

МИНОБРНАУКИ РОССИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ
ВЫСШЕГО ОБРАЗОВАНИЯ
“ВОРОНЕЖСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ”

Факультет компьютерных наук

Кафедра программирования и информационных технологий

Мобильное приложение «PicStorm»

Курсовой проект

09.03.04 Программная инженерия

Информационные системы и сетевые технологии

Зав. кафедрой _____ *С.Д. Махортов, д.ф.-м.н., доцент* _____.20__

Обучающийся _____ *Е.Г. Баркалов, 3 курс, д/о*

Обучающийся _____ *К.Ю. Скофенко, 3 курс, д/о*

Обучающийся _____ *Е.М. Закаблуков, 3 курс, д/о*

Руководитель _____ *В.С. Тарасов, ст. преподаватель*

Воронеж 2023

Содержание

| | |
|--|----|
| Введение..... | 4 |
| 1 Постановка задачи..... | 6 |
| 1.1 Требования к разрабатываемой системе | 6 |
| 1.1.1 Функциональные требования | 6 |
| 1.1.2 Требования к интерфейсу | 7 |
| 1.2 Задачи, решаемые в процессе разработки | 7 |
| 2 Анализ предметной области | 9 |
| 2.1 Анализ рынка приложений фотоблогов | 9 |
| 2.2 Обзор аналогов..... | 10 |
| 2.2.1 Instagram..... | 10 |
| 2.2.2 Flickr | 11 |
| 2.2.3 500px..... | 13 |
| 3 Реализация..... | 15 |
| 3.1 Средства реализации..... | 15 |
| 3.2 Языковые версии приложения..... | 16 |
| 3.3 Состав команды и распределение задач | 16 |
| 3.4 Диаграмма IDEF0..... | 18 |
| 3.5 Диаграмма вариантов использования | 19 |
| 3.6 Диаграммы состояний | 20 |
| 3.7 Диаграмма активностей..... | 22 |
| 3.8 Диаграммы последовательностей | 23 |
| 3.9 Диаграмма сотрудничества..... | 26 |

| | |
|--|----|
| 3.10 Реализация серверной части приложения | 27 |
| 3.10.1 Архитектура серверной части приложения | 27 |
| 3.10.2 Схема базы данных приложения | 28 |
| 3.10.3 Диаграмма классов сущностей | 29 |
| 3.10.4 Диаграмма классов репозиториев | 30 |
| 3.10.5 Диаграмма классов сервисов | 30 |
| 3.10.6 Диаграмма классов контроллеров | 32 |
| 3.10.7 Диаграмма классов-конфигураций | 32 |
| 3.10.8 Диаграмма классов для передачи данных | 33 |
| 3.11 Реализация клиентской части приложения | 34 |
| 3.11.1 Макеты интерфейса | 34 |
| 3.11.1.1 Экраны входа и регистрации | 36 |
| 3.11.1.2 Экран ленты публикаций | 37 |
| 3.11.1.3 Экран поиска пользователей | 39 |
| 3.11.1.4 Экран профиля пользователя | 40 |
| 3.11.1.5 Экран ленты публикаций конкретного пользователя | 42 |
| 3.11.1.6 Экран списка подписчиков/подписок | 43 |
| 3.11.2 Архитектура клиентской части приложения | 43 |
| 3.11.3 Обработка фотографий при загрузке | 46 |
| 3.12 Сценарии воронок | 47 |
| 3.13 А/В тестирование | 51 |
| Заключение | 53 |
| Список использованных источников | 54 |

Введение

Социальные медиа и приложения для публикации фотографий достигли своего пика популярности в последние годы. Instagram, Flickr, Pinterest и другие платформы сделали процесс публикации фотографий проще, чем когда-либо. С помощью этих приложений, пользователи могут с легкостью поделиться своими творческими работами с миллионами людей по всему миру.

Однако, наверное, каждый из нас столкнулся с проблемой того, что приложения для публикации фотографий не всегда удобны в использовании или недоступны в вашей стране. Они могут быть неудобными, непонятными или не иметь нужных функций. Именно поэтому многие люди начинают искать альтернативные варианты, которые будут соответствовать их ожиданиям.

Таким образом, идея создания собственного приложения для публикации собственных фотографий становится все более популярной среди пользователей. В нашем современном мире, где технологии играют важную роль в повседневной жизни, публикация и обмен красивыми и интересными фотографиями в таком приложении могут быть не единственным занятием. Сервисы фотоблога могут быть использованы как источник дохода, творческий проект или просто в качестве другого способа делиться своими моментами с людьми.

Однако создание собственного приложения для публикации своих фотографий может быть связано со многими сложностями. Это требует много времени, труда и денег. Разработчики могут столкнуться со сложностями технического характера, они могут попасть под основные индексы поисковых систем или не обладать необходимым опытом и знаниями в создании приложений.

Тем не менее, преимущества создания собственного приложения для публикации своих фотографий огромны. Это дает разработчикам возможность

создавать свой бренд и получить уникальный опыт. Разработчики могут создавать свой собственный стиль и фирменный дизайн приложения. А пользователи могут использовать свои навыки и знания в фотографии, чтобы создать уникальный контент, который привлечет аудиторию и даст возможность заработать деньги на продаже своих фотографий и принесет прибыль разработчикам на рекламе и прочем. Независимо от того, какие мотивы и цели разработчиков, создание своего приложения для публикации фотографий может быть прекрасным способом открывать новые возможности и расширять мир вокруг нас. Сочетание технологий и творческих способностей может сделать создание приложения настоящим творческим проектом и открыть ряд новых возможностей для развития.

1 Постановка задачи

Целью данного курсового проекта является разработка мобильного приложения для просмотра и публикации фотографий «PicStorm»

1.1 Требования к разрабатываемой системе

1.1.1 Функциональные требования

Разрабатываемое приложение должно обладать следующими возможностями:

- Просмотр списка публикаций всех пользователей
- Оценка выбранной публикации
- Фильтрация общего списка публикаций по дате, сортировка по пользовательским оценкам
- Поиск пользователей по имени
- Просмотр профиля пользователя
- Просмотр и оценка публикаций конкретного пользователя
- Подписка и отписка на другого пользователя
- Просмотр списка публикаций, основанного на подписках
- Загрузка и замена фотографии профиля
- Загрузка своих публикаций
- Удаление своих публикаций
- Блокировка отдельных публикации и пользователей в целом администраторами

1.1.2 Требования к интерфейсу

Все экраны приложения должны быть оформлены в едином стиле:

- Фон всех экранов должен быть розовым
- Основной текст надписей – черный, вспомогательный – серый
- Наполнение приложения должно располагаться на белых скругленных прямоугольниках
- В качестве цвета-акцента внимания должен использоваться фиолетовый
- Кнопки приложения могут быть серыми, черными, зелеными или фиолетовыми в зависимости от назначения

Дизайн приложения должен быть адаптирован для корректного отображения при различных размерах экрана и поддерживать портретную ориентацию экрана.

1.2 Задачи, решаемые в процессе разработки

Были поставлены следующие задачи:

- Анализ предметной области
- Анализ аналогов
- Постановка задачи
- Разработка сценариев работы системы в Miro
- Разработка макетов интерфейса в сервисе Figma
- Написание технического задания
- Построение UML диаграмм
- Проектирование БД

- Реализация бизнес-логики
- Подключение Swagger
- Написание Unit-тестов
- Реализация пользовательского интерфейса
- Создание сценариев воронок
- Размещение backend-части и БД на хостинге
- Описание процесса разработки и результата

2 Анализ предметной области

2.1 Анализ рынка приложений фотоблогов

Анализ рынка приложений с функциями социальной сети по публикации фотографий показывает, что данный сегмент весьма конкурентоспособный. Множество приложений предлагают пользователям возможность создавать и делиться своими фотографиями, комментариями и сторис на платформах социальных сетей, таких как Instagram, Facebook, VK, Snapchat, TikTok и др.

Среди наиболее популярных приложений фотоблогов можно выделить Instagram, который предлагает пользователям широкую функциональность для обработки фотографий, использования фильтров, создания историй и взаимодействия с другими пользователями. На момент анализа рынка приложений, он является запрещенным на территории Российской Федерации, поэтому не может конкурировать с доступными приложениями. Также к заметным аналогам можно отнести VSCO, Flickr, Tumblr, 500px, и другие.

Основными трендами на рынке приложений фотоблогов являются:

- Развитие функциональности и улучшение качества загрузки фотографий
- Развитие функциональности для создания контента, включая возможность создания историй, видеороликов, трансляций в режиме реального времени и т. д.
- Возможности для взаимодействия пользователей между собой - лайки, подписки, и т. д.
- Усиление защиты конфиденциальности пользователей, борьба с нежелательным контентом и блокировка пользователей, не соблюдающих правила приложения, публикующих запрещенные материалы и т.п.

Рынок приложений фотоблогов продолжает расти и развиваться, привлекая все больше пользователей по всему миру.

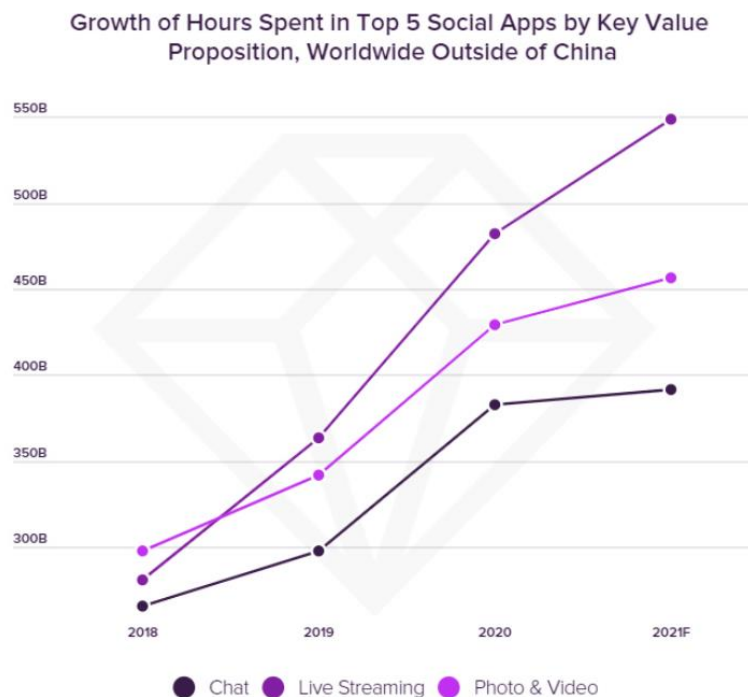


Рисунок 1 - Статистика часов, проведенных в социальных сетях

На графике изображено, сколько часов каждый год люди тратят на определенные вещи в социальных сетях. Можно заметить, что за три года пользователи стали тратить в полтора раза больше времени на фото и видео контент [1]. Тенденция роста популярности социальных сетей для публикации фотографий имеет положительный характер и по сей день.

2.2 Обзор аналогов

2.2.1 Instagram

Instagram - одно из самых популярных приложений для фотоблоггеров. Он позволяет пользователям загружать фотографии и видео, настраивать свои профили, следить за другими пользователями, комментировать их посты, ставить лайки и использовать хэштеги для поиска популярных тем.

Плюсы:

- Удобно вести блог с упором на фото
- Огромный простор для коммерции и ведения бизнеса
- Реклама идеально подгоняется под конкретного пользователя
- Фотографии, картинки, видео разной длины — все в одном приложении
- Встроенные инструменты для редактирования контента

Минусы:

- Недоступен на территории Российской Федерации
- Социальная сеть не предназначена для лонгридов и объемных текстов

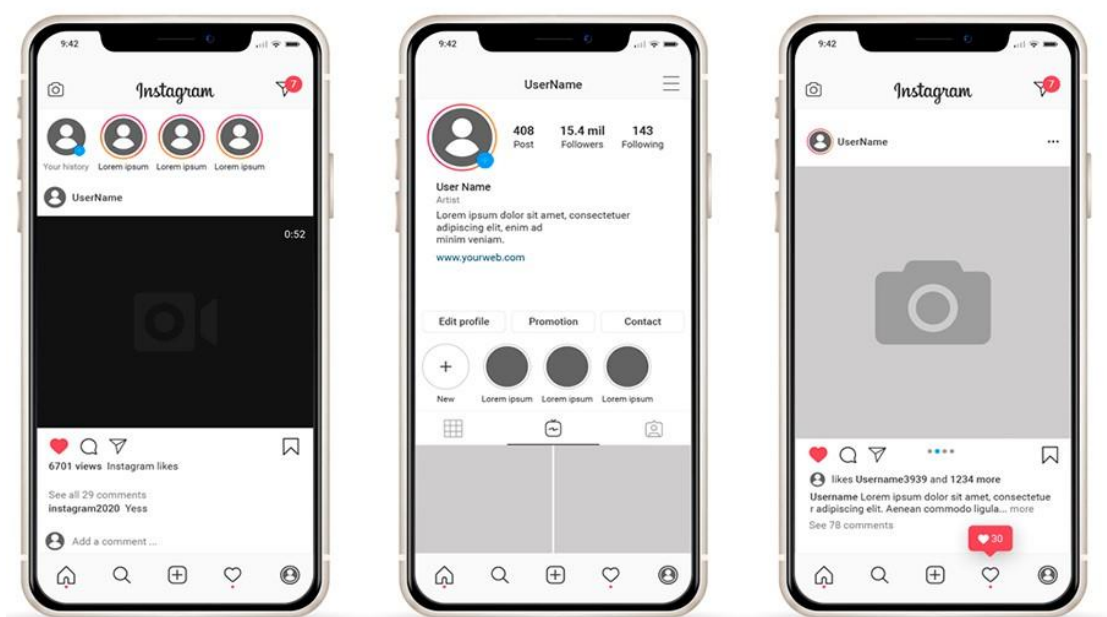


Рисунок 2 - Интерфейс приложения Instagram

2.2.2 Flickr

Flickr - это социальная сеть для фотографов и фотоблоггеров. В ней пользователи могут делиться своими фотографиями, создавать альбомы и

комментировать работы других фотографов. Это хороший выбор для профессиональных фотографов, которые хотят демонстрировать свои работы и получать отзывы от сообщества.

