# ВВЕДЕНИЕ

Целью данного проекта является изучение, проектирование и написание собственной программы для выбранной ЭВМ. В качестве ЭВМ была выбрана система под управлением Android.

Android — операционная система для смартфонов, планшетов, электронных книг, цифровых проигрывателей, наручных часов, фитнес-браслетов, игровых приставок, ноутбуков, нетбуков, смартбуков, очков Google Glass, телевизоров и других устройств. Основана на ядре Linux и собственной реализации виртуальной машины Java от Google. Изначально разрабатывалась компанией Android, Inc., которую затем купила Google[4].

Эта система была выбрана не случайно, так как она на данный момент является одной из самых популярных систем в мир, а также для нее есть написана хорошая документация, которая будет понятна даже новичку в данной области. Благодаря распространённости данной системы, сообщество разработчиков велико, что позволяет найти необходимый совет, когда это необходимо.

Для разработки было выбрано приложение под названием «Покормите студента». Приложение позиционируется как книга несложных кулинарных рецептов под любой случай. При этом, важным условием приготовления любого из этого рецепта является то, что оно не требует особых навыков в области кулинарии или специфических инструментов.

Идея социального приложения очень проста – давать людям общаться между собой. Но у каждой соцсети есть своя уникальная идея. Instagram – это приложение для публичного обмена фотографиями. Запечатление своих жизненных моментов и сохранение их. Twitter – создан для публичного высказывания своих идей, мыслей, где каждый может написать свое мнение на чьей-то ещё.

SelMe – это приложение для обмена фотографиями с элементами социальной сети. Данное приложение предназначено для совместного голосования за ту или иную фотографию. Люди часто не могут сделать правильный выбор или не знают, что выбрать из двух зол: кто-то не может выбрать какое платье надеть на свидание, кто-то не знает какую марку автомобиля купить, или самое страшное это не знать, какие же пельмешки лучше купить в магазине. Для таких случаев и существует данное приложение.

# АНАЛИЗ И ПОСТАНОВКА ЗАДАЧИ

Цель курсового проекта создание социальной сети. Любая социальная сеть должна отвечать некоторым стандартным требованиям:

* регистрация пользователя;
* создание собственного профиля;
* вход через логин и пароль;
* создание поста.

Как было написано выше, что каждая соцсеть должна иметь свой индивидуальную составляющую. В данном приложение должны быть осуществлены следующие функциональные характеристики:

* только авторизованные пользователи имеют доступ к функционалу приложения, за исключение входа и регистрации;
* пользователь может добавлять фотографии из своей галереи на телефоне;
* пользователь может делать описание к своему посту;
* каждый пользователь может голосовать за ту или иную картинку;
* каждый пользователь может оставить комментарий под любым постом;
* у каждого пользователя есть свой профиль с личными данными;
* пользователь может восстановить свой пароль.

Так же в приложении были выставлены некоторые ограничения:

* пользователь может оставить только один комментарий под постом;
* пользователь может добавить только две фотографии в пост, больше или меньше не допускается;
* пользователь может только один раз проголосовать, без возможности поменять свой выбор;
* пользователь не имеет возможности отредактировать пост;
* пользователь не имеет возможности удалить пост.

# ПРОГРАММНОЕ ПРОЕКТИРОВАНИЕ

## 2.1 Функциональное проектирование

При проектировании программы была разработана архитектура данных MVP.

MVP - схема разделения данных приложения, пользовательского интерфейса и управляющей логики на три отдельных компонента: model, view, presenter. У каждого из них есть свои обязанности и связь между ними происходит через presenter.

**Model** содержит бизнес-логику приложения. Он контролирует как данные создаются, сохраняются и изменяются.

**View** это интерфейс, который отображает данные и направляет действия пользователя в Presenter.

**Presenter** выступает в роли посредника. Он извлекает данные из Model и показывает их в View. Он также обрабатывает действия пользователя, переданные ему из View.

MVP является аналогом MVC, но с более современной парадигмой, которая создаёт лучшее разделение ролей и максимизирует тестируемость приложения[3]. Схема работы MVP представлена на рисунке 2.1.1.

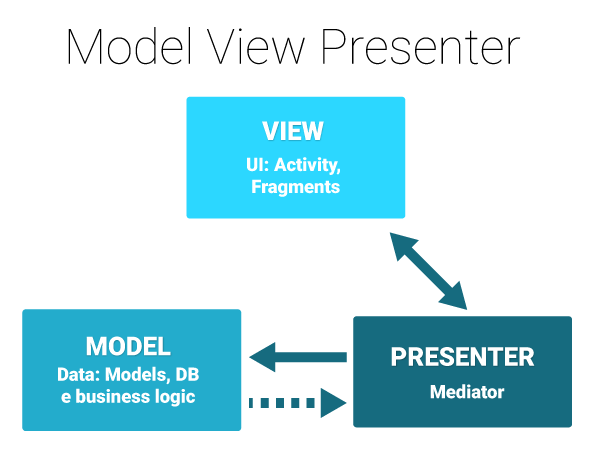


Рисунок 2.1.1 – Схема MVP

В данной схеме разделения проекта данных существуют следующие модули, у каждого модуля есть своя роль:

* Model;
  1. Dto;
  2. Entity;
* Presenter;
  1. Dao;
  2. Interfaces;
  3. Service;
* View;
  1. Activity;
  2. Adapter;
  3. Fragments.

**Entity.** Entity – это сущность которая является отображением в базе данных. В проекте есть две сущности: PostEntity и UserEntity. В этих объектах не может быть ничего кроме полей, которые аналогичны полям в базе данных. Все поля сделаны приватными и для них реализованы геттеры, сеттеры и вспомогательная функции equals. Entity объекты не могут напрямую взаимодействовать с компонентами view.

**DTO**. Dto (data transfer object) объекты используется для передачи данных между подсистемами приложения. Data Transfer Object, в отличие от data access object не должен содержать какого-либо поведения. Так же в отличии от entity dto объекты могут напрямую взаимодействовать с view объектами. Dto объекты реализуют Parcelable интерфейс, чтобы данный объект можно было передавать например через intent

**DAO.** DAO (data access object) объекты реализую возможность взаимодействия с базой данных. Dao объекты напрямую взаимодействую с entity. В данном объекте может быть реализованы методы сохранения данных, удаления данных, изменения данных и обновление данных из базы данных. Схема работы DAO объекта представлена на рисунке 2.1.2.

