Новосибирский Государственный Университет
Факультет Информационных Технологий

Техническое описание проекта по курсу ООАД

“NSUMedia бот”

Студент ФИТ НГУ
Петров Сергей
группа 21211

Студент ФИТ НГУ
Авдеев Виталий
группа 21211

Версия 0.4.8

Содержание

Техническое описание проекта по курсу ООАД

1. Введение

1.1 Цель

Данный документ представляет собой техническое описание проекта "NSUMedia бот" и содержит основные требования к разрабатываемой в рамках проекта программной системе и описание архитектуры программного решения.

1.2 Область действия

Документ разработан в рамках проекта "NSUMedia бот" на основе стандартного шаблона и предназначен для использования студентами ФИТ и преподавателями дисциплины ООАД.

1.3 Определения и сокращения

Таблица 1: Определения и сокращения

Термин	Описание
Бот	Специальные аккаунты в Telegram, созданные для того, чтобы автоматически обрабатывать и отправлять сообщения
LongPollingBot	Вид ботов, работающих по принципу периодического опроса Телеграм сервера о наличии обновлений.
WebhookBot	Вид ботов, работающих по принципу регистрации бота на Телеграм сервере и получении от него обновлений.

1.4 Ссылки

В тексте содержатся ссылки на следующие документы:

[1] diagrams.asta - файл с UML диаграммами;
[2] sample.env - файл с переменными среды, необходимыми для развертывания бота;
[3] docker-compose.yml - файл утилиты Docker Compose, содержащий инструкции для запуска и настройки сервисов;

Ссылки приводятся в виде [N], где N – номер документа в вышеприведенном списке.

1.5 Краткое описание

Содержание данного документа построено таким образом, чтобы дать ответ на следующие вопросы:

- Какие проблемы предметной области должен решать будущий программный продукт
- Посредством какой функциональности системы будут достигнуто решение проблем предметной области
- Какова архитектура программного решения

Описание предметной области и проблем, для решения которых предназначен будущий программный продукт, приведены в разделе 2.

Раздел 3 содержит описание требований к программному решению, раздел 4 – описание архитектуры выбранного решения.

2. Предметная область проекта

Обучение и учебный процесс студентов требуют доступа к обширным объемам учебного материала, не всегда имеющего в библиотеках или общедоступных сайтах. Студенты всегда ищут удобные способы доступа к этой информации, чтобы успешно учиться и готовиться к экзаменам. Однако поиск, организация и сортировка учебных материалов может быть вызовом, а существующие платформы и ресурсы не всегда удовлетворяют потребности студентов.

2.1 Существующие проблемы

Студенты часто сталкиваются с неудобствами при поиске и доступе к необходимым учебным материалам. Ресурсы, предоставляемые учебными заведениями, могут быть неструктурированными, и важные материалы могут быть разбросаны по разным источникам. Это приводит к потере времени и снижению эффективности обучения. Кроме того, не всегда существует простой способ обновления или дополнения материалов, а также обмена ими между студентами.

2.2 Предлагаемое решение

Мы предлагаем создать Telegram-бота, который будет решать эти проблемы, предоставляя студентам удобный доступ к учебным материалам, организованным по курсам, семестрам и дисциплинам. Студенты смогут легко загружать, обновлять и обмениваться учебными материалами через этого бота. Наш бот станет надежным партнером студентов в учебном процессе, помогая им легко находить учебные ресурсы и управлять ими.

3. Требования к программному решению

Данный раздел описывает требования к программной системе, разрабатываемой в рамках проекта "NSUMedia бот".

3.1 Роли

Основные роли и их иерархия описаны в *diagrams/1. Use-case модель/Основные UC*[1] (см. Рис.1).

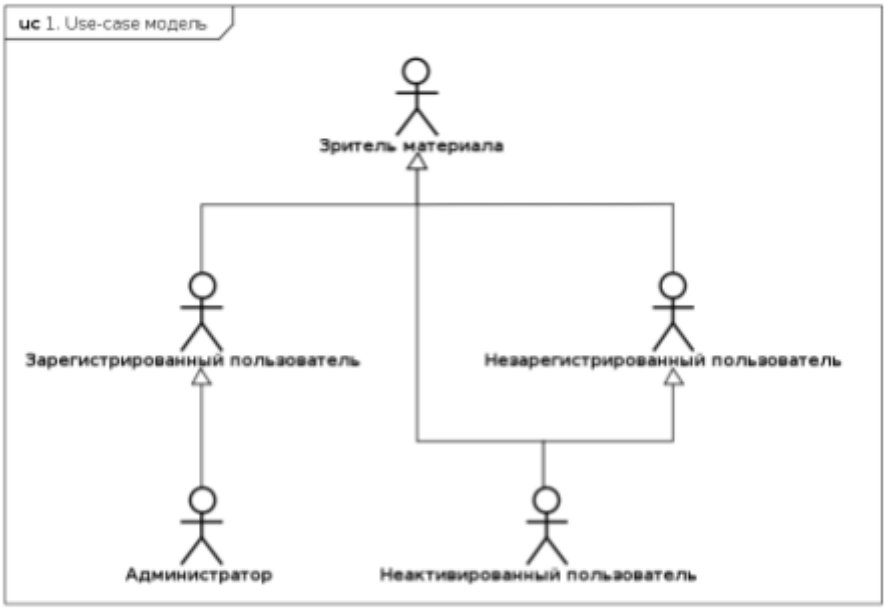


Рис.1. Иерархия ролей

1. **Зритель материала.** Зритель материала - фиктивный актёр, имеющий возможность просматривать учебный материал. От зрителя материала наследуются все остальные роли.

2. **Незарегистрированный пользователь.** Пользователь без регистрации может начать процесс регистрации и получить больше привилегий.

3. **Неактивированный пользователь.** Неактивированный пользователь - промежуточная роль между незарегистрированным и зарегистрированным пользователем. Возникает при начале регистрации и заканчивается после активации аккаунта. Имеет возможность активироваться. Также наследует возможности незарегистрированного пользователя, т.е. способен заново начать регистрацию.

4. **Зарегистрированный пользователь.** Зарегистрированный пользователь получает возможность предлагать новый материал, адресуя его администраторам.

5. **Администратор.** Администратор подписан на рассылку ботом нового материала для добавления, может принимать или отклонять его. Также имеет возможность добавлять или удалять материал напрямую.

3.2 Функциональные требования для роли «Зритель материала»

Use-case и их сценарии для роли «Зритель материала» описаны в *diagrams/1. Use-case модель/UC зрителя материала*[1] (см. Рис. 2).

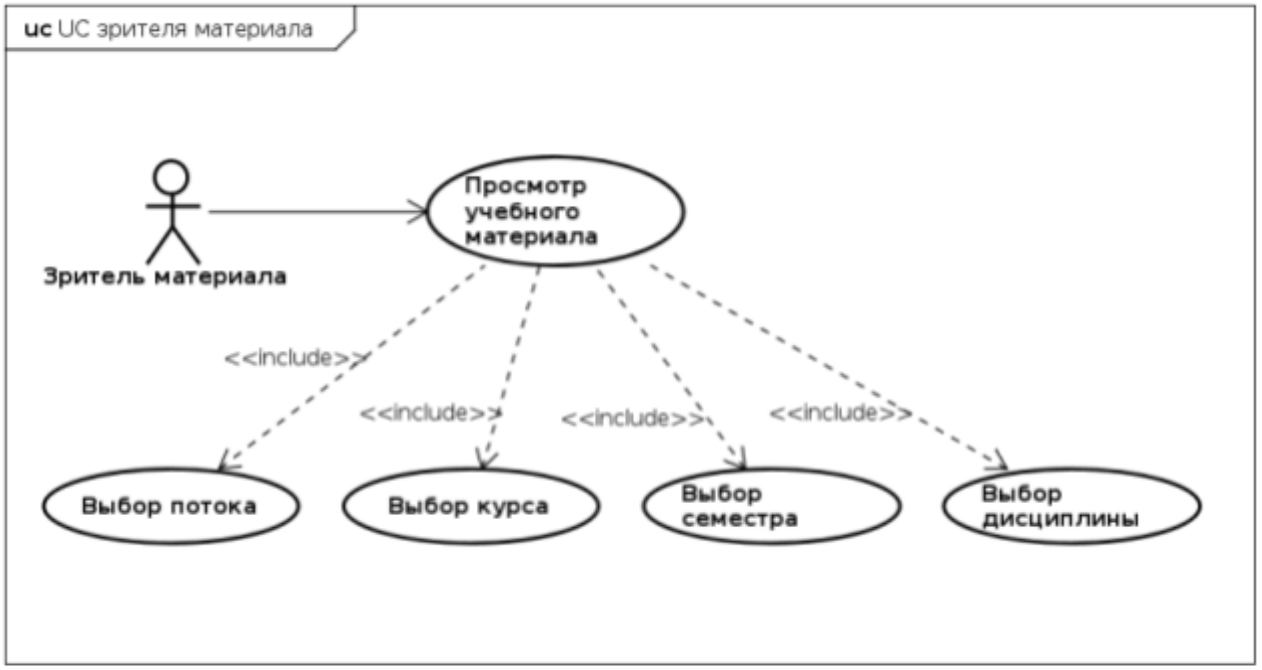


Рис. 2. Use cases Зрителя материала

3.2.1 Просмотр учебного материала

Описывает процесс просмотра учебного материала на платформе. Для удобства нахождения необходимого пособия Зритель материала последовательно выбирает ту или иную кнопку выбора потока/курса/семестра/дисциплины (см. Рис 3-7).

