



Universitatea Tehnică a Moldovei

**Информационная система для организации волонтерской
деятельности**

**Sistem informațional pentru organizarea activităților de voluntariat
Information system for organizing volunteer activities**

Student: **gr. TI-196,**

Nigai Lucia

Coordonator: **Scorohodova Tatiana**
asistentă universitară

Chișinău, 2023

MINISTERUL EDUCAȚIEI ȘI CERCETĂRII AL REPUBLICII MOLDOVA

Universitatea Tehnică a Moldovei

Facultatea Calculatoare Informatică și Microelectronică

Departament Ingineria Software și Automatică

Admis la susținere

Șef de departament:

Fiodorov I. dr., conf. univ.

“ ____ ” _____ 2023

Sistem informațional pentru organizarea activităților de voluntariat

Proiect de licență

Student: _____

Nigai Lucia, TI-196

Coordonator: _____

Scorohodova Tatiana, asist. univ.

Consultant: _____

Cernei Irina, asist. univ.

Chișinău, 2023

Universitatea Tehnică a Moldovei

Facultatea Calculatoare, Informatică și Microelectronică

Departamentul Ingineria Software și Automatică

Programul de studii Tehnologia Informației

Aprob

Șef de departament

Fiodorov Ion, dr., conf. univ.

“ 30 ” _____ 2022

CAIET DE SARCINI

pentru proiect de licență al studentului

Nigai Lucia

(numele și prenumele studentului)

1. Tema proiectului de licență Sistem informațional pentru organizarea activităților de voluntariat

confirmată prin hotărârea Consiliului facultății nr. 1 nr. “23” decembrie 2023

2. Termenul limită de prezentare a proiectului de licență 20.05.2023

3. Date inițiale pentru elaborarea proiectului de licență *Sarcina pentru elaborare proiectului de diploma*

4. Conținutul memoriului explicative

Введение

1 Анализ предметной области

2 Моделирование и проектирование информационной системы

3 Реализация системы

4 Оценка стоимости проекта

5 Документирование

Заключение

5. Conținutul părții grafice a proiectului de licență

Image holder для изображения пользователя и изображения поста. Элемент wave для страниц регистрации аутентификации и профиля пользователя.

6. Lista consultanților:

Consultant	Capitol	Confirmarea realizării activității	
		Semnătura consultantului (data)	Semnătura studentului (data)
I. Cernei	Standarde tehnologice, Controlul calității, Estimarea costului proiectului		

7. Data înmânării caietului de sarcini 02.09.2022

Coordonator *Scorohodova Tatiana* _____
semnătura

Sarcina a fost luată pentru a fi executată de către studentul *Nigai Lucia*
_____ 02.09.2022
semnătura, data

PLAN CALENDARISTIC

Nr. crt.	Denumirea etapelor de proiectare	Termenul de realizare a etapelor	Nota
1	<i>Анализ предметной области. Постановка задачи.</i>	<i>07.09.22 – 26.09.22</i>	<i>8%</i>
2	<i>Моделирование и проектирование информационной системы</i>	<i>27.09.22 – 21.11.22</i>	<i>20%</i>
3	<i>Реализация информационной системы</i>	<i>22.11.22 – 30.04.23</i>	<i>45%</i>
4	<i>Тестирование информационной системы</i>	<i>25.03.23 – 10.04.23</i>	<i>10%</i>
5	<i>Стоимостная оценка системы</i>	<i>10.04.23 – 20.04.23</i>	<i>10%</i>
6	<i>Документирование</i>	<i>21.04.23 – 15.05.23</i>	<i>7%</i>

Studentă _____ *Nigai Lucia* (_____)

Coordonator de proiect de licență _____ *Scorohodova Tatiana* (_____)

DECLARAȚIA STUDENTULUI

Subsemnata Nigai Lucia, declar pe proprie răspundere că lucrarea de față este rezultatul muncii mele, pe baza propriilor cercetări și pe baza informațiilor obținute din surse care au fost citate și indicate, conform normelor etice, în note și în bibliografie. Declar că lucrarea nu a mai fost prezentată sub această formă la nici o instituție de învățământ superior în vederea obținerii unui grad sau titlu științific ori didactic.

Semnătura autorului

UNIVERSITATEA TEHNICĂ A MOLDOVEI

FACULTATEA CALCULATOARE, INFORMATICĂ ȘI MICROELECTRONICĂ DEPARTAMENTUL INGINERIA SOFTWARE ȘI AUTOMATICĂ PROGRAMUL DE STUDII TEHNOLOGII INFORMAȚIEI

AVIZ

la proiectul de licență

Titlul: Информационная система для организации волонтерской деятельности

Студентка Нигай Лучия гр. TI-196

1. Actualitatea temei: Для функционирования современного общества важны многие составляющие, такие как законодательная база, сильная экономика, и многое другое. Но у каждого человека есть возможность оказать влияние на общество в целом и на жизнь отдельных людей, данную возможность может предоставить волонтерская деятельность. Разработанное в ходе выполнения данного дипломного проекта мобильное приложение, внесет свой вклад в развитие волонтерской деятельности.

2. Caracteristica proiectului de licență: Система была разработана для организации волонтерской деятельности, и является доступной для всех слоев населения и различных типов пользователей через мобильное устройство.

3. Analiza prototipului: Приложение станет своего рода посредником между людьми, нуждающимися в помощи, и теми, кто готов ее оказать, причем один и тот же пользователь сможет публиковать, как и запросы на оказание помощи так и на ее получение.

4. Estimarea rezultatelor obținute: Интерфейс приложение был разработан максимально простым, предназначенным для пользователей с любым, даже самым минимальным опытом использования мобильных устройств.

5. Corectitudinea materialului expus: Материал, использованный в данном дипломном проекте представлен со ссылками на источники, написанные людьми с опытом работы в сфере информационных технологий.

6. Calitatea materialului grafic: Для наглядности, проект был представлен с использованием диаграмм, таблиц а также изображений интерфейса.

4. Valoarea practică a proiectului: Система предназначена для устройств под руководством операционной системы Android.

5. Observații și recomandări: Требования к дипломному проекту были полностью соблюдены, Замечаний нет.

6. Caracteristica studentului și titlul conferit: Студент, Нигай Лучия, проявила индивидуальность и профессионализм при создании дипломного проекта, доказала целесообразность разработки продукта, смог соблюсти все поставленный департаментом цели и требования.

Din cele relatate, urmează că lucrarea de licență poate fi admisă spre susținere, cu nota _____

Lucrarea în forma electronică corespunde originalului prezentat către susținere publică.

Coordonatorul proiectului de licență _____ Scorohodova Tatiana, asist. univ.

semnătura, data

АННОТАЦИЯ

Целью данного дипломного проекта было создание информационной системы для организации волонтерской деятельности, которое предназначено для всех слоев населения. Разработанное приложение обладает минималистичным дизайном и простым понятным интерфейсом, и реализовано с учетом того, что пользователями приложения будут люди с разным опытом использования мобильных устройств. Проект состоит из 5 глав, в которых пошагово описан процесс его создания.

Целью первой главы по анализу предметной области стало определение важности темы разрабатываемого проекта, а также изучение, анализ и сравнение существующих, на данный момент на рынке аналогов. Первые два шага затем стали основой для определения четких целей, задач и требований к разрабатываемой системе. Вторая глава, моделирование и проектирование системы позволяет получить лучшее представление будущего проекта, а также помогает лучше понять и проанализировать функциональность разрабатываемой системы, входящих в нее компонентов и взаимодействия между ними. Для выполнения данной задачи был выбран язык моделирования UML. Глава “Реализация системы”, третья по счету, содержит в себе практическую часть работы над проектом, а именно, описание структуры приложения, его дизайн, используемые инструменты, базу данных, а также примеры кода. Неотъемлемой частью разработки любой системы является ее тестирование на обнаружение потенциальных недочетов в реализации, и наша система не стала исключением, в третьей главе также было проведено тестирование системы и последующее исправление обнаруженных недочетов. Четвертая глава в свою очередь стала завершающей в плане разработки проекта, в ней представлена документация к разработанной системе для будущих пользователей приложения.

В пятой главе описывается взгляд на данную систему с финансовой точки зрения, а именно была проведена оценка стоимости разрабатываемого проекта, также для наглядности была использована диаграмма. Завершением проекта стало подведение итогов всем этапам работы над проектом в виде краткого заключения.

REZUMAT

Scopul acestui proiect de absolvire a fost crearea unui sistem informațional pentru organizarea activităților de voluntariat, care este destinat tuturor segmentelor de populație. Aplicația dezvoltată are un design minimalist și o interfață simplă intuitivă și este implementată ținând cont de faptul că utilizatorii aplicației vor fi persoane cu experiență diferită în utilizarea dispozitivelor mobile. Proiectul este format din 5 capitole, care descriu pas cu pas procesul de creare a acestuia.

Scopul primului capitol privind analiza domeniului subiectului a fost acela de a determina importanța temei proiectului în curs de dezvoltare, precum și de a studia, analiza și compara analogii existenți pe piață în acest moment. Primii doi pași au devenit apoi baza pentru definirea unor scopuri, obiective și cerințe clare pentru sistemul în curs de dezvoltare. Al doilea capitol, Modelare și proiectare a sistemului, vă permite să vă faceți o idee mai bună despre viitorul proiect și, de asemenea, ajută la înțelegerea și analizarea mai bună a funcționalității sistemului în curs de dezvoltare, a componentelor sale și a interacțiunilor dintre ele. Pentru a îndeplini această sarcină, a fost ales limbajul de modelare UML. Capitolul „Implementarea sistemului”, al treilea la rând, conține partea practică a lucrării la proiect, și anume, o descriere a structurii aplicației, designul acesteia, instrumentele utilizate, baza de date și exemple de cod. O parte integrantă a dezvoltării oricărui sistem este testarea acestuia pentru a detecta potențialele deficiențe în implementare, iar sistemul nostru nu a făcut excepție, în capitolul al treilea, sistemul a fost și testat și corectarea ulterioară a deficiențelor detectate. Al patrulea capitol, la rândul său, a devenit cel final în ceea ce privește dezvoltarea proiectelor, acesta oferă documentație pentru sistemul dezvoltat pentru viitorii utilizatori ai aplicației.

Capitolul cinci descrie o vedere a acestui sistem din punct de vedere financiar și anume, a fost efectuată o evaluare a costului proiectului în curs de dezvoltare și a fost folosită și o diagramă pentru claritate. Finalizarea proiectului a fost însumarea rezultatelor tuturor etapelor de lucru ale proiectului sub forma unei scurte concluzii.

ABSTRACT

The purpose of this graduation project was to create an information system for organizing volunteer activities, which is intended for all segments of the population. The developed application has a minimalistic design and a simple intuitive interface, and is implemented taking into account the fact that the users of the application will be people with different experience of using mobile devices.

The project consists of five chapters, which describe step by step the process of its creation. The purpose of the first chapter on the analysis of the subject area was to determine the importance of the topic of the project being developed, as well as to study, analyze and compare existing analogs on the market at the moment. The first two steps then became the basis for defining clear goals, objectives and requirements for the system under developed. The second chapter, System Modeling and Design, allows you to get a better idea of the future project, and also helps to better understand and analyze the functionality of the developed system, its components and the interactions between them. To accomplish this task, the UML modeling language was chosen. Chapter "Implementation of the system", the third in a row, contains the practical part of the work on the project, namely, a description of the structure of the application, its design, the tools used, the database, and code examples. An integral part of the development of any system is its testing to detect potential shortcomings in the implementation, and our system was no exception, in the third chapter, the system was also tested and the subsequent correction of the detected shortcomings. The fourth chapter, in turn, became the final one in terms of project development, it provides documentation for the developed system for future users of the application.

The fifth chapter describes a view of this system from a financial point of view, namely, an assessment of the cost of the project being developed was carried out, and a diagram was also used for clarity. The completion of the project was summing up the results of all stages of work on the project in the form of a brief conclusion.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ	12
1 АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ.....	13
1.1 Важность темы.....	13
1.2 Существующие аналоги.....	14
1.3 Цель, задачи и требования к системе	19
2 МОДЕЛИРОВАНИЕ И ПРОЕКТИРОВАНИЕ СИСТЕМЫ	21
2.1 Поведенческое описание системы	22
2.1.1 Взаимодействие пользователя с системой.....	22
2.1.2 Моделирование потоков управления	24
2.1.3 Взаимодействие объектов системы	26
2.1.4 Описание системы на уровне отдельных объектов	28
2.2 Структурное описание системы.....	29
2.2.1 Классовая структура системы	29
2.2.2 Аппаратная составляющая системы	32
3 РЕАЛИЗАЦИЯ СИСТЕМЫ.....	34
3.1 Дизайн приложения.....	34
3.2 База данных.....	42
3.3 Серверная сторона приложения.....	45
3.4 Основная Activity и навигация в приложении.....	50
3.5 Классы адаптеры для постов, комментариев и отзывов	52
3.6 Отображение постов	54
3.7 Профиль пользователя	55
3.8 Публикация и изменение постов	58
3.9 Публикация и изменение комментариев.....	59
3.10 Публикация и изменение отзывов о пользователе.....	60
3.11 Тестирование информационной системы	61
4 СТОИМОСТНАЯ ОЦЕНКА СИСТЕМЫ.....	64

5 ДОКУМЕНТИРОВАНИЕ.....	67
ЗАКЛЮЧЕНИЕ.....	81
СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ	82

ВВЕДЕНИЕ

Для функционирования современного общества важны многие составляющие, такие как законодательная база, сильная экономика, и многое другое. Большинство из этих критериев несут глобальный характер и роль одного человека не так уж и значительна. Но у каждого человека есть возможность оказать влияние на общество в целом и на жизнь отдельных людей, данную возможность может предоставить волонтерская деятельность.

Развитие волонтерской деятельности является важным как для общества в целом, так и отдельных его секторов, а также самих волонтеров. Для отдельного человека участие в волонтерской деятельности способствует самореализации и самосовершенствованию, дает возможность получить новые знания и опыт, что, безусловно, является важным особенно для молодых людей, а также возможность почувствовать себя социально значимым и социально полезным. Государству волонтерский труд помогает эффективнее решать задачи, стоящие перед ним и обществом. Развитие волонтерства способствует становлению гражданского общества, служит повышению роли некоммерческих и общественных организаций. Волонтерство положительно влияет на социальное и экономическое развитие страны в целом, помогая решить социально значимые проблемы. [1]

В большинстве развитых стран до половины населения страны, а то и больше учувствовало в волонтерской деятельности, но в нашей стране она лишь набирает обороты. В данной работе будет создано приложение для организации волонтерской деятельности, которое должно внести свой вклад в развитие волонтерства на территории республики Молдова. Оно будет доступно любым пользователям, став посредником между волонтерами и гражданами, нуждающимися в помощи.

1 АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ

В данной главе проводится анализ важности и актуальности, темы дипломного проекта. Рассматриваются и анализируются существующие на рынке информационные системы по различным критериям для определения их особенностей, а также сильных и слабых сторон. На основе проведенного анализа составлена таблица, для наглядного сопоставления по ключевым характеристикам, функционирующих на данный момент времени решений с разрабатываемой информационной системой. Данный анализ позволяет учесть положительные и отрицательные особенности при проектировании нового решения.

1.1 Важность темы

Волонтерство является ключевым фактором социальных преобразований как в развитых, так и в развивающихся странах. Оно поддерживает активную гражданскую позицию, устойчивость сообщества и социальную активность, а также продвигает совместную ответственность. Волонтерство дает возможность каждому гражданину в независимости от его положения вносить непосредственный вклад в развитие страны и приносит пользу не только людям, которые в нем нуждаются, но и тем, кто оказывает эту помощь.

Несмотря на все преимущества волонтерства оно получило широкое распространение далеко не во всех странах мира. Глобальная волонтерская организация FDIP провела исследование для определения количества людей, которые в какой-либо период своей жизни принимали участие в волонтерской деятельности. Согласно результатам данного исследования на 1 января 2023 года сред, принимавших в нем участие стран лидирующую позицию, занимает США, результаты исследования представлены на рисунке 1.1. [2]

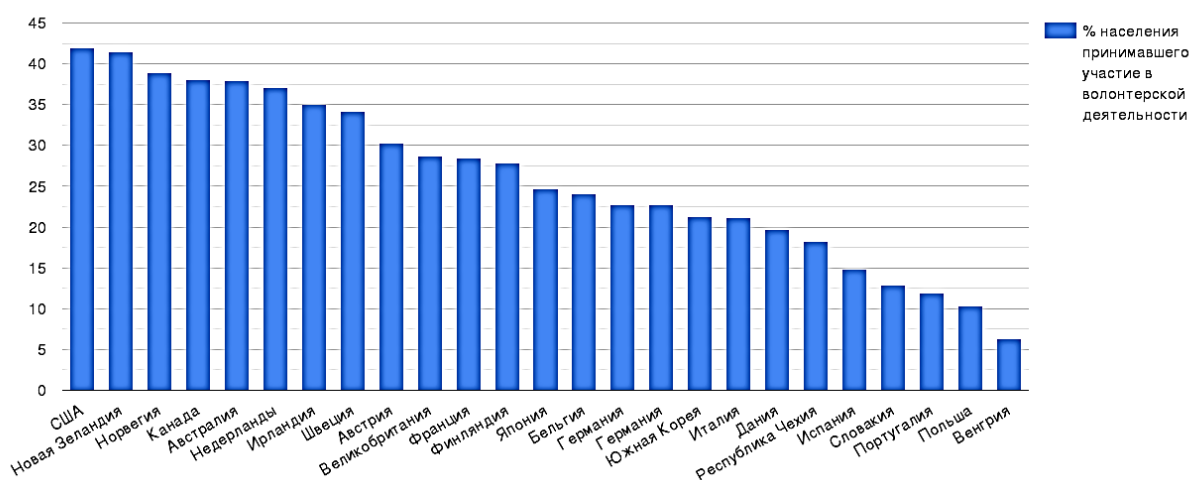


Рисунок 1.1 - Процент людей, задействованных в волонтерской деятельности[2]

В данном опросе принимало участие ограниченное количество стран, но можно с уверенностью утверждать, что волонтерство занимает значимую роль в жизни общества. К сожалению, во многих странах, включая Молдову, оно не получило такой широкой популяризации во многом из-за незначительной численности волонтерских организаций, недостаточной информированности граждан, а также отсутствия доступных централизованных источников информации по данной теме.

Одним из способов решения данных проблем будет создание приложения, которое станет посредником между людьми, желающими заниматься волонтерской деятельностью, а также теми, кто в ней нуждается. Мобильное приложение будет предоставлять удобный и своевременный доступ к информации за счет широкого распространения мобильных устройств. Пользователю не нужно будет тратить огромное количество времени на поиски организаций, занимающихся волонтерской деятельностью в той сфере, в которой они заинтересованы.

Приложение будет предоставлять возможность пользователям следить за запросами в интересующей их сфере волонтерства, и оказывать помощь без посредников. Преимуществом данного приложения является то, что оно не имеет узкой специализации и направлено на централизацию запросов на волонтерскую деятельность различных направлений.

До сих пор множество организаций сосредоточены преимущественно в Европе и Америке и не предоставляют информацию или возможность участия в волонтерской деятельности гражданам других стран, а информационные решения в данной сфере в большинстве своем являются узкоспециализированными. Приложение, созданное в процессе реализации данного дипломного проекта, внесет свой вклад в более широкое распространение волонтерской деятельности в Молдове и поможет объединить людей различных слоев населения, способствуя развитию социальной активности.

1.2 Существующие аналоги

На настоящий момент в сфере волонтерства существуют различные информационные системы, но они отличаются по специализации и функционалу, а также не ориентированы на Молдавский рынок. Все информационные системы могут быть разделены на две группы:

- информационные системы для менеджмента, они направлены на организацию волонтерской работы и используется только самими волонтерами;
- информационные системы для волонтерства направленные на участие волонтеров в различных мероприятиях, а также на сбор средств, они могут обхватывать широкий круг сфер или же быть узкоспециализированными.

Приложения из первой группы больше напоминают “to do” приложения с напоминаниями о запланированном участии в каком-либо мероприятии, или позволяющие организаторам распределять задачи между волонтерами и отслеживать их выполнение. Данные приложения направлены на отслеживание работы волонтера и на взаимодействие между ними. Вторая группа включает в себя более широкий спектр приложений и сайтов, которые позволяют заинтересованным людям и волонтерам оказывать помощь различным группам людей, как в повседневной жизни, так и за счет пожертвований. Так как разрабатываемая информационная система относится ко второй группе приложений для сравнительного анализа также были выбраны лишь приложения из второй группы.

