments/1212874143947554918/1252766401933086870/image.png?ex=66736921&is=667217a1&hm=6d133ca31edb482c9a9acac771e3d26f2a5c4d9d0c081ef2db4d5d445d7644a3&=

1. Результаты интеграционного тестирования
   1. Unit тестирование

Unit тестирование направлено на проверку отдельных модулей или компонентов системы. Каждый модуль тестируется изолированно, без взаимодействия с другими частями системы, что позволяет выявить ошибки и дефекты в конкретных функциональных блоках. Unit тесты обычно разрабатываются параллельно с кодом и часто выполняются автоматически. Цель данного вида тестирования — убедиться, что каждый компонент работает правильно сам по себе.

https://cdn.discordapp.com/attachments/1212874143947554918/1252766300682453032/image.png?ex=66736909&is=66721789&hm=fbffb066ad82e03dfb9bccc5096698af91b4b6006fe8eed299c8185d79e1eced&=

1. Результаты Unit тестирования

Заключение

В ходе выполнения данного курсового проекта был проведен тщательный анализ предметной области и существующих аналогов приложения. Были созданы макеты интерфейса, выбрана подходящая платформа, и построены UML диаграммы. Для контроля версий использовался репозиторий на GitHub. В процессе разработки приложения были выполнены следующие задачи:

* авторизация и регистрация пользователей;
* предоставление ознакомительной информации о приложении;
* просмотр и настройка профиля, включая изменение аватара;
* просмотр календаря и заданий (проектов), а также детальной информации о них;
* добавление учащимися ответов на задания и проекты, а также отслеживание состояния выполнения домашних заданий и проектов;
* управление заданиями и проектами преподавателями, включая создание, удаление, изменение, приписку и отписку учащихся, оценку и комментирование работ, а также проверку общего прогресса выполнения;
* управление профилями пользователей администраторами, включая создание, удаление и изменение;
* обработка жалоб администраторами.

Backend часть приложения и база данных были развернуты в контейнере Docker Compose на хостинге. Разработанное приложение полностью соответствует поставленным требованиям, и все задачи были успешно выполнены.

Список используемых источников

1. Spring Boot Documentation [Электронный ресурс]. – Режим доступа: https://spring.io/projects/spring-boot – Заглавие с экрана. – (дата обращения: 19.06.2024).
2. Spring Security Documentation [Электронный ресурс]. – Режим доступа: https://spring.io/projects/spring-security – Заглавие с экрана. – (дата обращения: 17.06.2024).
3. Spring Mail Documentation [Электронный ресурс]. – Режим доступа: https://spring.io/projects/spring-mail – Заглавие с экрана. – (дата обращения: 18.06.2024).
4. SpringDoc OpenAPI Documentation [Электронный ресурс]. – Режим доступа: https://springdoc.org – Заглавие с экрана. – (дата обращения: 04.06.2024).
5. Flyway Documentation [Электронный ресурс]. – Режим доступа: https://flywaydb.org/documentation/ – Заглавие с экрана. – (дата обращения: 09.06.2024).
6. Java JWT (JJWT) Documentation [Электронный ресурс]. – Режим доступа: https://github.com/jwtk/jjwt – Заглавие с экрана. – (дата обращения: 20.06.2024).
7. MapStruct Documentation [Электронный ресурс]. – Режим доступа: https://mapstruct.org/documentation/stable/reference/html/ – Заглавие с экрана. – (дата обращения: 20.06.2024).