Рис. 3. Основной сценарий просмотра материала

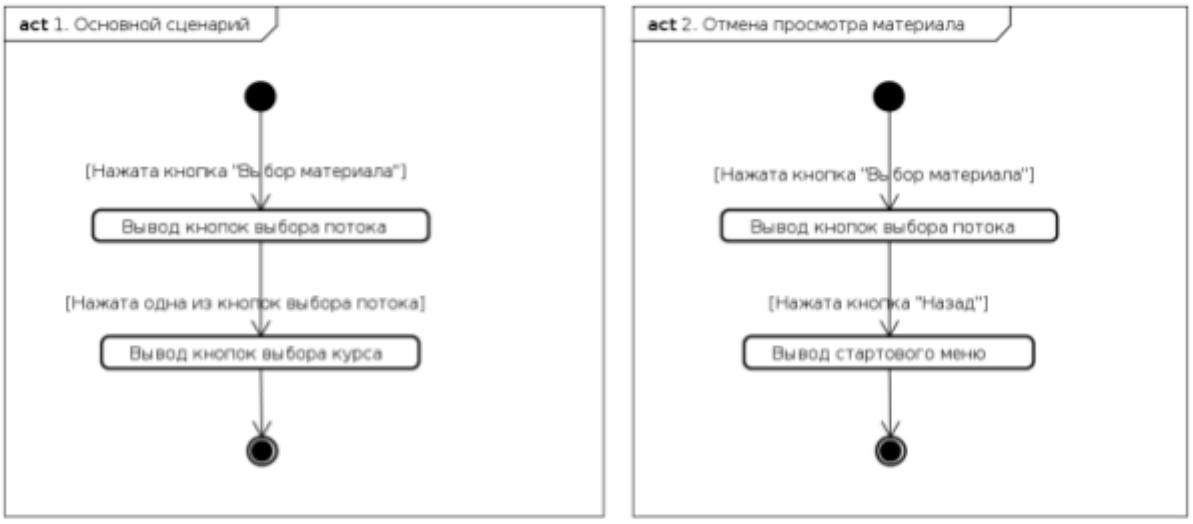


Рис. 4. Основной сценарий при выборе потока и сценарий отмены

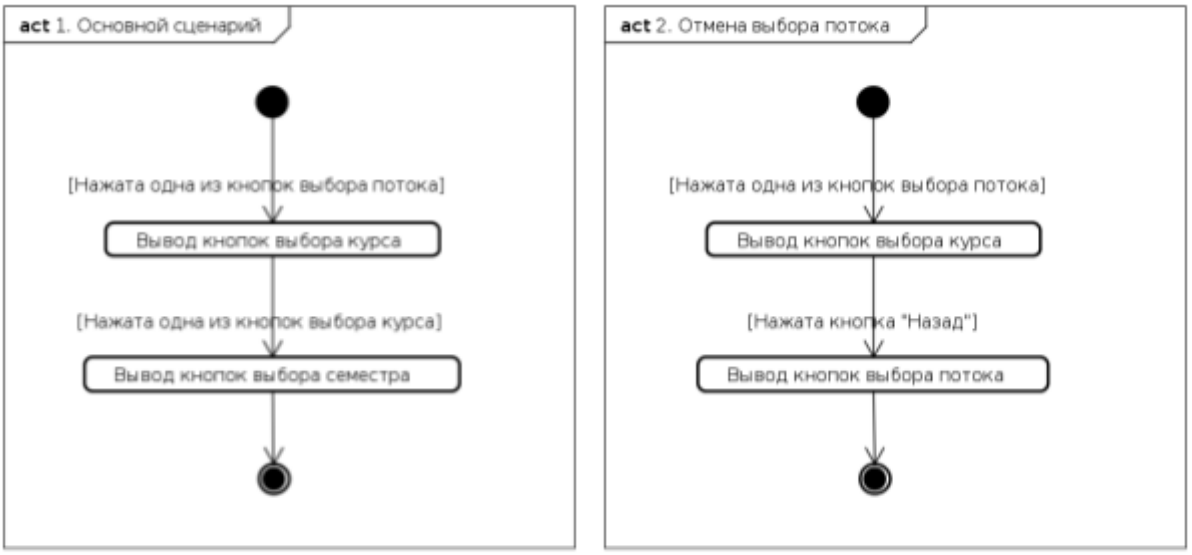


Рис. 5. Основной сценарий при выборе курса и сценарий отмены

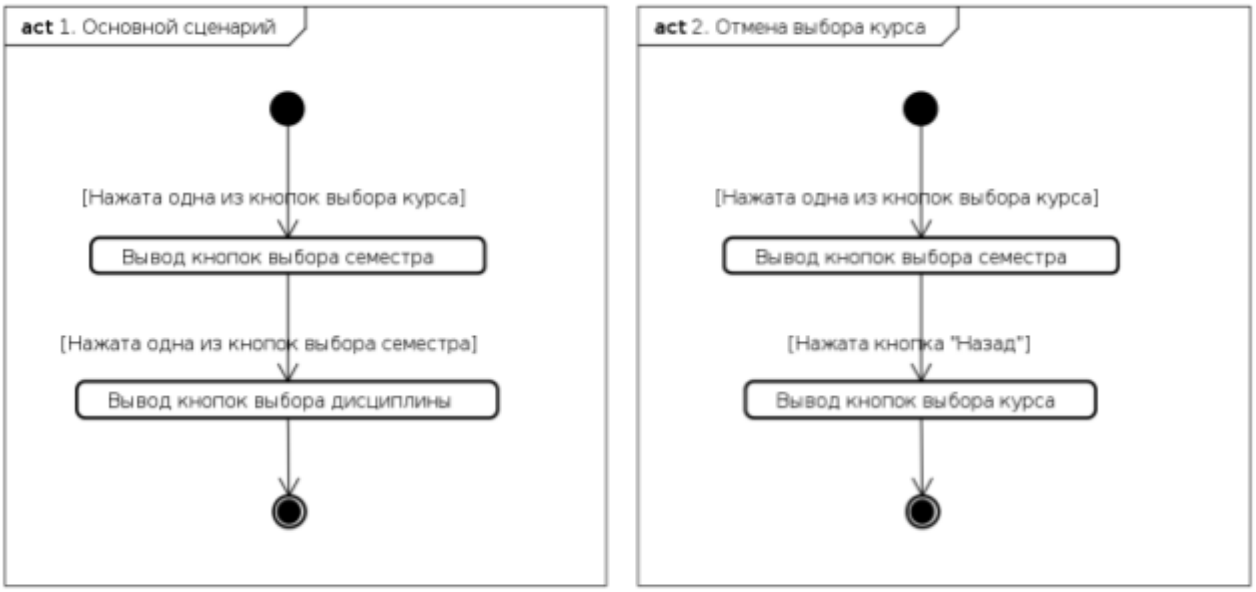


Рис. 6. Основной сценарий при выборе семестра и сценарий отмены

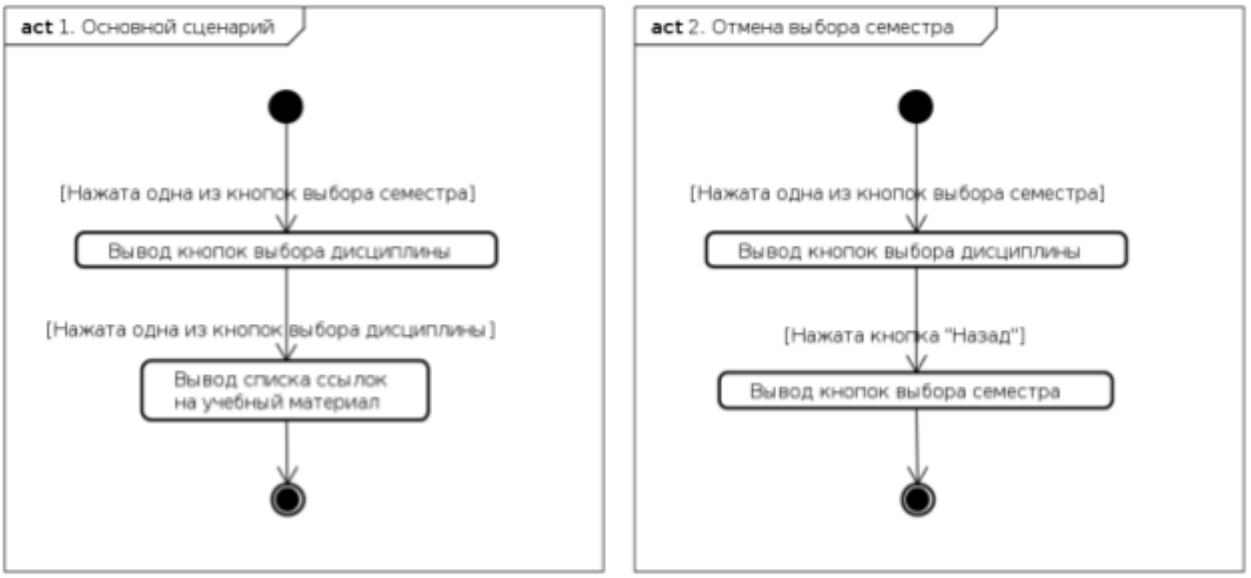


Рис. 7. Основной сценарий при выборе дисциплины и сценарий отмены

3.3 Функциональные требования для роли «Незарегистрированный пользователь»

Use-case и их сценарии для роли «Незарегистрированный пользователь» описаны в *diagrams/1. Use-case модель/UC незарегистрированного пользователя*[1] (см. Рис. 8).