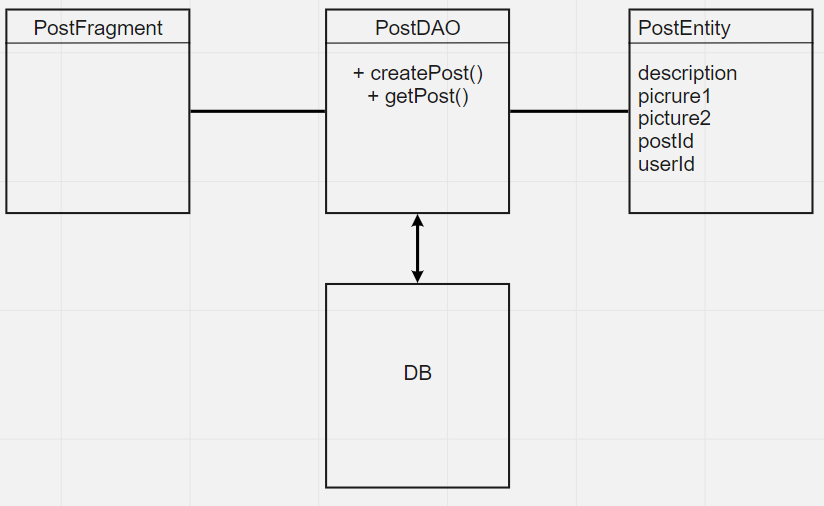


Рисунок 2.1.2 – Схема работы DAO объекта

**Services.** В данном модуле реализованы утилитные объекты, которые помогают работать с другими объектами. Например, существует класс DataMapper. Данный класс помогает переводить данные, которые приходят с сервера в DTO объекты. Схема работы объекта DataMapper представлена на рисунке 2.1.3.

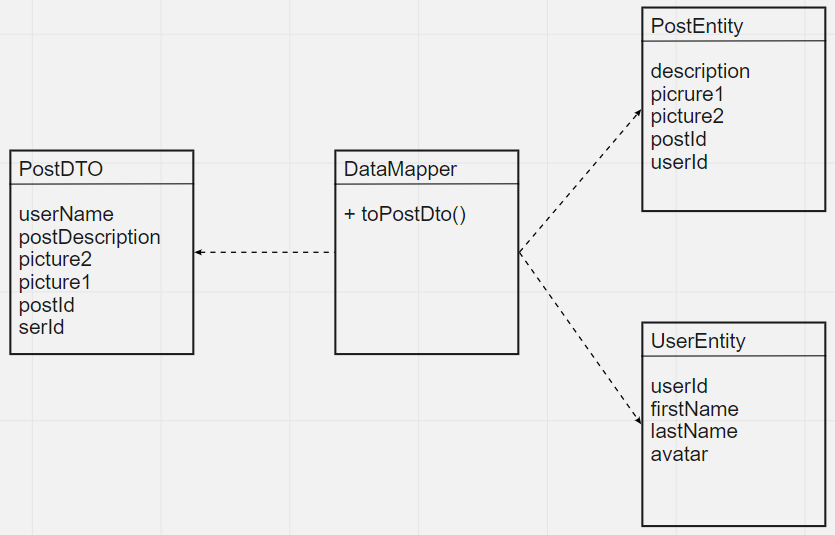


Рисунок 2.1.3 – Схема работы DataMapper

**View.** View модуль реализует объекты, которые дают возможность взаимодействия с интерфейсом. В данном модуле хранятся классы фрагментов, активити и адаптеров. У каждого класса есть привязка к xml документу, в котором находится верстка интерфейса.

Для удобства откладки в проекте используется стандартная библиотека Android android.util.Log. Очень часто программисту нужно вывести куда-то промежуточные результаты, чтобы понять, почему программа не работает. Некоторые временно размещают на экране текстовую метку и выводят туда сообщение при помощи метода textView.setText("Здесь был Васька"), но это не совсем правильное решение и очень неудобное.

Класс android.util.Log позволяет разбивать сообщения по категориям в зависимости от важности. Для разбивки по категориям используются специальные методы, которые легко запомнить по первым буквам, указывающие на категорию:

* Log.e() - ошибки (error);
* Log.w() - предупреждения (warning);
* Log.i() - информация (info);
* Log.d() - отладка (degub);
* Log.v() - подробности (verbose);
* Log.wtf() - очень серьезная ошибка! (What a Terrible Failure!, работает начиная с Android 2.2).

Но надо помнить, что множественное использование логов также пагубно влияет. Компания Google в своей документации настойчиво рекомендует удалять логи при сборке релиза и разумно этим рекомендациям следовать[1].

Диаграмма вариантов использования представлена в приложении Б.

## 2.2 Проектирование базы данных

В качестве базы данных была выбрана Firebase Firestore database. Данная база основана на NoSQL, но со своими примочками. Firesore - это гибкая, масштабируемая база данных для разработки на мобильных устройствах, веб и на серверах от Firebase и Google Cloud Platform.

Данная система была выбрана, потому что она легка в освоении, для нее написана понятная документация, которая ответит на всевозможные вопросы по реализации. Так же так как это разработка от компании Google, то она легко интегрируется в приложения под управлением Android.

Структура немного отличается от традиционной базы данных, таких как MySQL, PostgreSQL или MSSQL. База содержит коллекции. Коллекции – это своего рода таблицы в традиционных бд. В каждой коллекции есть документ с уникальным именем и уже в документе находятся поля.

В данном проекте будет две коллекции: users и post.

Коллекция users содержит информацию о данных пользователя. Под каждого пользователя создается уникальных документ с набор полей одинаковый для всех. Названия полей:

* description (тип String) – здесь содержится описание которое дал пользователь о себе;
* firstName (тип String) – имя пользователя;
* lastName (тип String) – фамилия пользователя;
* profilePhoto (тип String) – название файла фотографии в хранилище;
* userId (тип String) – уникальный идентификатор пользователя.