Наиболее близким аналогом разрабатываемого приложения является приложение Добро.рф. Оно является платформой для организации волонтерской деятельности, основными пользователями которой являются волонтерские организации и сами волонтеры. Данное приложение обладает простым интерфейсом, представленным на рисунке 1.2 Оно позволяет отслеживать организуемые волонтерские активности в вашем регионе, принимать в них участие, а также отслеживать рейтинг волонтера, количество мероприятий, в которых он участвовал и время участия. Данное приложение является доступным для скачивания и использования только на территории РФ. [3]

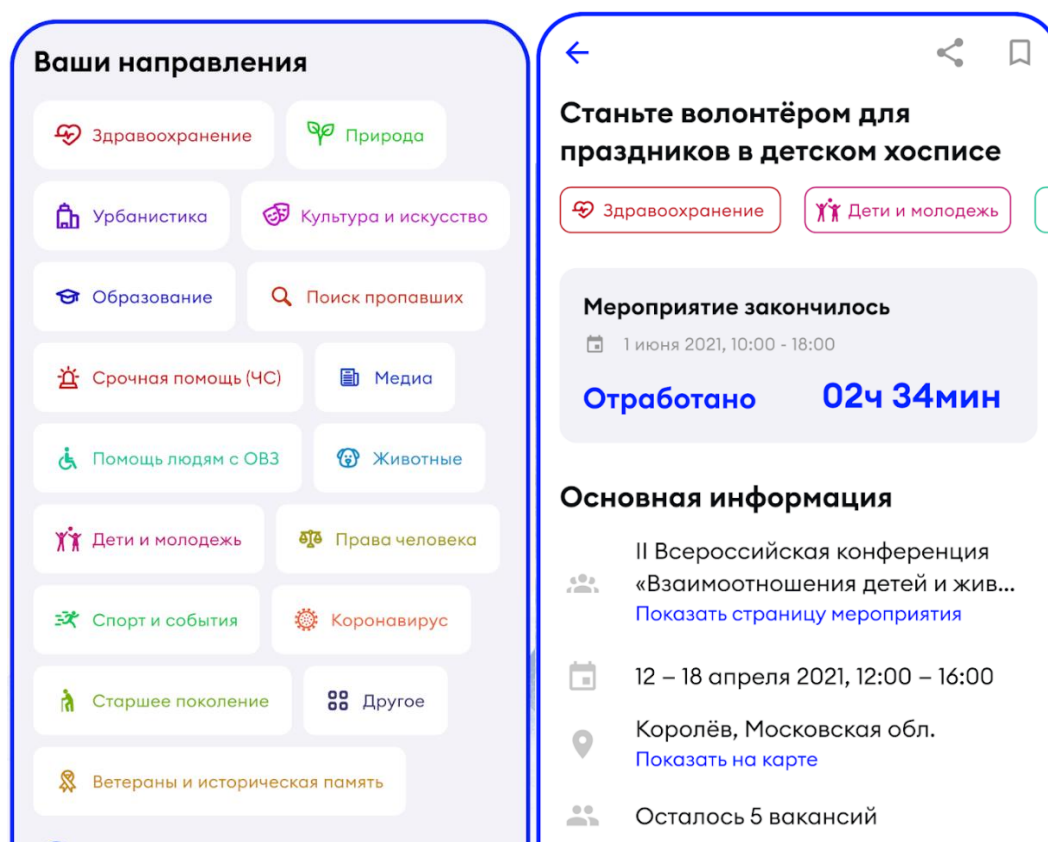


Рисунок 1.2 – Интерфейс Добро.рф [3]

VolunteerMatch – один из самых известных некоммерческих сайтов, а также связанное с ним приложение сопоставляет и связывает пользователей с некоммерческими организациями на основе предоставленных интересов, навыков, времени, которое вы готовы посвящать волонтерству а также местоположения. Пользователь легко может найти практически любой тип волонтерской деятельности в его регионе и принять в нем участие, например стать донором, оказать физическую или материальную помощь, в этом помогает и простой интерфейс сайта, рисунок 1.3. Данные сайт и приложение не ориентированы на Молдавский рынок и не предоставляет информацию о волонтерских активностях в наших регионах. [4]



Рисунок 1.3 – Интерфейс VolunteerMatch [4]

Be my Eyes – данное приложение является узкоспециализированным и направленно на помощь людям с проблемами со зрением. Люди, нуждающиеся в помощи, могут обратиться за помощью к волонтерам и связаться с ними по видеосвязи для оказания помощи от прочтения документов, до проверки срока годности на продуктах. Для того чтобы стать волонтером необходимо указать язык, предоставить доступ к микрофону и уведомлениям, когда кому-то

понадобиться помощь появится уведомление и доступный волонтер окажет помощь. Интерфейс данного приложения представлен на рисунке 1.4. В отличие от предыдущего оно может быть использовано в независимости от местоположения пользователя и привязано только к языку, на котором он общается. [5]

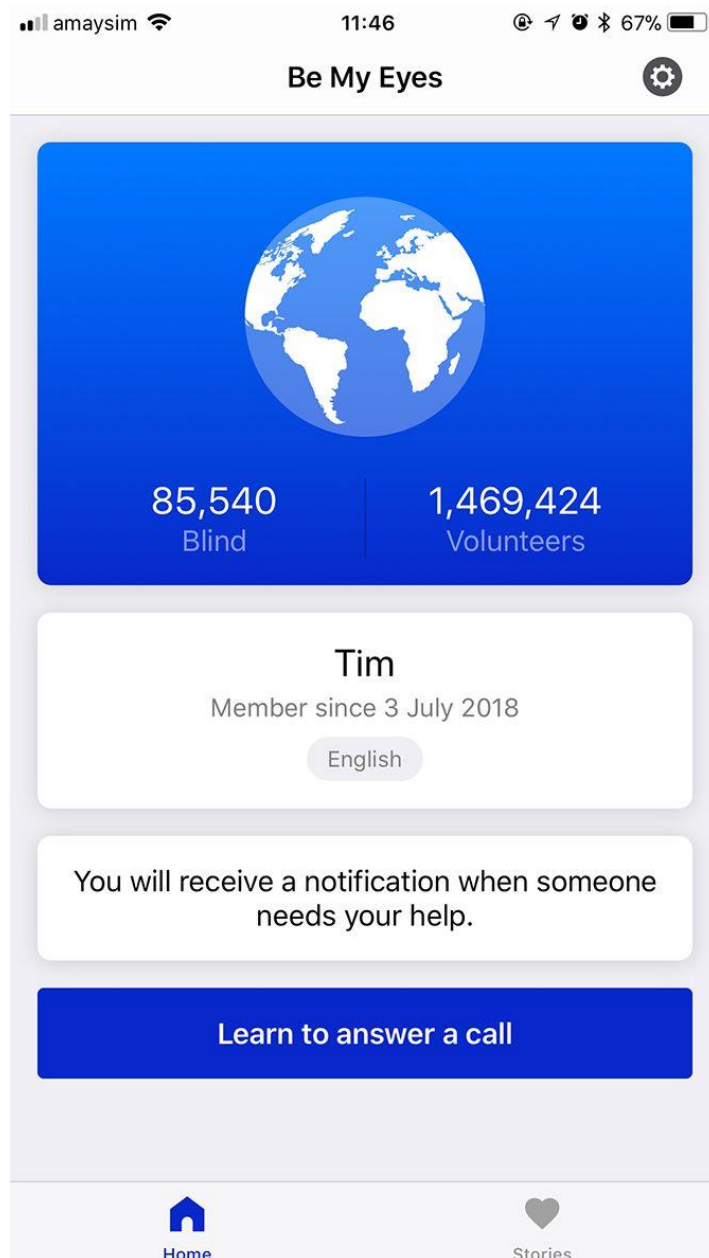


Рисунок 1.4 – Интерфейс Be my Eyes [5]

OLIO – приложения для обмена продуктами питания и непродовольственными товарами. Можно зарегистрироваться и размещать свои товары, такие как еду, одежду мебель и т.д. и те, кто в них нуждаются запрашивают эти предметы и приходят за ними в назначенное время и место. Данное приложение обладает очень простым интерфейсом и использует местоположение для

нахождения людей, находящихся ближе всего к вам интерфейс приложения представлен на рисунке 1.5. Оно является абсолютно бесплатным и работает как на Android, так и на iOS [4]

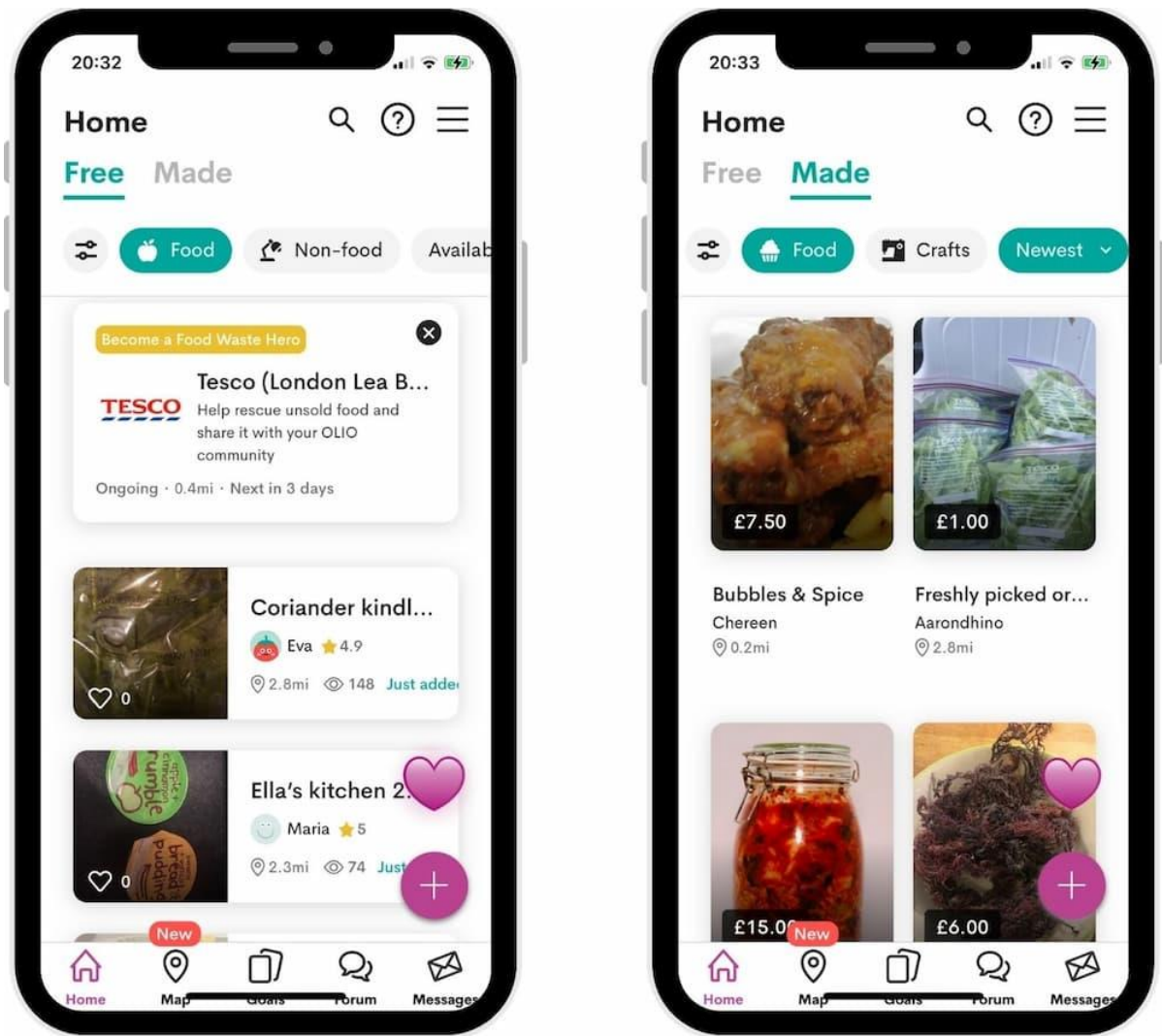


Рисунок 1.5 – Интерфейс OLIO [4]

Ниже в таблице 1.1. представлен сравнительный анализ разрабатываемого приложения с существующими аналогами по основным критериям, что позволяет наглядно рассмотреть достоинства и недостатки систем.

Таблица 1.1 – Сравнительный анализ аналогов

Критерии	Добро.рф	VolunteerMatch	Be my Eyes	Разрабатываемое приложение
Мобильное приложение	да	да	да	да
Используется волонтерами	да	да	да	да

Используется организациями	да	да	да	да
Используется обычными пользователями	нет	да	да	да
Обязательно подключение к интернету	да	да	да	да
Предоставляет информацию о регионе Молдова	нет	нет	да	да
Узкоспециализированное	нет	нет	да	нет

Как можно заметить из таблицы выше, большинство существующих аналогов имеют один большой недостаток, они не предназначены для использования на территории Молдовы. Из рассмотренных выше приложений лишь *Be my Eyes* может быть использовано так как не привязано к определенной стране, а лишь к языку. Но более масштабные приложения, которые охватывают различные сферы волонтерства привязаны к одной или нескольким странам и, к сожалению, не предоставляют возможности использовать их на территории Молдовы.

Основываясь на проведенном анализе, было решено создать мобильное приложение, которое будет охватывать различные направления волонтерства и будет доступно различным типам пользователей, таким как индивидуальные волонтеры и обычные пользователи. Приложение, которое будет предоставлять возможность публиковать запросы, как от желающих помочь, так и от тех, кто в этой помощи нуждается. Благодаря наличию различных категорий, приложение будет понятно пользователям с любым уровнем опытности и станет посредником между волонтерами и людьми.

1.3 Цель, задачи и требования к системе

Целью данного дипломного проекта является создание информационной системы для организации волонтерской деятельности, а именно приложения для мобильных устройств под управлением операционной системы Android, так как она является наиболее распространенной.

Наш проект подразумевает взаимодействие между пользователями и предоставление быстрого и своевременного доступа к информации именно поэтому для его реализации была

выбрана клиент-серверная архитектура, которая позволяет синхронизировать данные между клиентскими устройствами. Выбор данной архитектуры дает следующие преимущества [8]:

- позволяет легко модифицировать и конфигурировать приложения;
- упрощает взаимодействие пользователей с системой;
- имеет хорошую масштабируемость и производительность;
- более безопасна, так как все данные пользователей и программы находятся на удаленном сервере.

В клиент-серверной архитектуре используется три компонента:

- клиент — программа, которую мы используем в интернете. Чаще всего это браузер, но может быть и другая отдельная программа;
- сервер — компьютер, на котором хранится сайт или приложение. Когда мы заходим на сайт магазина, мы обращаемся к серверу, на котором находится сайт;
- база данных — программа, в которой хранятся все данные приложения. Для магазина это будет база клиентов, товаров и заказов.

Требования к дизайну

Дизайн также является важной составляющей приложения, оно является его визитной карточкой и от того насколько он продуман зависит легкость в использовании приложения, а также внешняя опрятность. Ключевым моментом будет выбор расположения элементов на экране, а также приятной палитры цвета, для улучшения общего впечатления от приложения.

Требования к программному обеспечению

Так как приложение создается для мобильных устройств под управлением системы Android, была выбрана следующая среда разработки:

- среда моделирования Enterprise Architect;
- интегрированная среда разработки Android Studio;
- система управления базами данных MySQL.

2 МОДЕЛИРОВАНИЕ И ПРОЕКТИРОВАНИЕ СИСТЕМЫ

Моделирование системы – это процесс разработки абстрактных моделей системы, при этом каждая модель представляет собой отдельный вид или перспективу данной системы. Моделирование системы помогает лучше понять и проанализировать функциональность системы, ее компоненты и взаимодействие между ними. В настоящее время под моделированием системы все чаще подразумевается представление системы с использованием какой-либо графической нотации. Наиболее известным инструментом для моделирования на данный момент является язык UML.

UML (United Modeling Language) – представляет собой стандартизированный язык моделирования, состоящий из интегрированного набора диаграмм и разработанный, чтобы помочь разработчикам систем и программного обеспечения для определения, визуализации, создания и документирования артефактов программных систем, а также для бизнес-моделирования и других непрограммных систем.[6]

Моделирование системы является очень важной частью разработки программного обеспечения, и важность данного этапа увеличивается пропорционально по мере увеличения сложности системы. Создавая диаграммы с использованием языка UML необходимо следовать правилам моделей и синтаксиса. Нотация UML позволяет визуализировать систему и впоследствии улучшать ее в процессе работы по мере необходимости.

Одним из наиболее известных и используемым инструментом для разработки UML диаграмм является Enterprise Architect, инструмент визуального моделирования и проектирования. Интерфейс программы представлен на рисунке 2.1.

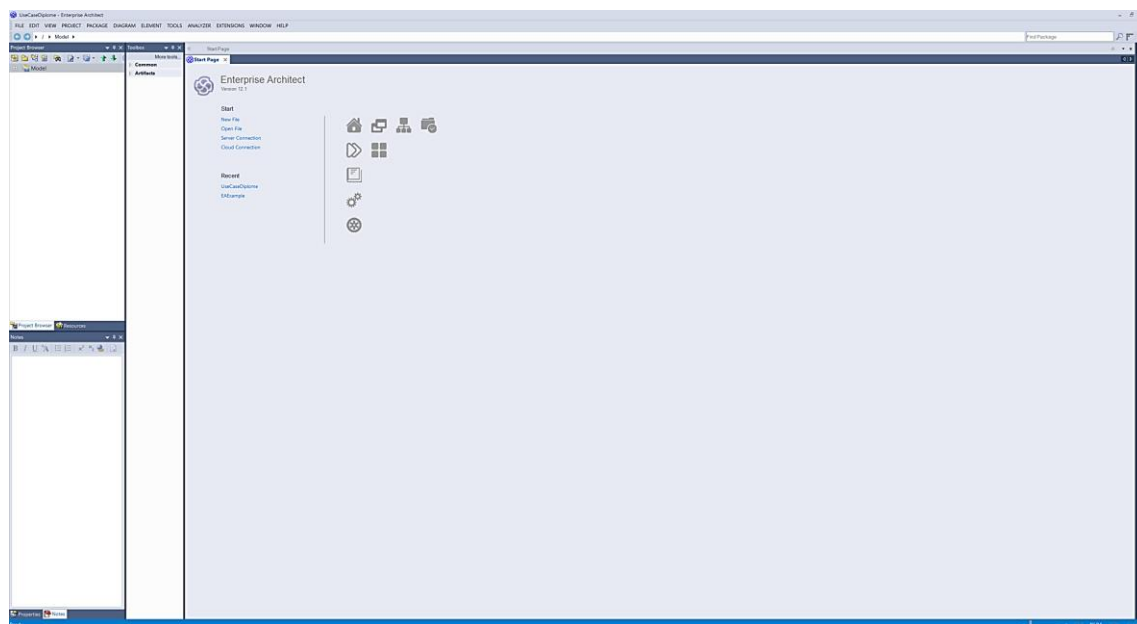


Рисунок 2.1 – Интерфейс Enterprise Architect

Стандарт UML определяет 12 видов диаграмм, которые разделены на две группы: поведенческие и структурные диаграммы. Поведенческие диаграммы позволяют визуализировать то, как система ведет себя и взаимодействует со своими компонентами и другими пользователями, другими системами и компонентами. Структурные диаграммы, как следует из названия, демонстрируют структуру системы, включая классы, объекты, пакеты, компоненты и т.д., а также взаимодействие между ними. [7]

2.1 Поведенческое описание системы

Поведенческие диаграммы визуализируют элементы системы, которые зависят от времени и передают динамические концепции системы и то, как они соотносятся друг с другом. В данную группу входят следующие виды диаграмм:

- диаграмма вариантов использования позволяют моделировать поведение системы и связь между ними;
- диаграмма последовательности представляет собой структурированное представление поведения в виде последовательности шагов во времени;
- диаграмма кооперации демонстрирует взаимодействие между элементами во время выполнения, визуализируя отношения между объектами.
- диаграмма состояний демонстрирует то, как элементы могут переходить из одного состояния в другое, в соответствии с триггерами перехода и ограничениями защиты.
- диаграмма деятельности моделирует поведение системы и способы, которым эти поведения связаны в общем потоке системы.

2.1.1 Взаимодействие пользователя с системой

Одним из способов наглядно представить взаимодействие пользователя и системы являются диаграммы вариантов использования. Каждый вариант использования представляет собой отдельную задачу, которая включает внешнее взаимодействие с системой. Основными элементами диаграммы вариантов использования являются: актер, вариант использования и связи между ними. Актер может представлять собой как человека, так и другую систему, другими словами, объект, взаимодействующий с системой. Данный тип диаграмм позволяет получить представление о том какие типы пользователей смогут пользоваться системой, а также доступный отдельному типу пользователя функционал. Несмотря на то, что это один из самых простых типов диаграмм они являются информативными, с точки зрения взаимодействия пользователя и системы, при этом они не раскрывают внутренне устройство системы.

В разрабатываемой системе присутствует только один тип пользователя так как система является общедоступной и нет необходимости в существовании различных типов пользователей. Так что всем пользователям пользующимся системой доступен одинаковый функционал.

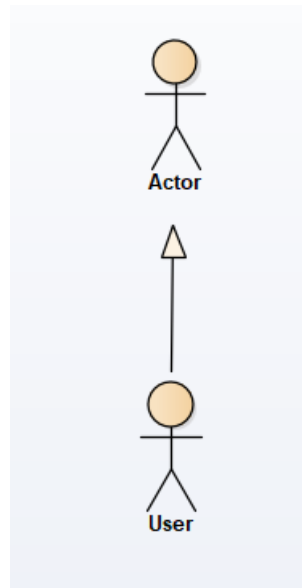


Рисунок 2.2 – Типы актеров системы

Далее можно рассмотреть действия доступные пользователю определенного типа. Для начала рассмотрим диаграмму вариантов использования для обычного пользователя, представленную на рисунке 2.3.

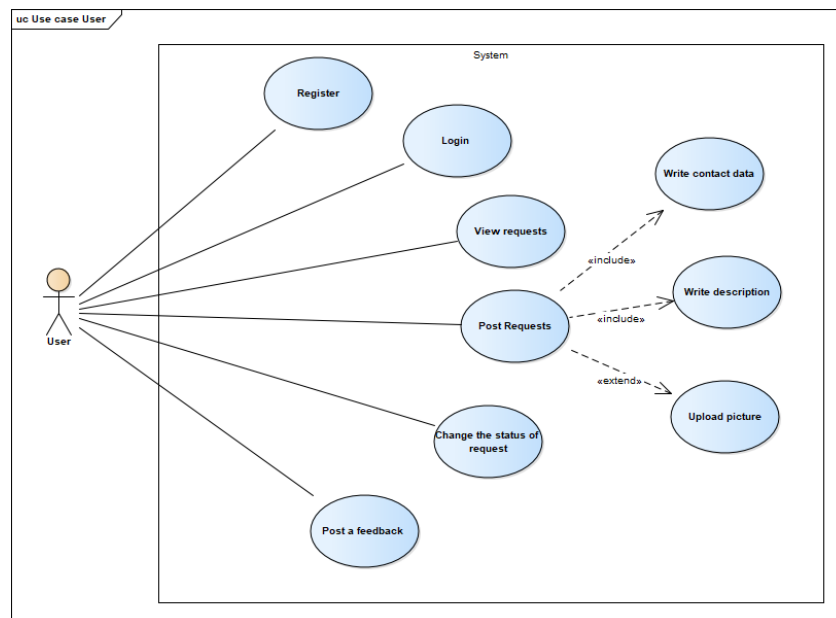


Рисунок 2.3 – Диаграмма вариантов использования пользователя

Как можно увидеть на представленной диаграмме пользователю доступен следующий основной функционал: просмотр постов, возможность выложить пост и изменять его статус, а также реагировать на посты других пользователей.

При входе в приложение пользователю открывается страница с его профилем, и для того, чтобы просмотреть актуальные запросы он должен перейти в соответствующий раздел нажав на него в нижней части приложения или перейдя используя меню. Также в процессе просмотра постов он сможет оставлять свою реакцию на них в виде лайков или комментариев.