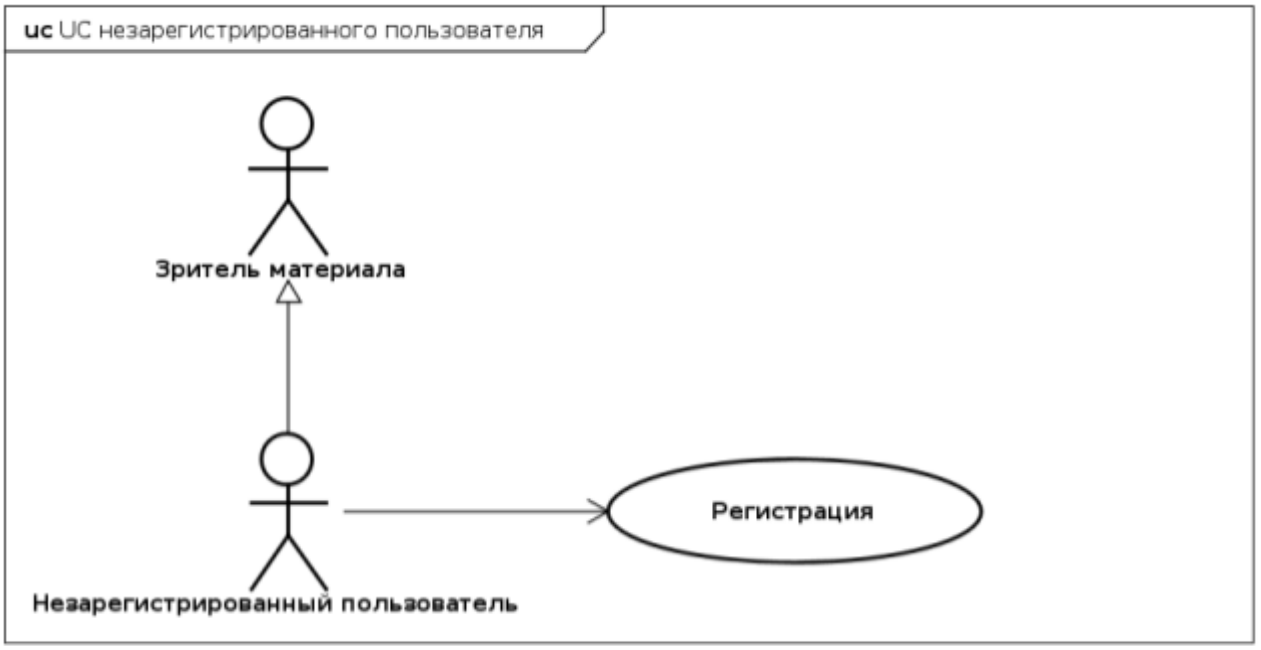


Рис. 8. Use cases Незарегистрированного пользователя

3.3.1 Регистрация

Незарегистрированный пользователь имеет возможность зарегистрироваться для получения доступа к дополнительным функциям системы (см. Рис. 9). Для регистрации необходимо ввести почту НГУ и подтвердить ее при помощи токена из письма.

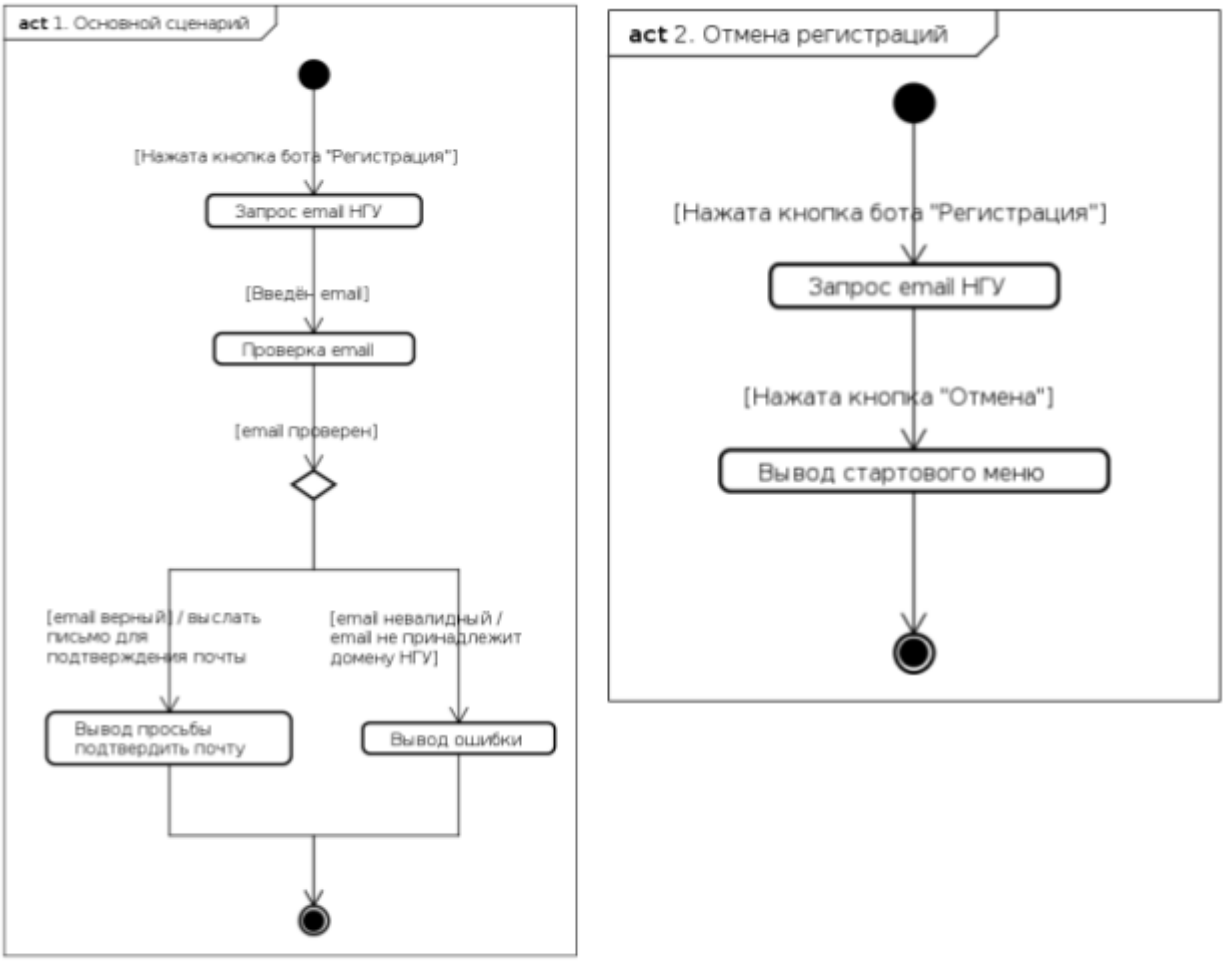


Рис. 9. Основной сценарий регистрации и сценарий отмены

3.4 Функциональные требования для роли «Неактивированный пользователь»

Use-case и их сценарии для роли «Неактивированный пользователь» описаны в *diagrams/1. Use-case модель/UC неактивированного пользователя*[1] (см. Рис. 10).