Плюсы:

- Бесплатное использование: Flickr предоставляет бесплатные аккаунты с ограниченным пространством хранения
- Хранилище фотографий: Flickr предлагает возможность хранить фотографии в высоком разрешении в облаке
- Возможность общения: Flickr позволяет общаться и делиться фотографиями с другими пользователями платформы
- Качественная статистика: Flickr предоставляет высококачественную статистику, которая может помочь в улучшении качества вашего контента
- Хорошая SEO-оптимизация: Flickr представляет собой отличный ресурс для SEO, что помогает устанавливать ваш авторитет и репутацию в своей сфере деятельности

Минусы:

- Ограниченное пространство хранения: Бесплатные аккаунты Flickr имеют ограниченное пространство хранения, что может позволить вам загружать не более 1 000 фотографий
- Реклама: Flickr иногда может отображать рекламные объявления на страницах пользователей
- Ограниченная интеграция: Flickr не всегда полностью интегрируется с другими социальными сетями и приложениями

- Интерфейс: Flickr имеет сложный интерфейс, что может отпугнуть новых пользователей, но он привычен к использованию со временем

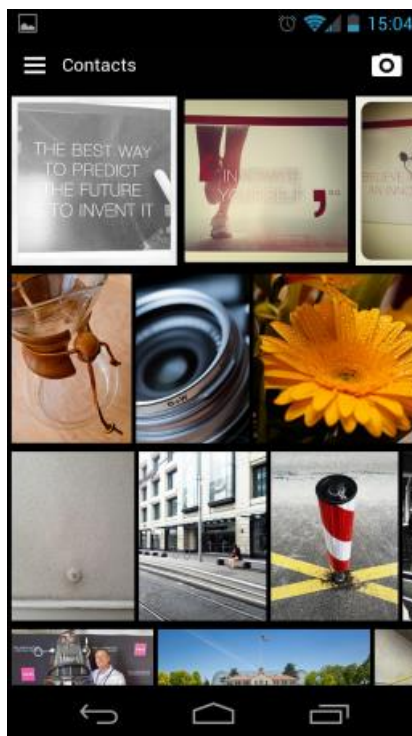


Рисунок 3 - Интерфейс приложения Flickr

2.2.3 500px

500px - это платформа для профессиональных фотографов и фотоблоггеров. В этом приложении пользователи могут загружать свои работы, создавать свои портфолио и продавать свои фотографии. Это хороший выбор, если вы занимаетесь фотографией на профессиональном уровне.

Плюсы:

- Высокое качество изображений
- Большое количество креативных фотографий от профессионалов со всего мира
- Удобный и приятный дизайн приложения
- Возможность создания собственной коллекции изображений
- Различные фильтры для поиска фотографий по тематике

Минусы:

- Большинство функций доступно только с подпиской на платный тариф
- Непонятные права на использование фотографий при скачивании
- Неудобный процесс загрузки фотографий
- Очень большое количество фотографий, которые могут перегружать приложение и затруднять поиск нужных
- Не всегда удастся найти нужную фотографию из-за ограничений по поиску



Рисунок 4 - Интерфейс приложения 500px

3 Реализация

3.1 Средства реализации

Клиентская часть реализована с помощью:

- Язык программирования Kotlin
- Android SDK 32
- Система автоматизации сборки проектов Gradle
- Библиотеки Retrofit2 и OkHttp3 для реализации коммуникации с сервером
- Библиотека для внедрения зависимостей Hilt

Язык программирования Kotlin был выбран в связи с его активным развитием и поддержкой, простотой синтаксиса и высокой безопасностью, благодаря работе с пустыми ссылками (null safety).

Серверная часть использует:

- Язык программирования Java
- JDK 17
- Система автоматизации сборки проектов Apache Maven
- Фреймворк Spring Boot
- СУБД PostgreSQL
- Yandex Object Storage
- Docker для развёртывания приложения

Сочетание языка программирования Java и Spring Boot было выбрано из-за большого выбора дополнительных библиотек, возможности быстро разворачивать разрабатываемое приложение. Также данное сочетание

технологий обеспечивает быструю разработку за счет уже реализованного внедрения зависимостей и авто конфигурации.

СУБД PostgreSQL обладает рядом плюсов, среди которых можно особенно выделить открытый исходный код, свободную лицензию и широкую функциональность.

Загруженные пользователем фотографии хранятся в облачном хранилище от Yandex, взаимодействие с которым происходит при помощи Amazon S3 SDK для Java. Выбор данного сервиса для сохранения фото обусловлен возможностью его автоматического масштабирования и удобством взаимодействия.

3.2 Языковые версии приложения

Приложение реализовано с поддержкой русского языка.

3.3 Состав команды и распределение задач

Команда, выполнявшая проект, состоит из трех студентов очного отделения кафедры программирования и информационных технологий: Скофенко Кирилла Юрьевича, Закаблукова Егора Михайловича и Баркалова Евгения Геннадьевича.

Скофенко Кирилл был ответственен за следующие задачи:

- Ведение таск менеджера
- Разработка сценариев приложения в Migo
- Создание UML диаграмм: прецедентов, последовательностей, сотрудничества, классов, объектов, состояний
- Создание репозиторий для документации и серверной части
- Проектирование базы данных

- Разработка серверной части приложения: слоя для работы с БД, API для авторизации, действий с публикациями, поиска пользователей и администрирования
- Интеграция с Яндекс Yandex Object Storage для хранения фото
- Добавление Swagger документации
- Развертывание сервера
- Помощь другим участникам команды

Закаблукоев Егор выполнял следующее:

- Создание диаграмм: IDEF0, активностей и состояний
- Дизайн приложения: окон регистрации и входа, поиска пользователей и ленты публикаций, списка подписчиков, профиля пользователя
- Создание интерактивного макета в Figma
- Создание репозитория для клиентской части
- Написание клиентской части приложения: верстка макетов в соответствии с техническим заданием, настройка навигации, взаимодействие с сервером и отображение результатов, обработка ошибок, настройка диалоговых окон

Баркалов Евгений выполнял ниже перечисленные задачи:

- Составление технического задания к приложению
- Создание Unit-тестов серверной части
- Написание курсового проекта
- Ручное тестирование приложения
- Настройка сервиса аналитики, разработка воронок конверсии
- Обновление Readme файлов репозитория
- Создание презентации для выступления
- Составление отчета по ролям

3.4 Диаграмма IDEF0

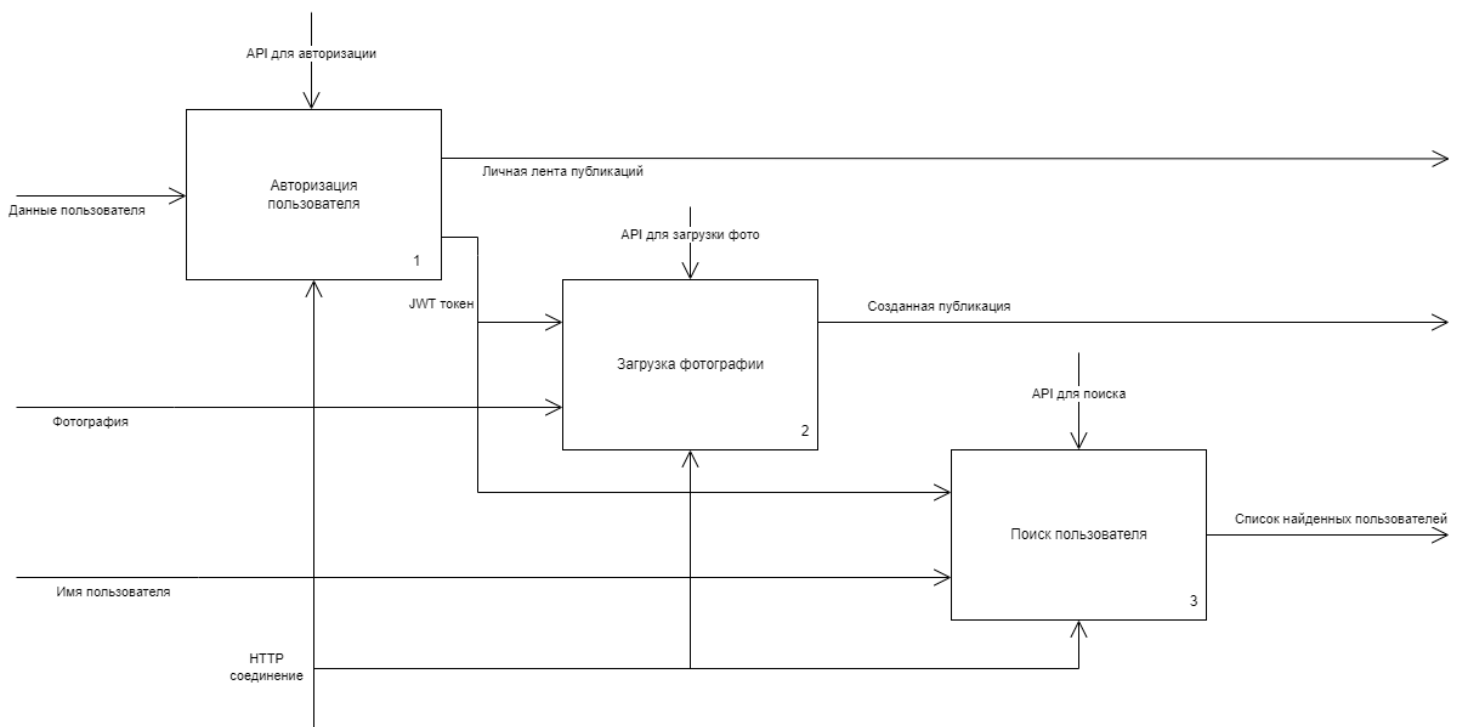


Рисунок 5 - Диаграмма IDEF0

На диаграмме представлены процессы, происходящие в приложении. Показано какие объекты подаются на входе и на выходе, какие средства используются, а также какие ресурсы требуются для определенного процесса.

3.5 Диаграмма вариантов использования

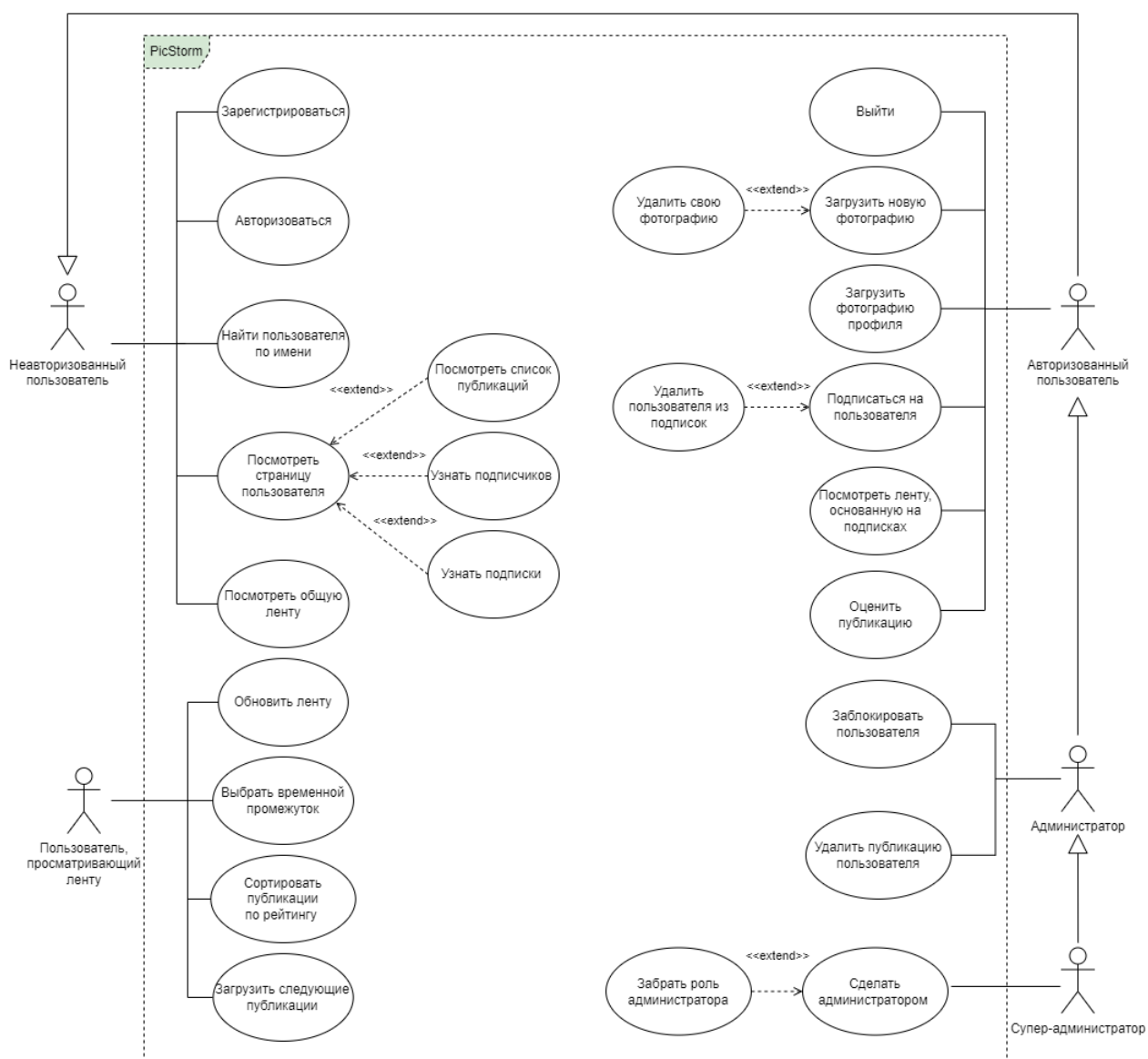


Рисунок 6 - Диаграмма вариантов использования

На диаграмме прецедентов показаны все варианты использования приложения для каждой роли (неавторизованные и авторизованные пользователи, администраторы и супер-администраторы). Каждая следующая роль расширяет список доступных вариантов использования.