Для того чтобы самому загрузить пост пользователь должен нажать на соответствующий раздел в нижней части экрана, после чего будет открыта страница, на которой он должен будет ввести доступные контактные данные, а также описание запроса, при желании он также может загрузить фото. У самого запроса будет несколько состояний, которые пользователь сможет изменять впоследствии для того, чтобы остальные пользователи смогли получить лучшее представление о том на каком этапе, находится данный запрос.

2.1.2 Моделирование потоков управления

Следующим типом диаграмм, используемым для моделирования данной системы, является диаграмма активностей, которая показывает переход потока управления от одной деятельности к другой как реакция на внешнее взаимодействие со стороны пользователя.

Данный тип диаграммы позволяет рассмотреть реакцию системы на каждое действие со стороны пользователя, будь то загрузка изображения или нажатие на кнопку. Поэтому была создана диаграмма для рассмотрения процесса входа пользователя в приложения и добавления им нового поста.

Первым шагом как нетрудно догадаться является вход в систему, в случае неправильно введенных данных, или отсутствии каких-то данных пользователю выводится сообщение об ошибке с соответствующей ошибкой, а также предлагается ввести их заново или исправить уже введенные данные. В случае успешной аутентификации пользователю открывается возможность публикации поста для этого он должен выбрать опцию добавить пост. Пользователь может выбрать из представленных опций категорию, состояние поста, а также ввести его описание, где он может ввести как сам запрос на оказание или получение помощи, но также и ввести необходимые контактные данные, которые он почитает нужным и последним шагом загружает изображение для данного поста. После заполнения полей он нажимает на кнопку “post” и при успешном добавлении запроса клиент увидит соответствующее сообщение и сможет увидеть его в очереди самым первым так как посты сортируются по дате их добавления.

На рисунке 2.4 представлен диаграмма для данного процесса, завершением активности является добавление поста в очередь постов, но пользователь может в дальнейшем выполнять другие действия.

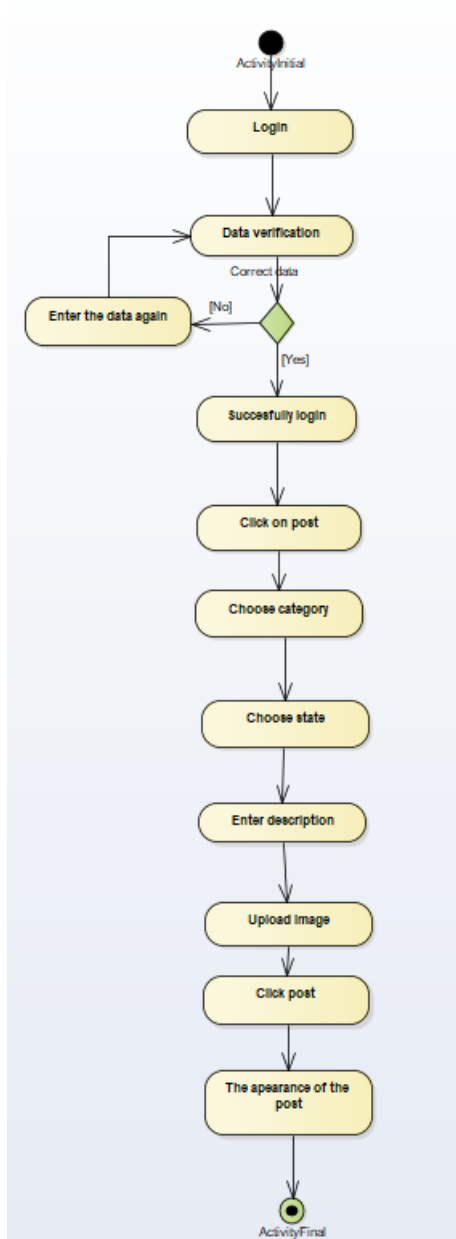


Рисунок 2.4 – Процесс публикации нового поста

Также пользователь может работать с уже загруженным постом, он может делать с любой страницы на который опубликованы посты, будь то домашняя страница, страница одной из категорий или страница с постами самого пользователя. При нажатии на пост производится проверка на то является ли текущий пользователь автором данного поста, если результат положительный пользователю открывается окно редактирования поста, в котором он может

изменить введенную ранее информацию или фото, если же он не хочет редактировать эту информацию он может закрыть данную страницу. На рисунке 2.5 представлена диаграмма активности для работы с уже опубликованными постами.

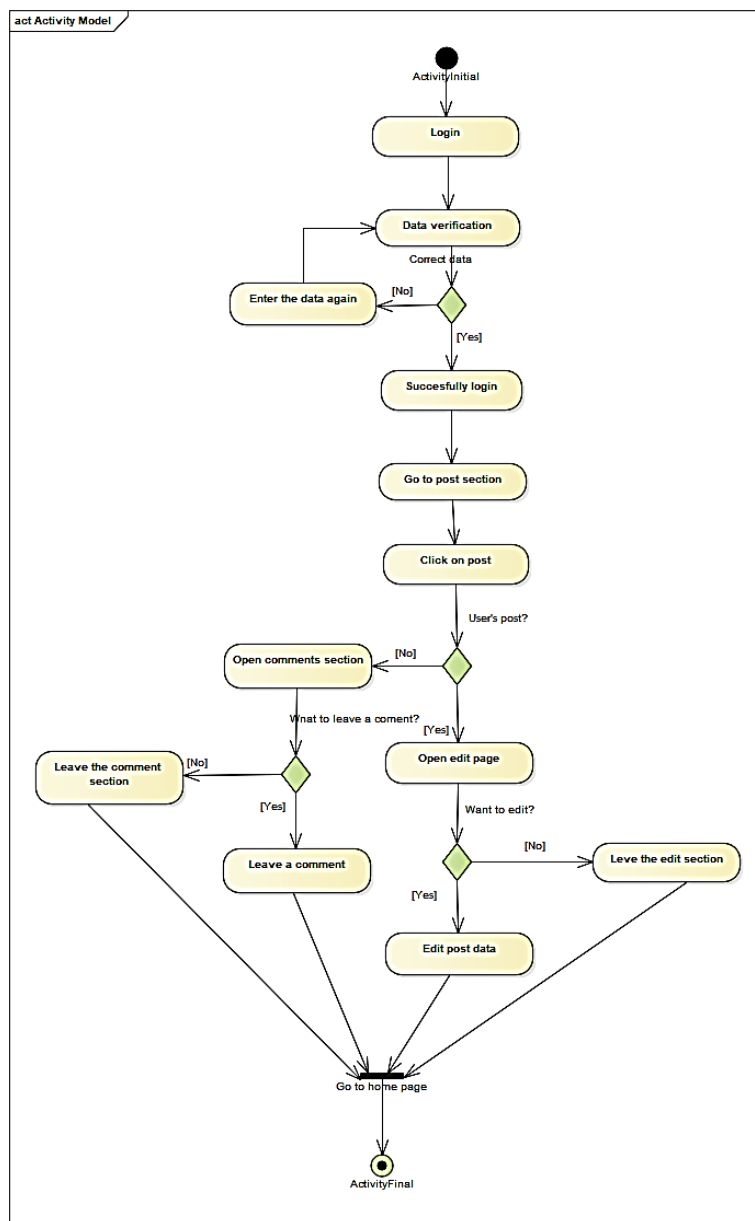


Рисунок 2.5 – Взаимодействие пользователя с опубликованным постом

В случае же если он не является его автором ему откроется секция с комментариями, где он может ознакомиться с комментариями к данному посту, а также оставить свой комментарий. Итогом всех предыдущих действий является переход пользователя на домашнюю страницу.

2.1.3 Взаимодействие объектов системы

Любая система состоит из множества объектов, взаимодействующих между собой посредством обмена некоторой информацией. Одним из способов описания данного

взаимодействия являются диаграммы взаимодействия. Для рассмотрения различных сценариев взаимодействия объектов системы во времени при выполнении конкретной функции или операции чаще всего применяют диаграммы последовательности.

При рассмотрении определенного сценария мы можем наглядно увидеть взаимодействие объектов друг с другом посредством обмена сообщениями различного типа. На рисунке 2.6 представлен сценарий добавления нового поста пользователем с момента входа в приложение.

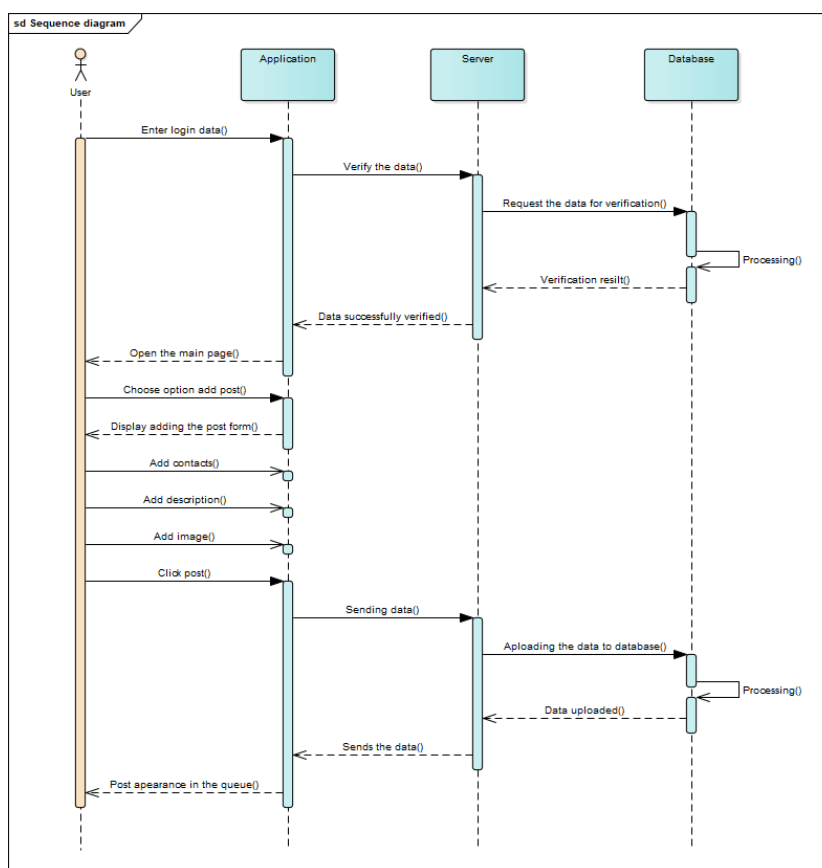


Рисунок 2.6 – Сценарий публикации поста

Можно наглядно рассмотреть взаимодействие основных составляющих объектов системы с пользователем во времени. При выполнении данной операции пользователь взаимодействует с интерфейсом приложения, которое в свою очередь отправляет запросы серверу, который взаимодействует с базой данных. Для публикации нового поста пользователь должен выполнить аутентификацию, и лишь после ее успешного прохождения он может загрузить пост нажав на конкретную опцию. После успешной загрузки поста, пользователь может сразу видеть его в очереди постов.

2.1.4 Описание системы на уровне отдельных объектов

В предыдущей главе для описания взаимодействия объектов системы мы использовали диаграмму последовательности, но это лишь один из возможных вариантов рассмотрения системы с точки зрения входящих в нее объектов и их взаимодействия. Для описания поведения системы на основе отдельных объектов также широко используются диаграммы кооперации или коммуникации. На данном типе диаграмм объекты изображаются как прямоугольники, между взаимодействующими объектами устанавливаются ассоциации. Само взаимодействие между объектами описывается при помощи сообщений, которыми они обмениваются. Сообщения нумеруются порядковыми номерами и, как и на диаграмме последовательности бывают трех видов: синхронные которые включают в себя вызов и ответ, и асинхронное сообщение.

На рисунке 2.7 представлена диаграмма коммуникации, на которой можно наглядно рассмотреть процесс публикации пользователем нового поста.

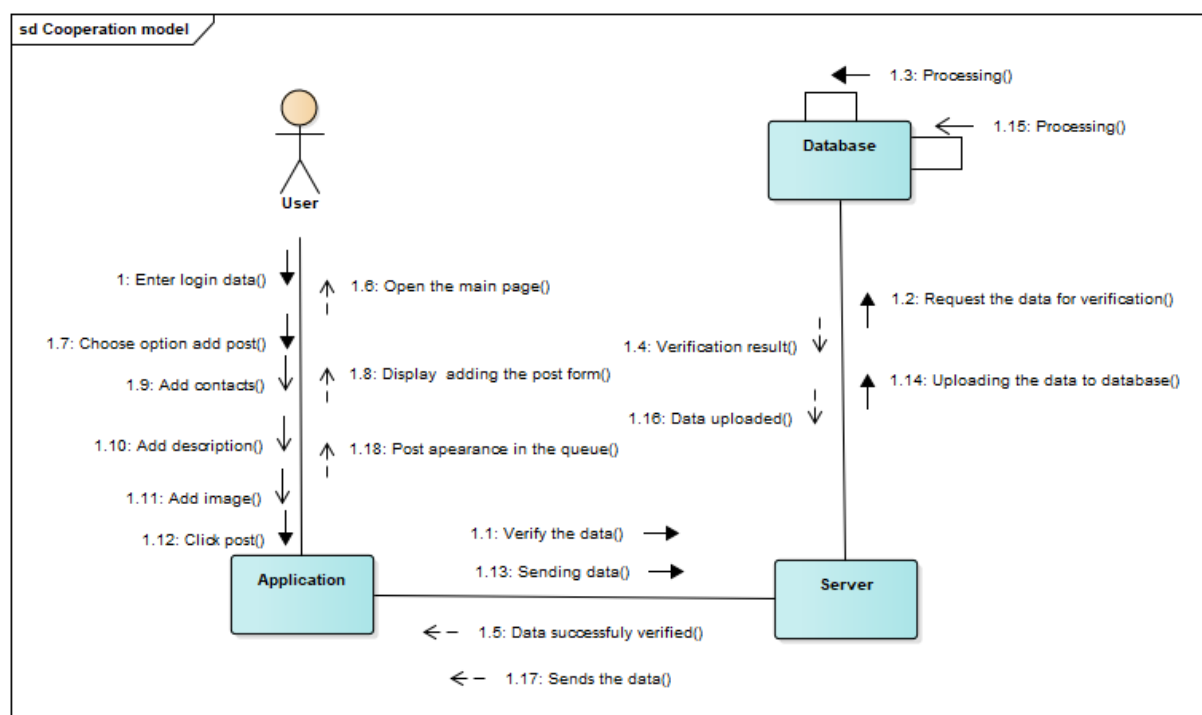


Рисунок 2.7 – Процесс публикации поста

На диаграммах последовательности как несложно понять исходя из названия легче рассмотреть последовательность взаимодействий между объектами, в свою очередь на диаграмме коммуникации в свою очередь легче понять сам поток событий и отношения между объектами.

На данной диаграмме мы можем рассмотреть последовательность шагов в виде сообщений, которыми обмениваются объекты входящие в состав системы. Пользователь лишь выполняет аутентификацию в приложении, выбирает опцию опубликовать пост, и загружает пост с нужной

информацией нажав на кнопку загрузки. В это время приложение обменивается информацией с базой данных для выполнения проверки входных данных, а также для загрузки поста в базу данных. В итоге пользователь лишь увидит свой пост добавленным в череду других.

2.2 Структурное описание системы

Как было отмечено ранее в нотации UML существует 12 типов диаграмм, которые в свою очередь поделены на 2 категории или группы, поведенческие и структурные диаграммы. На предыдущем этапе было проведено моделирование системы с использованием поведенческих диаграмм, то есть рассматривали систему с точки зрения ее взаимодействия с пользователем, а также взаимодействия объектов, входящих в состав системы между собой. Но данных диаграмм недостаточно для получения полноценного представления о системе, они не вдаются в подробности внутреннего устройства и реализации, а именно эти аспекты в основном используются для последующего конструирования архитектуры и составления документации.

Именно структурные диаграммы позволяют наглядно рассмотреть архитектуру будущей системы, входящие в нее элементы, такие как классы, интерфейсы и компоненты системы.

Данная категория диаграмм включает в себя следующие типы:

- диаграмма классов - основное большинство системы созданы с использованием объектно-ориентированного программирования, то есть система организована в виде классов, взаимодействующих между собой, диаграммы классов позволяют представить структуру будущей системы, входящие в нее классы и интерфейсы;
- диаграмма компонентов – так как сложные системы порой состоят из 1000 классов и простой диаграммы классов может быть недостаточно для описания подобной системы, диаграмма компонентов позволяет рассмотреть комплексную систему разделив ее на составляющие, что позволяет легче рассмотреть существующие связи между компонентами системы.;
- диаграмма развертывания – позволяет рассмотреть аппаратные компоненты, входящие в состав сложной системы, данный аспект позволяет получить представление о том насколько система надежна и масштабируема, а также определить расположение компонентов относительно друг друга.

2.2.1 Классовая структура системы

Система была разработана на широко распространенном объектно-ориентированном языке программирования Java, другими словами, основой системы являются классы. Диаграммы классов позволяют представить систему с точки зрения входящих в ее состав классов, а также отношений

между ними. Данный вид диаграмм включает в себя такие виды элементов как классы, интерфейсы и пакеты.

На рисунке 2.8 представлена диаграмма классов для адаптера, который используется для работы с элементами RecyclerView.

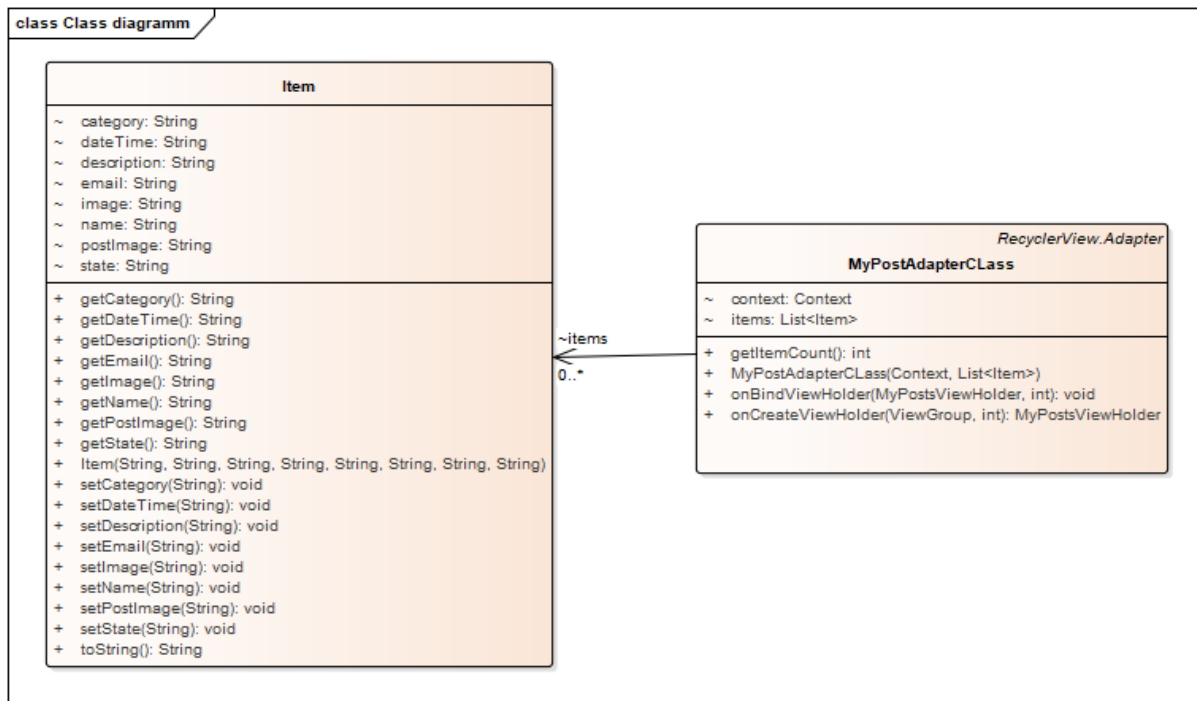


Рисунок 2.8 – Диаграмма классов адаптера для RecyclerView

В нем происходит создание List с объектами класса Item, который впоследствии используется для хранения и данных о постах пользователей. Как видно из диаграммы классов в нем определен конструктор который получает в качестве параметров контекст а также элемент List<Item>. Класс Item в свою очередь содержит в себе атрибуты, которые соответствуют информации о посте и включает в себя дату его создания, категорию, состояние, описание и так далее.

В приложении посты пользователей разделены на категории и в меню приложения пользователь может перейти в интересующую его категорию постов. Именно поэтому в каждом фрагменте, представляющем соответствующую категорию, используется класс MyPostAdapterClass. Атрибуты и операции классов фрагментов очень схожи между собой, так как они выполняют схожий функционал, а именно отвечают за загрузку и публикацию данных из базы данных и основное их различие находится на стороне сервера состоит в категории публикуемых данных. Сами же классы не имеют больших отличий друг от друга. В каждом классе происходит создание объекта класса MyPostAdapterClass, в который впоследствии происходит запись постов из базы данных, для их последующего отображения информация из базы данных будет загружаться в

List объект который затем будет передаваться адаптеру. На рисунке 2.9 представлена диаграмма классов, описывающая те фрагменты, в которых представлены посты пользователей, включая общие посты, а также посты, разделенные по категориям.