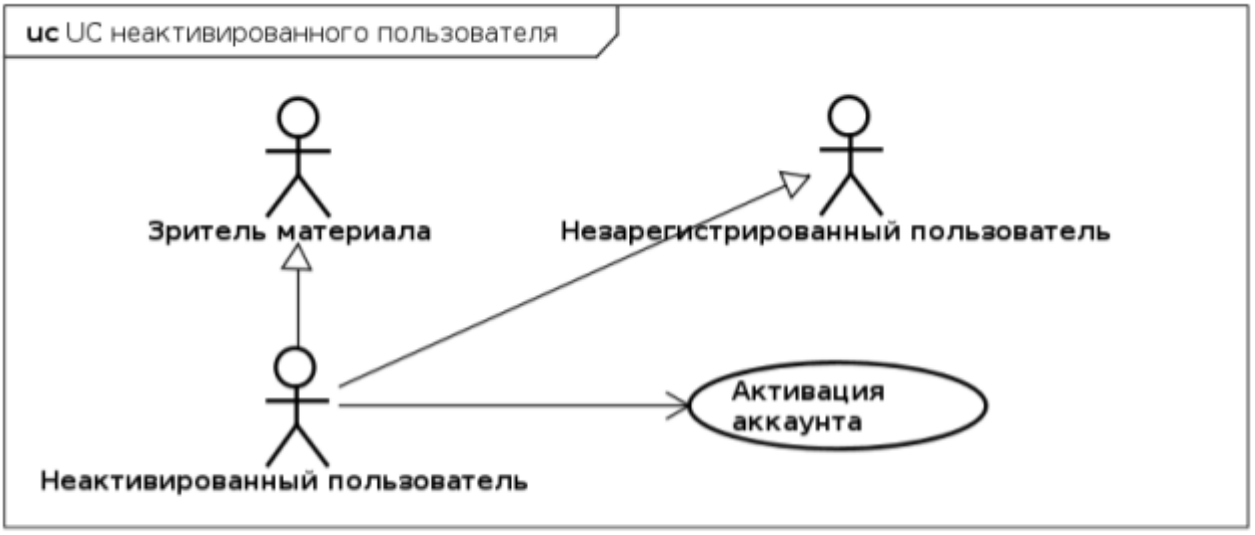


Рис. 10. Use cases Неактивированного пользователя

3.4.1 Активация аккаунта

Неактивированный пользователь может активировать аккаунт при помощи токена, отправленного на ранее введенную почту (см. Рис. 11).

Рис. 11. Основной сценарий активации аккаунта

3.5 Функциональные требования для роли «Зарегистрированный пользователь»

Use-case и их сценарии для роли «Зарегистрированный пользователь» описаны в *diagrams/1. Use-case модель/UC зарегистрированного пользователя*[1] (см. Рис. 12).

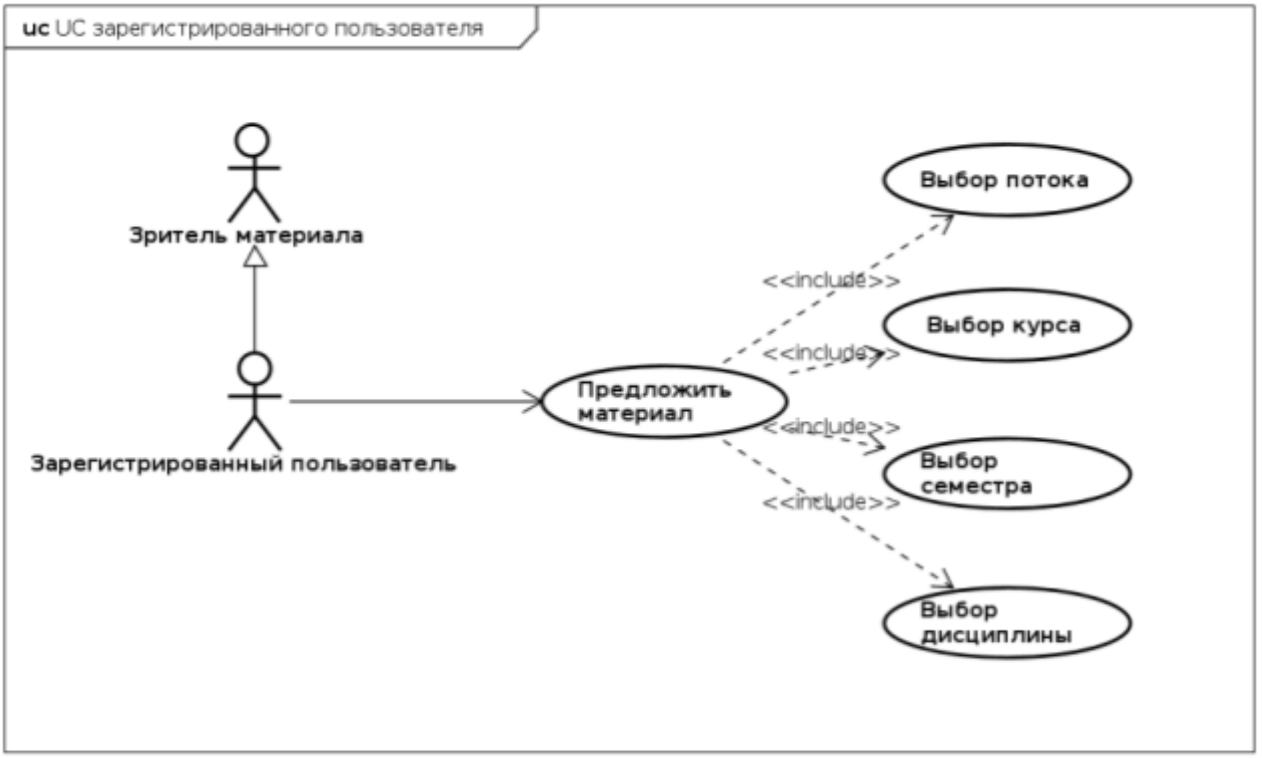


Рис. 12. Use cases Зарегистрированного пользователя

3.5.1 Предложение материала

Зарегистрированный пользователь может предложить новый учебный материал или ресурс для системы. Необходимо последовательно выбирать ту или иную кнопку выбора потока/курса/семестра/дисциплины и загрузить файл пособия (см. Рис 4-7, 13).

Рис. 13. Основной сценарий предложения материала

3.6 Функциональные требования для роли «Администратор»

Use-case и их сценарии для роли ««Администратор»» описаны в *diagrams/1. Use-case модель/UC администратора*[1] (см. Рис. 14).

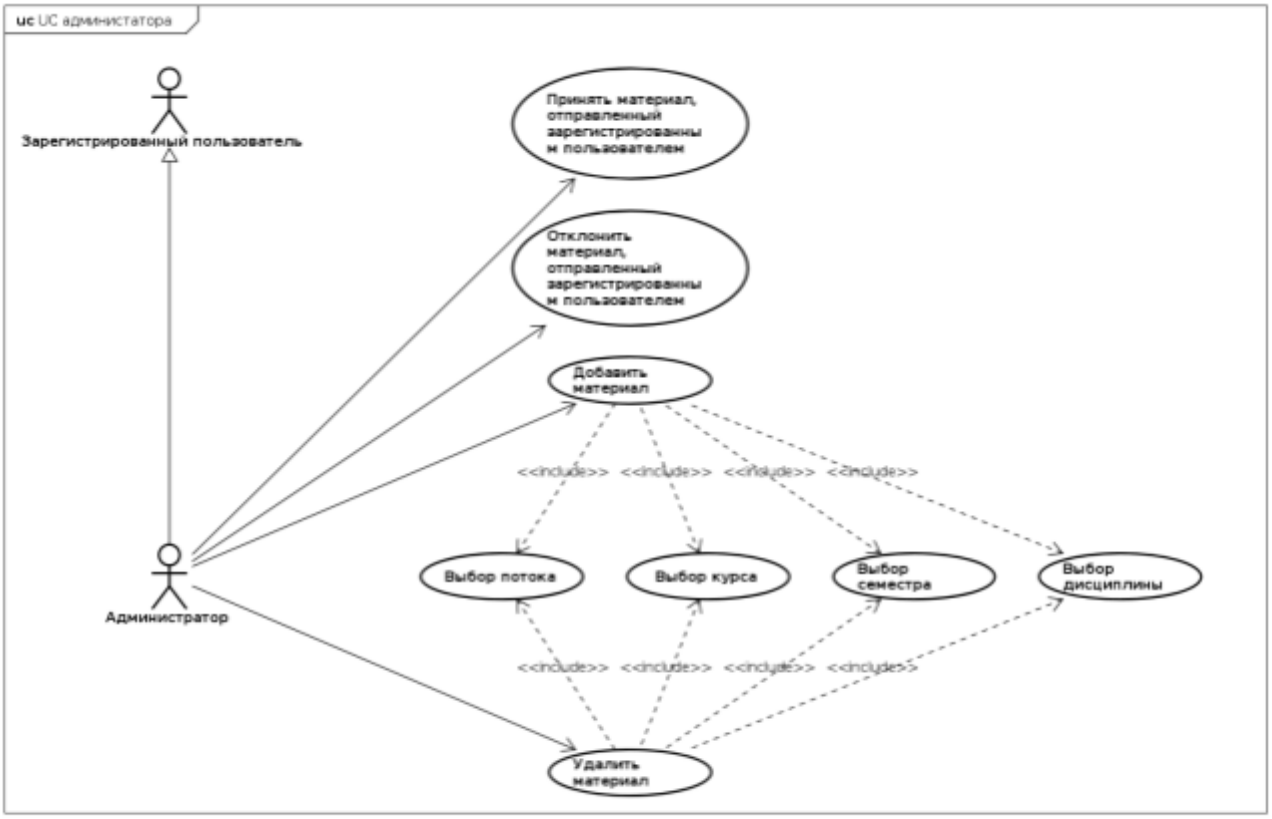


Рис. 14. Use cases Администратора

3.6.1 Принятие и отклонение материала, отправленного пользователем

Администратор может эффективно управлять материалами, предоставленными пользователями, и обеспечивает процесс принятия или отклонения материалов в зависимости от их соответствия критериям и требованиям ресурса (см. Рис. 15).