Отдельно рассмотрен пользователь, просматривающий ленту. В качестве этого актора может выступать любой из перечисленных выше. При этом просматриваемая лента может отличаться – быть общей или основанной на подписках.

3.6 Диаграммы состояний

На диаграммах состояний изображены схемы всех возможных состояний и путей к ним. Рассмотрим возможные состояния пользователей и публикаций.

Публикация может быть видима – сразу после создания, удалена заблокирована или не видима из-за того, что заблокирован пользователь, сделавший ее.

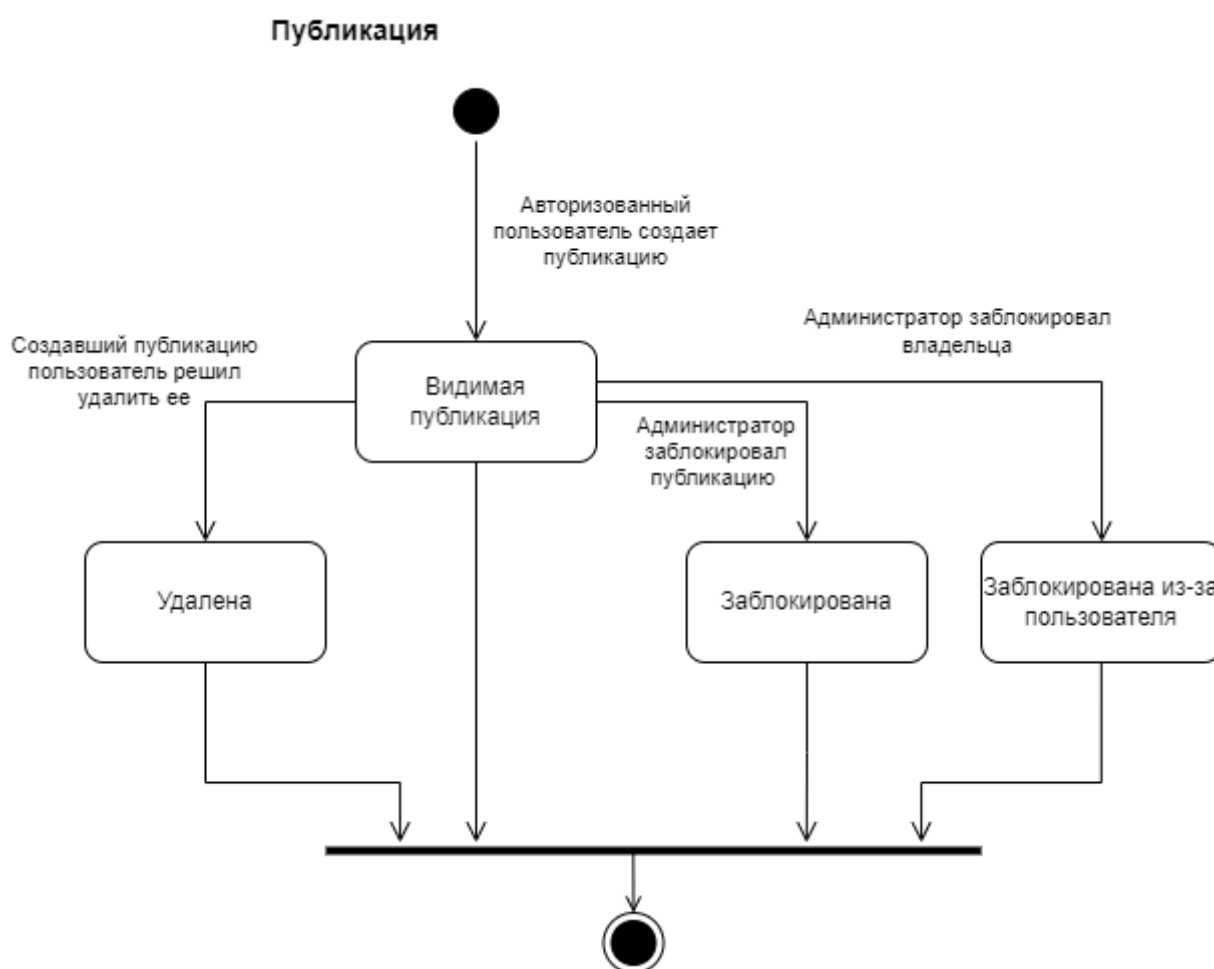


Рисунок 7 - Диаграмма состояний публикации

Соответственно пользователь может быть не авторизован (если до этого не проходил регистрацию, ввел при ней неверные данные или вышел из приложения), находиться в обычном состоянии, стать администратором, или войти в приложение в качестве супер-администратора.

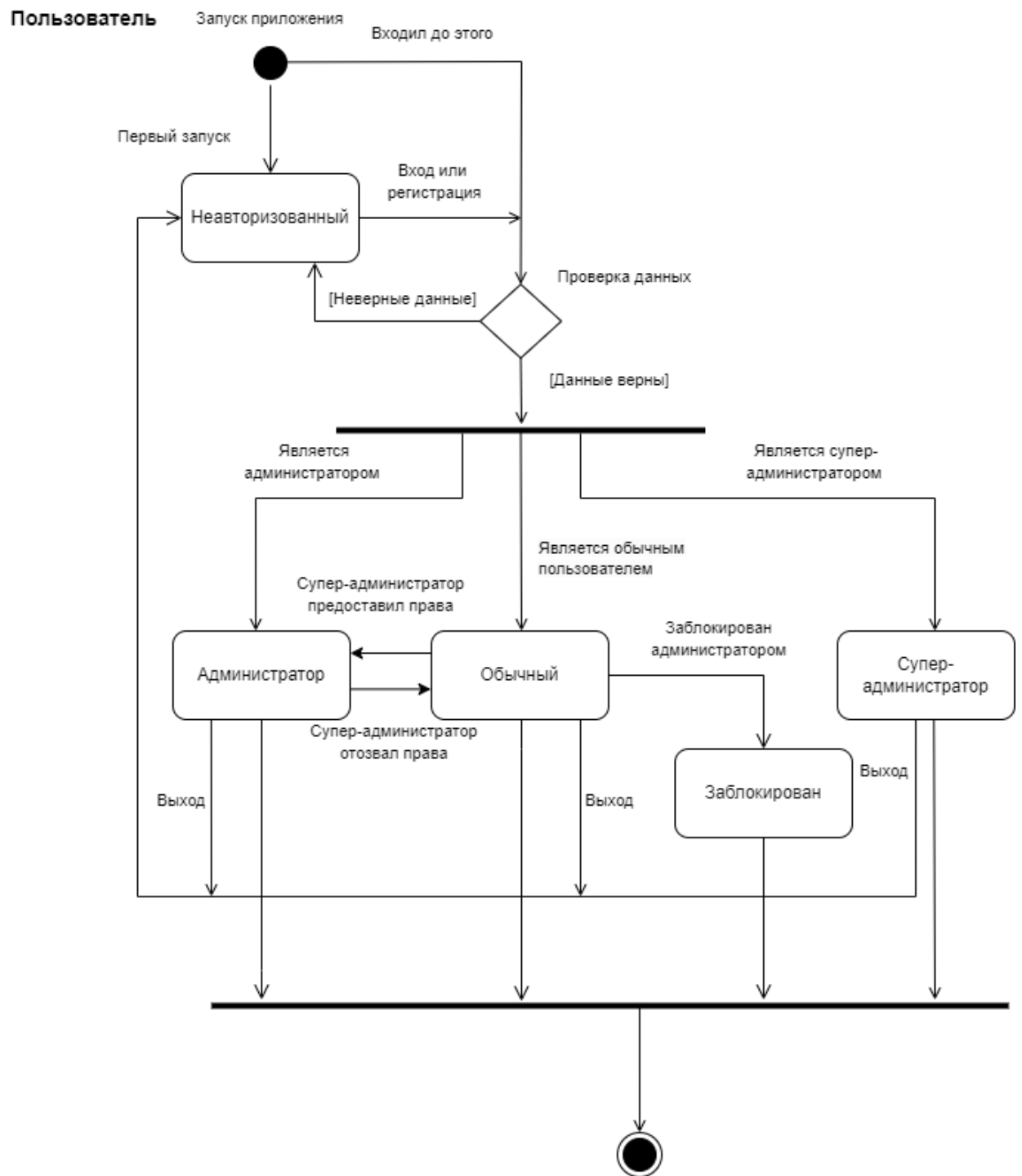


Рисунок 8 - Диаграмма состояний пользователя

3.7 Диаграмма активностей

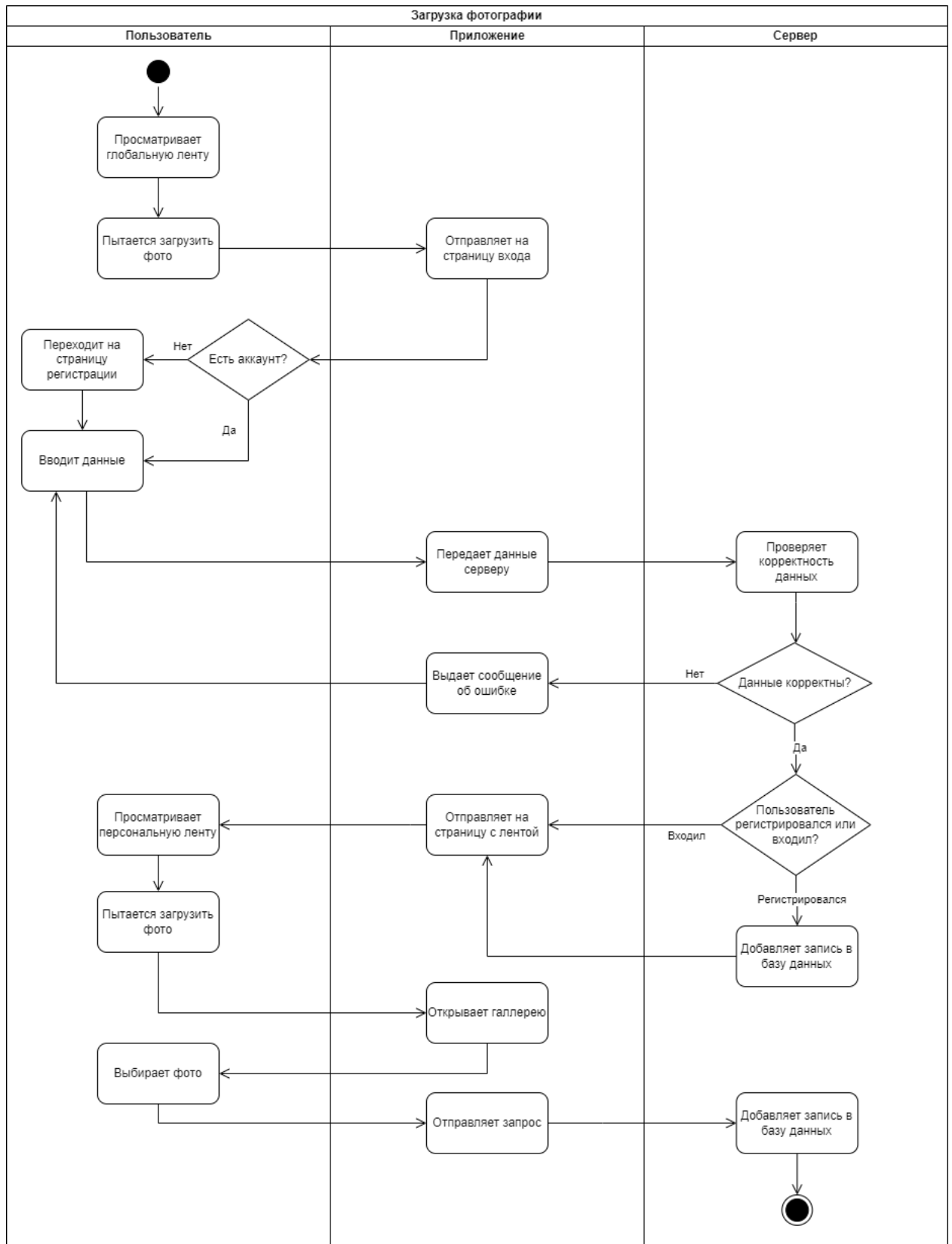


Рисунок 9 - Диаграмма активностей

На диаграмме активностей изображена главная активность пользователя, скачавшего приложение – загрузка фотографии. Подразумевалось, что пользователь впервые запустил приложение на своем смартфоне и еще не зарегистрирован либо не авторизован (мог иметь аккаунт на другом смартфоне).

3.8 Диаграммы последовательностей

На диаграммах последовательностей были изображены последовательности действий для различных ситуаций сценариев трёх разных групп пользователей: администраторов, авторизованных и неавторизованных пользователей.

Для неавторизованных показана последовательность при регистрации. При этом учтены возможные ситуации, когда были введены не валидные данные или пользователь уже существует.

Для авторизованных пользователей была показана последовательность при отображении личной ленты, основанной на подписках. В этом случае сервер взаимодействует не только с базой данных, но и с облачным хранилищем. А для администратора – последовательность блокировки пользователя.