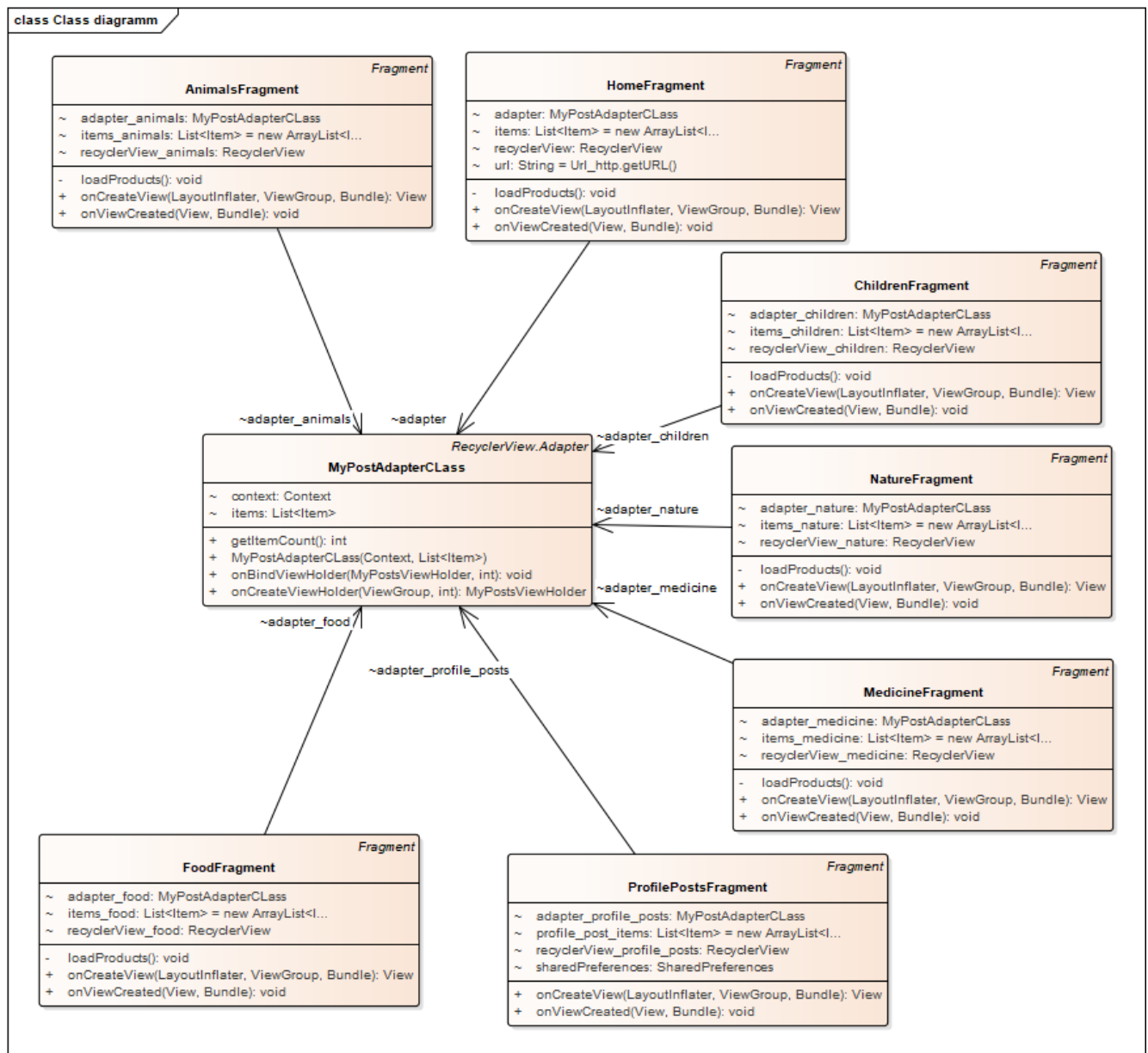


Рисунок 2.9 – Диаграмма классов фрагментов системы

Для более наглядного представления была создана диаграмма классов показывающая связь между классами Item, MyPostAdapterClass и фрагментом HomeFragment. В классе адаптера конструктору передается лист объектов класса Item, который впоследствии заполняется соответствующими значениями, при этом не все они обязательно выводятся для пользователя, элементы могут использоваться в самой программе для написания запросов или последующего использования данных в других классах, как например в классах представленных выше или, в тех которые будут созданы позже, например для расширения допустимых категорий приложения.

Аналогичные связи существуют и с другими классами фрагмента приложения, на рисунке ниже представлена более подробно связь с классом `HomeFragment`.

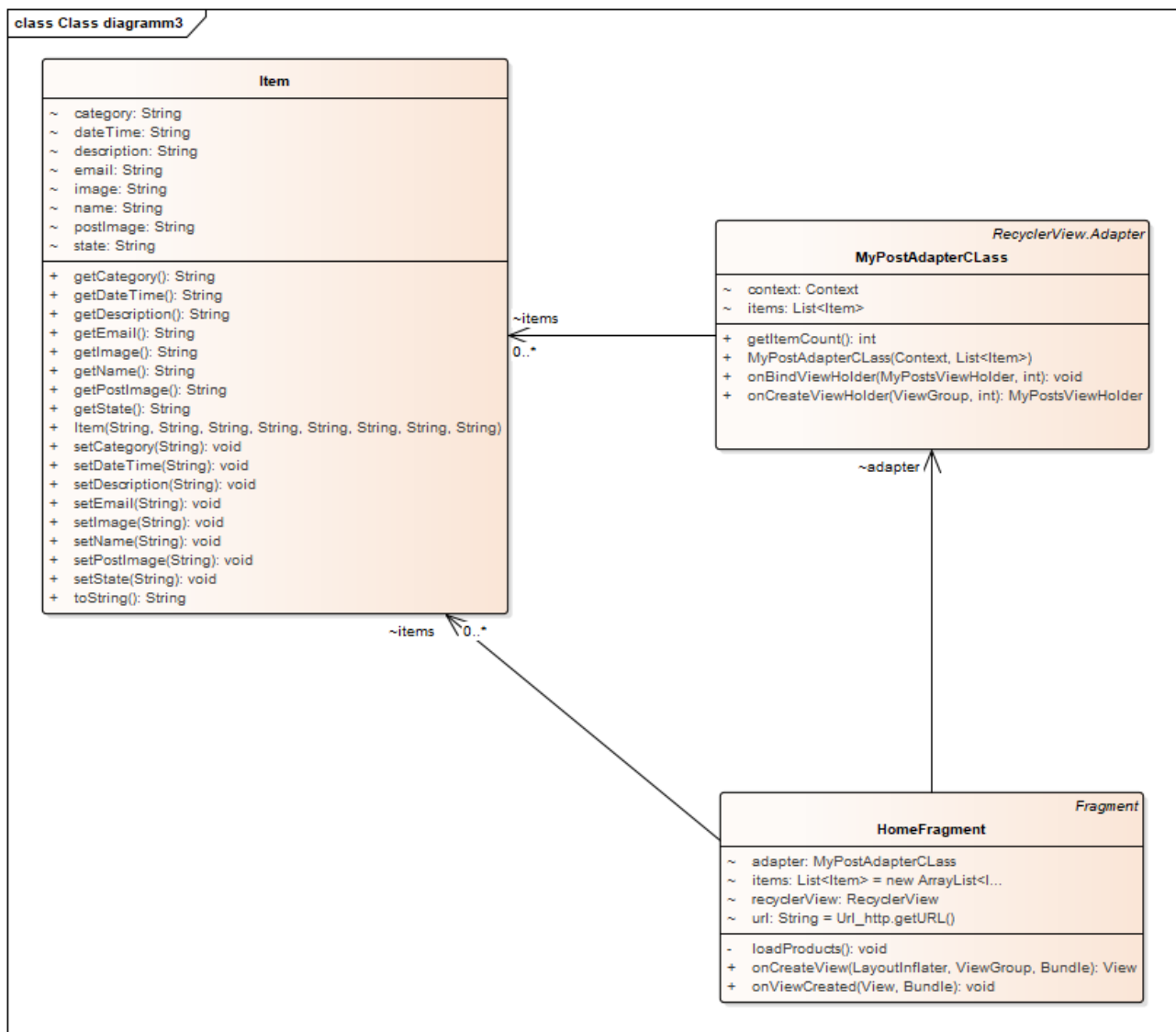


Рисунок 2.10 – Отношения между классами фрагментов и адаптером

Как можно увидеть из данной диаграммы в классе фрагмента `HomeFragment` создается объект класса `Item`, который затем через передается конструктору класса `MyPostAdapterClass`.

2.2.2 Аппаратная составляющая системы

Для успешной разработки системы необходимо понять какие компоненты будут в нее входить и какое аппаратное обеспечение будет использоваться. Диаграммы развертывания помогают справиться с этой задачей, в отличие от других UML диаграмм, которые описывают в основном логическую составляющую системы, они помогают получить представление о физической составляющей, то есть об устройствах, которые будут использоваться данной системой.

На рисунке 2.11 представлена диаграмма развертывания для разрабатываемой системы. Данная диаграмма позволяет получить представление о том из каких компонентов состоит система и как они связаны между собой.

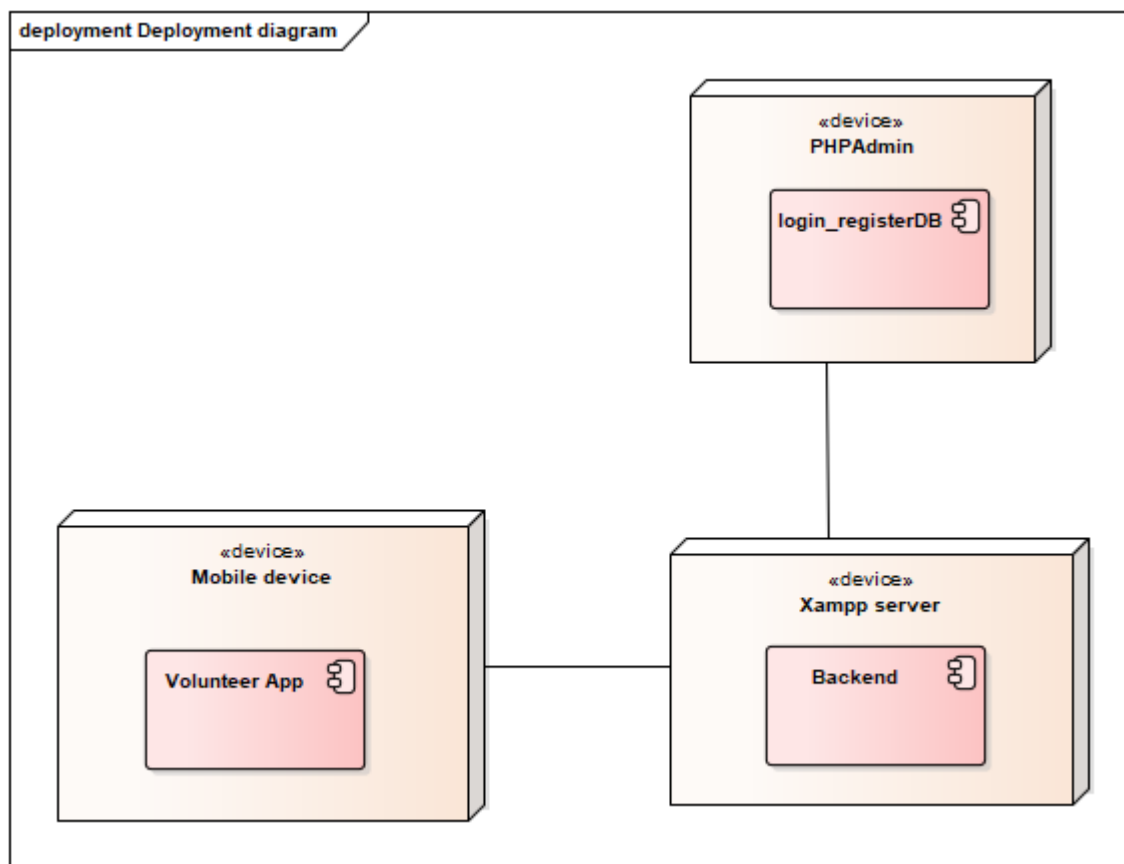


Рисунок 2.11 – Компоненты развертывания системы

При работе приложения для получения данных или их загрузки приложение взаимодействует с сервером, и на стороне Backend производятся запросы к базе данных и результаты данных запросов через Backend возвращаются в приложение. Данная диаграмма позволяет лучше рассмотреть, как именно устройства входящие в систему связаны между собой, сколько уровней в разрабатываемой архитектуре и насколько она масштабируема.

3 РЕАЛИЗАЦИЯ СИСТЕМЫ

В данной главе описывается процесс создания информационной системы, а именно дизайна приложения, его функционала и базы данных. Основная работа над приложением была проведена с использованием интегрированной среды разработки Android Studio. Дизайн приложения написан на языке XAML, функционал приложения на языке Java, а взаимодействие с сервером на языке PHP. В процессе разработки помимо интегрированной среды разработки также был использован локальный сервер Xampp, который использовался для взаимодействия с MySQL базой данных. Далее приведено подробное пошаговое описание процесса разработки информационной системы по организации волонтерской деятельности.

3.1 Дизайн приложения

В процессе работы над данным проектом были определены основные цели и требования к системе, а также был проведен анализ существующих аналогов, что позволило получить более точное представление о типе разрабатываемой системы. Далее для определения ее функционала, входящих в нее компонентов и их взаимодействия, а также аппаратного обеспечения необходимого для реализации системы было проведено моделирование при помощи различных типов UML диаграмм. Совокупность данных этапов позволила получить план действий и примерные шаги для реализации приложения.

Первым этапом стала разработка дизайна приложения. Как уже упоминалось ранее основной средой для разработки была выбрана Android Studio, она содержит в себе встроенные инструменты для создания дизайна, но при создании дизайна данного приложения было принято решение прописать его вручную на языке XAML. Цветовая схема приложения должна была быть приятной и минималистичной. Поэтому на первом этапе создания дизайна была выбрана цветовая схема, основным цветом приложения является белый, также были использованы оттенки голубого и темно-серый цвет. Основные цвета представлены на рисунке 3.1.



Рисунок 3.1 – Цветовая схема приложения

Таже в дизайне приложения использовался онлайн генератор волн, который далее использовались на таких страницах как регистрация, вход и профиль пользователя. Первыми созданными страницами стали страницы входа и регистрации так как это первые страницы, с которыми взаимодействует пользователь. Дизайн данных страниц представлен на рисунке 3.2.

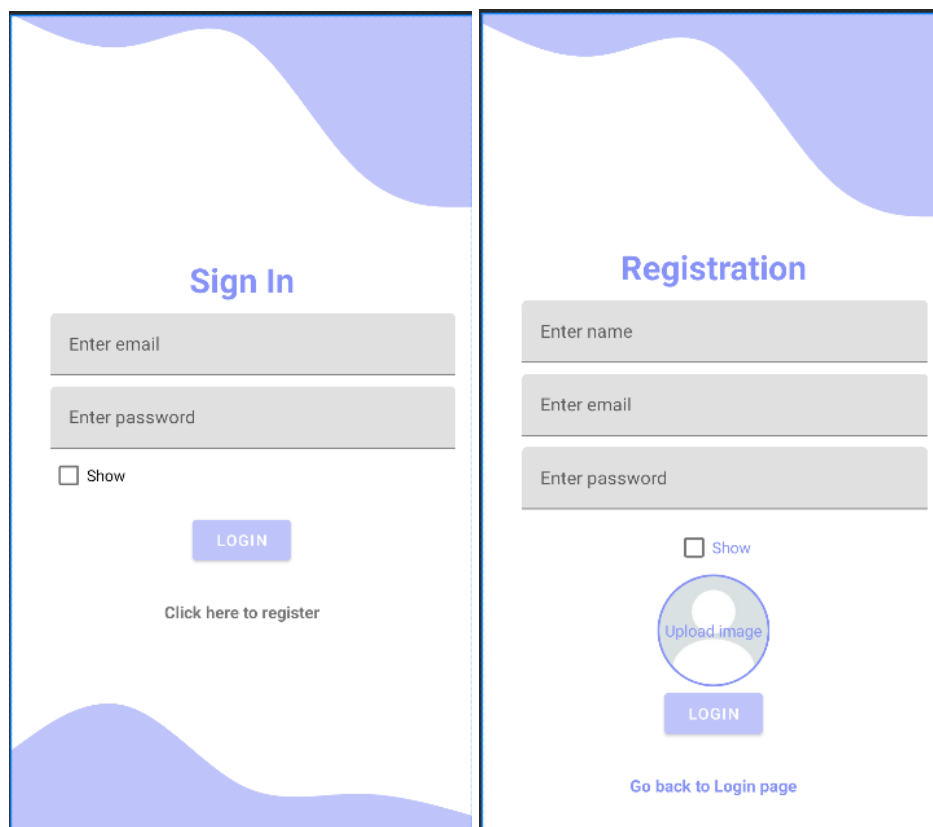


Рисунок 3.2 – Дизайн страниц регистрации и авторизации

Как видно из рисунка выше у данных страниц очень схожий дизайн, основное отличие лишь в наличии на странице регистрации дополнительных элементов для ввода имени и фото пользователя.

Как уже упоминалось ранее при помощи онлайн генератора были созданы волны, которые используются в дизайне этих двух страниц, с помощью удобного редактора были выбраны цвет, и форма волны и сгенерирован xml файл содержащий нужный код для использования в дизайне. Данные файлы затем были загружены в папку drawable и использованы при создании дизайна. Основными элементами этих страниц являются View, TextView, TextInputEditText, CircleImageView и Button. Элемент TextView использовался для вывода текста, TextInputEditText предназначен для ввода пользователем информации, CircleImageView используется как для загрузки изображения пользователем, так и для его вывода на экран, и последний кнопки. Волна,

которая использовалась в дизайне каждой страницы дважды была добавлена при помощи элемента View, как представлено в коде ниже.

```
<View  
  
    android:layout_width="wrap_content"  
  
    android:layout_height="246dp"  
  
    android:layout_gravity="start"  
  
    android:layout_marginTop="-40dp"  
  
    android:background="@drawable/wave__1_" />
```

Далее был создан дизайн страницы профиля пользователя, как и основной страницы, так и страницы, которая будет видна остальным пользователям, основным функционалом приложения является публикация постов, но также пользователи могут оставлять помимо комментариев отзывы на других пользователей. Данные страницы очень схожи между собой и отличаются лишь тем, что на странице профиля, который доступен всем пользователям отсутствует кнопка для выхода из приложения, а также элемент для просмотра и изменения информации пользователя. Основными элементами стали отзывы, для быстрого доступа к отзывам о самом пользователе, посты, чтобы было легче найти нужный пост из опубликованных и отредактировать его или удалить, личная информация, в случае если пользователь захочет отредактировать ее, а также кнопка для выхода из приложения.

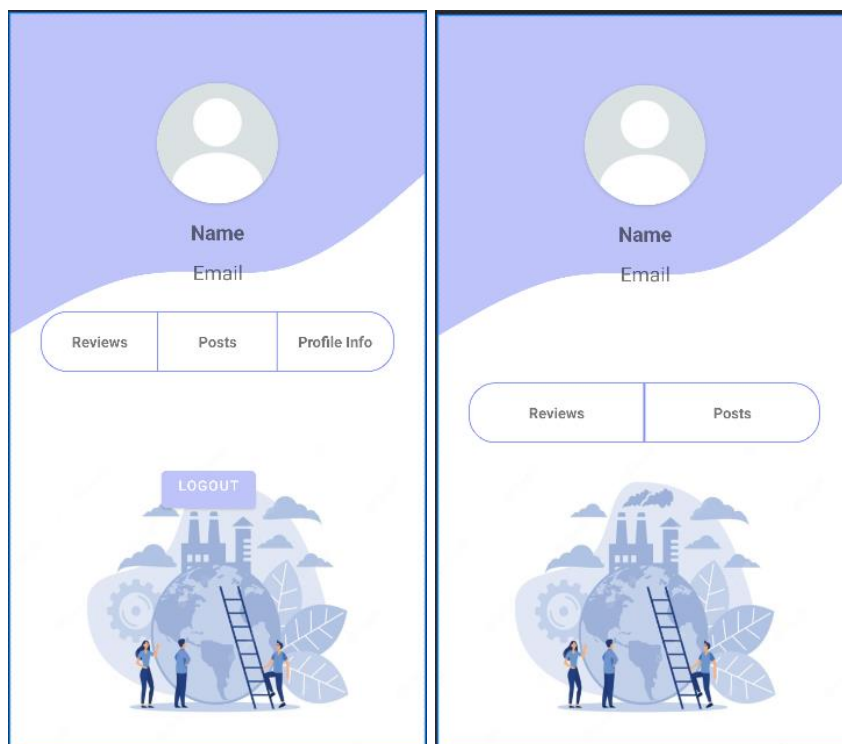


Рисунок 3.3 – Профиль пользователя

При создании данной страницы в качестве элементов дизайна были использованы волна и изображение, подходящее тематике приложения. Самым сложным и объемным элементом дизайна является элемент для переключения на отзывы, посты и информацию профиле. Он состоит из 4 различных элементов контейнеров `LinearLayout`, которые используются для группировки. Был создан один основной контейнер и три других являются вложенными и в свою очередь содержат в себе `TextView`. Для каждого из трех контейнеров были созданы файлы `xml` который использовались как фоновое изображение, каждый из которых представляет собой фигуру. Ниже представлен код `xml` файла использованный для контейнера `reviews`.

```
<?xml version="1.0" encoding="utf-8"?>

<shape xmlns:android="http://schemas.android.com/apk/res/android"

    android:shape="rectangle" >

    <solid android:color="@color/white" >

    </solid>

    <stroke android:width="1dp"

        android:color="@color/default_blue" >

    </stroke>

    <corners android:bottomLeftRadius="26dp"

        android:topLeftRadius="26dp">

    </corners>

</shape>
```

В данном файле прописан основной цвет фигуры, цвет границ, а также радиус. Каждый контейнер включает в себя элемент с текстом, который останется неизменным, в свою очередь имя, почта пользователя и изображение впоследствии будут заменены на соответствующие данные пользователя, вошедшего в систему, или пользователя на страницу которого вошел текущий пользователь.

На следующем этапе были созданы дизайны домашней страницы, а также страниц для вывода постов по категориям, таким как еда, дети, природа, животные, медицина. У них идентичный дизайн, который состоит лишь из трех элементов контейнера, кнопки для загрузки постов и элемента `RecyclerView`, код которого представлен ниже. Данные страниц будут отличаться лишь категорией выводимых постов, который будет определяться на стороне `backend`.

```
</LinearLayout>

<androidx.recyclerview.widget.RecyclerView
```

```

android:id="@+id/recycler_view_posts"

android:layout_width="match_parent"

android:layout_height="wrap_content"

android:layout_below="@id/add_post_button_layout"

android:layout_alignParentBottom="true"

android:scrollbars="vertical"

app:layout_behavior="@string/appbar_scrolling_view_behavior"/>

```

Далее был создан дизайн самого поста, который будет динамически вставляться как объект RecyclerView. Было решено, что пост будет содержать в себе следующую информацию: имя, электронную почту и фото пользователя, выложившего пост и данные о посте такие как категория, состояние, дата, когда он был выложен, описание и фото, а также элемент меню, который будет впоследствии использоваться для вывода таких опций как редактирование и удаление поста. Дизайн самого поста, представлен на рисунке 3.4. Все элементы дизайна включены в контейнер RelativeLayout что позволило вручную установить их положение как по отношению к самому контейнеру, так и по отношению друг к другу. Дизайн поста включает в себя лишь элементы для отображения текста и изображений, даже элемент меню в правом верхнем углу создан при помощи TextView.