На рисунке 2.2.1 представлен пример заполнения базы данных данными.

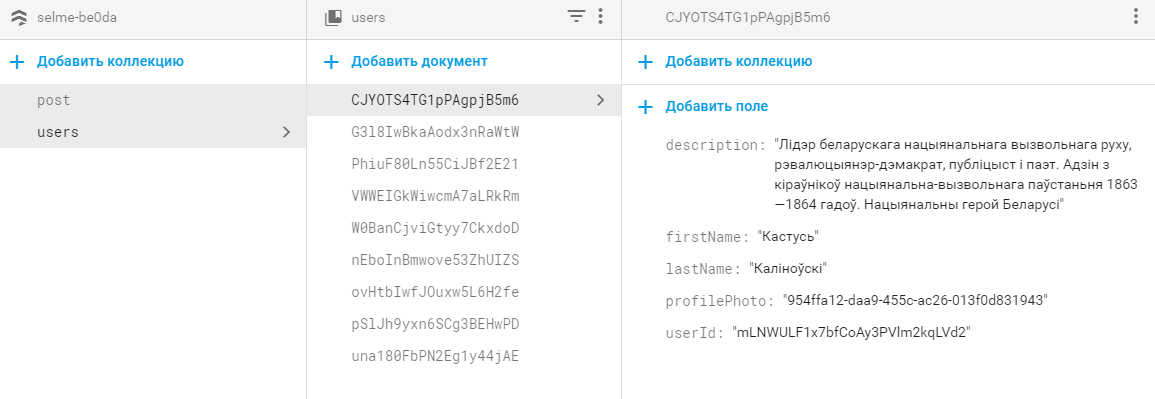


Рисунок 2.2.1 – Пример заполнения базы

Коллекция post содержит информацию о посте. Который создал авторизованный пользователь. Каждый пост — это отдельный документ с уникальным именем в базе данных. В данной коллекции реализованы следующие поля:

* comments (тип Map<String, String>) – содержит комментарии к посту, где ключ это userId, а значение это комментарий;
* createdDate (тип timestmp) – содержит дату создания поста;
* description (тип String) – содержит описание поста;
* docId (тип String) – содержит название документа поста в бд;
* likes (тип Map<String, Boolean>) - содержит пользователь, кто оставил лайк, где ключ это userId, а значение это поставлен лайк или нет
* photo1 (тип String) – название файла фотографии в хранилище;
* photo2 (тип String) – название файла фотографии в хранилище;
* pickPic1 (тип int) – количество проголосовавших за первую картинку
* pickPic2 (тип int) – количество проголосовавших за вторую картинку
* userId (тип String) – уникальный идентификатор пользователя.
* votedUserIds (тип Array<String>) – userId пользователей, кто проголосовал.

На рисунке 2.2.2 представлен пример заполнения коллекции post данными.

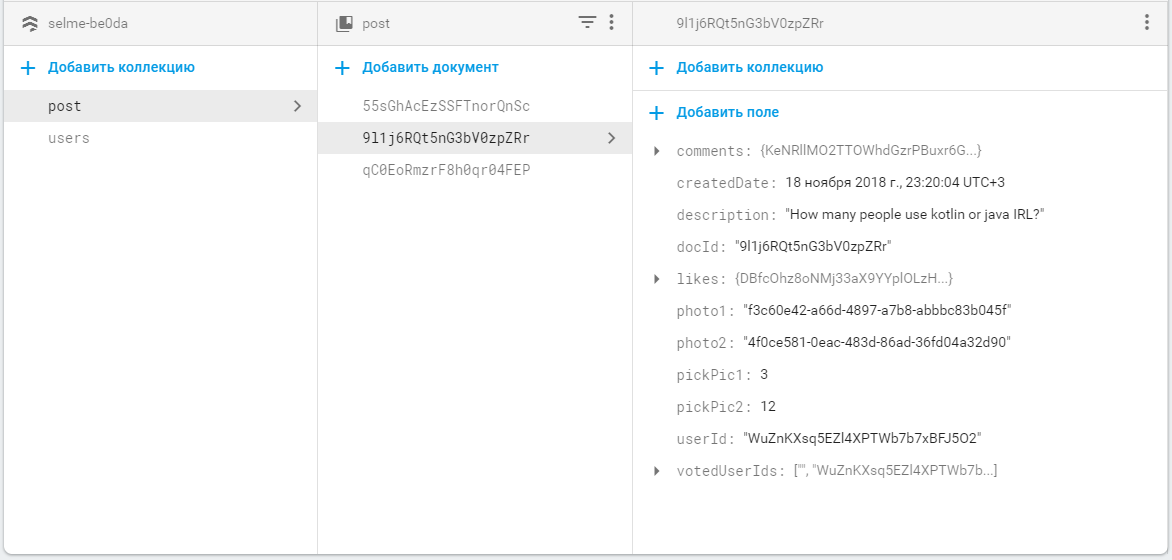


Рисунок 2.2.2 – Пример заполнения коллекции post данными

# ПРОГРАММНАЯ РЕАЛИЗАЦИЯ И ТЕСТИРОВАНИЕ

## Детальная реализация функциональных частей ПО

Для реализации основного функционала был выбран шаблон BottomNavigationView (Листинг 3.1.2). Этот шаблон реализует все необходимые требования Material Design и рекомендован к применению Google[1]. Этот шаблон легок для понимания пользователем и легок в применении.

В MainActivity реализован данный шаблон, так как это Аctivity взято за главное и прописано в файле Manifest (Листинг 3.1.1).