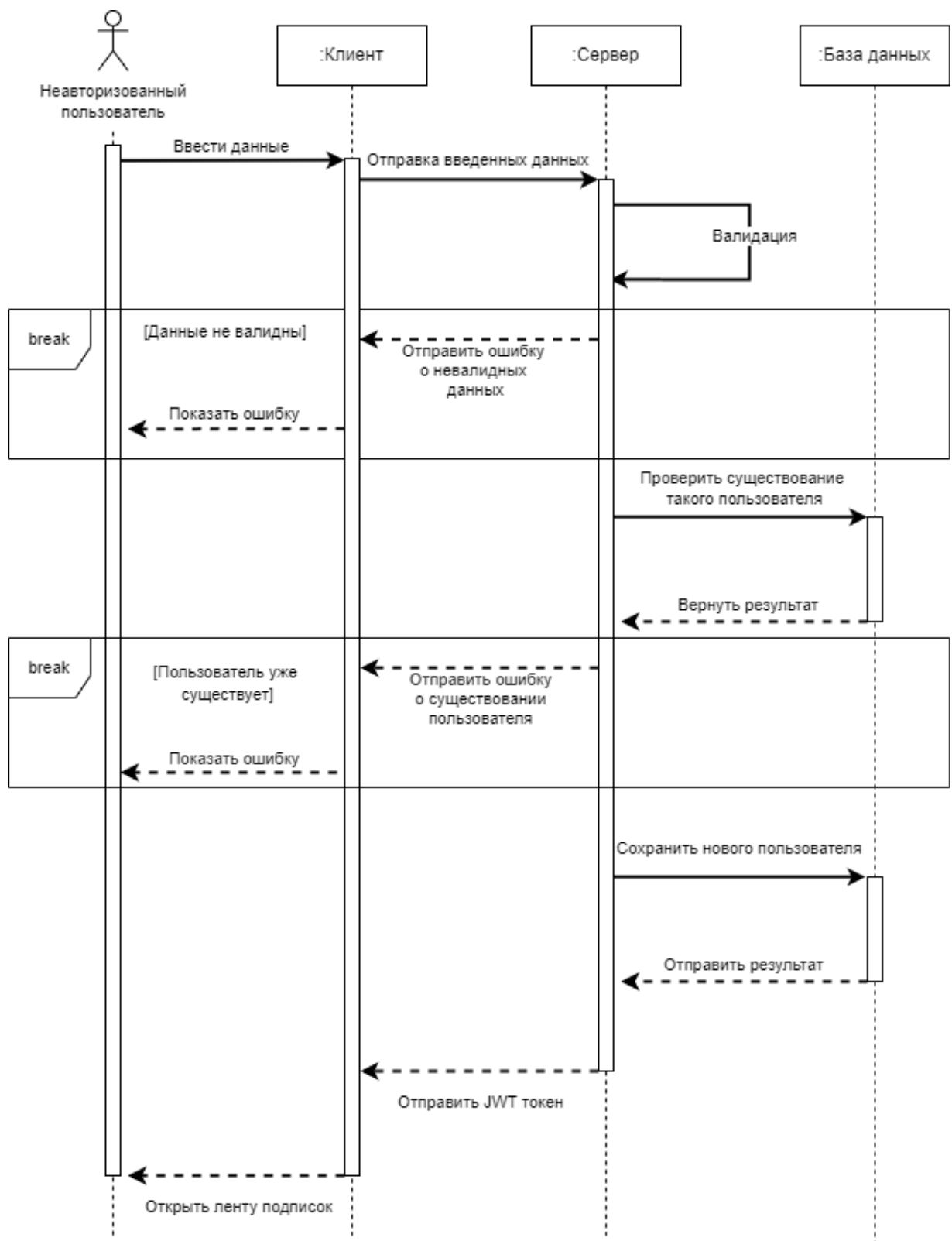


Рисунок 10 - Диаграмма последовательности неавторизованного пользователя

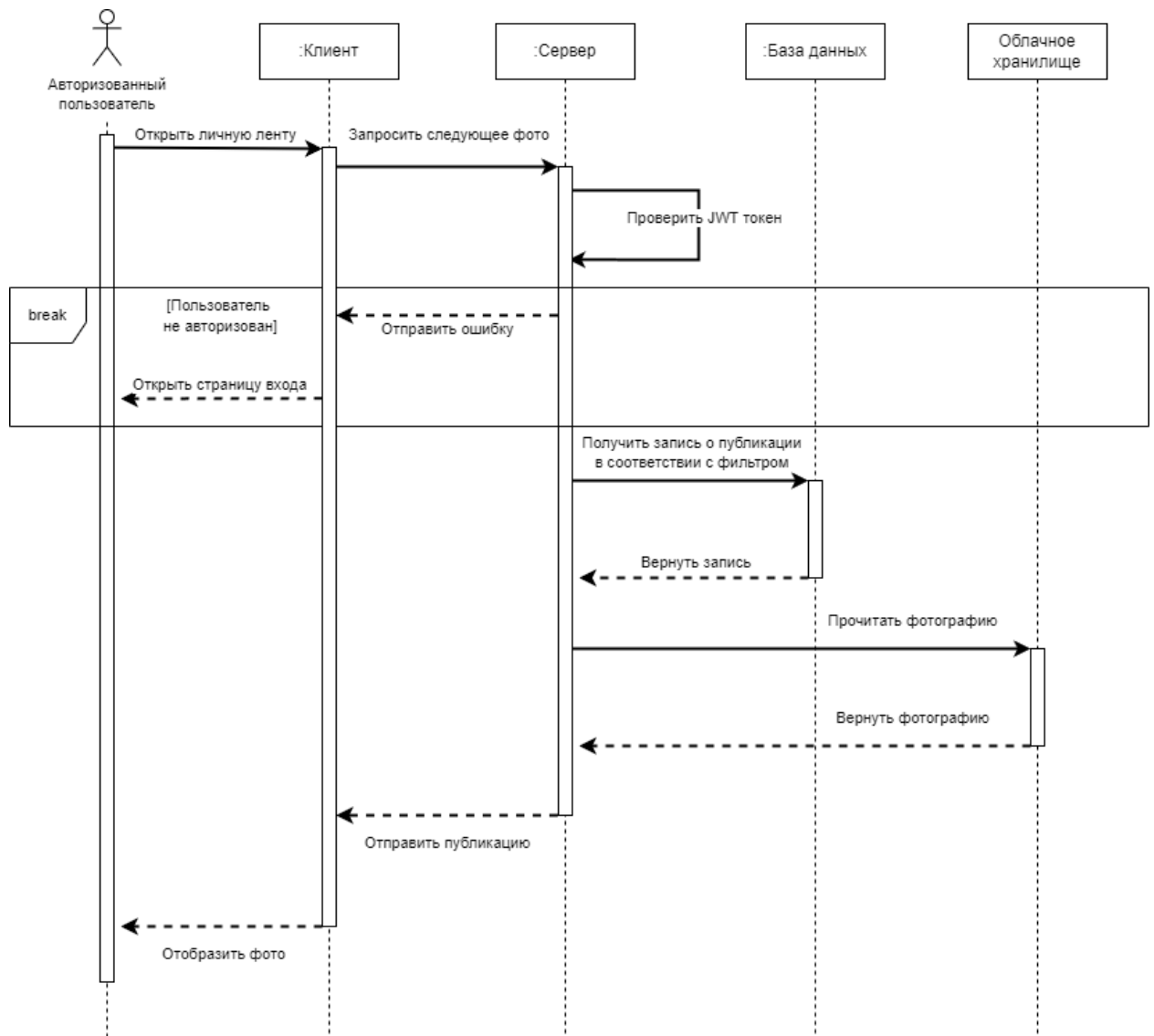


Рисунок 11 - Диаграмма последовательности авторизованного пользователя

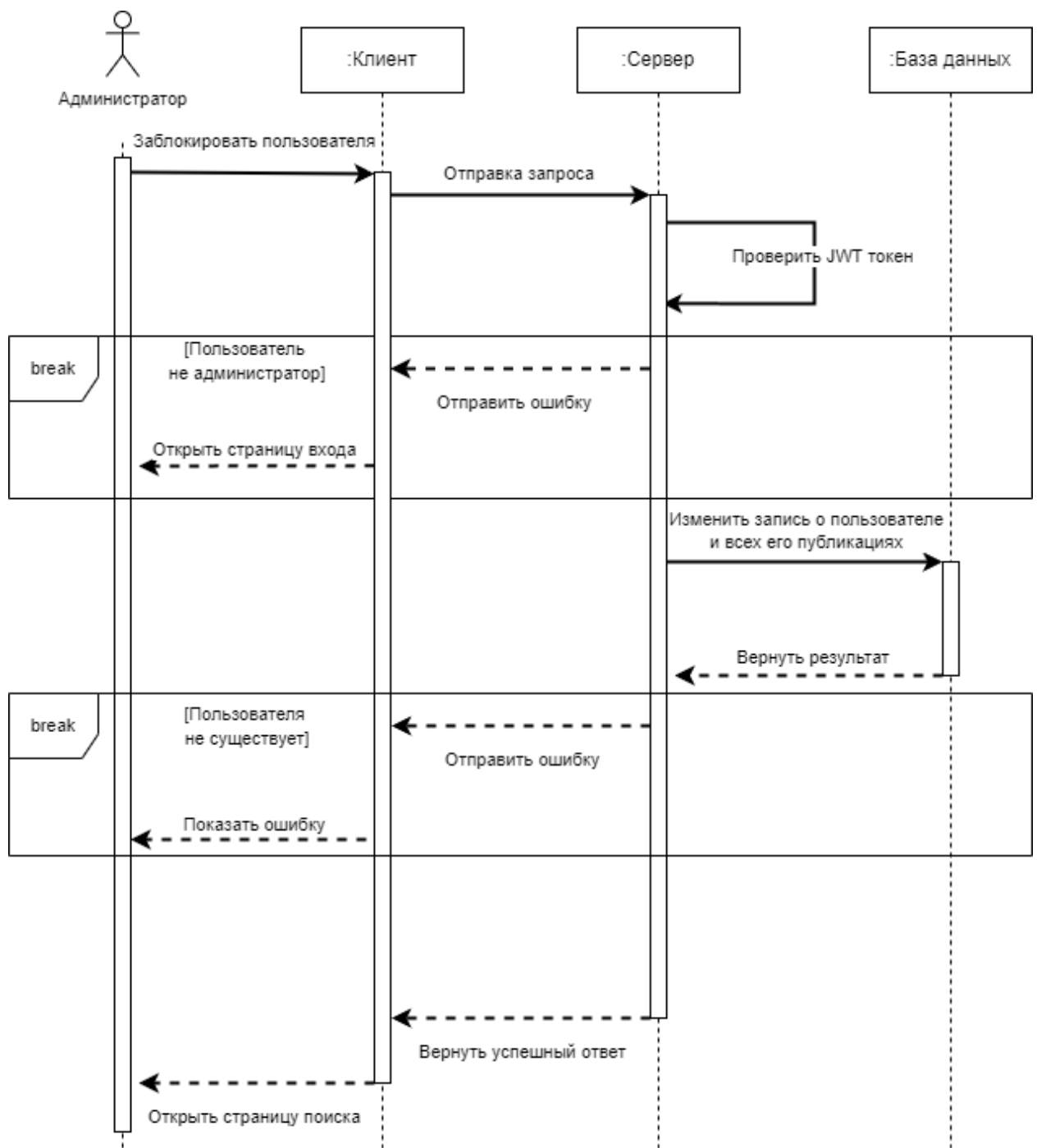


Рисунок 12 - Диаграмма последовательности администратора

3.9 Диаграмма сотрудничества

На диаграмме сотрудничества показано общее взаимодействие объектов при попытке авторизации пользователя. Представлены такие объекты, как клиент, сервер, база данных, неавторизованный пользователь.

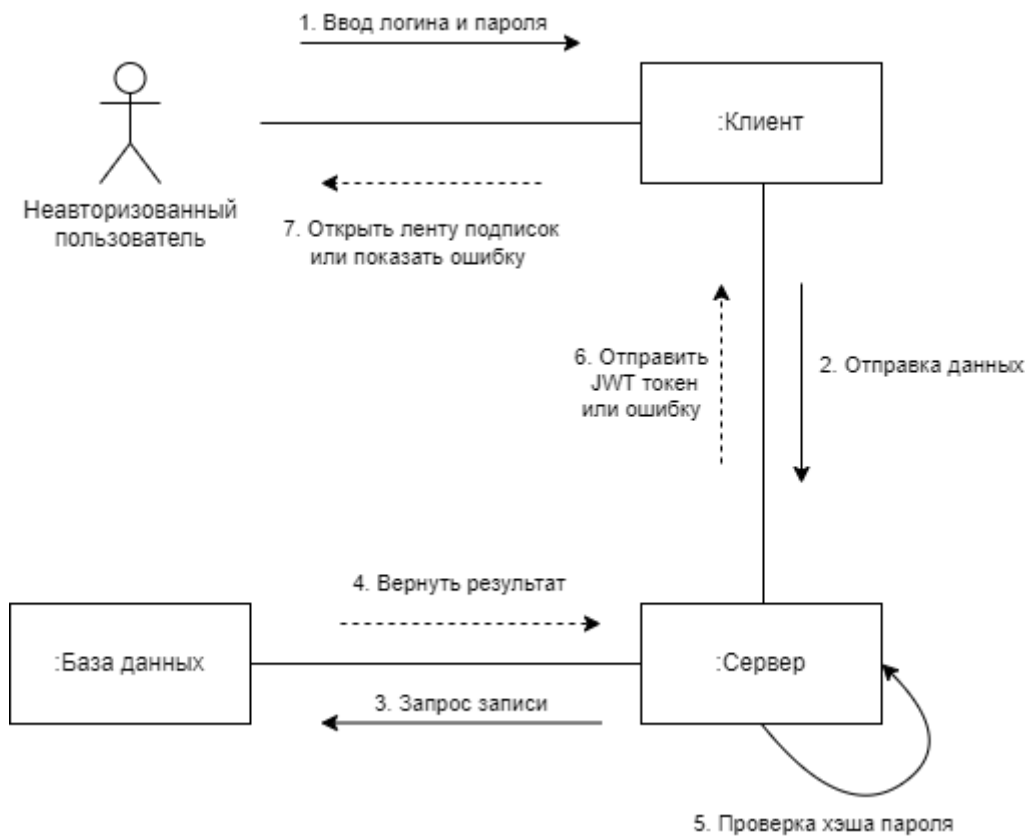


Рисунок 13 - Диаграмма сотрудничества

3.10 Реализация серверной части приложения

3.10.1 Архитектура серверной части приложения

Серверная часть приложения написана, используя архитектуру, предлагаемую Spring фреймворком. В соответствии с ней сервер разделен на 3 слоя:

- Контроллеры, обрабатывающие входящие запросы клиента
- Сервисы, реализующих бизнес-логику, также к этому слою относятся репозитории - интерфейсы для взаимодействия с базой данных, предоставляемые фреймворком
- Модели, описывающих сущности, используемые в приложении

Сервер использует принцип инверсии управления, при котором поток управления программы контролируется фреймворком. Реализуется он с помощью механизма внедрения зависимостей (DI). Такой подход позволяет

уменьшить связность кода и упростить расширение функциональности и придерживаться принципа единой ответственности.