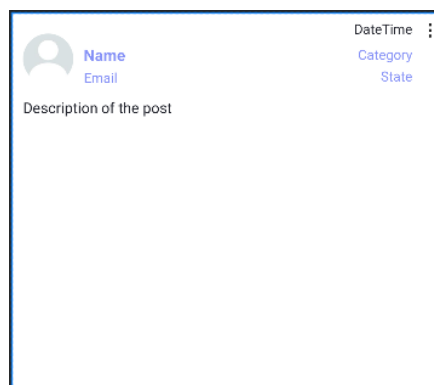


Рисунок 3.4 – Дизайн поста

Следующими созданным дизайном стал дизайн для комментария и отзыва о пользователе. Комментарий и отзыв в отличие от поста включают в себя лишь имя и посту пользователя, сам комментарий и кнопку меню, которая, как и меню на посте создана при помощи элемента TextView код которого представлен ниже.

```

<TextView

    android:id="@+id/commentMenu"

    android:layout_width="wrap_content"

```

```

android:layout_height="wrap_content"

android:layout_alignParentRight="true"

android:layout_alignParentTop="true"

android:layout_marginRight="20dp"

android:text="#8942;"

android:textAppearance="?android:textAppearanceLarge" />

```

Содержание комментария и отзыва абсолютно идентично, именно поэтому было принято решение создать один xml файл с дизайном, который позже используется по-разному за счет отличающегося Java кода и кода на стороне сервера, дизайн представлен на рисунке 3.5.

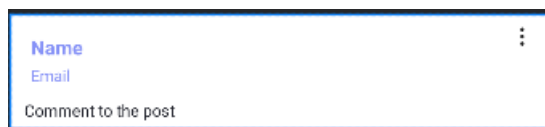


Рисунок 3.5 – Дизайн комментария и отзыва

На следующем этапе был создан дизайн страницы с информацией пользователя, которая предназначена для отображения текущей информации, такой как фото, почта и имя пользователя и кнопками для изменения данной информации. Для данного дизайна, представленного на рисунке 3.6, как и в случае с дизайном профиля пользователя были созданы два xml файла с фигурами которые затем использовались в качестве заднего фона для элемента TextView и для контейнера содержащего данные пользователя и кнопки для изменения этих данных.

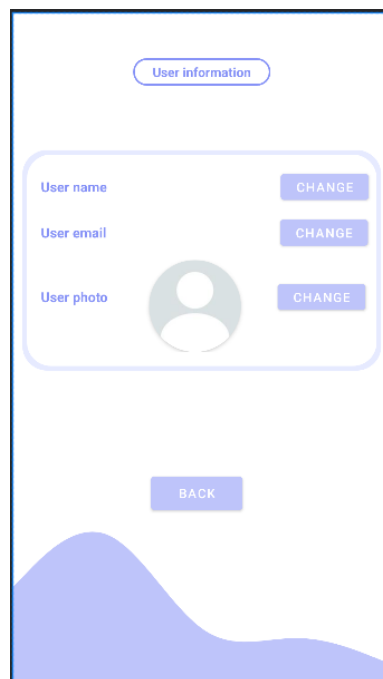


Рисунок 3.6 – Дизайн страницы с информацией о пользователе

В дизайне также был использован элемент волна для заполнения свободного пространства и добавления эстетичности дизайну. В данном случае помимо основного контейнера страницы были использованы 4 контейнера, один основной с xml файлом в качестве фона, и три контейнера содержащие текст и кнопки для изменения каждого из элементов. На изображении выше не видны поля для имени и почты пользователя, а также фото профиля пользователя содержит временное изображение, но при использовании приложения эти данные будут отображаться динамически в зависимости от данных пользователя, вошедшего в систему.

Далее был создан дизайн для добавления и изменения пользователем постов, который приставлен на рисунке 3.7. Для добавления категории и состояния поста было принято решение добавить элементы Spinner, для них позже в коде были определены массивы, хранящие нужные нам значения. Это было сделано для того, чтобы облегчить пользователям ввод информации, избежать ошибок при добавлении постов, а также их сортировке на страницах с категориями. В случае с категорией это будут упомянутые ранее категории, а состояния поста включают в себя новый, в процессе и завершённый, в случае с новым постом все понятно, в процессе значит то, что помощь уже оказывается кем-то, но любой желающий все еще может помочь, а завершённый, то что помощь уже была оказана. Это сделано для пользователей, которые хотят отслеживать проделанную работу и по завершении оказания помощи решат не удалять сам пост.

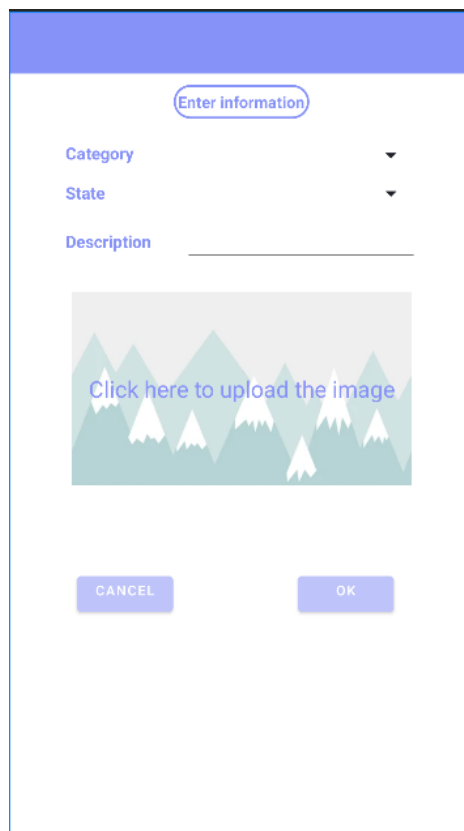


Рисунок 3.7 – Дизайн страницы добавления и изменения постов

Единственное что должен будет ввести пользователь это описание самого поста, а также выбрать изображение, которое он хочет добавить. Изначально было задумано добавить поля для ввода контактных данных таких как телефон или поста пользователя, но впоследствии было принято решение отказаться от этого. Пользователь сам сможет решить какие личные данные или контактные данные он готов предоставить и может внести их в описание самого поста. В зависимости от того хочет ли пользователь добавить новый пост или изменить уже имеющийся поля выглядят по-разному. В случае, когда пользователь хочет добавить новый пост он увидит дизайн, представленный на изображении выше в неизменном виде, но, если это уже существующий пост все поля включая изображение будут заполнены имеющимися данными.

Для удобства и легкости использования в поле с изображением также был добавлен текст `Click here to upload image` чтобы пользователи знали какие действия им нужно выполнить. Это было реализовано путем вложения элемента текст в элемент с изображением при помощи таких параметров как `layout_alignTop`, `layout_alignRight` и так далее, как представлено в коде ниже.

```
<ImageView
    android:id="@+id/post_edit_add_image" android:layout_width="300dp"
    android:layout_height="200dp" android:src="@drawable/blanc_post_pic"/>
<TextView    android:id="@+id/postImageText"
    android:layout_width="wrap_content" android:layout_height="wrap_content"
    android:layout_alignLeft="@+id/post_edit_add_image"
    android:layout_alignTop="@+id/post_edit_add_image"
    android:layout_alignRight="@+id/post_edit_add_image"
    android:layout_alignBottom="@+id/post_edit_add_image"
    android:gravity="center"
    android:text="Click here to upload the image"
    android:textColor="@color/default_text_blue" android:textSize="20dp" />
```

Само изображение представлено в дизайне является временным и после загрузки желаемого изображения пользователем заменяется на него,

Также для добавления пользователями комментариев к постам или отзывов о самих пользователях был создан идентичный дизайн, отличающийся лишь текстом в верхней части экрана. Несмотря на то, что и комментарий, и отзыв будут включать себя имя, почту и изображение пользователя эти данные будут автоматически заполняться из базы данных,

пользователю не нужно вводить их вручную. Поэтому страницы добавления комментария или отзыва содержат в себе лишь само поле для его добавления как представлено на рисунке 3.8.

The image shows two side-by-side mobile application screens. The left screen has a blue header bar. Below it is a white area with a blue button labeled 'Leave your review'. Underneath the button is a horizontal line representing a text input field. At the bottom of the screen are two blue buttons: 'CANCEL' on the left and 'OK' on the right. The right screen is identical in layout but has a blue button labeled 'Leave your comment' instead of 'Leave your review'.

Рисунок 3.8- Страницы добавления отзыва и комментария

3.2 База данных

После завершения работы над дизайном приложения была создана реляционная MySQL база данных для хранения информации, используемой в приложении. На основе имеющихся данных было принято решение создать 4 отношения, а именно: users, post_info, post_comments, user_reviews, которые связаны между собой связью один ко многим. Далее будет описано каждое отношение, а также подробнее рассмотрены связи между ними.

Первым и самым важным отношением, которое связано со всеми последующими является отношение users. Помимо таких данных как имя, почта, изображение и пароль пользователя оно также хранит его id и такой параметр как apiKey, который нужен для контроля доступа к приложению, по умолчанию он пустой и в случае, когда пользователь вошел в систему он заполняется уникальным значением.

Для данного отношения первичным ключом является id пользователя, именно поэтому он должен быть уникальным, также он заполняется автоматически за счет параметра AUTO_INCREMENT. Помимо первичного ключа поле с почтой пользователя должно быть уникальным, так мы контролируем чтобы два разных пользователя не смогли зарегистрироваться, используя один и тот же почтовый адрес. Пароль не сохраняется в базе данных в начальном виде, в котором его ввел пользователь, а шифруется. Несмотря на то, что MySQL база данных имеет отдельный формат для загрузки и хранения изображений, было принято решение сохранять лишь

путь к файлу, это уменьшает вес, хранящихся в базе данных. Отношение пользователи представлено на рисунке 3.9.

	#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
<input type="checkbox"/>	1	id	int(11)			No	None		AUTO_INCREMENT	Change Drop More
<input type="checkbox"/>	2	name	text	utf8mb4_general_ci		No	None			Change Drop More
<input type="checkbox"/>	3	email	varchar(300)	utf8mb4_general_ci		No	None			Change Drop More
<input type="checkbox"/>	4	password	text	utf8mb4_general_ci		No	None			Change Drop More
<input type="checkbox"/>	5	apiKey	text	utf8mb4_general_ci		Yes	NULL			Change Drop More
<input type="checkbox"/>	6	user_photo	text	utf8mb4_general_ci		No	None			Change Drop More

Рисунок 3.9 – Отношение users

Далее было создано отношение user_reviews хранящее отзывы пользователей о других пользователях, которое помимо таких данных как отзыв, id отзыва хранит id обоих пользователей, как и того, кто оставил отзыв так и того, кому отзыв предназначен, а также время, когда отзыв был оставлен, которое впоследствии понадобится для сортировки. Отношение user_reviews представлено на изображении 3.10.

	#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
<input type="checkbox"/>	1	review_id	int(11)			No	None		AUTO_INCREMENT	Change Drop More
<input type="checkbox"/>	2	user_id	int(11)			No	None			Change Drop More
<input type="checkbox"/>	3	user_review_id	int(11)			No	None			Change Drop More
<input type="checkbox"/>	4	review	varchar(1000)	utf8mb4_general_ci		No	None			Change Drop More
<input type="checkbox"/>	5	datetime	timestamp			No	current_timestamp()			Change Drop More

Рисунок 3.10 – Отношение user_reviews

Первичным ключом в данном отношении является review_id, вторичными ключами user_id и user_review_id, при помощи данных ключей отношение связано с отношением users на основе id пользователя.

Самым большим отношением является отношение post_info, оно хранит такие атрибуты как категория, состояние и описание поста, изображение, а также дату, которая впоследствии не только используется для сортировки постов, но также и для отображения на самом посте. Как и в случае с изображением пользователя изображение для поста хранится в виде пути к изображению, которое в момент публикации сохраняется в отдельной папке. Это сделано для того, чтобы

избежать утери изображения в случае, если пользователь решит удалить его с устройства.

Отношение post_info представлено на рисунке 3.11.

	#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
<input type="checkbox"/>	1	id	int(11)			No	None			Change Drop More
<input type="checkbox"/>	2	post_id	int(11)			No	None		AUTO_INCREMENT	Change Drop More
<input type="checkbox"/>	3	post_date	timestamp			No	current_timestamp()			Change Drop More
<input type="checkbox"/>	4	post_category	text	utf8mb4_general_ci		No	None			Change Drop More
<input type="checkbox"/>	5	post_state	text	utf8mb4_general_ci		No	None			Change Drop More
<input type="checkbox"/>	6	post_description	text	utf8mb4_general_ci		No	None			Change Drop More
<input type="checkbox"/>	7	post_photo	text	utf8mb4_general_ci		Yes	NULL			Change Drop More

Рисунок 3.11 – Отношение post_info

Первичным ключом в данном случае является post_id, который является уникальным для каждого поста, а вторичным ключом id при помощи которого реализована связь многие к одному по отношению к таблице users по атрибуту id.

Последним отношением является отношение post_comments, которое хранит комментарии пользователей к посту. Данное отношение очень схоже с отношением user_reviews только в отличие от него хранит не id пользователя, которому предназначается отзыв, а id поста, к которому этот комментарий относится, в остальном все поля идентичны. Отношение post_comments представлено на рисунке 3.12.

	#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
<input type="checkbox"/>	1	post_id	int(11)			No	None			Change Drop More
<input type="checkbox"/>	2	comment_id	int(11)			No	None		AUTO_INCREMENT	Change Drop More
<input type="checkbox"/>	3	comment	text	utf8mb4_general_ci		No	None			Change Drop More
<input type="checkbox"/>	4	datetime	timestamp			No	current_timestamp()			Change Drop More
<input type="checkbox"/>	5	user_id	int(11)			No	None			Change Drop More

Рисунок 3.12 – Отношение post_comments

Первичным ключом в данном случае является comment_id, а вторичными id пользователей, с помощью которых данное отношение связано с отношением users. Как уже упоминалось ранее все отношения связаны между собой связью один ко многим. Отношение users связано связью один ко многим со всеми отношениями по атрибуту id, а отношение post_info связано связью один ко многим с отношением post_comments, единственные отношения, не связанные между собой это

user_revies и post_info. Связи между всеми существующими отношениями представлены на рисунке 3.13.

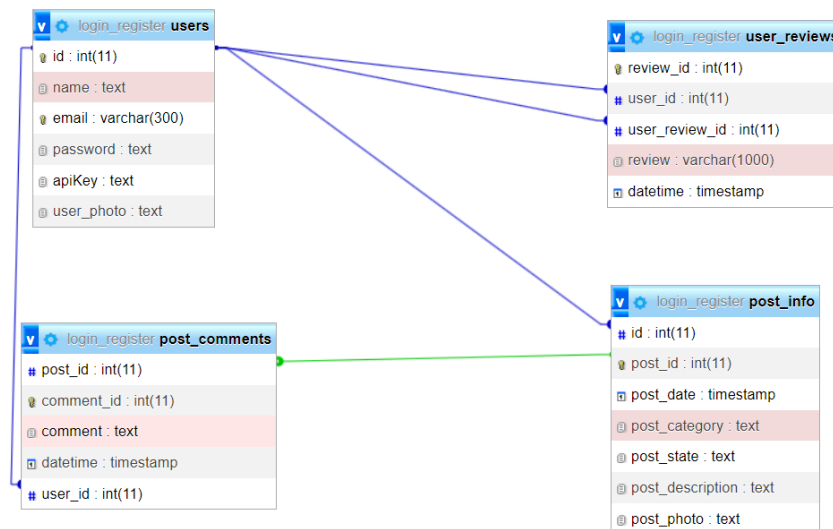


Рисунок 3.13 – Связи между отношениями базы данных

3.3 Серверная сторона приложения

Как уже упоминалось ранее серверная сторона приложения, то есть взаимодействие приложения с базой данных написаны на языке PHP. Практически для каждого действия пользователя будь то отображение данных в самом приложении или загрузка данных из приложения были прописаны скрипты.

Первыми реализованными скриптами стали скрипты для регистрации, входа пользователя в приложение, а также выхода из приложения. Далее каждый из этих скриптов рассмотрен по отдельности. Ниже представлен код для регистрации пользователя.

```

<?php

if(!empty($_POST['name']) && !empty($_POST['email']) && !empty($_POST['password']) &&
!empty($_POST['image'])) {

    $con = mysqli_connect("localhost", "root","", "login_register");

    $name = $_POST['name']; $email = $_POST['email'];

    $image1='image'.date("d_m_Y").'_'.rand(1000,100000);

    $image = 'D:/Android Projects/Volunteer/app/src/main/res/drawable/'.
$image1.'.png';

```

```

$password = password_hash($_POST['password'], PASSWORD_DEFAULT);

if($con){ if(file_put_contents($image,base64_decode($_POST['image']))) {

$sql = "insert into users (name, email, password, user_photo)
values ('".$_name."', '".$_email."', '".$_password."', '".$_image1."')";

if(mysqli_query($con,$sql)){ echo "Success";

} else echo "Registration Failed";

} else echo 'Failed to upload image';

}else echo "Database connection failed";

} else echo "All fields are required";

```

Первым шагом производится проверка если все нужные данные такие как имя пользователя, почта, пароль и изображение профиля были введены пользователем, затем эти данные присваиваются определенным переменным. В случае с изображением мы также устанавливаем путь куда именно мы его будем сохранять, в базу данных записывается лишь название самого изображения в формате String, а введенный пароль сразу кодируется. Если какие-то данные отсутствуют будет возвращено соответствующее сообщение об ошибке. Далее устанавливается соединение с базой данных и происходит его проверка если оно не установлено будет возвращено сообщение об ошибке. После успешного подключения к базе данных происходит проверка если изображение было загружено в указанную директорию, если нет возвращается сообщение об ошибке, а если да, то начнется выполнение скрипта на языке SQL. В данном случае выполняется запрос insert и в качестве значений передаются данные, хранящиеся в обретенных ранее переменных. В случае успешного завершения запроса будет возвращено сообщение success, которое далее будет обработано уже программой.

Следующим был написан скрипт для входа пользователя в приложение, который представлен ниже.

```

<?php

if(!empty($_POST['email']) && !empty($_POST['password'])) {

$email=$_POST['email']; $password =$_POST['password'];

```

Как и в случае с регистрацией сначала проводится проверка если пользователь ввел все нужные данные и введенные данные присваиваются определенным переменным. Далее устанавливается соединение с базой данных и проверяется установлено ли оно.

```

$result = array(); $con = mysqli_connect("localhost", "root","", "login_register");

```

Затем выполняется запрос и сравнивается если введенные данные соответствуют хранящимся в базе данных.

```
if($con){ $sql = "select * from users where email='".$email."'"; $res=mysqli_query($con, $sql);

if(mysqli_num_rows($res) != 0){ $row = mysqli_fetch_assoc($res);

if($email == $row['email'] && password_verify($password, $row['password'])){

try{ $apiKey = bin2hex(random_bytes(23));}

catch(Exception $e){ $apiKey = bin2hex(uniqid($email, true));}
```

В случае успешного прохождения проверки устанавливается уникальный `apiKey` для почты, введенной пользователем, и результат возвращается в программу в виде массива, результат зависит от того выполнен ли каждый этап успешно или нет. Далее, как и в случае с регистрацией каждое сообщение обрабатывается программой.

```
$sqlUpdate = "update users set apiKey='".$apiKey.'" where email = '".$email."'";

if(mysqli_query($con,$sqlUpdate)){ result=array("status"=> "success","message"=> "Login successful","name"=> $row['name'], "email"=> $row['email'], "password"=> $row['password'], "apiKey"=> $apiKey);

}else $result =array("status"=> "failed", "message"=> "Login failed, try again");

}else $result=array("status"=> "failed", "message"=> "Retry with correct email and password");

}else $result=array("status"=> "failed", "message"=> "Retry with correct email and password");

}else $result=array("status"=> "failed", "message"=> "Database connection failed");

}else $result=array("status"=> "failed", "message"=> "All fields are required");

echo json_encode($result, JSON_PRETTY_PRINT);
```

Выход из приложения осуществляется схожим образом из приложения скрипт получает почту пользователя проверяет если установлен `apiKey`, в случае успешного прохождения проверки, он обнуляется и происходит выход из приложения.

Далее будут рассмотрены скрипты по добавлению, изменению и удалению поста пользователем. Так как добавление, изменение и удаление комментариев и отзывов выполняется схожим образом они не будут рассмотрены. Ниже представлен скрипт для добавления поста пользователем.

```
<?php

if(!empty($_POST['post_category']) && !empty($_POST['post_state']) &&
!empty($_POST['post_description']) && !empty($_POST['image'])){
```



```

$con = mysqli_connect("localhost", "root","", "login_register");

$post_category = $_POST['post_category']; $post_state = $_POST['post_state'];

$post_description=$_POST['post_description'];
$image1='image'.date("d_m_Y").'_'.rand(1000,100000);

$image = 'D:/Android Projects/Volunteer/app/src/main/res/drawable/'.
$image1.'.png';

```

Первым шагом выполняется загрузка данных из приложения, которые затем присваиваются определенным переменным, в случае с изображением также определяем путь куда именно оно должно сохраниться. Далее проверяется если подключени к базе данных было выполнено успешно, а также если изображение было загружено в указанную папку.

```

if($con){

if(file_put_contents($image,base64_decode($_POST['image']))) {

```

Следующим шагом выполняется запрос insert для данных введенных пользователем и производится проверка если запрос был выполнен успешно.

```

$sql = "insert into post_info (id, post_category, post_state, post_description,post_photo)
values ((select id from users where apiKey<>'), '$_POST['post_category'].', '$_POST['post_state'].',
'$_POST['post_description'].', '$_POST['image'].')";

if(mysqli_query($con,$sql) && $con->affected_rows>0){ echo "success"; } else echo "Post
publication Failed";

} else echo 'Failed to upload image'; }else echo "Database connection failed";

} else echo "All fields are required";

```

В зависимости от результата каждого шага происходит либо переход к следующему, либо возвращается соответствующий результат, который затем обрабатывается программой.