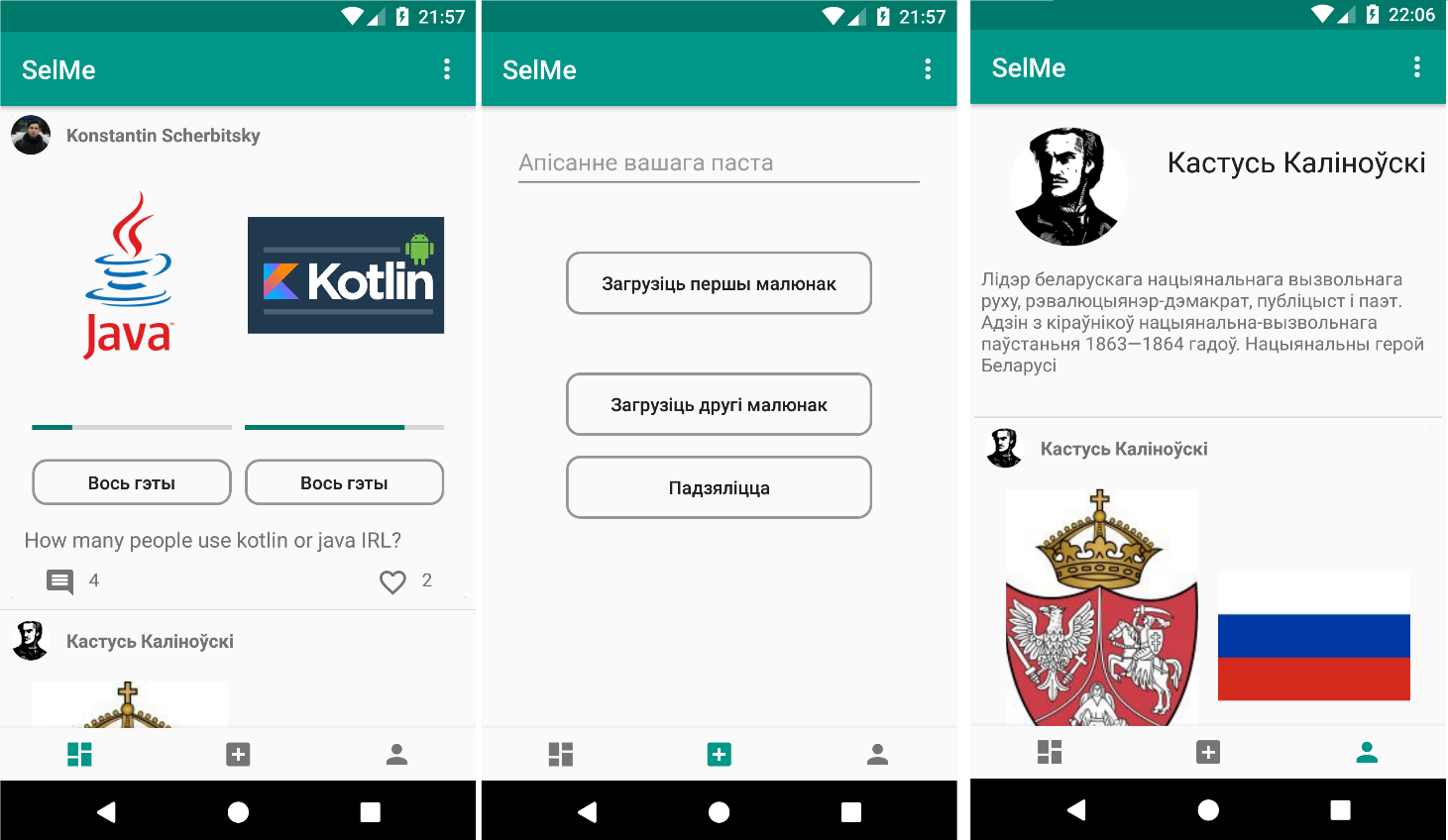
**Листинг 3.1.1** – Определение MainActivity как главной в AndroidManifest

1. <activity
2. android:name=".view.activity.MainActivity"
3. android:label="@string/app\_name" >
4. <intent-filter>
5. <action android:name="android.intent.action.MAIN" />
6. <category android:name="android.intent.category.LAUNCHER" />
7. </intent-filter>
8. </activity>

**Листинг 3.1.2** – Реализация BottomNavigationView в activity\_main.xml

1. <FrameLayout
2. android:id="@+id/fragment\_container"
3. android:layout\_width="match\_parent"
4. android:layout\_height="match\_parent"
5. android:layout\_marginBottom="40dp"
6. android:text="@string/title\_home"
7. app:layout\_constraintLeft\_toLeftOf="parent"
8. app:layout\_constraintTop\_toTopOf="parent" />
9. <android.support.design.widget.BottomNavigationView
10. android:id="@+id/navigation"
11. android:layout\_width="0dp"
12. android:layout\_height="40dp"
13. android:layout\_marginStart="0dp"
14. android:layout\_marginEnd="0dp"
15. app:layout\_constraintBottom\_toBottomOf="parent"
16. app:layout\_constraintLeft\_toLeftOf="parent"
17. app:layout\_constraintRight\_toRightOf="parent"
18. app:labelVisibilityMode="unlabeled"
19. app:menu="@menu/navigation" />

Как можно там присутствует FrameLayout он существует для того, чтобы в этот layout можно было заносить фрагменты, в которых реализованы views приложения. На рисунке 3.1.1 показаны скриншоты главных экранов приложения