3.10.2 Схема базы данных приложения

Таблица `app_user` хранит информацию о зарегистрированных пользователях приложения: их имена, электронные почты, роли и даты регистрации и внешний ключ – `id` аватара.

`Publication` – таблица, содержащая описание публикации: ее рейтинг, состояние, дату создания, а также внешние ключи для пользователя, сделавшего ее и опубликованной картинки.

Таблица `reaction` хранит описание реакций на публикации, оставленных пользователями. Она состоит из ее типа (лайк или дизлайк), даты, когда она была поставлена. Еще присутствуют внешние ключи, ссылающиеся на пользователя, оставившего ее, и саму публикацию.

`Subscription` – таблица, описывающая подписку пользователя. Она содержит дату создания и внешние ключи: того, кто подписался и на кого.

Таблица `Picture`, содержащая дату загрузки картинки, служит для хранения информации о фото, находящихся в Yandex Object Storage.

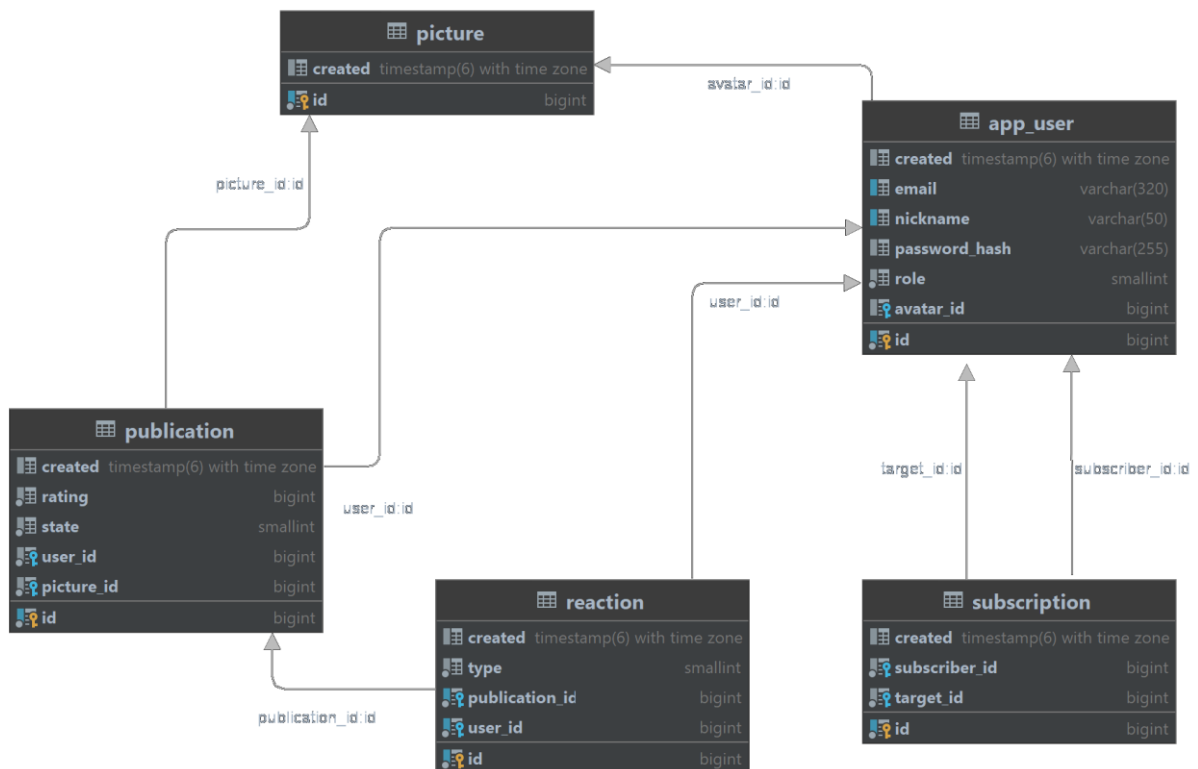


Рисунок 14 - Схема базы данных

3.10.3 Диаграмма классов сущностей

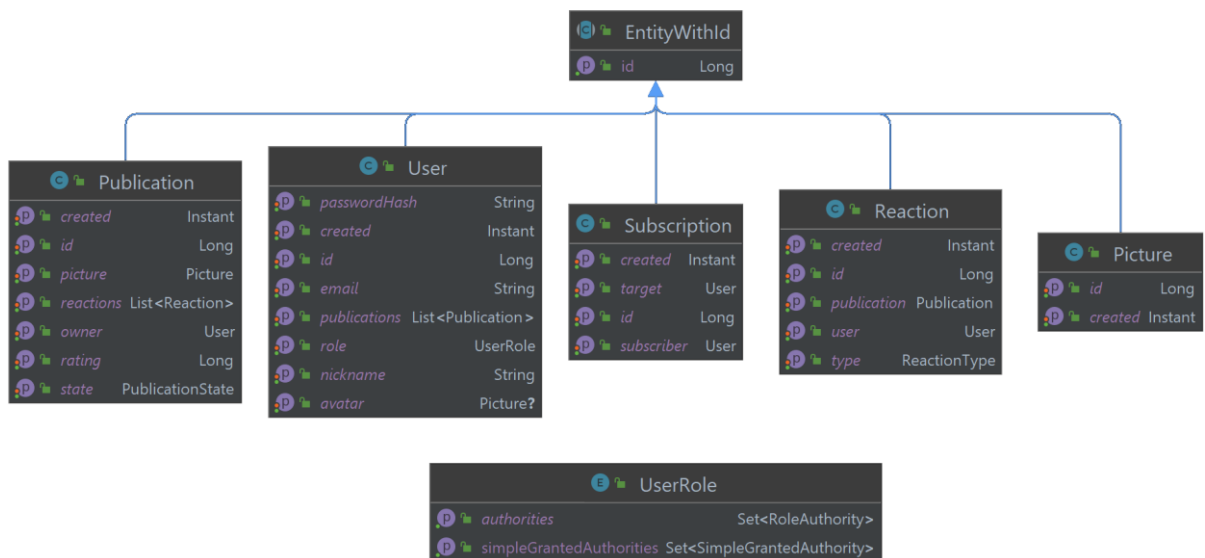


Рисунок 15 - Диаграмма классов сущностей

Абстрактный класс `EntityWithId` переопределяет методы для сравнения сущностей хранимых в БД, чтобы избежать ненужной загрузки lazy-полей. Поля каждого из его наследников эквиваленты атрибутам одноименных

таблиц. Перечисление UserRole хранит в себе полномочий, доступных для каждой роли.

3.10.4 Диаграмма классов репозиториев

Интерфейсы репозиториев, представленные ниже, являются наследниками JpaRepository, предоставляемого фреймворком Spring Boot. Каждый такой интерфейс имеет соответствующие методы для CRUD операций (поиск по id, сохранение и удаление сущностей), а также методы необходимые для реализации пагинации.

Отдельно необходимо остановиться на классе PublicationFeedSpecification. Он представляет собой динамический фильтр для поиска ленты публикаций, формируемый на основе переданных при создании ограничений для пользователей и дат. При последующем использовании он передает соответствующему репозиторию вместе с необходимой сортировкой по рейтингу и номером страницы.

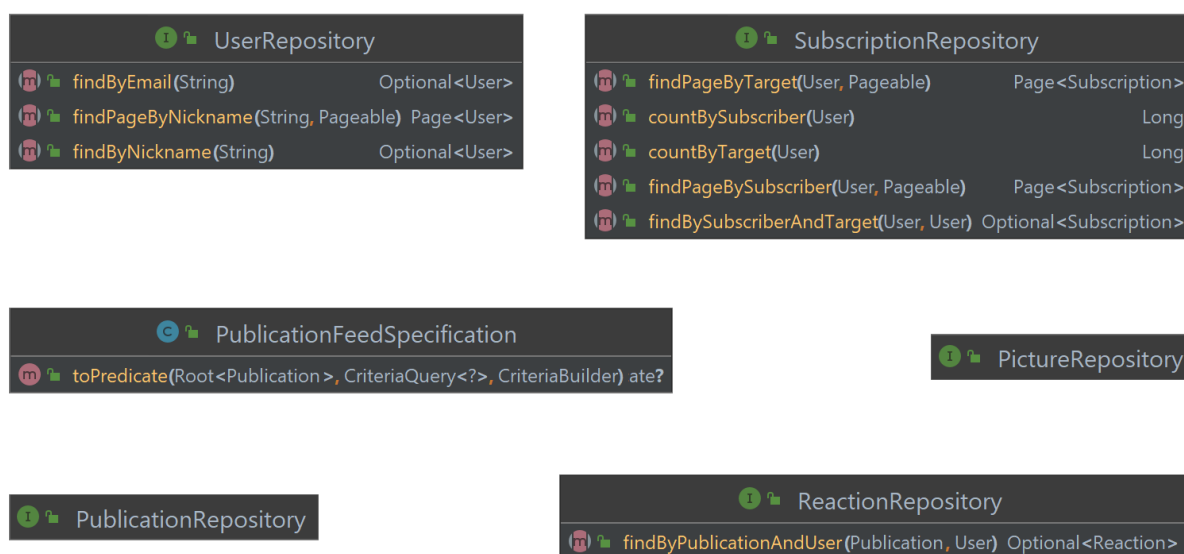


Рисунок 16 - Диаграмма классов репозиториев

3.10.5 Диаграмма классов сервисов

Каждый из этих классов является частью слоя бизнес-логики, то есть

выступают связующим звеном между слоем доступа к данным и контроллерами. Все вычисления и алгоритмы находятся в этих классах.

Класс `YandexCloudPictureStorageService` реализует взаимодействие с облачным хранилищем для сохранения, получения и удаления фотографий.

`AuthService` служит для регистрации и аутентификации пользователей, `SubscriptionService` – для получения списка подписок/подписчиков и изменения подписок.

Класс `PublicationService` отвечает за создание, удаление и блокировку публикаций, получения ленты, а также для постановки реакций. В свою очередь, `UserService` нужен для поиска пользователей, их блокировки и выдачи прав администратора, получения профилей.

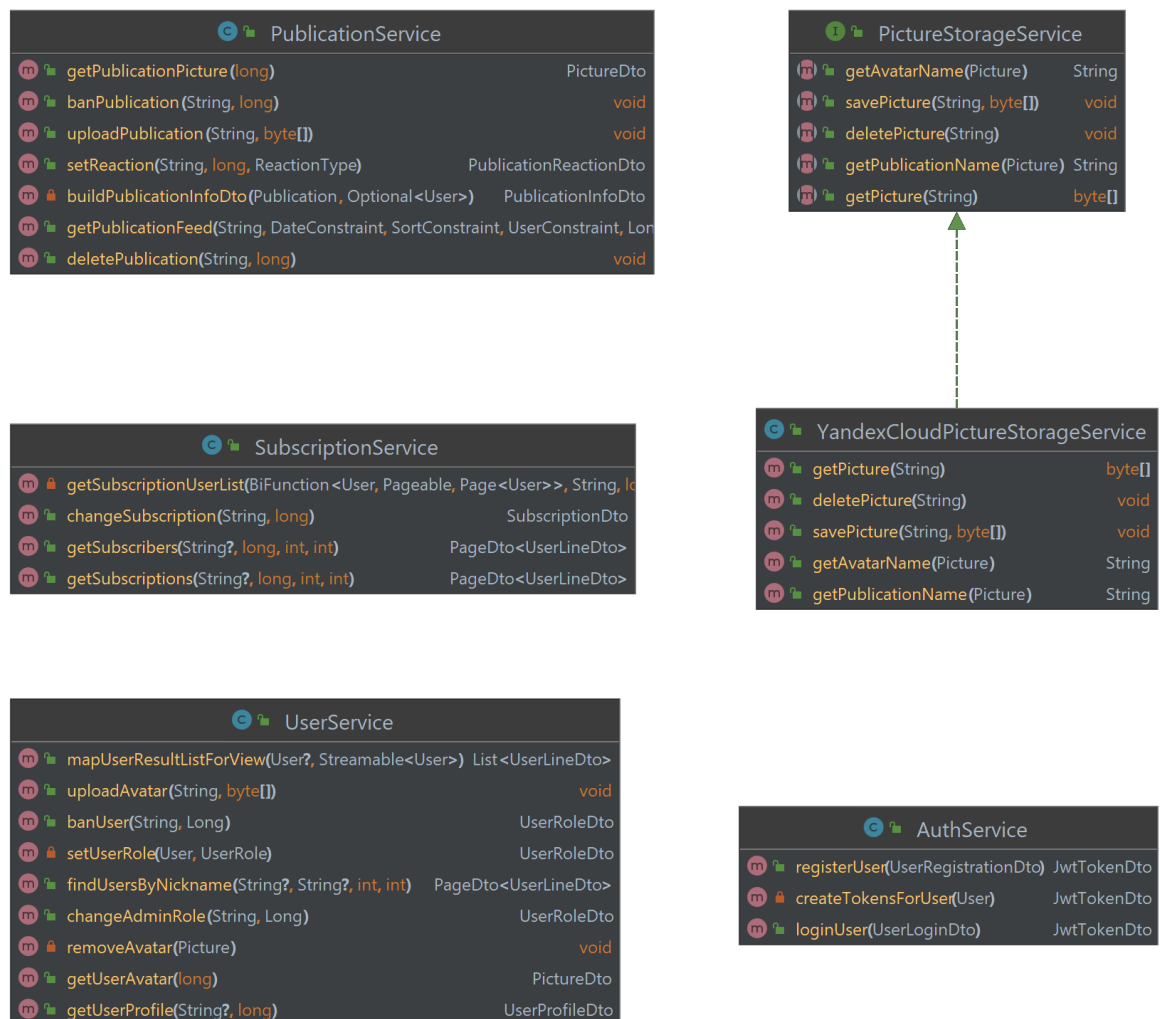


Рисунок 17 - Диаграмма классов сервисов

3.10.6 Диаграмма классов контроллеров

Данные классы необходимы для обработки общения с клиентами. После получения запросов в них проверяется имеет ли пользователь необходимые полномочия для совершения этого действия, вызываются методы слоя бизнес-логики и отправляются ответы назад на клиент.