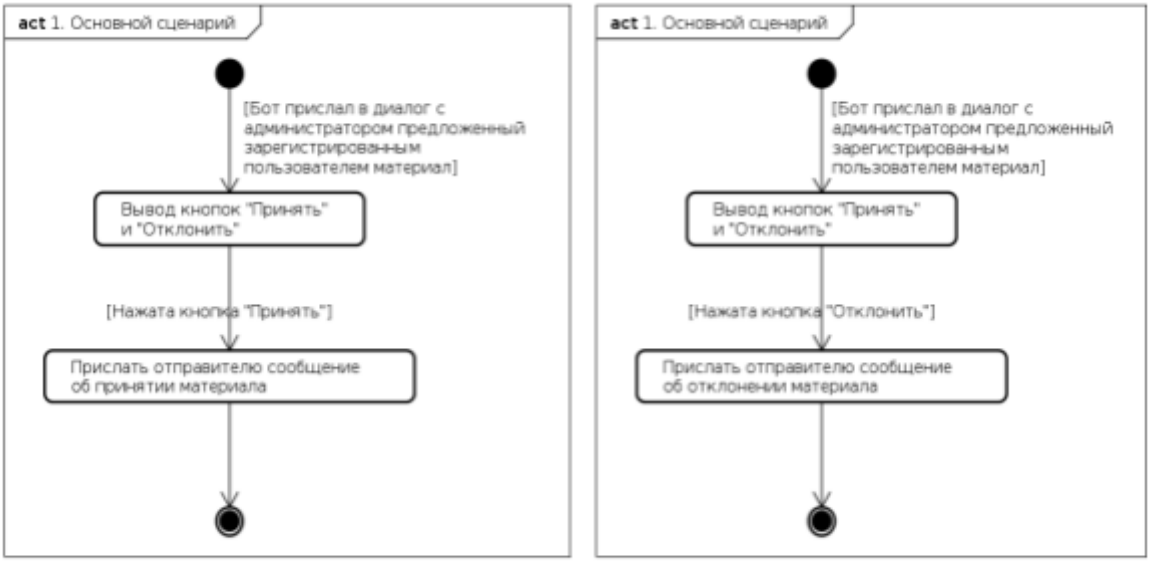


Рис.15. Основные сценарии принятия и отклонения материала пользователя

3.6.2 Добавление материала

Администратору предоставлена возможность добавлять материал на диск через бота. Необходимо указать поток, курс, семестр и дисциплину и прикрепить файл пособия. После подтверждения файл запишется на диск (см. Рис 4-7, 16).

Рис.16. Основной сценарий добавления материала

3.6.3 Удаления материала

Администратору предоставлена возможность удалять материал с диска через бота. Необходимо указать поток, курс, семестр, дисциплину и файл пособия. После подтверждения файл удалится с диска (см. Рис 4-7, 17).

Рис.17. Основной сценарий удаления материала

3.7 Нефункциональные требования

3.7.1 Производительность

Ресурсы (например, вычислительная мощность и хранилище) должны быть масштабируемыми и достаточными для обслуживания пользователей в количестве студентов ФИТ НГУ. Хранилище диска должно иметь возможность хранить учебный материал в объеме 50 ГБ.

3.7.2 Необходимость миграции данных из legacy системы

Данные из legacy системы должны быть корректно и безопасно перенесены в новую систему, сохраняя целостность и доступность. Должна быть выстроена необходимая файловая структура диска для использования ботом.

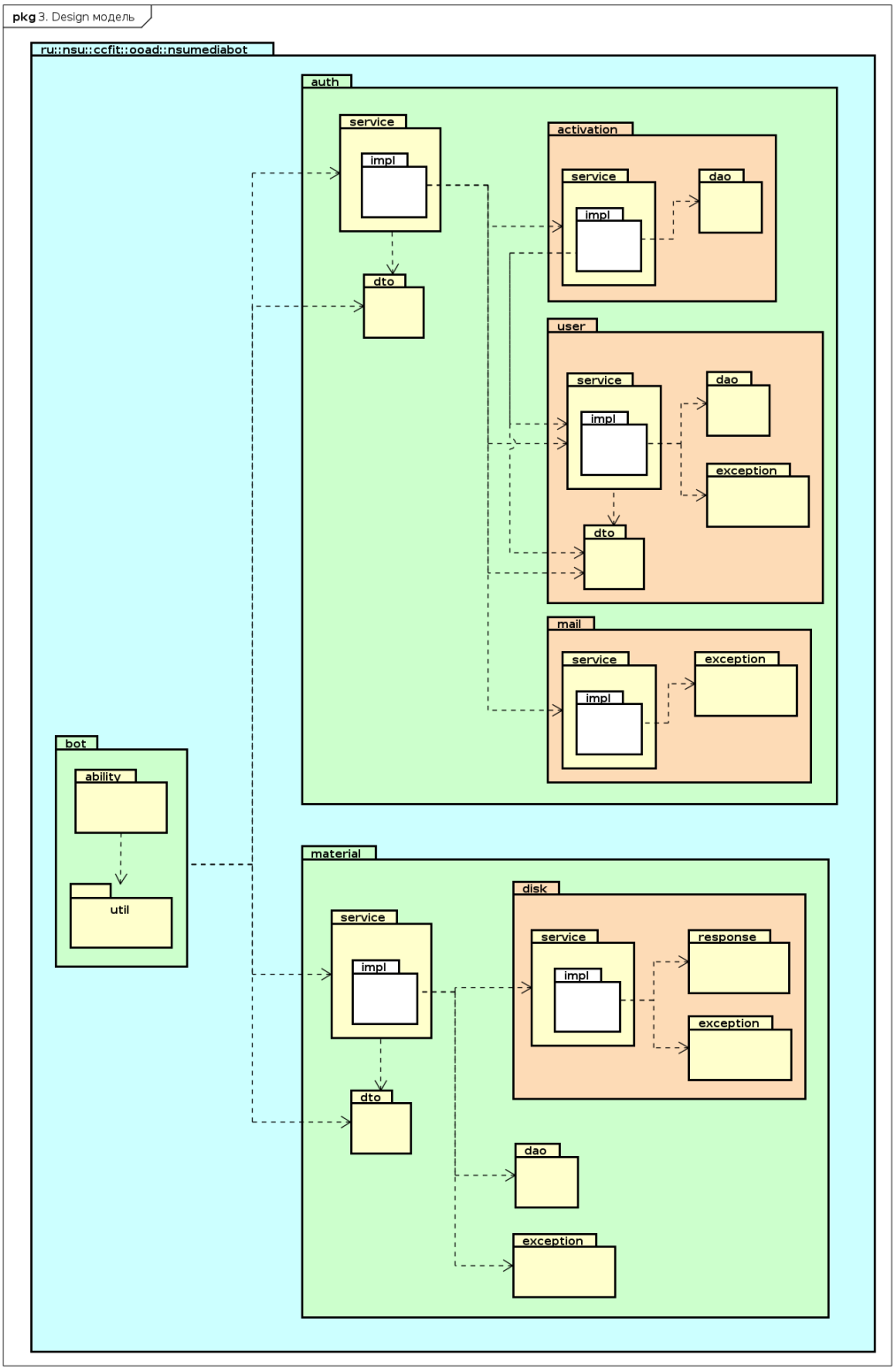
3.7.3 Использование Yandex Диск API

Бот должен интегрироваться с API Yandex диска для возможности хранения, получения и обновления учебного материала.

4. Обзор архитектуры

Этот раздел описывает архитектуру системы.