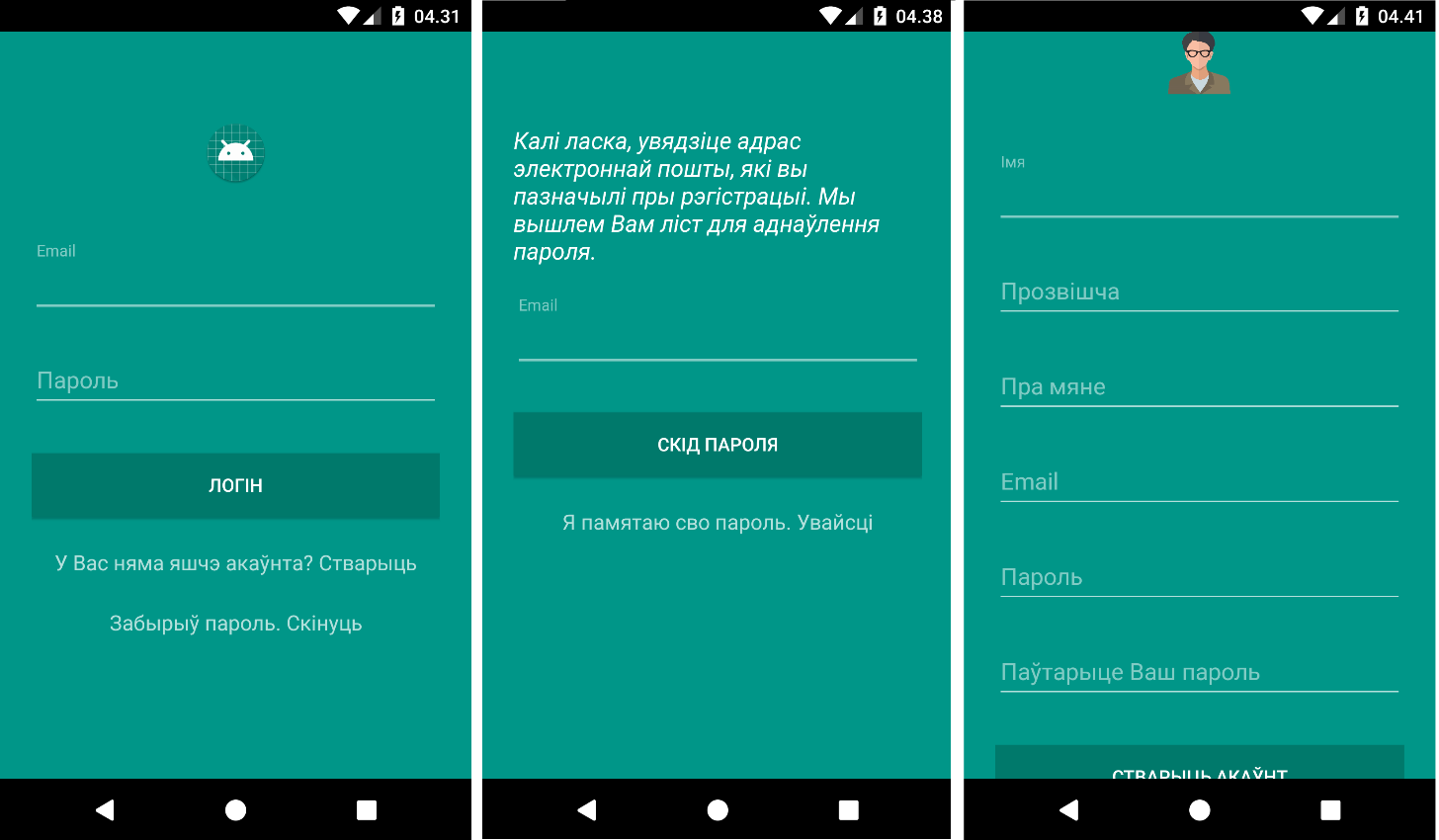
**Рисунок 3.1.1** – Скриншоты главных экранов приложения

В MainActivity реализован метод loadFragment() который заменяет FrameLayout на Fragment (Листинг 3.1.3).

**Листинг 3.1.3** – Метод загрузки фрагмента

1. private boolean loadFragment(Fragment fragment) {
2. Log.d(TAG, "loadFragment() called with: fragment = [" + fragment + "]");
3. try {
4. getSupportFragmentManager()
5. .beginTransaction()
6. .replace(R.id.fragment\_container, fragment)
7. .commit();
8. return true;
9. } catch (NullPointerException ex) {
10. Log.e(TAG, "loadFragment: Cannot to load fragment" + fragment, ex);
11. return false;
12. }
13. }

В приложении реализован функционал для регистрации нового пользователя и для входа, если аккаунт был уже создан. Так же есть возможность восстановления пароля. Для каждой действия была реализована отдельная активити. Все активити выдержаны в одном стиле, который придерживаются рекомендациям Material Design[1]. На рисунке 3.1.2 представлены экраны входа, восстановления и регистрации пользователя.



**Рисунок 3.1.2** – Скриншоты экранов входа, восстановления пароля, регистрации соответственно

Начиная с Android 6.0 Google сделал обязательным запрос на разрешения к разрешениям, которые используются в приложении[1]. Поэтому в SignUpActivity были реализованы два методы, которые за это отвечают, также в AndroidManifest были прописаны запрашиваемые разрешения, представленные в листинге 3.1.4 и листинге 3.1.5.

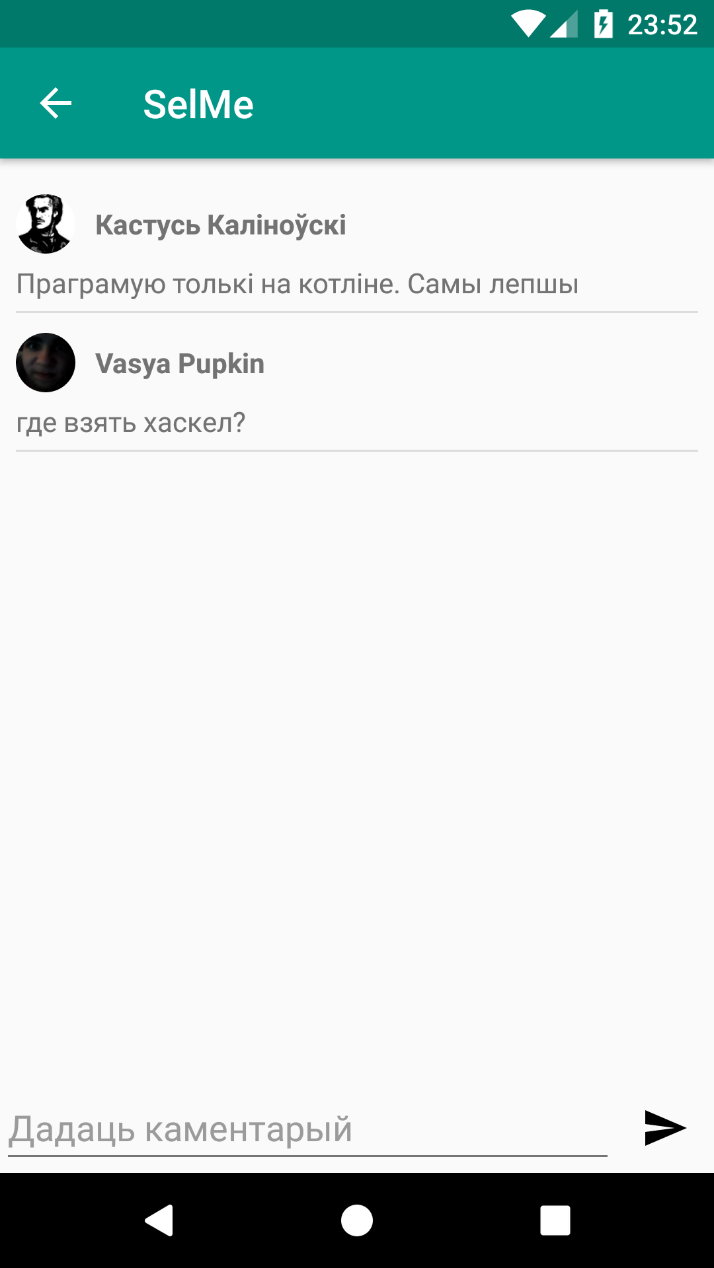
**Листинг 3.1.4** – Запрашиваемые разрешения у пользователя