Класс `ExceptionHandlerController` служит для обработки ошибок, возникающих в ходе работы сервера, и отправки соответствующих им ответов клиенту.

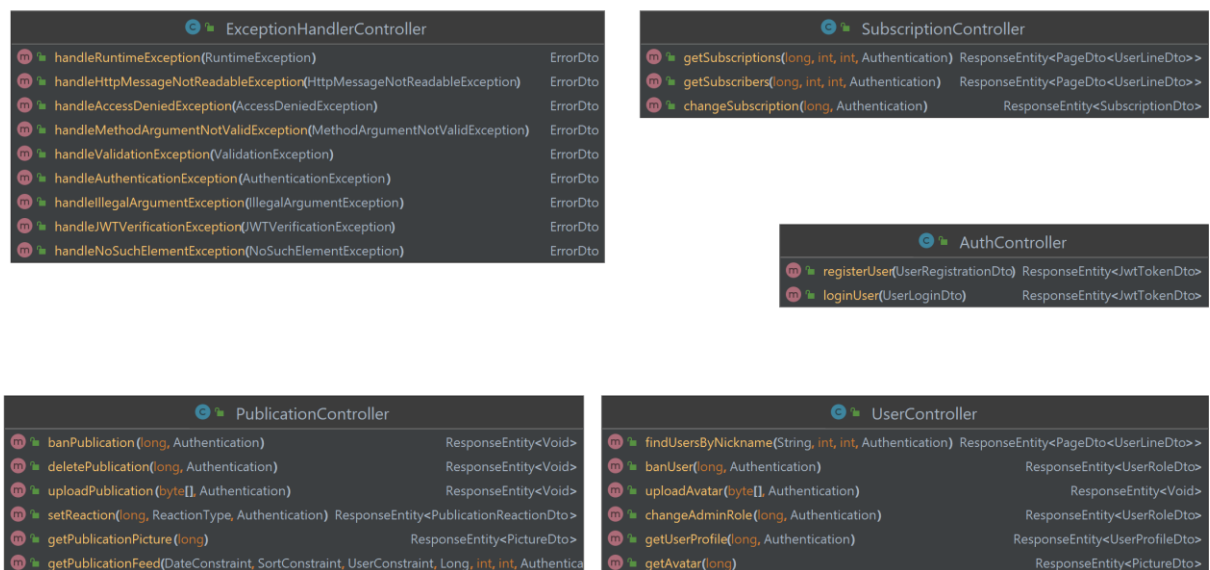


Рисунок 18 - Диаграмма классов контроллеров

3.10.7 Диаграмма классов-конфигураций

Классы конфигурации служат для создания некоторых сервисных объектов и настройки приложения. `SwaggerConfig` предоставляет общую информацию для документации API, а также обрабатывает отображение необходимых полномочий для эндпоинтов.

`ModelMapperConfig` создает объект для преобразования между классами сущностей и объектами передачи данных (DTO), а `YandexCloudConfig` предоставляет данные для подключения к облачному хранилищу.

SecurityConfig настраивает CORS, ставит фильтр для авторизации с помощью JWT токенов и предоставляет экземпляр объекта для хэширования паролей.

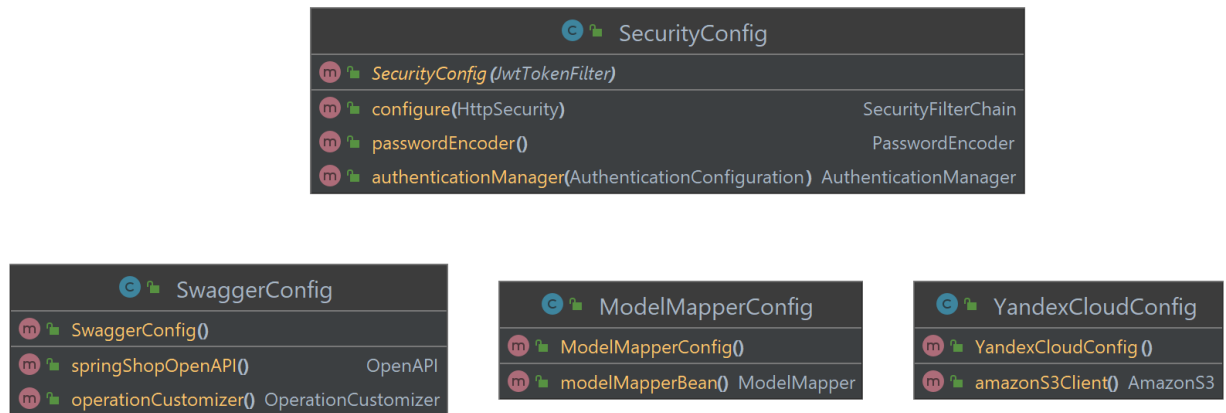


Рисунок 19 - Диаграмма классов конфигураций

3.10.8 Диаграмма классов для передачи данных

Эти классы были созданы в соответствии с шаблоном проектирования Data Transfer Object и нужны для возможности изменения моделей без необходимости менять API.

| UserLineDto | |
|-------------------|---------|
| <i>userId</i> | Long |
| <i>subscribed</i> | Boolean |
| <i>nickname</i> | String |

| UserRoleDto | |
|----------------|----------|
| <i>userId</i> | Long |
| <i>newRole</i> | UserRole |

| PageDto<T> | |
|---------------|---------|
| <i>last</i> | boolean |
| <i>index</i> | int |
| <i>size</i> | int |
| <i>values</i> | List<T> |

| PublicationInfoDto | |
|----------------------|--------------|
| <i>uploaded</i> | Instant |
| <i>ownerNickname</i> | String |
| <i>rating</i> | Long |
| <i>ownerId</i> | Long |
| <i>userReaction</i> | ReactionType |
| <i>publicationId</i> | Long |

| SubscriptionDto | |
|-----------------|---------|
| <i>id</i> | Long |
| <i>created</i> | Instant |

| UserProfileDto | |
|---------------------------|----------|
| <i>subscribersCount</i> | Long |
| <i>role</i> | UserRole |
| <i>nickname</i> | String |
| <i>id</i> | Long |
| <i>subscriptionsCount</i> | Long |
| <i>subscribed</i> | Boolean |

| ErrorDto | |
|----------------|--------|
| <i>message</i> | String |

| PictureDto | |
|----------------|--------|
| <i>picture</i> | byte[] |

| JwtTokenDto | |
|--------------------|--------|
| <i>accessToken</i> | String |

Рисунок 20 - Диаграмма классов для передачи данных

3.11 Реализация клиентской части приложения

3.11.1 Макеты интерфейса

При запуске приложения перед пользователем появляется экран-заставка приложения с логотипом в центральной части.



Рисунок 21 - Экран-заставка

Сразу после этого пользователь попадает на экран ленты публикаций. Стоит упомянуть, что на всех экранах кроме экрана-заставки, экранов входа и регистрации, есть навигационная панель, имеющая три кнопки.

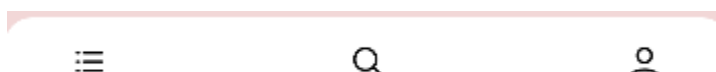


Рисунок 22 - Навигационная панель

Кнопка в виде списка ведет на экран ленты публикаций, кнопка в виде лупы – на экран поиска пользователей, кнопка в виде человека – на экран своего профиля в случае, если пользователь авторизован, или на экран входа, если пользователь не авторизован.

3.11.1.1 Экраны входа и регистрации

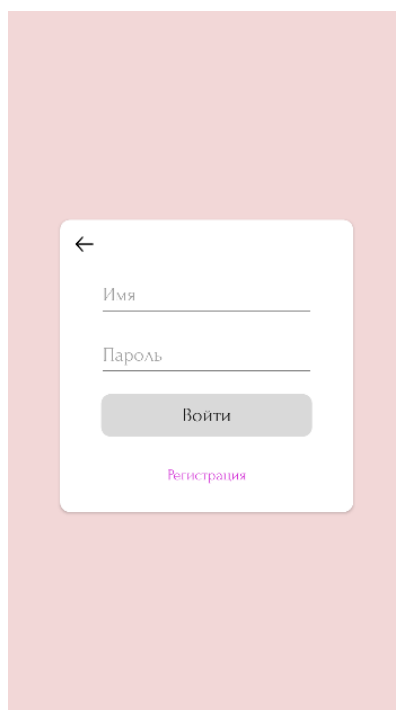


Рисунок 23 - Экран входа

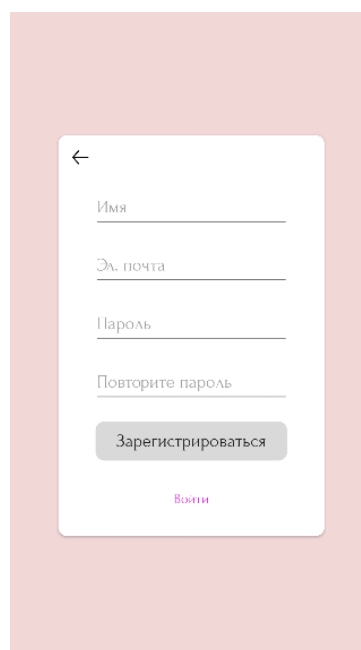


Рисунок 24 - Экран регистрации

Экраны входа и регистрации имеют похожий интерфейс. Кнопка стрелки назад возвращает пользователя на прошлый экран. С помощью кнопки снизу экрана можно переходить с одного экрана на другой, в зависимости от нужды

пользователя. Когда пользователь введет данные и нажмет серую кнопку, будет отправлен запрос на сервер. В случае какой-либо ошибки появится диалоговое окно с сообщением о ней. Если же пользователя ожидает успех, он будет перенаправлен на экран ленты публикаций. Теперь он авторизован.

3.11.1.2 Экран ленты публикаций

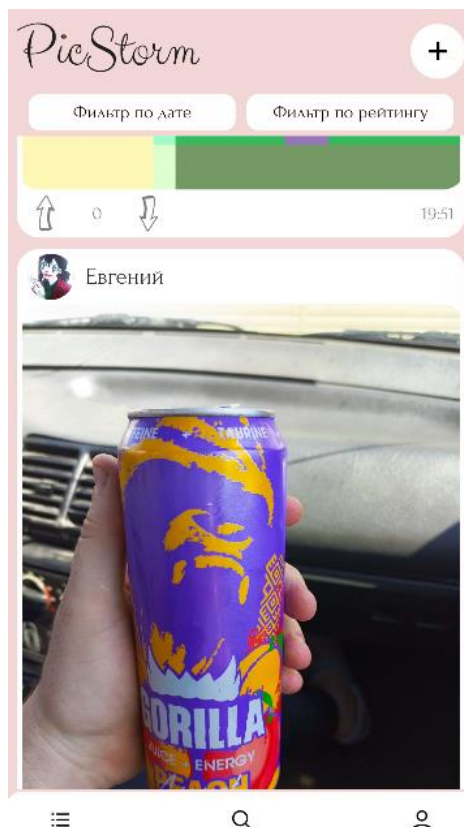


Рисунок 25 - Экран ленты публикаций неавторизованного пользователя



Рисунок 26 - Экран ленты публикаций авторизованного пользователя

На данном экране отображаются публикации пользователей, слева сверху логотип, справа сверху – кнопка для публикации своей фотографии. Если пользователь авторизован, то при нажатии на нее откроется галерея, иначе его перенаправит на экран входа.

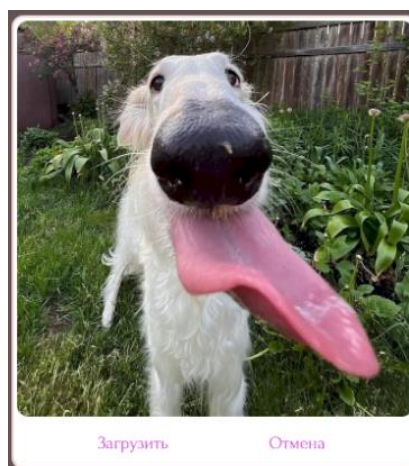


Рисунок 27 - Диалоговое окно подтверждения загрузки фотографии

Чуть ниже находятся выпадающие списки с выбором фильтров ленты, где пользователь может выбрать, что ему по душе.



Рисунок 28 - Выпадающие списки фильтров

Стоит отметить, что авторизованный пользователь может переключать ленту публикаций на личную (основанную на подписках) и общую.

Неавторизованному пользователю отключена возможность ставить оценки на посты – у него кнопки не активны и помечены серым цветом.