Добавление комментариев и отзывов о пользователе происходит схожим образом. Отличие заключается лишь в данных, которые передаются из программы, а также в самих sql запросах. В случае с добавлением комментариев к постам скрипт получает в качестве данных id поста и сам комментарий, а в случае с добавлением отзыва сам отзыв и почту пользователя.

Изменение поста происходит схожим образом единственное отличие состоит в типе запроса SQL, если в случае с добавлением поста это запрос INSERT, то в случае его изменения это запрос UPDATE. Скрипт получает данные, проверяет если они все были введены, выполняет проверку подключения к базе данных и если изображение было загружено и выполняет запрос update, который выглядит следующим образом.

```
$sql = "update post_info set post_category='".$post_category."', post_state='".$post_state."',
post_description='".$post_description."', post_photo='".$image1.'"

where id in(select post_info.id from post_info LEFT JOIN users on post_info.id=users.id where
users.apiKey<>'')and post_info.post_id='".$post_id_int.'";
```

При добавлении поста происходит проверка по id поста, а также по id пользователя, проверяется если id пользователя, добавившего пост соответствует id пользователя, который хочет изменить пост. В случае изменения комментария или отзыва о пользователе запросы очень схожи отличаются лишь данные, которые будут отправлены в скрипт и сами запросы. Но в независимости запроса, будь то добавление или удаление всегда происходит проверка если пользователь, который хочет выполнить эти действия идентичен пользователю, который изначально создал запрос.

Следующим были реализованы скрипты для загрузки постов из базы данных, как для основной страницы, куда выводятся все имеющиеся на данный момент скрипты, так и для страниц с категориями, разница состоит лишь в коде РНР, а именно в запросе. В случае, когда нужно вывести посты для определенной категории в запрос добавляется условие where с соответствующей категорией, например ниже представлен запрос для вывода постов из категории природа.

```
$stmt = $con->prepare("select users.name, users.email, users.user_photo,
post_info.post_category, post_info.post_state, post_info.post_description,
post_info.post_photo, post_info.post_date, post_info.post_id from users
right join post_info using(id) where post_info.post_category='Nature' order by
post_info.post_date");
```

Посты сразу сортируются по дате, кроме основной информации о посте выводятся также данные пользователя, загрузившего пост, это делается путем добавления right join с таблицей users в запрос. Единственное отличие при выводе запросов других категорий — это значение post_category. А в случае когда пользователь хочет посмотреть посты конкретного пользователя или свои в запрос добавляется условие where для почты пользователя запросы которого нас интересуют.

Помимо вышеперечисленных запросов, также были созданы запросы для вывода информации и пользователя, вывода информации о посте, который пользователь хочет изменить, это сделано для того, чтобы пользователю не пришлось вводить уже введенные данные вновь. Также скрипты для изменения имени, почты и фото пользователя загрузки комментариев и отзывов о пользователе. Все эти скрипты далее использованы в программе и в зависимости от того

выполняются они или нет пользователю предлагается выполнить конкретные действия. Как в случае с входом в приложение при неправильно введенных данных или их отсутствии пользователь увидит соответствующее сообщение.

3.4 Основная Activity и навигация в приложении

Первым шагом в разработке самого приложения в Android Studio стала работа с имеющейся MainActivity. Было принято решение реализовать в ней меню как боковое, так и нижнее, а остальные страницы сделать в виде Fragments. Одним из преимуществ фрагментов является то, что всю логику переключения между ними можно написать в основной Activity и скорость такого переключения будет выше чем скорость переключения между Activity. А также меню определяется лишь один раз, и вся логика прописывается в MainActivity.

В данном случае были созданы два меню нижнее для переключения между самыми используемыми страницами так и боковое меню для переключения между категориями постов. Для этого была определена отдельная папка menu содержащая xml файл с желаемыми пунктами меню, а также изображением и текстом которые будут выводиться. Код для нижнего меню представлен ниже.

```
<?xml version="1.0" encoding="utf-8"?>

<menu xmlns:android="http://schemas.android.com/apk/res/android"

      xmlns:app="http://schemas.android.com/apk/res-auto">

    <item android:title="Profile"

          android:icon="@drawable/ic_baseline_person_24"

          android:id="@+id/profile"/>

    <item android:title="Home"

          android:icon="@drawable/ic_baseline_home_24"

          android:id="@+id/home" />

    <item android:title="My Posts"

          android:icon="@drawable/ic_baseline_text_snippet_24"

          android:id="@+id/settings" />

</menu>
```

Боковое меню отличается лишь количеством элементов. Также для бокового меню отдельно был создан header в котором будет выводиться логотип, имя и почта пользователя. Далее дизайн, как и бокового меню так и нижнего определен в файле activity_main.xml. Для нижнего меню отдельно был создан xml файл с фигурой, которая затем была установлена в качестве фона.

Далее были созданы фрагменты для разных категорий постов, а также для домашнего экрана куда будут выводиться все посты.

В MainActivity были определены NavigationView и BottomNavigationView которым были присвоены названия созданных ранее меню. Для реализации навигации между фрагментами был объявлен onNavigationItemSelectedListener для каждого объявленного ранее меню, а также была создана функция для перехода между фрагментами, код которой представлен ниже.

```
private void replaceFragment(Fragment fragment)
{
    FragmentManager fragmentManager = getSupportFragmentManager();
    FragmentTransaction fragmentTransaction = fragmentManager.beginTransaction();
    fragmentTransaction.replace(R.id.container, fragment);
    fragmentTransaction.commit();
}
```

Далее в каждом onNavigationItemSelectedListener использовался switch (case) для перехода между фрагментами, ниже представлены условия для нижнего меню.

```
case R.id.home:
    replaceFragment(new HomeFragment());
    return true;
case R.id.profile:
    replaceFragment(new ProfileFragment());
    return true;
case R.id.settings:
    replaceFragment(new ProfilePostsFragment());
    return true;
```

В зависимости от того, как какой странице хочет перейти пользователь вызывается функция, которой передается фрагмент. Переходы между элементами бокового меню выполнены схожим образом и отличаются лишь количеством и составом фрагментов. Также так как в боковом меню помимо перехода к странице есть еще опция выхода из приложения, она реализована в соответствующей опции case.

3.5 Классы адаптеры для постов, комментариев и отзывов

Для работы с постами было создано три вспомогательных класса, Item MyPostAdaperClass и MyPostViewHolder, MyCommentsViewHoder. В классе Item определены все переменные для хранения информации о посте, такие как description, state, image, name, email и так далее, также был определен базовый конструктор, а также getter и setter для каждой переменной. В свою очередь класс MyPostViewHolder имплементирует класс RecyclerView.ViewHolder и выполняет роль контейнера для элементов нашего xml файла. В нем определены переменные, которые впоследствии будут использоваться в классе adapter для взаимодействия с постом. Ниже представлен пример того, как определены переменные для изображения пользователя, даты, имени и почты пользователя.

```
imageViewProfile=(ImageView) itemView.findViewById(R.id.image_view_post_item);

textViewDateTime=itemView.findViewById(R.id.item_post_datetime);

textViewName=itemView.findViewById(R.id.post_item_name);

textViewEmail=itemView.findViewById(R.id.post_item_email);
```

MyPostAdapterCladd хранит в себе всю логику взаимодействия пользователя с постом. В нем определен лист объектов класса Item, который затем заполняется соответствующими данными для каждого поста. В данном классе определен onClickListener для Layout который содержит в себе фото, имя и почту пользователя, в котором прописана логика, а именно при нажатии на область где находятся эти данные пользователю открывается профиль пользователя выложившего пост, для этого создается Intent для перехода к нужной нам Activity, в данном случае это ProfileForOtherUserActivit, который при помощи опции putExtra передается почта пользователя выложившего пост, код представлен ниже.

```
Intent intentProfileForOtherUsers = new Intent(context, ProfileForOtherUsersActivity.class);
intentProfileForOtherUsers.addFlags(Intent.FLAG_ACTIVITY_NEW_TASK);

intentProfileForOtherUsers.putExtra("userEmail", items.get(holder.getAdapterPosition()).getEmail());

context.startActivity(intentProfileForOtherUsers);
```

Схожим образом выполняется переход на страницу, содержащую комментарии к данному посту. Информация о посте такая как описание и изображение заранее были заключены в Layout на который мы устанавливаем onClickListener и выполняем переход к странице с комментариями в случае нажатия пользователем на данную область, в данном случае мы также используем опцию putExtra но передаем id поста, который используется на странице с комментариями для выполнения запроса к базе данных.

Также в классе `MyPostAdapterClass` определено меню нажав на опции которого пользователь сможет редактировать или удалит созданный им пост. В обоих случаях перед выполнением этих действий пользователь увидит предупреждение с вопросом о том действительно ли он этого хочет. Это сделано для того, чтобы избежать случайного удаления поста. Это было реализовано при помощи создания `AlertDialog.Builder` с двумя условиями в случае, если пользователь нажал на «No» диалоговое окно просто закроется, и пользователь останется на той же странице что и был. В противном случае если он нажмет «Yes» при выборе опции удаление, профиль удалится и пользователь увидит обновленную страницу с постами, где удаленный пост отсутствует. Если же он выбрал опцию изменения поста он будет перенаправлен на страницу редактирования поста, как и в предыдущем случае при помощи `Intent` и с использованием `putExtra` для передачи `id` поста, чтобы затем выполнить запрос к базе данных для вывода введенной ранее информации.

Подобные классы также были созданы для комментариев и отзывов о пользователе, а именно `ItemComment`, `ItemReview`, `ReviewsViewHolder`, `MyPostCommentViewHolder`, `MyPostCommentAdapterClass` и `ReviewsAdapter`, которые выполняют схожие функции для каждой категории информации.

Класс `ItemComment` содержит в себе такие переменные как `name`, `email`, `comment`, `dateTime`, `commentId`, а также конструктор и геттеры, и сеттеры. В классе `MyPostCommentViewHolder` определены все элементы `xml` документа, которые впоследствии используются в классе `MyPostCommentAdapterClass` для взаимодействия с комментариями. В самом же классе `MyPostCommentAdapterClass` определен лист элементов класса `MyPostCommentViewHolder`, и определены действия при нажатии пользователем на пункты меню для комментария, такие как удаление комментария и его изменение, как и в случае с постами, пользователь увидит всплывающее окно с подтверждением его действий. Но в отличие от постов. При нажатии на сам комментарий не будет выполнено никаких действий.

Классы для отзывов схожи с классами, созданными для комментариев, отличие состоит в хранимых данных и в том, что они привязаны к разным `Activity`, но также как и в случае с комментариями в адаптере прописана логика при нажатии на пункты меню с опциями удаления или изменения отзыва.

Описанные выше классы для постов, комментариев и отзывов являются вспомогательными и впоследствии используются при загрузке или выводе данных в нужные нам `Activity` и `Fragment`.

3.6 Отображение постов

Посты в приложении отображаются как уже упоминалось ранее не только на основной странице, а также на страницах по категориям, а также на стране пользователя, как для самого пользователя, так и когда другие пользователи хотят посмотреть посты интересующего их пользователя. В о взаимодействии сервера с приложением уже упоминалось что запросы для вывода постов имеют схожую структуру, но выводят нужные нам данные именно поэтому на стороне приложения, для каждого фрагмента, которые выводят посты была реализована очень схожая между собой функциональность.

В каждый созданный ранее фрагмент был добавлен схожий код для вывода постов. Данный код будет рассмотрен на примере фрагмента HomeFragment. Сам файл fragment_home.xml содержит в себе лишь два элемента recyclerView и Button, первый для отображения постов и второй для их добавления. Именно поэтому в HomeFragment была прописана логика для этих двух элементов.

Для кнопки добавления постов был создан onClickListener и определена логика, что при нажатии на данную кнопку пользователь будет переходит на страницу для добавления нового поста. Вывод же постов выполняется в несколько шагов, сначала был создан массив из элементов класса Item, а также объект класса адаптер, и был объявлен элемент RecyclerView, который используется для отображения постов. В адаптер передается объект класса Item, и далее он устанавливает в качестве адаптера для объявленного ранее recyclerView, как представлено в фрагменте кода ниже.

```
adapter = new MyPostAdapterClass(getActivity(), items);

recyclerView.setLayoutManager(new LinearLayoutManager(getActivity()));

recyclerView.setAdapter(new
MyPostAdapterClass(getActivity().getApplicationContext(), items));
```

После этого создается StringRequest к базе данных с методом GET в который передается путь к скрипту url+"login-registration-android/post_get.php", url определена ранее и содержит api адрес устройства. Далее создается массив JSONObject в который записывается массив, полученный из базы данных путем присваивания к соответствующим переменным объектов itemsObject.getString(название переменной, которая передается из базы данных). Полученными элементами затем заполняется лист объектов Item и по окончании заполнения массива эти объекты передаются созданному ранее адаптеру и объекту recyclerView, как представлено ниже.

```
Item item = new Item(name, email, image, category, state, postDescription, postImage, dateTime,
postId);

items.add(item);

adapter = new MyPostAdapterClass(getActivity(), items);
```

```
recyclerView.setAdapter(adapter);
```

Затем при помощи метода `Collections.sort` выполняем сортировку по дате публикации поста. Страницы с постами по категориям реализованы схожим образом, лишь в случае с постами пользователя выполняется запрос `POST` и в базу данных отправляется почта пользователя, с использованием которой производится запрос с условием `where email = почта пользователя`, пользующегося приложением.

3.7 Профиль пользователя

Как упоминалось ранее было создано два дизайна профиля пользователя, один для пользователя, вошедшего в систему и один который отображается для других пользователей. Профиль, который видят другие пользователи отличается лишь отсутствием опции перехода на страницу с личными данными пользователя, а также кнопки для выхода из приложения, именно поэтому будет описана реализация профиля пользователя, вошедшего в систему.

На данной странице выводятся имя, почта и фото пользователя напрямую из базы данных. Поэтому был создан запрос к базе данных для вывода информации пользователя. Для этого создан `GET` запрос, ответ которого записывается напрямую в элементы на странице пользователя, как показано ниже.

```
textViewName.setText(name);
```

```
textViewEmail.setText(email);
```

```
imageViewProfilePic.setImageResource(getResources().getIdentifier(image, "drawable", getActivity().getPackageName()));
```

Для таких элементов как `Posts`, `Profile info` и `Reviews`, были определены `onClickListener`, который при нажатии на элемент открывает соответствующую страницу. Также был создан `onClickListener` для кнопки `Logout`. Как и в случае с выходом из приложения из бокового меню выводится всплывающее окно с подтверждением выхода пользователя из приложения, если пользователь нажимает “No”, всплывающее окно закрывается, и пользователь остается на странице профиля. Если он нажимает “Yes” отправляется `POST` запрос к базе данных и отправляется почта пользователя, который запрашивает выход из приложения. И в случае успешного выполнения скрипта устанавливает значение `shared preferences` как пустую строку, как показано в коде ниже.

```
if(response.equals("success")) {
```

```
    SharedPreferences.Editor editor = sharedPreferences.edit();
```

```
    editor.putString("logged", "");
```

```
    editor.putString("name", "");
```



```

editor.putString("email", "");

editor.putString("apiKey", "");

editor.apply();

Intent intent = new Intent(getActivity().getApplicationContext(), Login.class);

startActivity(intent);

getActivity().finish();

```

Пользователь может открыть страницы с постами, которые он выложил, с отзывами о нем, а также с информацией о пользователе. Ниже представлен фрагмент кода с `onClickListener` для перехода к отзывам о пользователе. Как видно из фрагмента выше для перехода к странице с отзывами пользователей создается `Intent` с соответствующим фрагментом и используется опция `putExtra` для передачи данных, первый для отправления почты пользователя, а второй для определения что это именно текущий пользователь открыл страницу. В дальнейшем этот параметр используется на странице отзывов в качестве условия.

```

profileReviews=view.findViewById(R.id.reviews);

profileReviews.setOnClickListener(new View.OnClickListener() {

@Override public void onClick(View v) {

Intent intentReviews = new Intent(getActivity(),ProfileForOtherUsersActivity.class);

intentReviews.putExtra("userEmail", sharedPreferences.getString("email",""));

intentReviews.putExtra("myprofile","aaaa");

startActivity(intentReviews);

}

});

```

При переходе на страницу отзывов пользователь сможет лишь посмотреть отзывы о себе, если это его страница или добавить отзыв к странице другого пользователя. На странице с постами пользователя есть кнопка добавить пост, а на странице, где они выводятся для других пользователей такой опции нет.

При нажатии на кнопку `Personal info` пользователь переходит на страницу, где отображаются его текущие данные. Напротив каждой информации находится кнопка `change` для ее изменения. Данные выводятся на страницу при помощи очередного `GET` запроса. Как упоминалось ранее при переходе из профиля пользователя на страницу с отзывами мы отправляем при помощи параметра `putExtra` почту пользователя и другую необходимую нам информацию, на страницу с

персональными данными мы также отправляем почту пользователя. Она используется в POST запросе при отображении данных пользователя.

Далее созданы `onClickListeners` для каждой кнопки `change`, в которых при помощи элемента `AlertDialog.Builder` создается всплывающее окно для пользователя с предупреждением об изменении данных как представлено ниже для изменения имени пользователя.

```
builderName.setTitle("Name change confirmation");  
  
builderName.setMessage("Are you sure you want to change your name?");
```

Затем созданы два `onClickListeners` который в зависимости от ответа пользователя выполняют соответствующие действия. При нажатии на `Yes` совершается переход на страницу с полем, в котором пользователь может увидеть текущие данные, а также изменить их при желании. При нажатии на `No` диалоговое окно просто закроется, и пользователь останется на странице с персональной информацией.

```
builderName.setNegativeButton("No", (dialog, which) -> dialog.dismiss());  
  
AlertDialog alert = builderName.create();  
  
//Setting the title manually  
  
alert.setTitle("Name change confirmation");  
  
alert.show();
```

После нажатия на эту `Yes`, что пользователь хочет изменить данные выполняется вызов соответствующего фрагмента, который содержит в себе два метода POST, первый для вывода данных пользователя, это сделано для того, чтобы пользователю не пришлось вводить свои данные заново, если он, например хочет изменить лишь одну букву. При загрузке данных из базы данных значение в поле для изменения данных сразу заполняется нужной информацией.

```
textInputEditTextChangeName.setText(name);
```

Далее были созданы `onClickListeners` для двух кнопок `Cancel` и `OK`, нажав на первую пользователь может вернуться на предыдущую страницу и данные останутся неизменными даже если он уже внес какие-либо изменения. После внесения изменений пользователь нажимает на кнопку `OK` повторно выводится всплывающее окно с подтверждением об изменении информации и выполняется второй POST запрос. При изменении имени пользователя или его изображения нет ограничений по данным, которые может вводить пользователь, однако при вводе почты она отправляется на проверку и в случае, если такая уже зарегистрирована отправляет соответствующее сообщение.

```
else Toast.makeText(getActivity().getApplicationContext(), "This email already exists, please  
enter another email", Toast.LENGTH_SHORT).show();
```

В противном случае если почта уникальна происходит ее изменение, и затем выполняется обратный переход на страницу с информацией пользователя, на которой отображаются новые данные. Пользователь может менять свои данные неограниченное количество раз, и так как для каждого поля установлены кнопки change он может изменять именно те данные которые захочет пользователь при этом не затрагивая остальные.

3.8 Публикация и изменение постов

Для публикации и изменения постов был создан класс AddEditPost, который принимает id поста в случае, если это уже существующий пост и создает новый если id не был передан. Как уже упоминалось ранее для работы с постами, отзывами и комментариями к посту были созданы дополнительные классы, одним из которых является адаптер, в котором прописаны условия для меню удаление и изменение поста. Удаление поста прописано в самом классе адаптере и было описано ранее, если же пользователь нажмет на опцию изменения поста откроется страница класса AddEditPost.

Было создано условие if для id поста, в случае, когда пользователь просто хочет добавить новый пост id поста будет пустым, и будет выполняться условие для данного случая. На странице для добавления поста пользователь увидит перед собой пустые поля для заполнения. Он может выбрать категорию и состояние поста используя элемент Spinner, для которого в классе AddEditPost был определен массив с заранее заданными значениями.

```
String[] PostCategory = new String[]{"Animals", "Medicine", "Food", "Nature", "Children"};  
String[] PostState = new String[]{"Need help", "Offer help"};
```

Данные массивы затем были отправлены в качестве параметра адаптера для каждого элемента Spinner, поэтому при нажатии на данный элемент пользователь увидит перед собой заранее определенные опции и сможет выбрать подходящую. Далее значения каждого Spinner будет присвоено переменной класса AddEditPost также как и значение текста введенного в качестве описания поста

```
String PostCategoryString = spinnerCategory.getSelectedItem().toString();  
String PostDescriptionString = String.valueOf(postDescription.getText());
```

Если с текстом и Spinner все проще, то в случае с изображением его загрузка выполняется в несколько шагов. Первым шагом был определен onClickListener для загрузки изображения из

памяти телефона. Затем это изображение преобразуется в байтовый массив, и преобразуется при помощи Base64.

После успешного введения всех данных создается POST запрос к базе данных, в который отправляются все введенные данные, изображение сохранится в отдельной папке, а его название, как и остальные данные о посте сохранятся напрямую в базу данных.

В случае изменения поста сам процесс загрузки поста в базу данных ничем не отличается, кроме как тем, что помимо основных данных в базу данных в качестве параметра передается id изменяемого поста. Также перед самой загрузкой измененных данных создается запрос POST в базу данных в который передается только id пользователя, и который выводит данные неизменного поста, включая изображение, описание поста и т.д. Пользователь может изменить лишь один параметр и загрузить измененный пост в базу данных.

3.9 Публикация и изменение комментариев

Пользователь может добавить комментарий как к чужому посту, так и к своему. Для этого ему нужно нажать на описание или изображение поста, комментарий к которому он хочет добавить. Логика для перехода к странице с комментариями как ранее уже упоминалось определена в адаптере для постов. При нажатии на пост перед пользователем открывается страница с уже существующими комментариями, либо если их нет с пустой страницей с надписью Comments и кнопкой внизу для добавления комментариев.