1. <uses-permission android:name="android.permission.WRITE\_EXTERNAL\_STORAGE" />
2. <uses-permission android:name="android.permission.READ\_EXTERNAL\_STORAGE" />
3. <uses-permission android:name="android.permission.CAMERA"/>

**Листинг 4.1.5** – Методы реализации запроса у пользователя

1. private void verifyPermissions(){
2. Log.d(TAG, "verifyPermissions: asking user for permissions");
3. String[] permissions = {Manifest.permission.READ\_EXTERNAL\_STORAGE,
4. Manifest.permission.WRITE\_EXTERNAL\_STORAGE,
5. Manifest.permission.CAMERA,
6. Manifest.permission.INTERNET};
7. if(ContextCompat.checkSelfPermission(this.getApplicationContext(), permissions[0])
8. == PackageManager.PERMISSION\_GRANTED &&
9. ContextCompat.checkSelfPermission(this.getApplicationContext(), permissions[1])
10. == PackageManager.PERMISSION\_GRANTED &&
11. ContextCompat.checkSelfPermission(this.getApplicationContext(), permissions[2])
12. == PackageManager.PERMISSION\_GRANTED &&
13. ContextCompat.checkSelfPermission(this.getApplicationContext(), permissions[3])
14. == PackageManager.PERMISSION\_GRANTED){
15. openGallery();
16. } else {
17. Log.wtf(TAG, "verifyPermissions: permissions was reject");
18. ActivityCompat.requestPermissions(SignUpActivity.this, permissions, REQUEST\_PERMISSIONS);
19. }
20. }
21. @Override
22. public void onRequestPermissionsResult(int requestCode, @NonNull String[] permissions, @NonNull int[] grantResults) {
23. Log.wtf(TAG, "onRequestPermissionsResult: try get permissions again");
24. verifyPermissions();
25. }
26. }

Так же в приложении присутствует отдельная активити для комментариев (Рисунок 3.1.3).



**Рисунок 3.1.3** – Скриншот экрана комментариев

## Сопроводительная документация

Сопроводительная документация по разработанному мобильному приложению представлена в приложении А.

## Анализ ПО

Приложение разрабатывалось, следуя современным тенденциям программирования. Были соблюдены все возможные рекомендации от компании Google, что делает его более безопасным, менее ресурс затратным и понятным для обычного пользователя.

Программа разработана с помощью MVP и VCS Git, поэтому поддержка данного ПО будет облегчена, а поиск багов будет удобен, так как ко структурирован и читабельный.

Приложение переведено на два языка: английский и белорусский. На английском велась разработка приложения поэтому, если на телефоне установлена локализация отличная от английской или белорусской, то приложение будет на английском языке, а если локализация соответствует белорусской, то и приложение будет на белорусском.

## Тестирование ПО

Тестирование программного обеспечения – процесс исследования, испытания программного продукта. Целью тестирования является выявить ситуации, в которых поведение программы является неправильным, нежелательным или не соответствующим спецификации.

Тестированием программного обеспечения проводилось в определённом порядке с целью получения информации о качестве работы.

Тестирование обеспечивает:

* + - Обнаружение ошибок.
    - Демонстрацию соответствия функций программы ее назначению.
    - Демонстрацию реализации требований к характеристикам.

Приложение было протестировано на нескольких устройствах с различными характеристиками. Основное тестирование проходило на телефоне Meizu m5s со следующими характеристиками:

* Операционная система: Android 6.0;
* Экран 5.3”;
* Разрешение экрана 720х1280;
* ОЗУ 3 гб.

Также параллельно тестирование проводилось на эмуляторе с характеристиками:

* Операционная система: Android 7.1.1;
* Экран 4.95”;
* Разрешение экрана 1080х1920;
* ОЗУ 1.5 гб.

Корректность выполнения определяется сравнением фактического и ожидаемого результатов. Если результаты совпали, то тестирование пройдено успешно. Если же результаты не равны, то произойдет сбой системы и пользователю выведется системное окно об ошибке. Все действия, проведённые в процессе тестирования, а также их результаты приведены в таблице 3.4.1.

**Таблица 3.4.1** – Тестирование программы

|  |  |  |
| --- | --- | --- |
| **Действие** | **Ожидаемый результат** | **Результат** |
| Запуск программы | Открытия окна входа, если вход не был произведен ранее или открытие окна «Сцяна» | Открытия окна входа, если вход не был произведен ранее или открытие окна «Сцяна» |
| Вход как зарегистрированный пользователь | Открытие окна «Сцяна» | Открытие окна «Сцяна» |
| Вход как не зарегистрированный пользователь | Получение сообщении об ошибки входа | Показан SnackBar с сообщением, что вход не возможен |
| Регистрация пользователя при заполнении всех полей | Открытие окна «Сцяна» | Открытие окна «Сцяна» |
| Регистрация пользователя при неправильном заполнении полей | Получение сообщении об ошибки регистрации и подсвечивания поля с ошибкой | Поле, которое было указано неправильно, либо не введено было подсвечно |
| Восстановление пароля | Получения письма с ссылкой для восстановления пароля на эл. почту | Получения письма с ссылкой для восстановления пароля на эл. почту |
| Переход на экран создания поста | Переход на экран создания поста | Переход на экран создания поста |
| Создание поста, если все картинки добавлены и все поля заполнены | Отображение поста на общей ленте и в своем профиле | Отображение поста на общей ленте и в своем профиле |
| Создание поста, если какое-то поле было пропущено | Получение сообщении об ошибки | Получение сообщении об ошибки |