3.11.1.3 Экран поиска пользователей

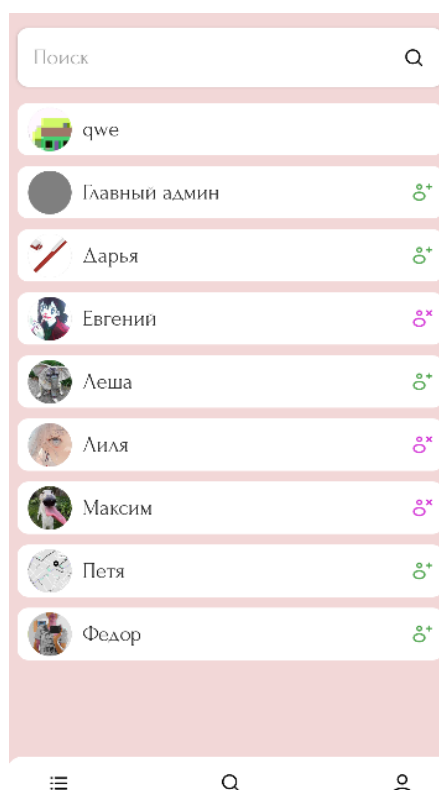


Рисунок 29 - Экран поиска пользователей

На экране поиска находится лента пользователей, при вводе текста в поле список будет обновляться, показывая найденных людей. По нажатию на элемент списка пользователь может перейти на профиль этого человека. Справа на элементе находится кнопка подписки или отписки от пользователя. Данной кнопки нет в случаях, если это ваш профиль, либо если вы – неавторизованный пользователь.

3.11.1.4 Экран профиля пользователя

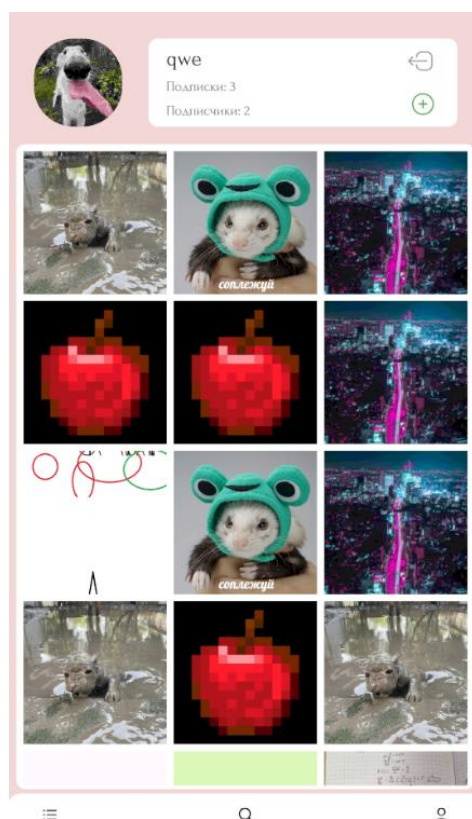


Рисунок 30 - Экран своего профиля

Слева сверху находится аватар пользователя. Если вы находитесь в своем профиле, то при нажатии на него вам будет предложено его заменить из своей галереи. На информационной панели отображается информация о имени пользователя, количестве его подписчиков и подписок. Так же, если это ваш профиль, то вы можете опубликовать новое фото, нажав на иконку с зеленым плюсом, или можете выйти из своего аккаунта, нажав на серую дверь со стрелочкой.

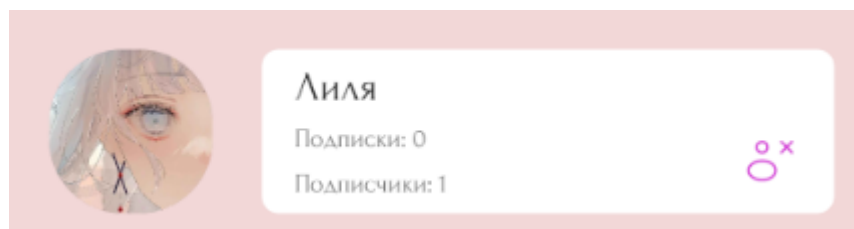


Рисунок 31 - Информационная панель чужого пользователя

На экране профиля чужого пользователя вместо этих кнопок есть кнопка, чтобы подписаться – зеленого цвета человек с плюсиком, либо отписаться – сиреневого цвета человек с крестиком.

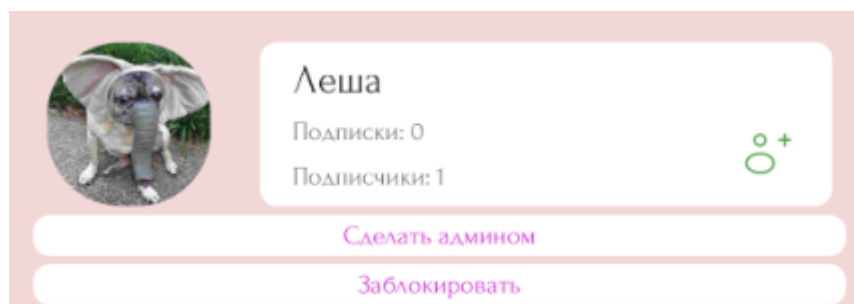


Рисунок 32 - Кнопки супер-администратора

Если роль пользователя – супер-администратор, под информационной панелью будет кнопка для того, чтобы сделать пользователя админом либо отнять у него эту роль, а также кнопка блокировки пользователя (есть и у обычного администратора).

При нажатии на одну из публикаций в ленте профиля пользователя перенаправит на конкретную публикацию в ленте публикаций конкретного пользователя.

При нажатии на подписчиков перенаправит на список подписчиков, при нажатии на подписки – на список подписок.

3.11.1.5 Экран ленты публикаций конкретного пользователя

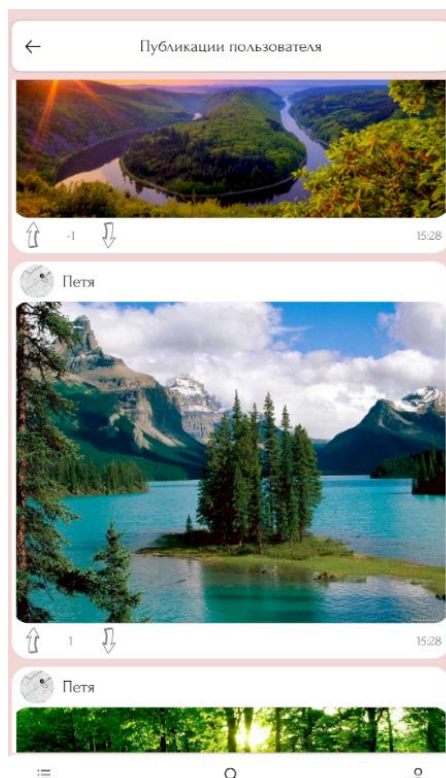


Рисунок 33 - Экран ленты публикаций конкретного пользователя

Данный экран похож на экран обычной ленты публикаций, разве что на верхней панели написано ваши ли это публикации или нет. При нажатии на стрелку назад пользователя вернет на предыдущий экран.

3.11.1.6 Экран списка подписчиков/подписок

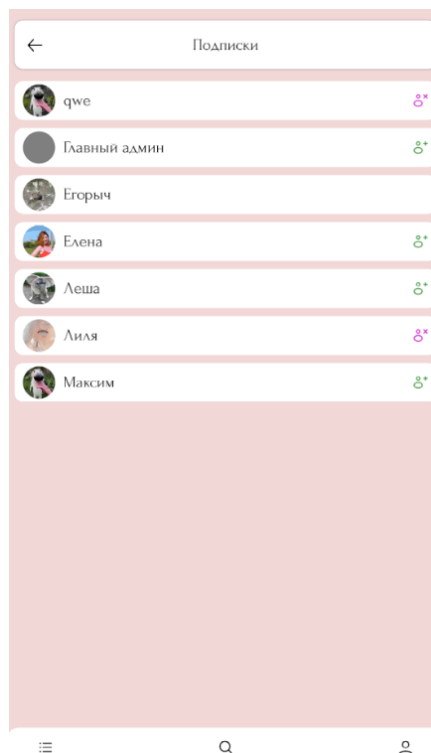


Рисунок 34 - Экран списка подписок

Внешне экран подписок такой же, как и экран подписчиков, разве что сверху написано его наименование. Стрелка назад при нажатии вернет пользователя на предыдущий экран. Элементы списка имеют такой же вид, как на экране поиска пользователей.

3.11.2 Архитектура клиентской части приложения

Клиентское приложение построено по архитектуре MVVM и состоит из 3-х слоев:

- Моделей, отражающих сущности, используемые в приложении
- Представлений, которые отвечают за отображение данных
- Моделей представлений, реализующих взаимодействие отображений с сущностями, используемыми в приложении

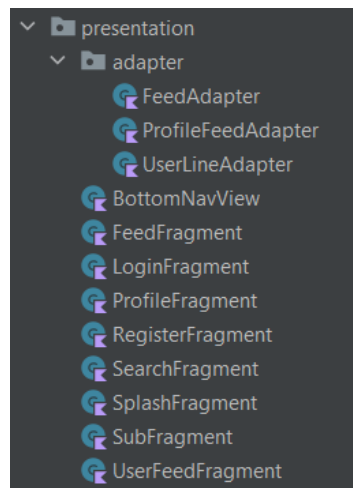


Рисунок 35 - Слой представлений

Фрагменты – это экраны приложения, в них инициализируются все элементы и их взаимодействие с пользователем. Адаптеры отвечают за поведение элементов различных списков, таких какие находятся в ленте публикаций (лента публикаций), профиле (лента публикаций в профиле), на экране поиска либо в списке подписчиков или подписок пользователя. Элементы этого слоя подписываются на обновление LiveData переменных в слое моделей представлений. Это значит, что любые их изменения будут приходить автоматически на экран, который должен их обрабатывать.

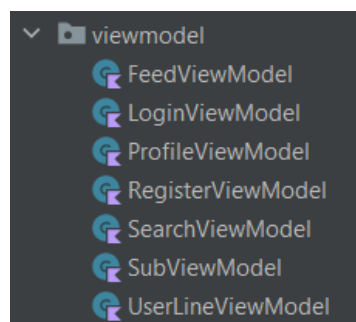


Рисунок 36 - Слой моделей представлений

Слой моделей представлений является связующим звеном двух других слоев. Там мы обращаемся к репозиторию, передавая данные с экрана. А он, в свою очередь, полученный объект отправляет запросом на сервер и дает ответ.

Слой моделей был разделен на зависимые и независимые от API сервера.

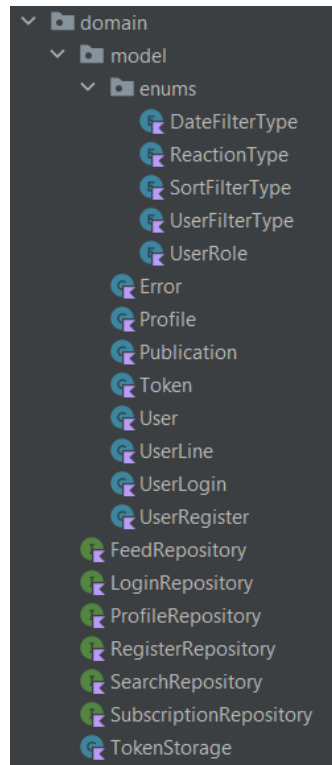


Рисунок 37 - Слой независимых моделей

Независимые — модели, которыми мы оперируем в приложении независимо от какого-либо сервера. Тут также находятся интерфейсы для всех репозиторий, хранилище токена пользователя, различные свои типы переменных: для типов фильтров, роли пользователя, типа реакции.

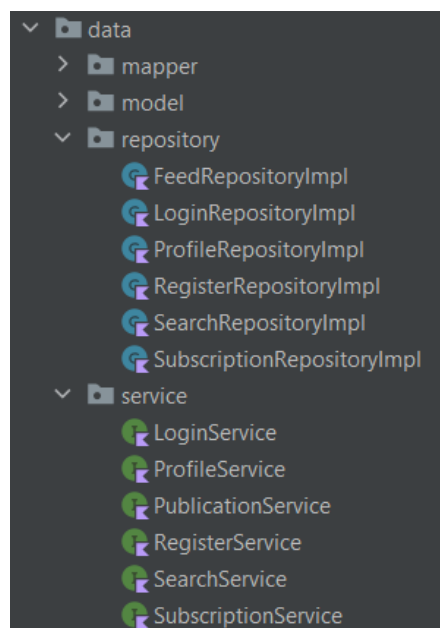


Рисунок 38 - Слой зависимых моделей: сервисы и репозитории

Слой зависимых моделей может изменяться в зависимости от сервера. Так, например, сервисы отвечают за структуру запросов, идущих на сервер. Имплементации репозитория отвечают за посылку этих запросов и прием данных от сервера.

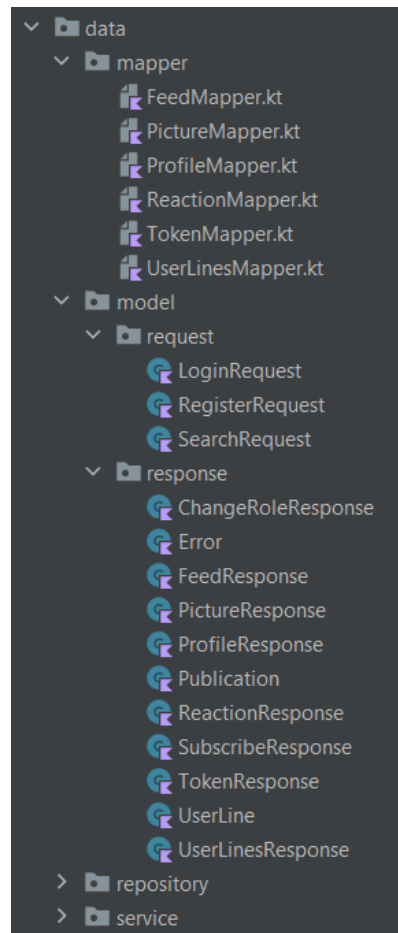


Рисунок 39 - Слой зависимых моделей: сущности и мапперы

В данном слое сущности имеют тот вид, в каком они пришли нам с сервера. Путем Json десериализации ответ с сервера преобразуется в такую сущность. Для того, чтобы с этими сущностями было удобно работать в приложении, реализованы мапперы. Они преобразуют десериализованную сущность в сущность нашего независимого слоя данных приложения.