4.1 Компонентная модель системы



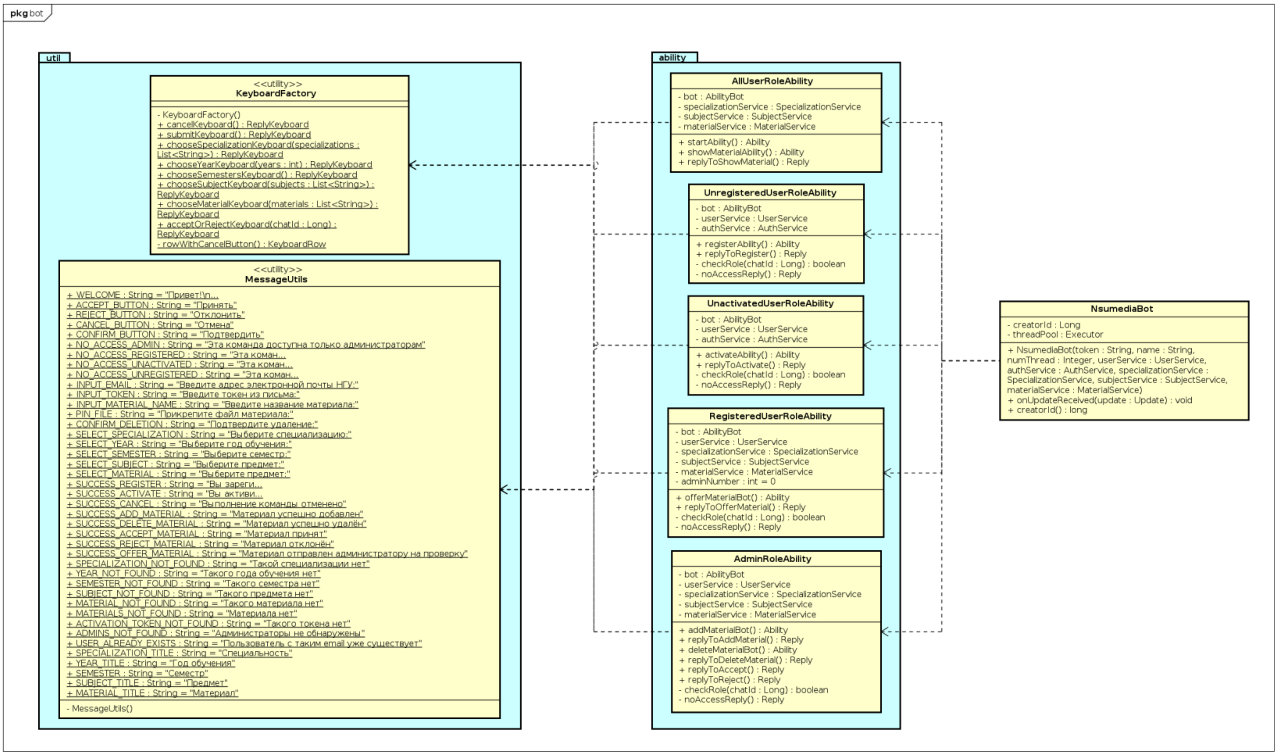
Bot - компонент, ответственный за реализацию и функционирование телеграм-бота.

Auth - компонент, предназначенный для регистрации пользователей и активации их учетных записей с использованием UUID токенов, отправляемых на почту.

Material - компонент, необходимый для просмотра, добавления и удаление учебных материалов.

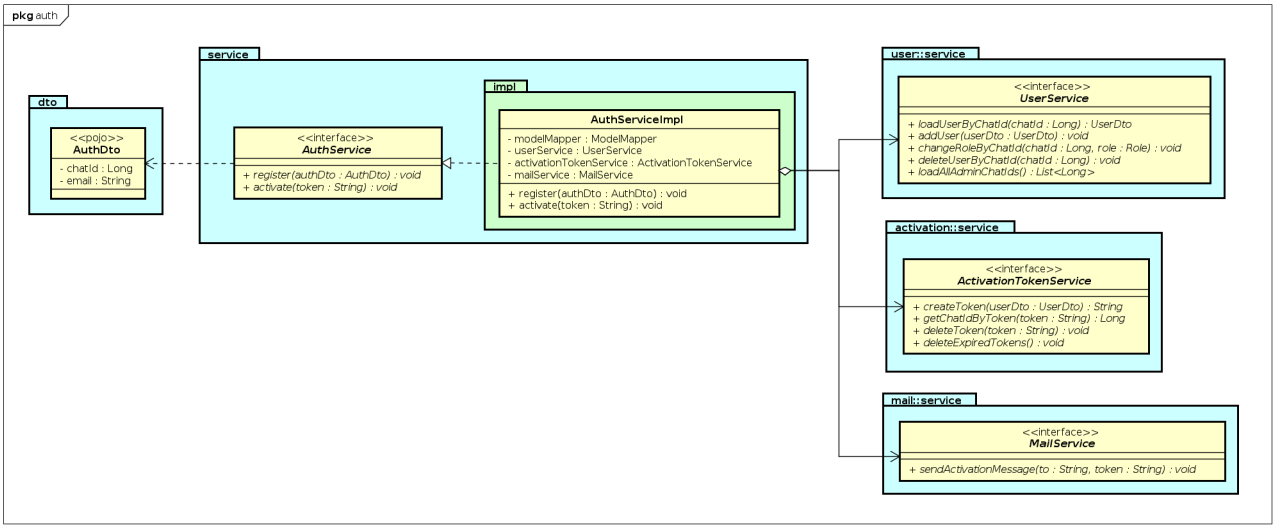
4.1.1 Компонент “Bot”

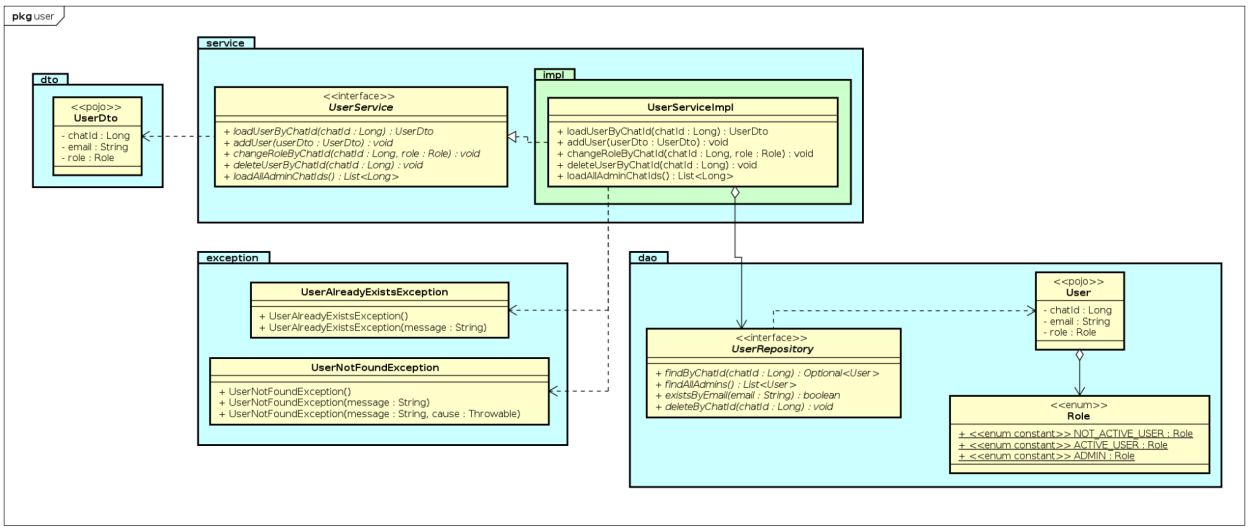
Компонент предназначен для приема обновлений от Телеграм сервера и их обработки в нескольких потоках. Также он отвечает за распознавание команд, поддержания сценариев общения, хранение промежуточных данных и отображение клавиатур для пользователей.