Продолжение таблицы 3.4.1

|  |  |  |
| --- | --- | --- |
| **Действие** | **Ожидаемый результат** | **Результат** |
| Переход на страницу с комментариями | Переход на страницу с комментариями | Переход на страницу с комментариями |
| Оставление комментария | Оставление комментария | Оставление комментария |
| Нажатие на кнопку лайка | Смена цвета сердечка и увеличение количества лайков | Смена цвета сердечка и увеличение количества лайков |
| Нажатие на кнопку лайка, когда он уже стоит | Смена цвета сердечка и уменьшение количества лайков | Смена цвета сердечка и уменьшение количества лайков |
| Нажатие на кнопку голосования | Отображение текущего статуса голосования | Отображение текущего статуса голосования |
| Нажатие на кнопку голосования, когда уже был сделан выбор | Получение сообщении об ошибки | Получение сообщении о том, что пользователь уже проголосовал |
| Переход на страницу личного профиля | Отображение своих личных данных и постов, которые были созданы данным пользователем | Отображение своих личных данных и постов, которые были созданы данным пользователем |
| Нажатие на кнопку «Выйсцi» | Завершение текущей сессии и переход на страницу входа | Завершение текущей сессии и переход на страницу входа |

В ходе тестирование не было выявлено багов, значит, можно сделать вывод, что приложение стабильно и полностью дееспособное.

# ЗАКЛЮЧЕНИЕ

Результатом выполнения курсового проекта является разработка мобильного приложения. Выбранные решения технически обоснованы для данного типа решаемых задач.

Приложение полностью готово к использованию. Оно рассчитано молодую аудиторию, которые любят вести соцсети.

Когда аудитория приложения будет стремительно расти, то лучше использовать другую базу данных, например PostgreSQL, а также собственные сервера. Тогда можно достичь более лучшего быстродействия и гибкости в решениях проектирования базы. Система Firebase, на которой сейчас работает приложение, больше подходит для проектов на начальном этапе или небольших проектах, где не будет “разбухания” базы данных.

Так как это соцсеть, то она может модифицироваться до бесконечности. Как пример можно посмотреть на приложение Instagram. Как оно выглядит сейчас и на начальном этапе.

Целью данной курсовой работы, являлось углубление знаний и расширение навыков по разработке алгоритмов и их реализации на OS Android.

# СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ

1. Официальная документация Android [Электронный ресурс] – режим доступа: https://developer.android.com/. Дата доступа 10.11.18.
2. Официальная документация Firebase [Электронный ресурс] – режим доступа: https://firebase.google.com/docs/. Дата доступа 14.11.18.
3. Интернет портал “Java-Help” [Электронный ресурс] – режим доступа: http://java-help.ru/model-view-presenter-android-part-1/. Дата доступа 14.11.18.
4. Интернет портал «Википедия» [Электронный ресурс] – режим доступа: https://ru.wikipedia.org/. Дата доступа 24.11.18.
5. Интернет портал «Освой программирование играючи» [Электронный ресурс] – режим доступа: <http://developer.alexanderklimov.ru/android/index.php>. Дата доступа 24.11.18.
6. Интернет портал «Habr» [Электронный ресурс] – режим доступа: https://habr.com/. Дата доступа 24.11.18.

# ПРИЛОЖЕНИЕ А

(обязательное)

Техническое задание

Введение

Наименование программного продукта – «SelMe» (*акр.*, select me). Данное приложение представляет из себя приложение для обмена фотографиями с элементами социальной сети. Данное приложение предназначено для совместного голосования за ту или иную фотографию, которую выложил пользователь. Люди часто не могут сделать правильный выбор или не знают, что выбрать из двух зол. Кто-то не может выбрать какое платье надеть на свидание, кто-то не знает какую марку автомобиля купить, или самое страшное это не знать, какие же пельмешки лучше купить в магазине. Для таких случаев и существует данное приложение.

А.1 Основание для разработки

Приложение разрабатывалось в рамках курсового проекта студента учреждения образования «Полоцкий государственный университет» Щербицкого К.О.

А.2 Назначение разработки

SelMe – приложение для обмена фотографиями с элементами социальной сети, позволяющее выкладывать в общий доступ и участвовать в голосовании за лучший вариант.

А.3 Требования к программе или программному изделию

А.3.1 Требования к функциональным характеристикам

При разработке данного приложения были выдвинуты следующие требования к функциональным характеристикам:

* Пользователь может добавлять фотографии из галереи;
* Пользователь может выставить в один пост только две фотографии;
* Пользователь может делать описание к своему посту;
* Каждый пользователь может голосовать за ту или иную фотографию;
* Каждый пользователь может оставить комментарий под любым постом;
* У каждого пользователя есть свой профиль с личными данными.

А.3.2 Требования к надежности

Мобильное приложение «SelMe» должно надежно функционировать. В случае программного или аппаратного сбоя, приложение должно завершить свою работу.

А.3.3 Условия эксплуатации

Для полноценной эксплуатации данного приложения, нужен смартфон под управлением операционной системы Android версией не ниже 5.0.

А.3.4 Требования к составу и параметрам технических средств

Для обеспечения устойчивости работы данного приложения требуется:

* Операционная система: Android;
* Версия Android не ниже 5.0;
* Базовые навыки пользования телефоном.

А.3.5 Требования к информационной и программной совместимости

Для успешной работы приложения, необходим смартфон под управлением операционной системы Android.

А.3.6 Требования к маркировке и упаковке

Требования к маркировке и упаковке отсутствуют.

А.3.7 Требования к транспортированию и хранению

Программное средство должно храниться на компакт-диске в виде исполняемых файлов программного продукта.

А.4 Требования к программной документации

В качестве программной документации по приложению должно быть техническое задание согласно ГОСТ 19.201-78.

Требования к перечисленным программным документам устанавливаются государственными стандартами ЕСПД.