3.11.3 Обработка фотографий при загрузке

Когда пользователь выбирает фото в своей галерее, выводится диалоговое окно с возможностью подтвердить загрузку либо отменить. Если загрузка

была подтверждена, фотография сжимается перед отправкой на сервер до 512 Килобайт, если это публикационная фотография, до 50 Килобайт, если это аватар пользователя. Сжатие фотографии до нужных размеров достигается путем постепенного уменьшения разрешения изображения и увеличения степени его сжатия.

```
class ImageCompressor {  
    companion object {  
        fun compress(image: Bitmap, size: Int): ByteArray {  
            val width = image.width  
            val height = image.height  
            val matrix = Matrix()  
            var scale = 1f  
            var newImage: Bitmap  
  
            val baos = ByteArrayOutputStream()  
            image.compress(Bitmap.CompressFormat.JPEG, quality: 100, baos)  
  
            var options = 100  
            while (baos.toByteArray().size / 1024 > size && options != 0 && scale != 0.0f) {  
                scale -= 0.05f  
                matrix.postScale(scale, scale)  
                newImage = Bitmap.createBitmap(image, x: 0, y: 0, width, height, matrix, filter: false)  
                baos.reset()  
                newImage.compress(Bitmap.CompressFormat.JPEG, options, baos)  
                options -= 5  
            }  
  
            return baos.toByteArray()  
        }  
    }  
}
```

Рисунок 40 - Функция для сжатия фотографий

3.12 Сценарии воронок

Сценарий воронки удаления публикаций из профиля определяет, какой процент от авторизованных пользователей использует данную функциональность.

Настройка воронки



Удаление публикации из профиля

| | | | |
|---|------------------------|---|---|
| 1 | Запуск приложения | Событие: Application started | + |
| 2 | Переход в свой профиль | Событие: User viewed his profile | + |
| 3 | Открытие публикации | Событие: User opened photo in his specified user feed | + |
| 4 | Нажатие на кнопку удал | Событие: User clicked delete photo in specified feed | + |
| 5 | Подтверждение удалени | Событие: User confirmed deletion | + |

+ Добавить шаг

Детальные настройки ^

| | | |
|----------------------|------------------|-----------------|
| События между шагами | Разрешены | Запрещены |
| Окно воронки | Без ограничений | Ограничить |
| Регистрация событий | Из любой сессии | Из одной сессии |
| Подсчет метрик | По пользователям | По устройствам |

Рисунок 41 - Настройка воронки «Удаление публикаций из профиля»

На следующем рисунке приведена диаграмма получившейся воронки после двух недель сбора данных.

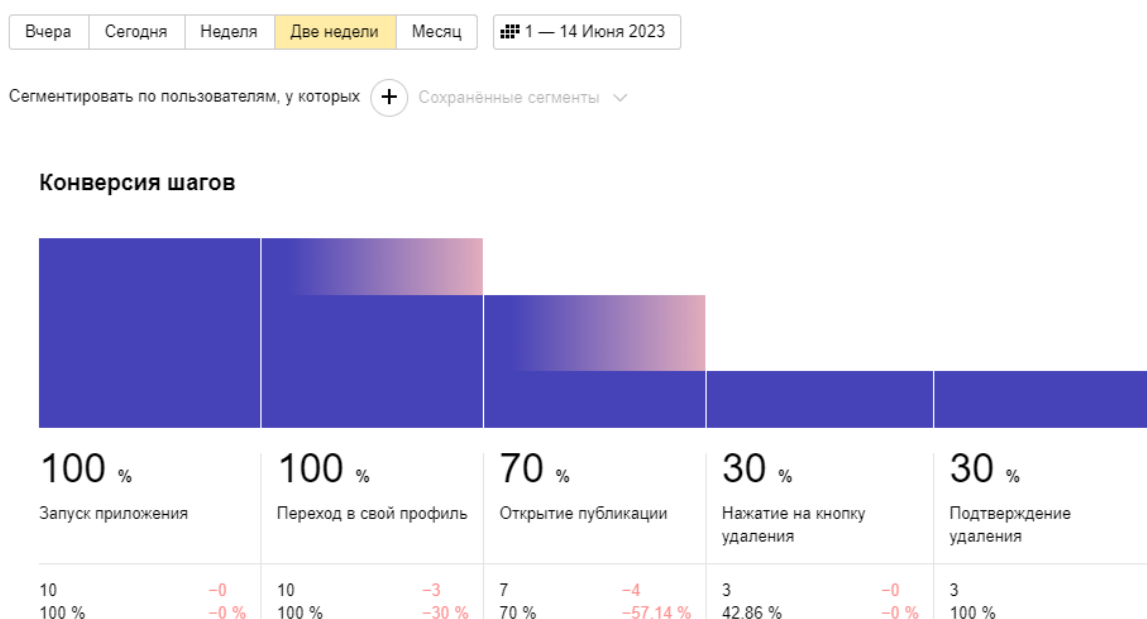


Рисунок 42 - Воронка «Удаление публикаций из профиля»

Следующий сценарий воронки определяет, какое количество пользователей решат узнать подписки того, чей профиль они просматривают.

Настройка воронки ✕

Просмотр подписок другого пользователя

| | | | |
|---|---------------------|---|---|
| 1 | Запуск приложения | Событие: Application started | + |
| 2 | Просмотр профиля | Событие: User viewed profile of another | + |
| 3 | Переход на подписки | Событие: User viewed subscriptions | + |

[+ Добавить шаг](#) [Детальные настройки ^](#)

События между шагами ? Разрешены Запрещены

Окно воронки ? Без ограничений Ограничить

Регистрация событий ? Из любой сессии Из одной сессии

Подсчет метрик ? По пользователям По устройствам

Сохранить Отмена Удалить воронку

Рисунок 43 - Настройка воронки «Просмотр подписок пользователя»

Получавшаяся в результате сбора статистики воронка приведена на следующем рисунке.

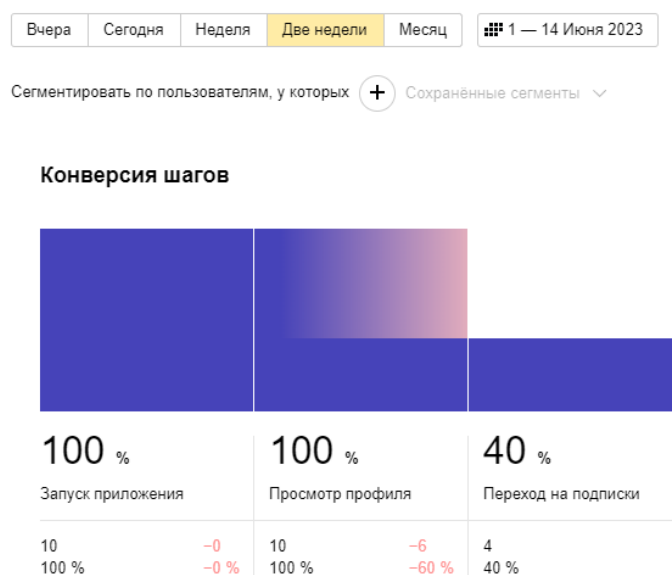


Рисунок 44 - Воронка «Просмотр подписок другого пользователя»

Последняя воронка призвана определить, сколько пользователей загружают фотографии из своего профиля, к ней также приведен получившийся результат.

Загрузка публикаций из профиля

1

Запуск приложения

Событие: Application started

+

2

Переход в свой профил

Событие: User viewed his profile

+

3

Выбор фото

Событие: User chosen photo to download in profile

+

4

Подтверждение

Событие: User confirms photo in profile

+

+

Добавить шаг

Детальные настройки ^

События между шагами ⓘ

Разрешены

Запрещены

Окно воронки ⓘ

Без ограничений

Ограничить

Регистрация событий ⓘ

Из любой сессии

Из одной сессии

Подсчет метрик ⓘ

По пользователям

По устройствам

Рисунок 45 - Настройка воронки «Загрузка публикаций из профиля»

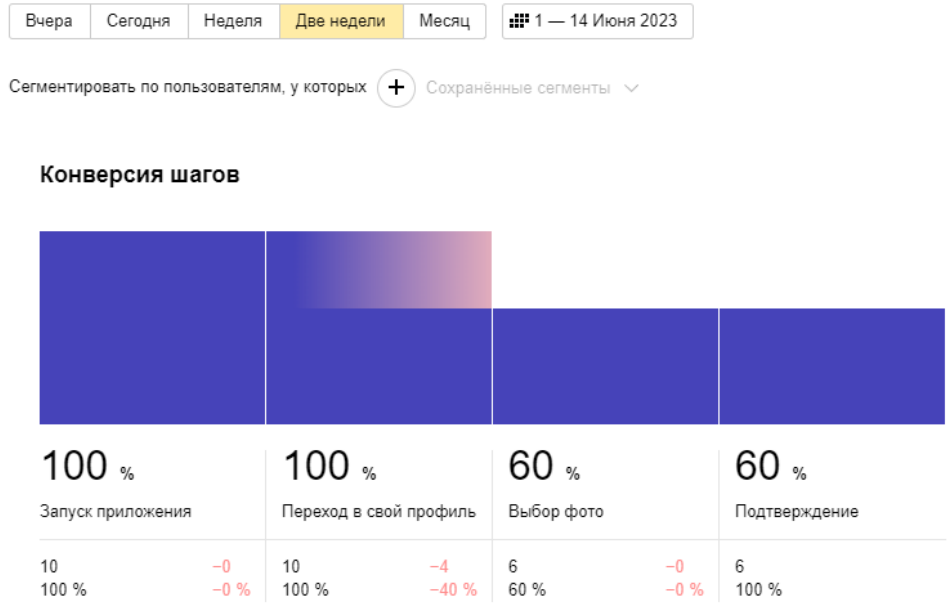


Рисунок 46 - Воронка «Загрузка публикаций из профиля»

К сожалению, собранной информации недостаточно для полноценного анализа результатов в силу малой выборки пользователей.

3.13 A/B тестирование

Используя сервис Firebase была настроена глобальная конфигурация для приложения. В ней содержится флаг `hasUploadPhotoButton`, показывающий нужно ли показывать кнопку загрузки фотографий (ведущую на экран входа) для неавторизованного пользователя. На основе нее было начато A/B тестирование.



Рисунок 47 - Объект A/B тестирования

Была выдвинута гипотеза, что присутствие данной кнопки повысит количество регистраций в приложении. Исходя из этого было настроено A/B тестирование.









| | |
|--|--|
|  Name and description | Upload photo button in feed |
|  Goals | Primary goal <code>unauthorized_enter_feed</code>  <code>user_registered</code>  Secondary goals <code>unauthorized_enter_feed</code> |
|  Targeting and distribution | App  <code>com.vsu.picstorm</code> Activation event <code>unauthorized_enter_feed</code> 100% of eligible users are in this experiment |
|  Test devices | 0 |
|  Variants | hasUploadPhotoButton |
| Baseline 50% | true |
| Variant A 50% | false |

Рисунок 48 - Настройка A/B тестирования

На момент написания курсового проекта данное тестирование все еще продолжается.

Заключение

В ходе выполнения данного курсового проекта был выполнен анализ предметной области и аналогов разрабатываемого приложения.

Для разработки приложения были разработаны макеты интерфейса, выбрана платформа приложения, построены UML диаграммы.

Для контроля версий были созданы репозитории на GitHub.

Были созданы сценарии воронок, подключен Swagger.

Back-end часть приложения и база данных были размещены на хостинге. Разработанное приложение удовлетворяет поставленным требованиям. Все поставленные цели были выполнены, а именно:

- Была создана платформа для публикации своих фотографий
- Обеспечена возможность просмотра фотографий на основе выбора пользователя
- Предоставлена возможность пользователем получать реакции на свои публикации

В качестве дальнейшего развития проекта рассматриваются следующие совершенствования:

- Создание IOS версии приложения
- Создание Web версии приложения
- Реализация возможности комментировать публикации пользователей
- Реализация возможности группировки фотографий по определенной тематике
- Добавить поддержку создания сообществ по интересам пользователей

Список использованных источников

1. Анализ рынка социальных сетей [Электронный ресурс]. – Режим доступа: URL: https://new-retail.ru/marketing/sotsialnye_seti/rynok_sotsialnykh_setey_v_pervoy_polive_2021_goda_situatsiya_v_mire_i_v_rossii8776/ - Заглавие с экрана. – (Дата обращения 30.05.2023).
2. Документация Android Developer [Электронный ресурс]. – Режим доступа: URL: <https://developer.android.com/docs> - Заглавие с экрана. – (Дата обращения 10.04.2023).
3. Интеграция Navigation Component [Электронный ресурс]. – Режим доступа: URL: <https://developer.android.com/guide/navigation/get-started> – Заглавие с экрана. – (Дата обращения 04.05.2023).
4. Руководство по Dependency Injection в Android [Электронный ресурс]. – Режим доступа: URL: <https://developer.android.com/training/dependency-injection> – Заглавие с экрана. – (Дата обращения 25.04.2023).
5. Руководство по созданию диалоговых окон в Android [Электронный ресурс]. – Режим доступа: URL: <https://developer.android.com/guide/fragments/dialogs> - Заглавие с экрана. – (Дата обращения 28.05.2023).
6. Документация Spring Data JPA [Электронный ресурс]. – Режим доступа: URL: <https://docs.spring.io/spring-data/jpa/docs/current/reference/html/> - Заглавие с экрана. – (Дата обращения 20.04.2023).
7. Документация Swagger [Электронный ресурс]. – Режим доступа: URL: <https://swagger.io/docs/> – Заглавие с экрана. – (Дата обращения 25.04.2023).
8. Документация Docker [Электронный ресурс]. – Режим доступа: URL: <https://docs.docker.com/get-started/overview/> - Заглавие с экрана. – (Дата обращения 10.04.2023).