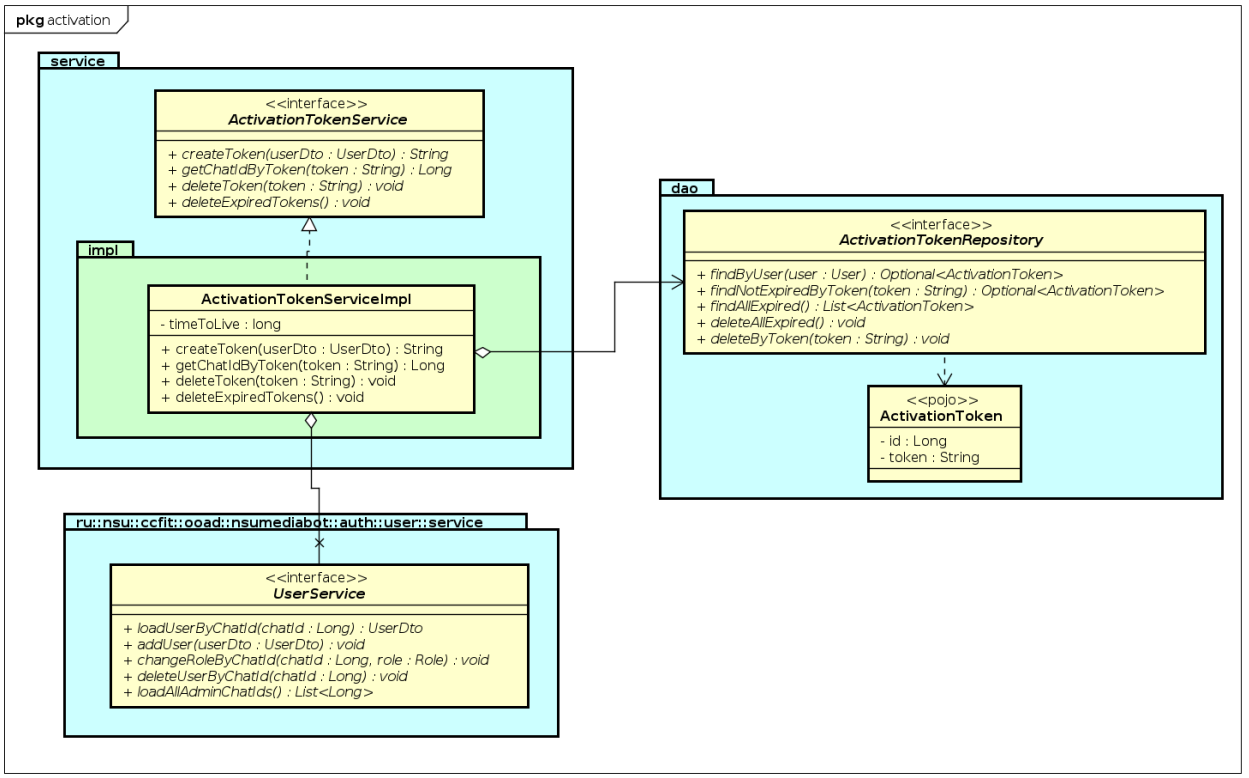


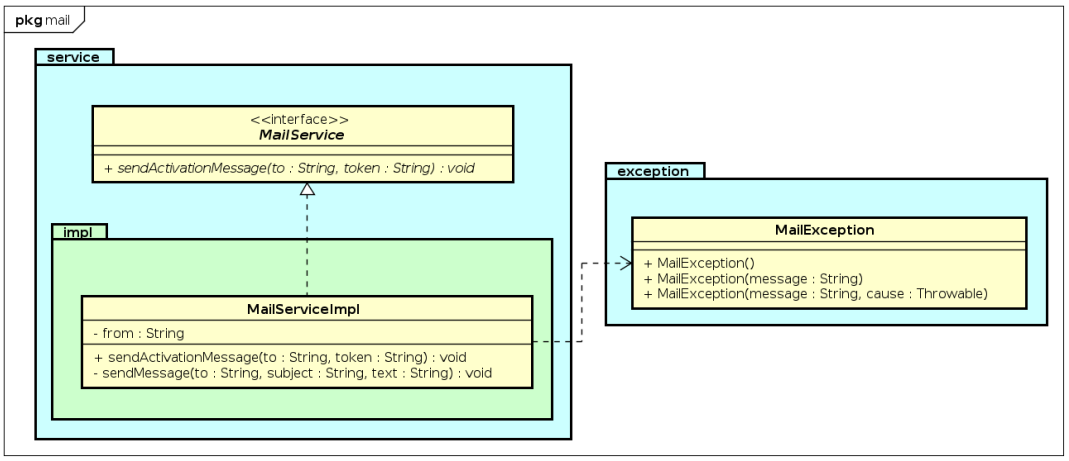
4.1.2 Компонент "Authentication”

Компонент предоставляет пользователям возможность регистрации и активации аккаунта. Включает в себя несколько подкомпонентов, ответственных за хранения пользователей в базе данных, за генерацию и валидацию токена активации и за отправку писем с токеном на почту.



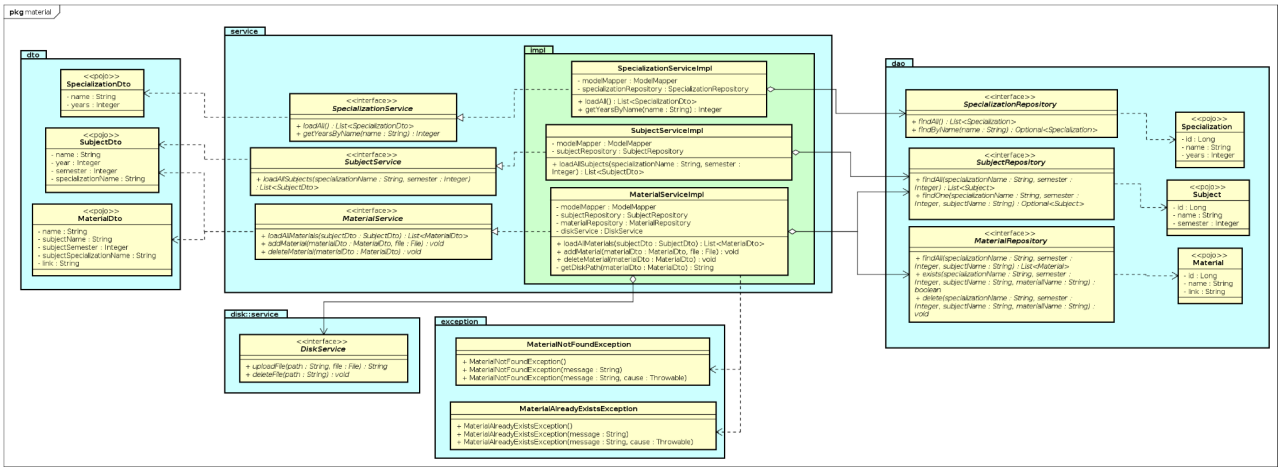

pkg user
service
dto
<<pojo>>
UserDto
- chatId : Long
- email : String
- role : Role
<<interface>>
UserService
+ loadUserByChatId(chatId : Long) : UserDto
+ addUser(userDto : UserDto) : void
+ changeRoleByChatId(chatId : Long, role : Role) : void
+ deleteUserByChatId(chatId : Long) : void
+ loadAllAdminChatIds() : List<Long>
impl
UserServiceImpl
+ loadUserByChatId(chatId : Long) : UserDto
+ addUser(userDto : UserDto) : void
+ changeRoleByChatId(chatId : Long, role : Role) : void
+ deleteUserByChatId(chatId : Long) : void
+ loadAllAdminChatIds() : List<Long>
exception
UserAlreadyExistsException
+ UserAlreadyExistsException()
+ UserAlreadyExistsException(message : String)
UserNotFoundException
+ UserNotFoundException()
+ UserNotFoundException(message : String)
+ UserNotFoundException(message : String, cause : Throwable)
dao
<<interface>>
UserRepository
+ findByChatId(chatId : Long) : Optional<User>
+ findAllAdmins() : List<User>
+ existsByEmail(email : String) : boolean
+ deleteByChatId(chatId : Long) : void
<<pojo>>
User
- chatId : Long
- email : String
- role : Role
<<enum>>
Role
+ <<enum constant>> NOT_ACTIVE_USER : Role
+ <<enum constant>> ACTIVE_USER : Role
+ <<enum constant>> ADMIN : Role

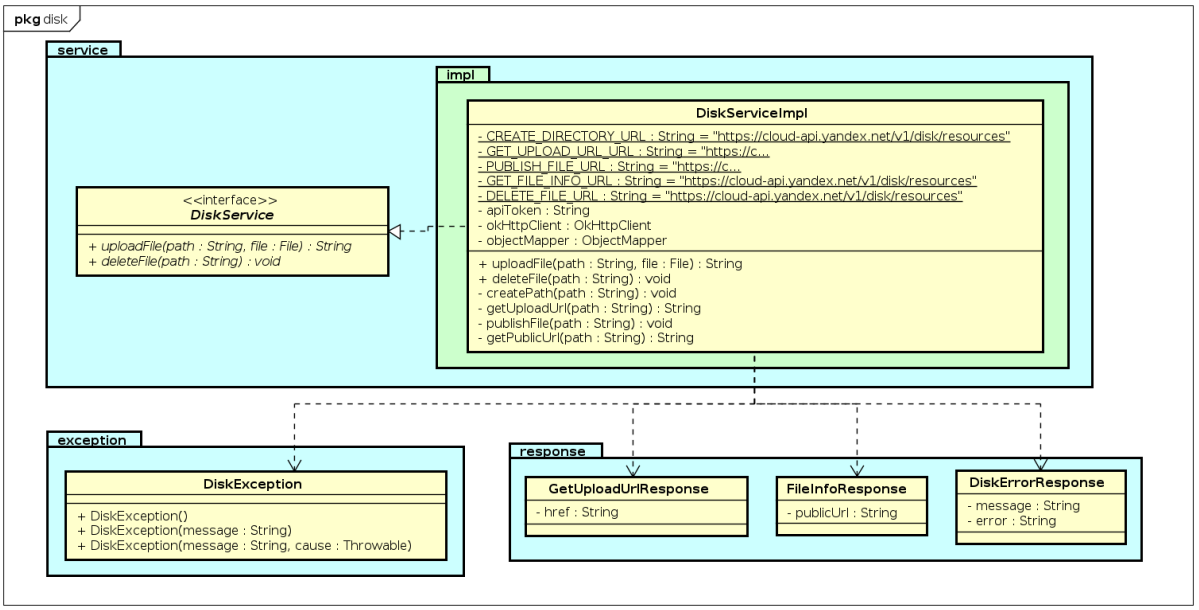

pkg activation
service
<<interface>>
ActivationTokenService
+ createToken(userDto : UserDto) : String
+ getChatIdByToken(token : String) : Long
+ deleteToken(token : String) : void
+ deleteExpiredTokens() : void
impl
ActivationTokenServiceImpl
- timeToLive : long
+ createToken(userDto : UserDto) : String
+ getChatIdByToken(token : String) : Long
+ deleteToken(token : String) : void
+ deleteExpiredTokens() : void
dao
<<interface>>
ActivationTokenRepository
+ findByUser(user : User) : Optional<ActivationToken>
+ findNotExpiredByToken(token : String) : Optional<ActivationToken>
+ findAllExpired() : List<ActivationToken>
+ deleteAllExpired() : void
+ deleteByToken(token : String) : void
<<pojo>>
ActivationToken
- id : Long
- token : String
ru::nsu::ccfit::ooad::nsumediabot::auth::user::service
<<interface>>
UserService
+ loadUserByChatId(chatId : Long) : UserDto
+ addUser(userDto : UserDto) : void
+ changeRoleByChatId(chatId : Long, role : Role) : void
+ deleteUserByChatId(chatId : Long) : void
+ loadAllAdminChatIds() : List<Long>



4.1.3 Компонент “Material”

Компонент позволяет получать списки специализаций, предметов и материалов, а также взаимодействует с API Яндекс Диска для загрузки и удаления файлов материалов.