А.5 Стадии и этапы разработки

Разработка приложения заключается в следующем:

* Анализ исходных данных и постановка задачи проектирования, разработка технического задания.
* Разработка пользовательского интерфейса.
* Разработка главного функционала загрузка фотографий, пользовательское голосование.
* Разработка второстепенного функционала.
* Тестирование приложений.
* Разработка программной документации.

А.6 Порядок контроля и приемки

Контроль и приемка программного средства осуществляется в соответствии с программой и методикой испытаний.

Тестирование программного обеспечения состояло из выполнения сценариев ранее перечисленных методов. Основным критерием оценки правильности проверки тестов, являлся визуальный контроль выполнения программой требующихся функций.

# ПРИЛОЖЕНИЕ Б

## Диаграмма вариантов использования

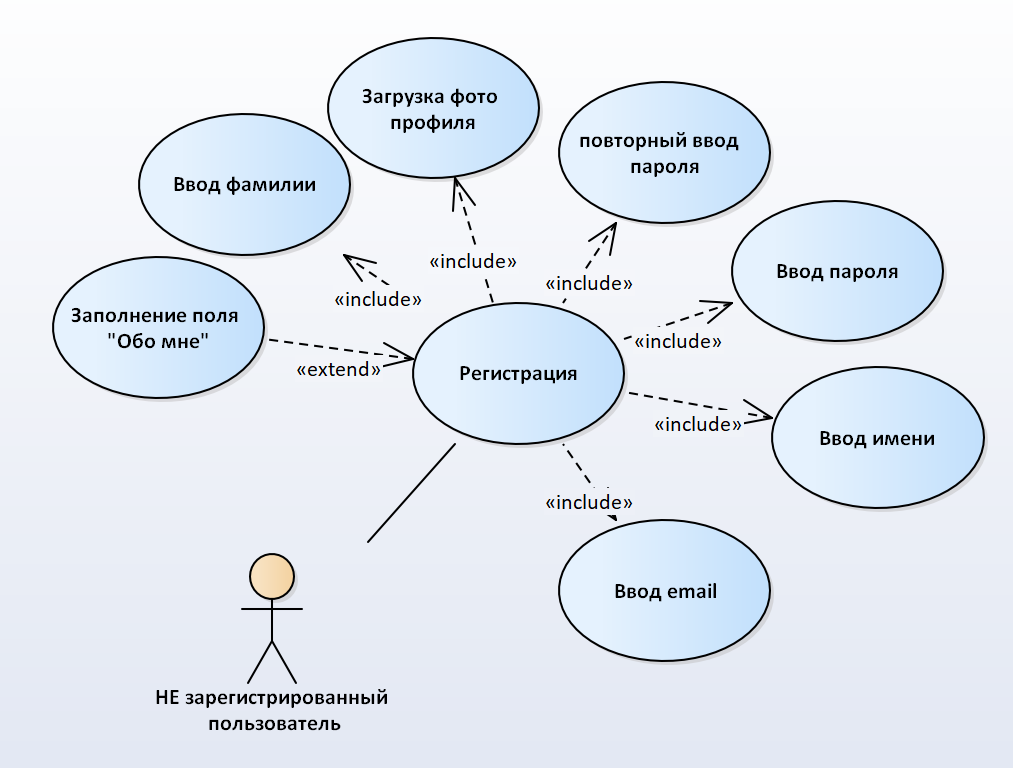


Рисунок Б.1 – Диаграмма вариантов использования для не зарегистрированного пользователя

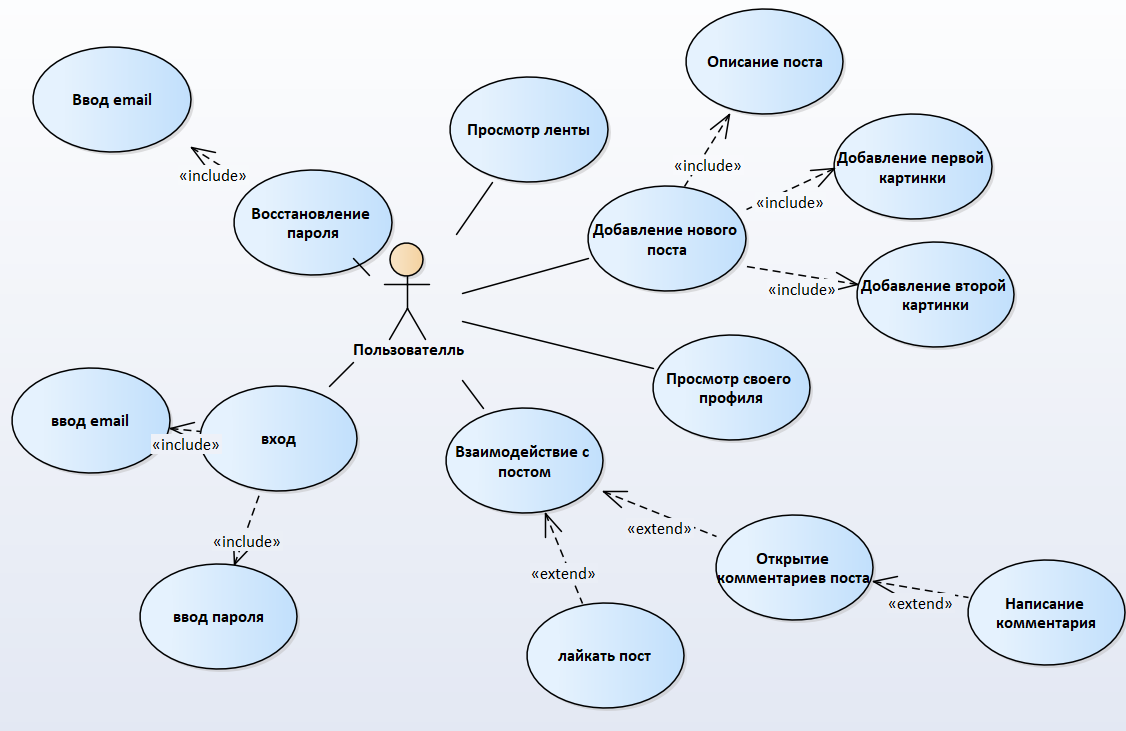


Рисунок Б.2 – Диаграмма вариантов использования для зарегистрированного пользователя