На странице с комментариями определена логика для двух элементов, это RecyclerView для вывода постов из базы данных и кнопка для их добавления. Для публикации комментариев, как и в случае с постами были созданы дополнительные классы, такие как ItemComment и MyPostCommentsAdapter, которые используются на данной странице. Первым шагом создается List с объектами класса ItemComment а также адаптер, далее создается POST запрос в базу данных с использованием id поста к которому данные комментарии относятся. Затем этими данными заполняется ранее созданный List объектов ItemComment который затем передается в качестве параметра адаптеру.

Далее определен onClickLitener для кнопки добавления комментария, которая при нажатии открывает пользователю страницу для добавления комментария, дизайн которой был продемонстрирован ранее. Комментарий загружается в базу данных с использованием POST запроса, но помимо самого комментария в качестве параметра в базу данных также отправляется id поста, к которому этот комментарий относится.

Если же пользователь хочет изменить ранее созданный комментарий, он может нажать на боковое меню с опцией удаления или изменения нужного комментария. Причем изменить или удалить пользователь может только свой комментарий, а не чужой. Логика для удаления поста определена в классе адаптере, также там определен переход на страницу с добавлением/изменением комментария, с дополнительным параметром в качестве id комментария. В случае если пользователь пытается изменить или удалить комментарий, добавленный другим пользователем, он увидит соответствующее сообщение о невозможности выполнения этой операции и будет возвращен на страницу с комментариями.

Как и в случае с изменением поста, для изменения комментария создается два POST запроса, первый для отображения текущего комментария и второй для внесения изменений в базу данных. Первый запрос отправляет в базу данных в качестве параметра id комментария и возвращает соответствующий ему комментарий, который пользователь впоследствии может изменить.

3.10 Публикация и изменение отзывов о пользователе

Как уже упоминалось ранее в адаптере для поста также прописана логика для перемещения на профиль пользователя, выложившего пост, для этого определен onClickListener для контейнера, в котором хранятся фото, имя и поста пользователя. После нажатия на этот контейнер пользователю открывается пост того, кто выложил интересующий его пост. На данной странице есть две опции первая посмотреть отзывы о пользователе и вторая посмотреть посты, выложенные пользователем. Реализация страниц для вывода постов была описана ранее, так что сейчас речь пойдет о странице с отзывами.

На данной странице, как и в случае с комментариями созданы лишь два элемента recyclerView и кнопка для добавления отзыва. Для вывода постов в элемент RecyclerView создается List с объектами ItemReview а также соответствующий адаптер для отзывов. И выполняется запрос в базу данных, в качестве параметра в который отправляется почта пользователя, отзывы о котором нужно открыть.

При нажатии на кнопку для добавления поста открывается поле для введения комментария, пользователь может ввести свой отзыв и при нажатии на кнопку ОК будет выполнен запрос к базе данных в качестве параметров которого, отправляются поста пользователя и текст отзыва, также пользователю выводится сообщение об успешном добавлении поста. Если же он по какой-то причине передумал оставлять отзыв он может нажать на кнопку Cancel и вернуться на основную страницу. Также было добавлено условие что пользователь не может оставить отзыв о самом себе,

при попытке это сделать пользователь увидит соответствующее сообщение и никаких действий не будет предпринято.

В случае же если пользователь хочет изменить или удалить добавленный ранее отзыв, как и в случае с комментариями эта логика прописана в классе адаптере для отзыва. В случае удаления поста пользователь увидит всплывающее окно с подтверждением удаления, и в случае попытки удаления поста чужого пользователя это окно закроется и выведется соответствующее предупреждение о невозможности удаления чужого отзыва. Если же пользователь нажмет на опцию изменения своего отзыва, ему откроется соответствующая страница с полем в котором будет выведен изначальный отзыв, который можно будет изменить. При попытке изменить чужой отзыв пользователь также увидит перед собой соответствующее сообщение о невозможности изменения чужого отзыва.

3.11 Тестирование информационной системы

Тестирование приложения является не менее важной частью процесса разработки чем создание дизайна, или серверной или backend частей приложения. Оно позволяет оценить работу приложения в различных сценариях, чтобы определить реагирует ли приложение так, как от него ожидает разработчик и в случае возникновения непредвиденной реакции приложения вносятся пометки об их исправлении. [9]

В процессе работы над приложением для его тестирования было создано три автоматических теста которые оценивают правильную работу программы при отправлении данных в базу данных, а также при работе с изображением и комментарием, который были выполнены успешно, результаты тестирования представлены на рисунке 3.14.



Рисунок 3.14 – Результаты тестирования

Но помимо этих основных тестов было проведено также ручное тестирование для оценки работы приложения. Оно было проведено в несколько этапов, пока все неисправности не были исправлены.

Далее будут представлены результаты первого и последнего этапа ручного тестирования различных процессов работы приложения с выводом экрана. Первым шагом был протестирован вход пользователя в приложение, при первом этапе тестирования было выявлено, что в случае неправильно введенных данных или их полного или частичного отсутствия выводилось сообщение из базы данных с кодом ошибки, при этом кнопка для входа смещалась за пределы видимости пользователя. В качестве решения данной проблемы был изменен php скрипт, так, чтобы в зависимости от результата выполнения он возвращал вместе с статусом запроса и сообщение с подсказкой для пользователя, как представлено ниже.

```
"status"=> "failed", "message"=> "Retry with correct email and password"
```

Далее был проведен финальный тест для проверки правильного функционирования входа. Ниже на рисунке 3.15 представлены скрины приложения при входе в него при различных сценариях.

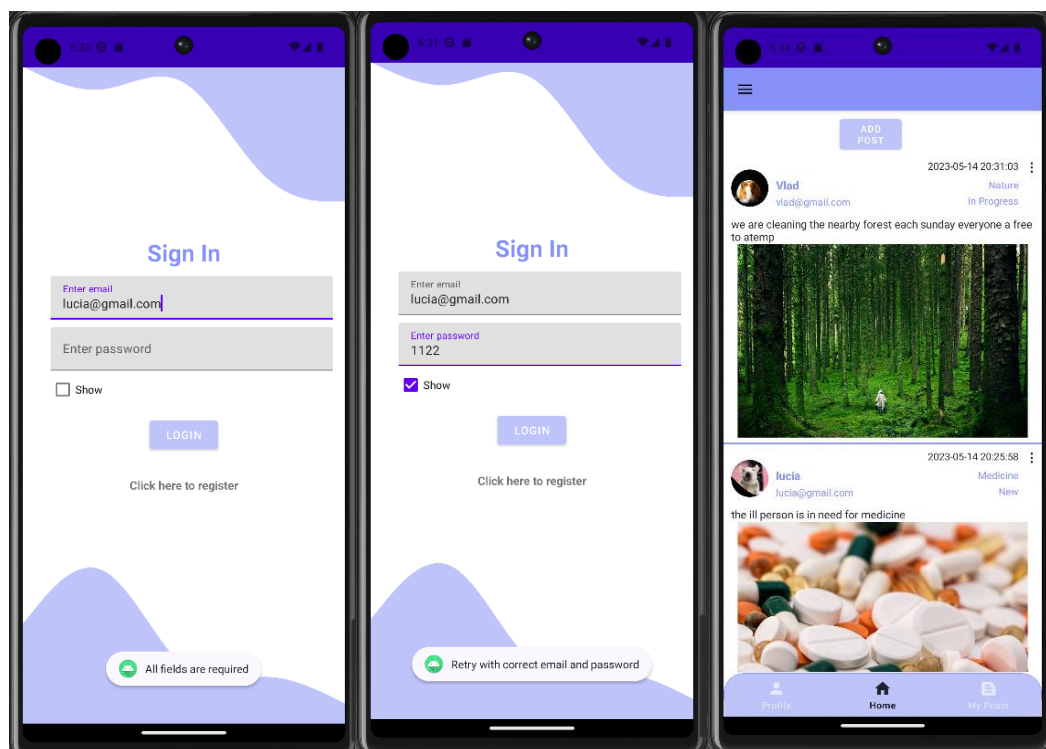


Рисунок 3.15 – Результаты ручного тестирования входа в приложение

В первом случае пользователь не ввел пароль, такой-же результат был получен в случае, когда пользователь не ввел почту или вообще не ввел никакие данные и нажал на кнопку входа. Во втором случае пользователь ввел неправильный пароль, такой-же результат был получен при вводе неправильной почты и правильного пароля, а также неправильного пароля и почты. И в последнем

случае все данные были введены правильно и пользователю открылась домашняя страница с существующими постами.

Далее было проведено тестирование корректного функционирования добавления, изменения и удаления отзыва о пользователе, при первом тестировании было выявлено что пользователь может добавить отзыв о самом себе, то есть результат был не тот, что был ожидаем. Так что были внесены изменения опять же в код php было добавлено условие что пользователь, который хочет добавить отзыв отличается от пользователя, которому отзыв предназначен. Изначально, пользователем, вошедшим в систему, является lucia@gmail.com, и вот результаты попыток добавления, изменения и удаления отзыва, представленные на рисунке 3.16.

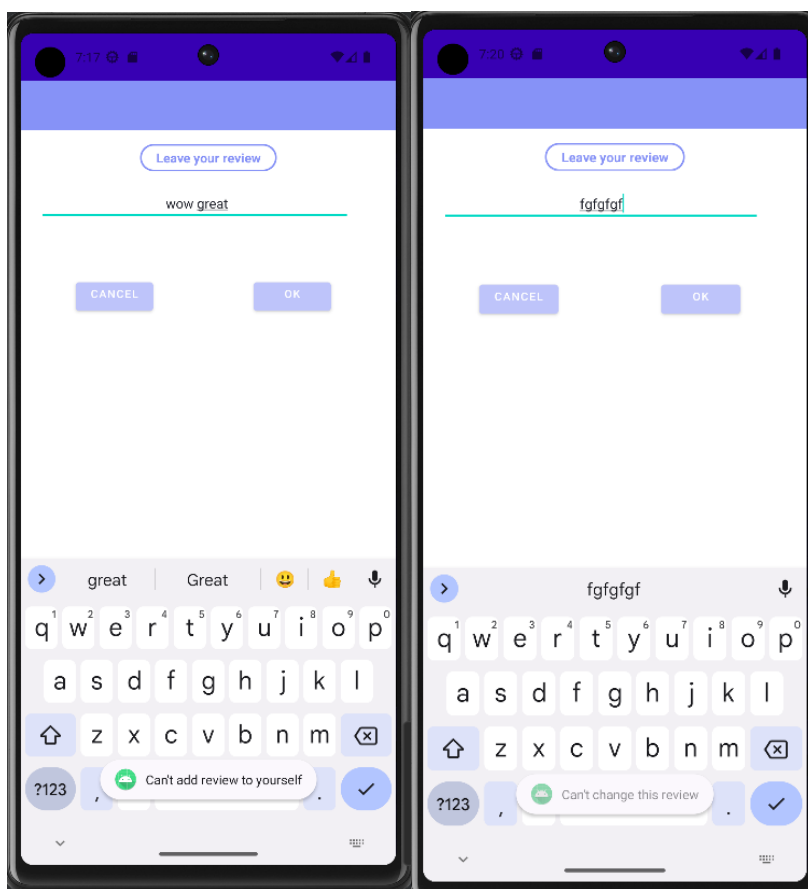


Рисунок 3.16 – Результаты проверки корректной работы с отзывами

В случае же если пользователь взаимодействует с созданным им же отзывом пользователю лишь выходят сообщения об успешном выполнении добавления, изменения или удаления поста, и выполняются соответствующие действия.

4 СТОИМОСТНАЯ ОЦЕНКА СИСТЕМЫ

Оценка стоимости проекта является не менее важной частью чем его документация, она позволяет получить правильную оценку затраченных усилий и ресурсов на создание проекта в совокупности с временем, затраченным на реализацию проекта

Одним из лучших способов для отслеживания временных рамок необходимых для реализации каждого этапа проекта является диаграмма Ганта, которая позволяет наглядно рассмотреть временные рамки для каждого этапа работы. Для ее создания были подробно рассмотрены этапы создания информационной системы, с временем затраченным на создание, а также датой начала этапа и его конца.[10]

Номер этапа	Название этапа	Продолжительность	Начало	Конец
1	Разработка информационной системы для организации работы волонтеров	178	07/09/2022	15/05/2023
1.1	Анализ предметной области. Постановка задачи.	14	07/09/2022	26/09/2022
1.1.1	Важность темы и существующие аналоги	7	26/09/2022	15/09/2022
1.1.2	Цель, задачи и требования к системе	7	16/09/2022	26/09/2022
1.2	Моделирование и проектирование информационной системы	40	27/09/2022	21/11/2022
1.3	Реализация информационной системы	113	22/11/2022	30/04/2023
1.3.1	Создание дизайна	22	22/11/2022	12/12/2022
1.3.2	Создание базы данных	1	13/12/2022	14/12/2022
1.3.3	Разработка серверной части	30	14/12/2022	25/01/2023
1.3.4	Разработка backend	60	26/01/2023	30/04/2023
1.4	Тестирование информационной системы	11	25/03/2023	10/04/2023
1.5	Стоимостная оценка системы	9	10/04/2023	20/04/2023
1.6	Документирование	19	21/04/2023	15/05/2023

На основе данной таблицы была создана диаграмма Ганта с наглядным представлением, времени затраченного на выполнение каждого этапа. Диаграмма представлена на рисунке 3.17.

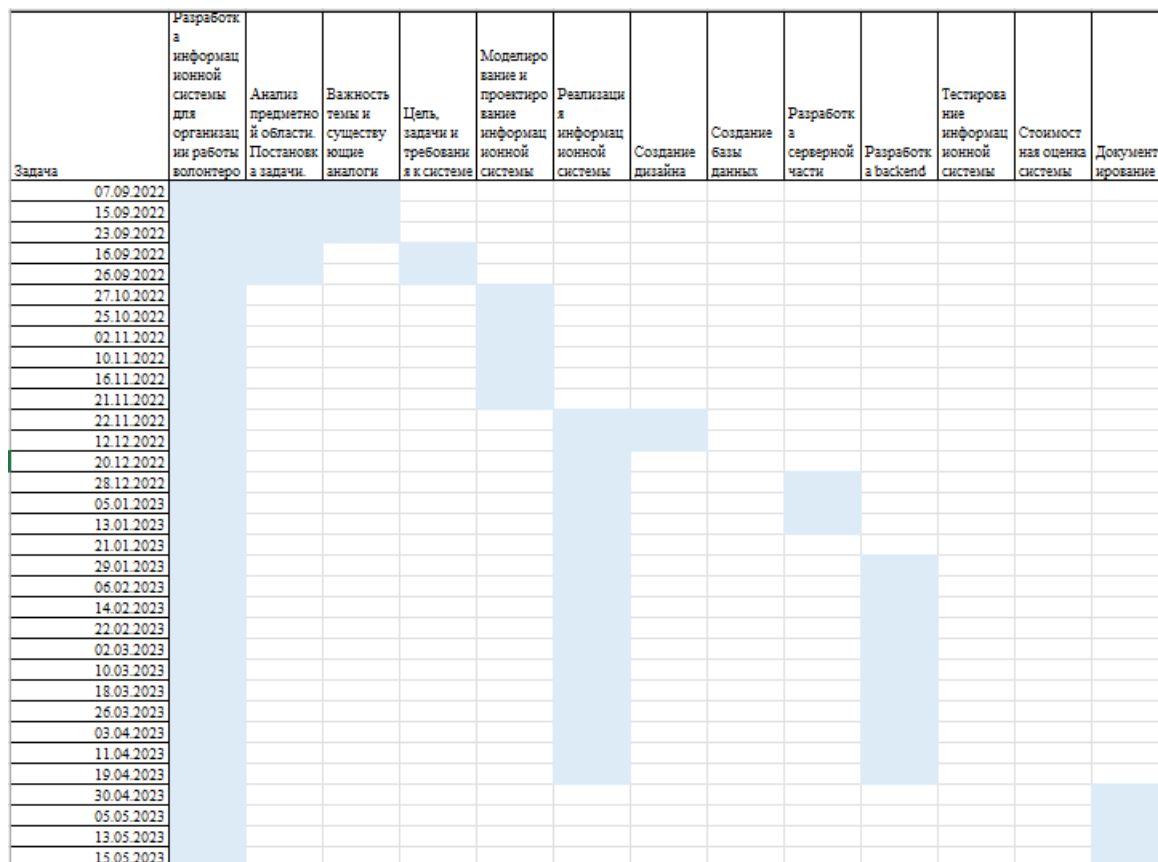


Рисунок 3.17 – Диаграмма Ганта

Далее на основе полученных данных о времени выполнения проекта, а также при подсчёте сторонних ресурсов была посчитана стоимость проекта, а также создана диаграмма с подробным распределением времени и финансов, потраченных на каждый этап работы.

В сумме на всю работу было потрачено 99 959 лей, с учетом выдачи заработной платы работникам с учетом что один рабочий день равняется 6 часам. Всего было затрачено 178 дней, то есть в сумме 1068 рабочих часа.

Затраты на электричество с учётом тарифа 1.73 лея за 1КВ/ч составили:

$q = N * W * t$, где

- q – расчет затрат на электричество;
- W – мощность одного компьютера, кв;
- N – количество компьютеров;
- t – время работы = 1068 ч;
- T – тариф на электроэнергию.

$$q = 4 \cdot 0.045_{\text{кв/ч}} \cdot 1068 = 192.24$$

$$\Xi = T \cdot q = 1,73 \cdot 192.24 = 332,58 \text{ лея}$$

Финальная оценочная стоимость проекта равно $999.959(\text{леев}) + 332.58(\text{леев}) = 100\,291.58(\text{леев})$.

График оценочной стоимости для с учетом выдачи заработной платы работникам представлен на рисунке 3.18.

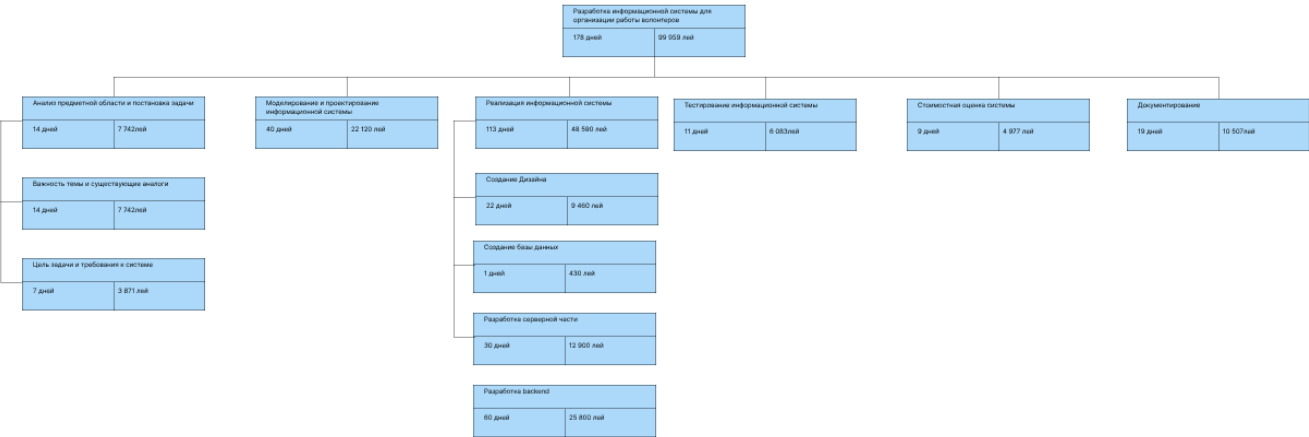


Рисунок 3.8 – Оценочная стоимость проекта

5 ДОКУМЕНТИРОВАНИЕ

Документирование системы очень важный и завершающий этап работы над приложением, оно подробно описывает этапы работы пользователя с приложением и его поведение в различных сценариях и позволяет определить сценарии при которых система ведет себя непредвиденно, и впоследствии исправить недочеты. В процессе разработки приложение могут применяться различные методы тестирования, причем каждый из них может повторяться, до тех пор, пока неисправности в работе приложения не будут исправлены.

При входе в приложение пользователю открывается страница для входа, если пользователь является новым он может зарегистрироваться в приложении нажав на надпись “Click here to register”. При этом откроется страница для ввода регистрационных данных, а именно имени пользователя, почты, пароля и изображения, как показано на рисунке 5.1.

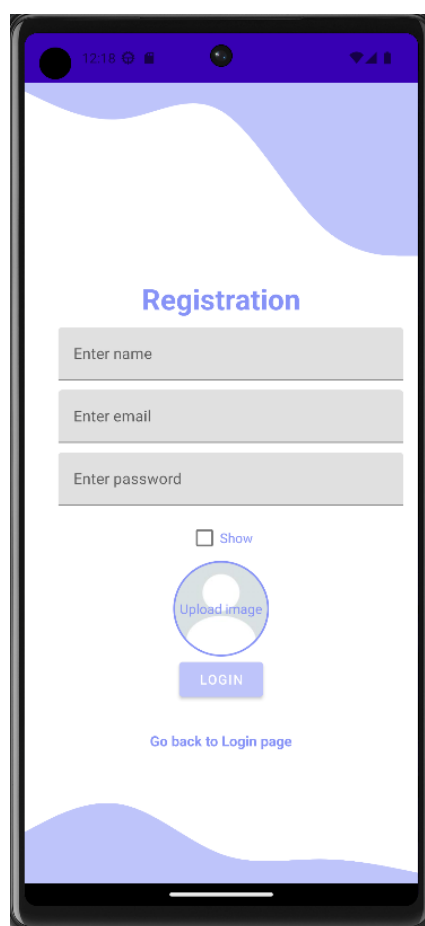


Рисунок 5.1 – Страница регистрации пользователя

Пароль при этом будет скрыт и при желании сделать его видимым пользователь может нажать на опцию Show под полем с паролем, если же хочет его вновь скрыть требуется повторное нажатие на данную опцию. Пользователь может внести свои данные нажав на соответствующее

поле. При нажатии на изображение пользователю откроется память устройства, и он может выбрать подходящее изображение, а также изменить его, повторно нажав на поле для ввода изображения. На данной странице присутствуют кнопка Login и также надпись “Go back to Login page” в случае, если пользователь случайно зашел на страницу регистрации, нажав на нее пользователь вернется на страницу входа в приложение. Если пользователь не ввел какие-либо данные при нажатии на кнопку “OK” он увидит сообщение “All fields are required” и сможет добавить нужные данные, если же пользователь с такой почтой уже существует отобразится соответствующее сообщение, и пользователь сможет поменять свою почту. Если же пользователь ввел все данные корректно, а также его почта является уникальной он увидит сообщение об успешной регистрации и будет направлен на страницу для входа в приложение.