4.2 Компоненты сторонних производителей

telegrambots-abilities - библиотека для создания телеграм ботов, реализующая новую абстракцию AbilityBot, позволяющую избавиться от шаблонного кода, например, распознавание команд, создание сценариев и др.

Spring Data JPA - фреймворк для работы с JPA и обеспечения абстракции уровня доступа к данным.

Liquibase - независимая от базы данных библиотека для отслеживания, управления и применения изменений схемы базы данных.

OkHttp - HTTP клиент, используемый для обращений к API Яндекс Диска.

Jackson - библиотека для сериализации/десериализации Java объектов в JSON.

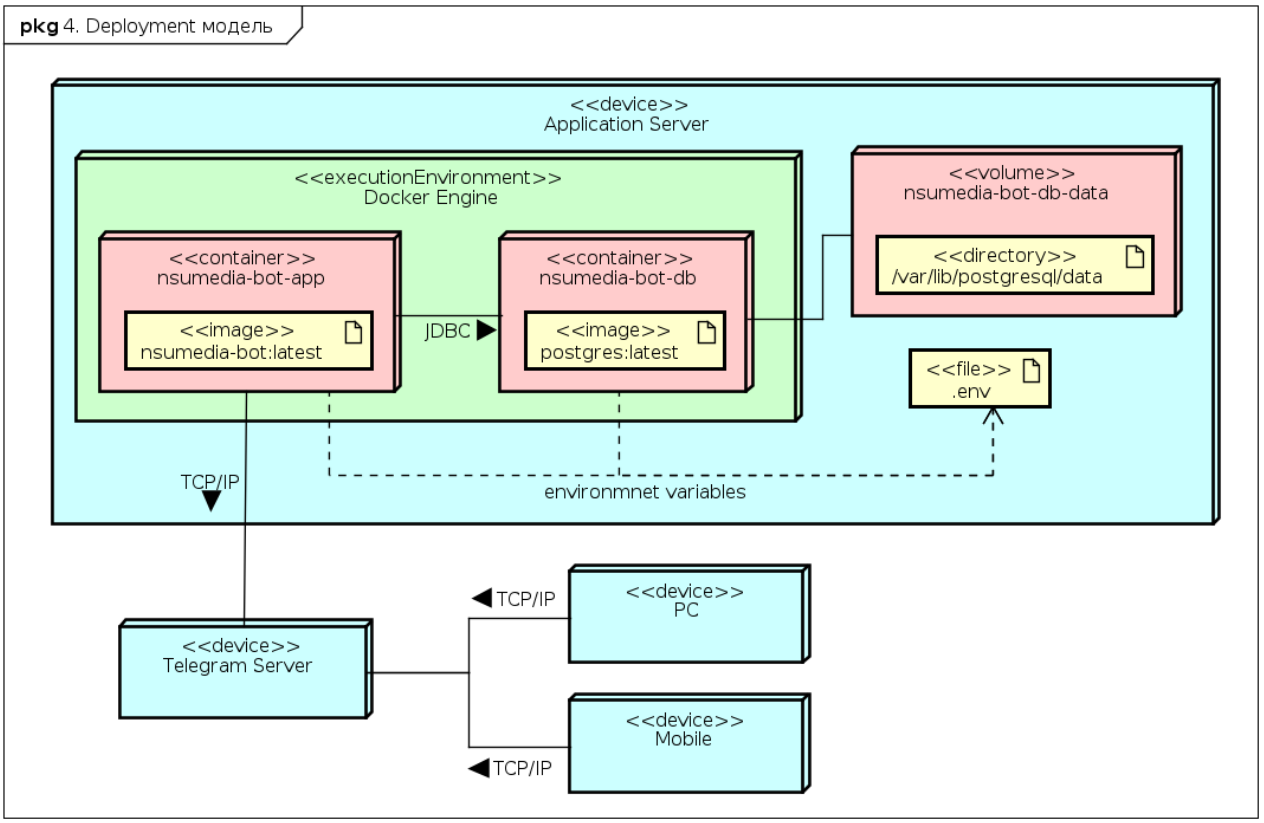
ModelMapper - библиотека маппинга моделей, которая автоматически сопоставляет Java объекты друг с другом.

JavaMailSender - интерфейс для отправки писем по электронной почте.

Thymeleaf - механизм шаблонов Java для создания и обработки HTML.

4.3 Схема развертывания приложения

Ниже приведена диаграмма развертывания приложения:



Приложении состоит из двух Docker контейнеров и развертывается при помощи утилиты docker-compose.

Чтобы запустить приложение, необходимо:

1. Создать файл .env в директории, содержащей docker-compose.yml[3], и определить в нем все необходимые переменные среды[2].
2. Запустить приложение командой:

```
user@host:~$ docker-compose up -d
```

5. Допущения и ограничения

При разработке проекта приняты следующие допущения:

- Ботом будут пользоваться русскоговорящие студенты;
- Количество пользователь будет примерно равняться количеству студентов ФИТ НГУ (≈800);
- Размер файл с учебным материалом не будет превышать 20 МБ. Причиной является ограничение размеров файла для скачивания ботом;

6. Известные проблемы

Ниже приводятся известные на данный момент проблемы и недоработки выработанного программного решения, а также возможные пути их устранения в последующих итерациях проекта.

6.1 Вероятная загруженность сервера

Проблема	Бот написан на базе LongPollingBot. Периодический опрос Телеграм сервера приводит к излишней нагрузке
Ранг	2 (низкий)
Влияние на проект	Вероятные проблемы с загруженностью сервера, на котором развернут бот
Пути решения	Переход к модели WebhookBot

Лист регистрации изменений

Дата	Версия	Описание	Автор
21.09.2023	0.0.0	Изучение предметной области. Написание vision	Авдеев В.С.
27.09.2023	0.1.0	Определение ролей, написание use-case и их сценариев	Авдеев В.С. Петров С.Е.
05.10.2023	0.1.1	Добавление ролей Зритель материала и Неактивированный пользователь, добавление связей между use-case	Авдеев В.С. Петров С.Е.
05.10.2023	0.1.2	Добавление вывода сообщения о результатах активации в use-case Активация аккаунта	Петров С.Е.
12.10.2023	0.1.3	Добавление use-case для Администратора	Петров С.Е.
12.10.2023	0.1.4	Описание функциональных требований в технической документации	Авдеев В.С.
12.10.2023	0.1.5	Описание нефункциональных требований в технической документации	Петров С.Е.
19.10.2023	0.2.0	Составление аналитической модели	Петров С.Е.
19.10.2023	0.2.1	Исправление Sequence диаграмм (добавление пользователя и взаимодействия с ним, изменение общения приложения с Телеграм сервером)	Петров С.Е.
09.11.2023	0.3.0	Конвертация аналитических классов. Составление диаграммы пакетов. Выстраивание связей между классами	Петров С.Е.
16.11.2023	0.3.1	Изменение диаграммы пакетов для зависимости от пакета в целом, а не от некоторой его части	Петров С.Е.
07.11.2023	0.4.0	Реализация сервиса для работы с Yandex Диском	Авдеев В.С.
11.11.2023	0.4.1	Исправление замечаний после code review	Авдеев В.С.
11.11.2023	0.4.2	Реализация сервиса для отправки писем на почту	Авдеев В.С.
18.11.2023	0.4.3	Реализация сервисов для регистрации, активации и работы с материалом	Петров С.Е.
18.11.2023	0.4.4	Реализация бота	Петров С.Е.
18.11.2023	0.4.5	Составление Deployment диаграммы. Описание развертывания бота	Петров С.Е.
20.11.2023	0.4.6	Написание файла README.md с описанием приложения, его технологий и с инструкцией по развертыванию	Петров С.Е.
21.11.2023	0.4.7	Добавление уведомления при ошибки работы с диском	Петров С.Е.
22.11.2023	0.4.8	Описание архитектуры приложения в технической документации	Авдеев В.С.

Лист регистрации проверок

Дата	Версия	Описание	Автор