Если пользователь уже зарегистрирован он может зайти в приложение со страницы входа, которая открывается при запуске приложения, либо если пользователь перешел случайно на страницу регистрации он может нажать “Go back to login page” и вернуться к странице входа, представленной на рисунке 5.2.

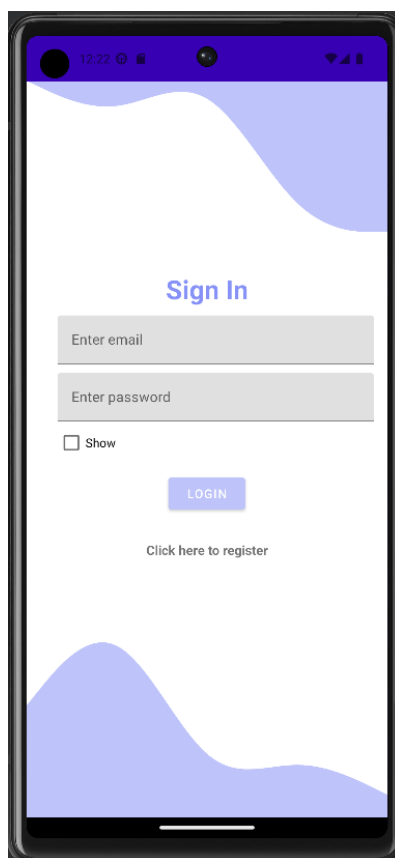


Рисунок 5.2 – Страница входа в приложение

Для входа в приложение пользователю нужно ввести свою почту и пароль, как и в случае с регистрацией пароль по умолчанию скрыт и для его отображения пользователь может нажать на

опцию Show под полем с паролем, повторное нажатие приведет к скрытию пароля. Если пользователь не заполнил все поля при нажатии на Login он увидит сообщение “All fields are required”, а если какие-то данные не верны он увидит сообщение “Retry with correct email and password”. Если же все данные введены правильно пользователь будет направлен на домашнюю страницу.

Навигация в приложении осуществляется за счет бокового и нижнего меню представленных на рисунке 5.3. Нижнее меню видно пользователю по умолчанию, а боковое скрыто и может быть открыто при нажатии пользователем на иконку меню в левом верхнем углу экрана, и закрыто либо при нажатии на элемент этого меню, либо на любую свободную часть экрана.

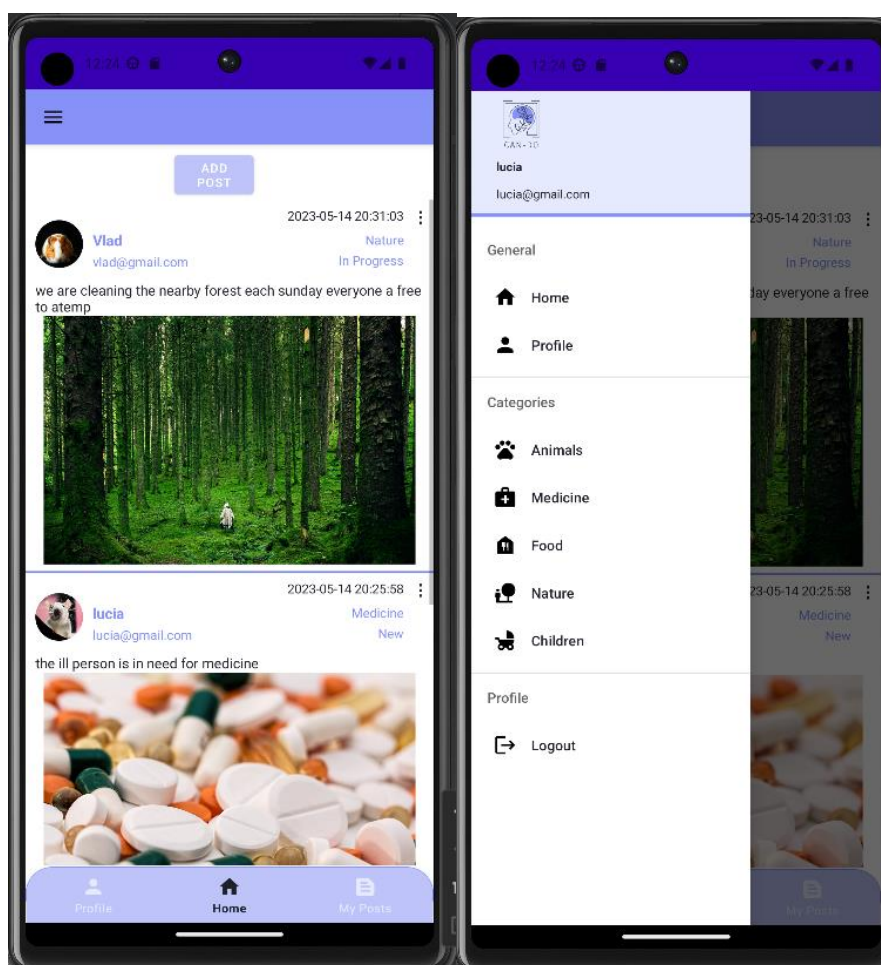


Рисунок 5.3 – Навигация в приложении

Используя меню внизу страницы, пользователь может перемещаться на три разные страницы, такие как домашняя, мои посты и страница профиля. Боковое меню помимо уже перечисленных опций также содержит страницы с категориями постов такими как животные, дети, природа, медицина, еда. Пользователь может сколько угодно раз перемещаться между нужными ему страницами нажимая на соответствующие пункты меню. Также боковое меню содержит опцию

выхода из приложения нажав на которую будет выполнен выход из приложения и пользователю откроется страница для входа в приложение.

Помимо выше перечисленных страниц пользователь также может перейти на страницу пользователя опубликовавшего пост, для этого он может нажать на одно из трех: на фото пользователя, его имя или почту, при этом он будет направлен на страницу содержащую две опции: страницу с отзывами пользователя и страницу с постами пользователя, нажав на которые он будет перенаправлен на соответствующие страницы, как представлено на рисунке 5.4. Вернуться обратно он сможет при помощи кнопки «Обратно» которая присутствует на всех устройствах.

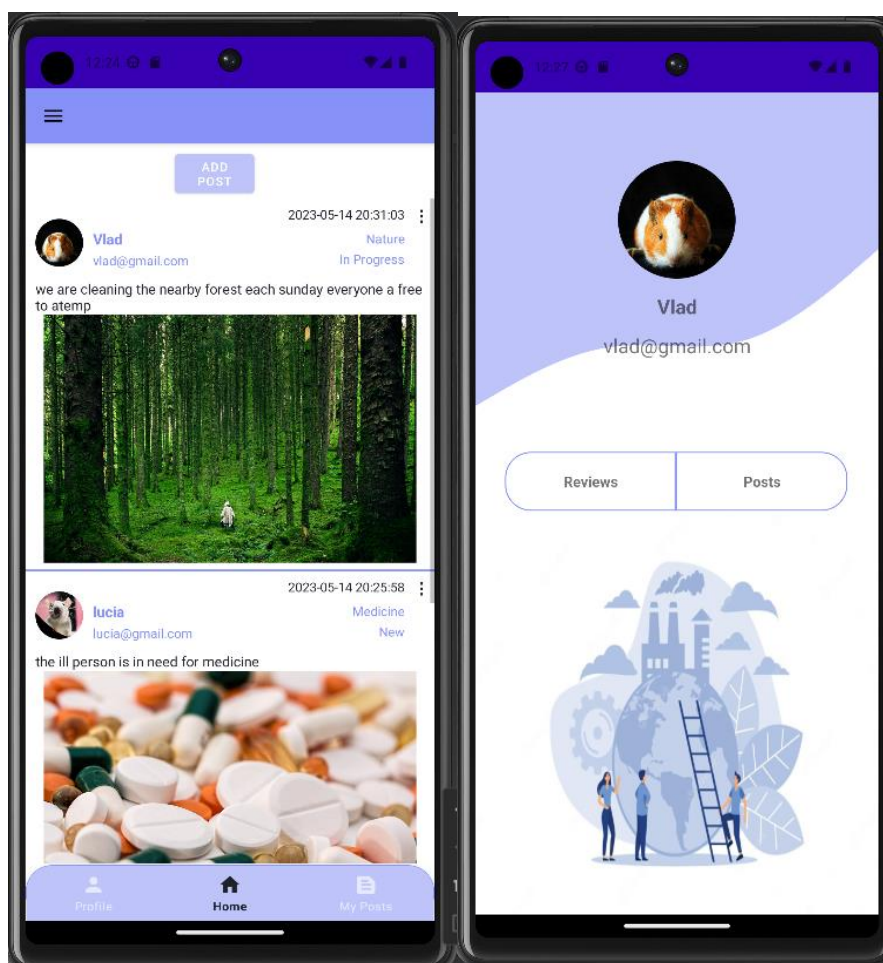


Рисунок 5.4 – Профиль пользователя, выложившего пост

Также есть страницы с комментариями и отзывами. Страницу с комментариями к посту пользователь может открыть, нажав на соответствующий пост. А страницу с отзывами о себе он может открыть, зайдя в свой профиль и нажав на соответствующую опцию. Как открыть отзывы о другом пользователе было описано выше.

И на страницу с информацией о пользователе, для пользователя, вошедшего в систему, он может перейти, нажав на страницу профиля в нижнем или боковом меню и выбрав опцию Profile

info, нажав на которую он будет направлен на страницу с информацией о профиле, где он сможет изменять данные, как представлено на рисунке 5.5.

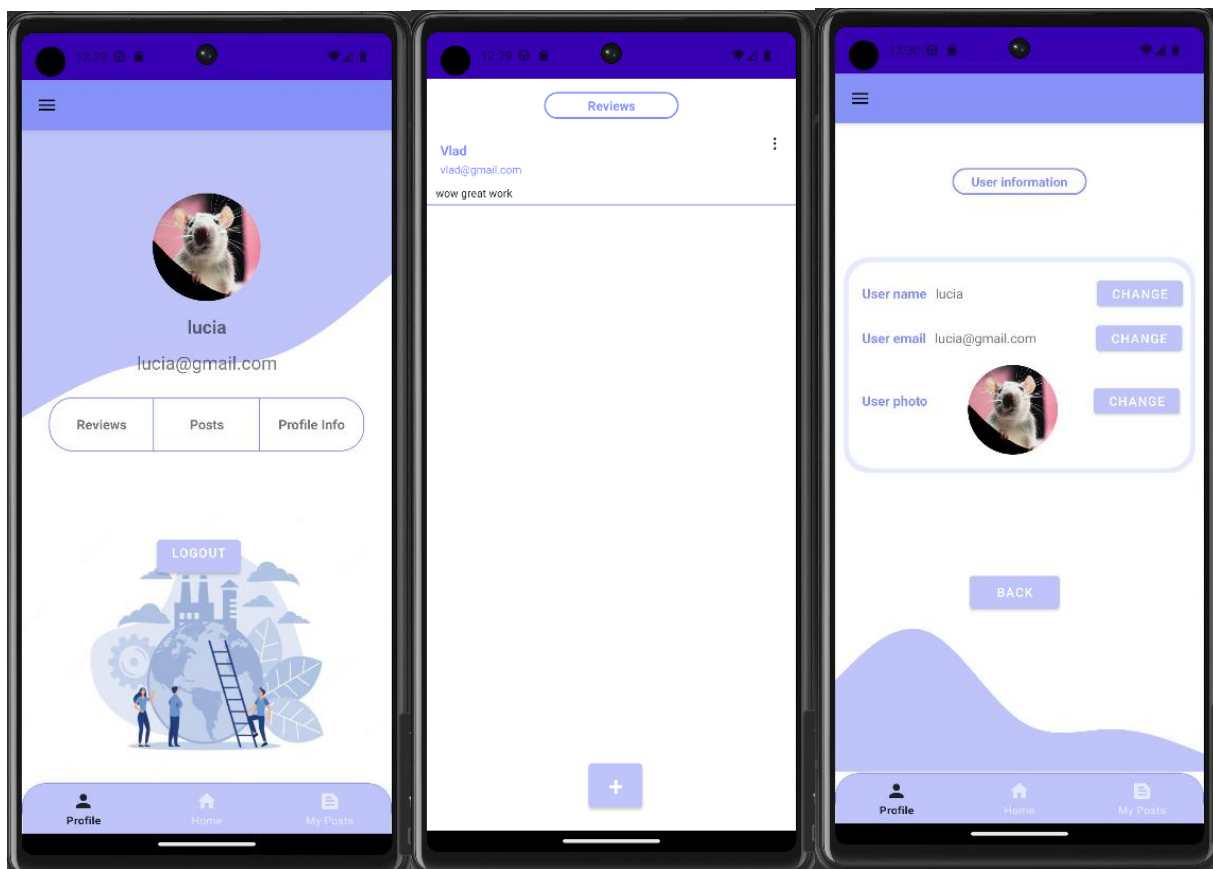


Рисунок 5.5 – Профиль пользователя, отзывы и персональная информация

Посты пользователей отображаются на 8 разных страницах: домашняя, животные, природа, еда, дети, медицина, посты пользователя, вошедшего в систему и посты пользователя на страницу которого перешел пользователь, вошедший в систему. Все операции, связанные с изменением, удалением и добавлением постов, а также с добавлением изменением или удалением комментариев могут быть выполнены с любой из этих страниц, кроме страницы пользователя на которую вошел текущий пользователь, на данной странице опция добавления поста отсутствует. Посты на этих страницах выводятся абсолютно одинаково и отличаются только по категориям, на домашней странице выводятся все посты, на страницах категорий выводятся только посты соответствующей категории и на страницах постов пользователей выводятся лишь посты, опубликованные конкретным пользователем. Все посты отсортированы по дате добавления от более новым к более старым.

Как упоминалось в предыдущем пункте добавление постов осуществляется с любой из 7 страниц, на которых опубликованы посты. Пользователь может добавить пост нажав на кнопку “Add post” находящейся в верхней части экрана страницы с опубликованными постами. При

нажатии на нее пользователю откроется окно для добавления поста как представлено на рисунке 5.6, где он может выбрать категорию и состояния поста нажав на соответствующий элемент, а также ввести описание и нажав на поле с изображением, как и в случае регистрации он будет направлен к изображениям хранящимся в памяти телефона и сможет выбрать желаемое, после чего будет возвращён на страницу добавления поста, пока не нажата кнопка ОК пользователь может изменять введенную информацию сколько угодно раз. Если пользователь не хочет публиковать пост он может вернуться на предыдущую страницу нажав на кнопку Cancel. Если же он нажимает на кнопку ОК, при этом не заполнив какое-то из полей он увидит соответствующее сообщение с предупреждением и после того, как все поля заполнены при нажатии на кнопку ОК пользователю откроется домашняя страница и он увидит сообщение об успешном добавлении поста.



5.6 – Страница добавления поста

Редактировать пост пользователь может, нажав на значок меню в правом верхнем углу поста и выбрать опцию change, в данном случае он увидит всплывающее окно с подтверждением того, что он хочет отредактировать пост. Если он нажмет на No, окно закроется, и он останется на той-же странице, если же нажмет на Yes перед ним откроется страница для редактирования поста. Если этот пост был создан другим пользователем перед желающим изменить пост пользователем предстанет страница для редактирования поста без введенных данных и при попытке ввода каких-либо данных и нажатии на ОК пользователь увидит сообщение о невозможности изменения чужого

поста. Он также может нажать на кнопку Cancel и вернуться к предыдущей странице. Если же этот пост принадлежит самому пользователю перед ним откроется страница с заполненными полями с изначальными данными поста, как представлено на рисунке 5.7, он может изменять их по своему усмотрению, для отмены изменений он может нажать Cancel и будет возвращен на предыдущую страницу, при нажатии на ОК внесенные изменения сохранятся при условии что пользователь заполнил все поля, в противном случае он увидит соответствующее предупреждение и должен будет ввести недостающие данные, после чего будет перенаправлен на домашнюю страницу и увидит сообщение об успешном изменении поста.

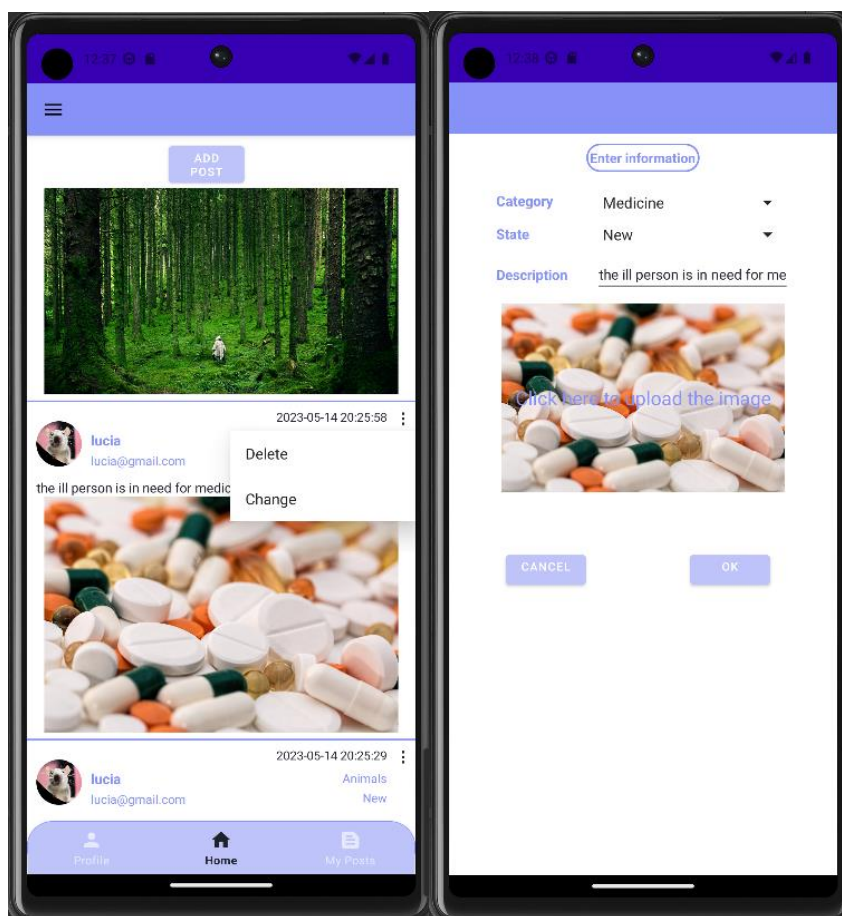


Рисунок 5.7 – Изменение поста

Для удаления поста пользователь должен нажать на боковое меню соответствующего поста в правом верхнем углу и выбрать опцию Delete, нажав на которую он увидит всплывающее окно с предупреждением об удалении поста. При нажатии на No, окно с предупреждением будет закрыто и никакие действия не будут предприниматься, если же пользователь нажмет на Yes, есть два варианта действий, если он не является пользователем создавшим пост, увидит сообщение с предупреждением о том что он не может удалить чужой пост и никакие действия не будут предприняты, если же это его пост он увидит сообщение с подтверждением удаления поста, как

представлено на рисунке 5.8 и будет направлен на домашнюю страницу с обновленными постами, а точнее без удаленного поста.

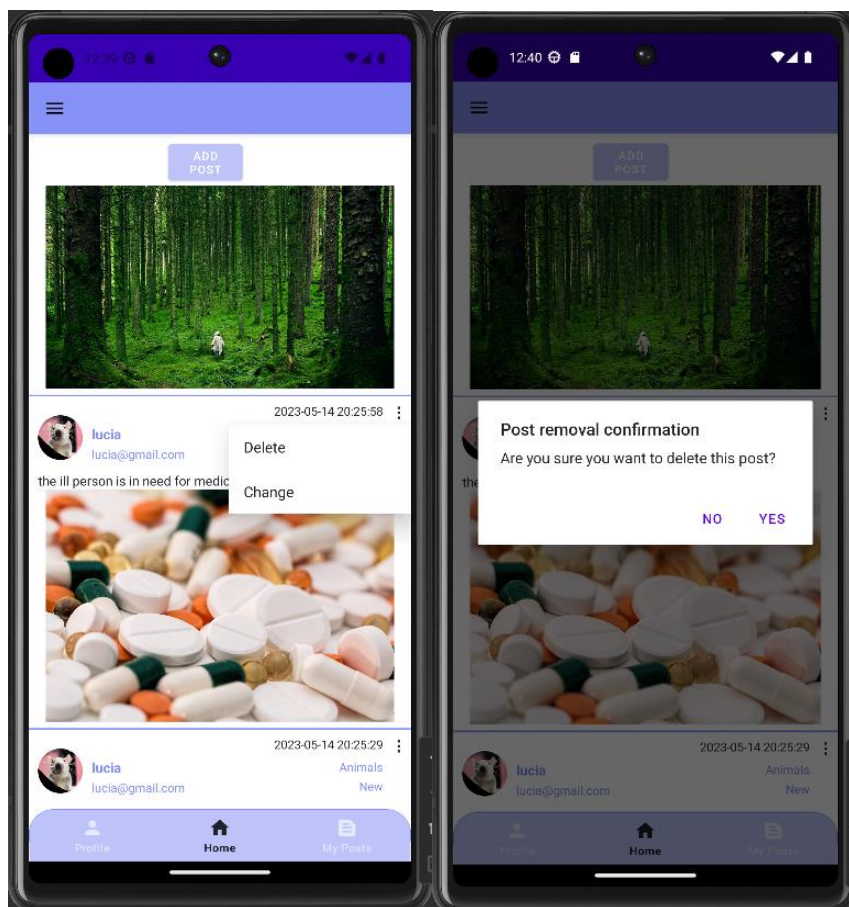


Рисунок 5.8 – Удаление поста

Для того чтобы открыть страницу с комментариями пользователю нужно нажать на описание или фото поста, комментарии к которому он хочет открыть. При нажатии на них пользователю откроется страница с комментариями, в случае же если комментарии отсутствуют он увидит пустую страницу с надписью вверху Comments и кнопкой с плюсом в нижней части экрана нажав на которую он может добавить свой пост, как представлено на рисунке 5.9.



Рисунок 5.9 – Пустая страница с комментариями

При нажатии на эту кнопку все что требуется от пользователя это ввести комментарий и для его публикации нажать на ОК, как представлено на рисунке 5.10, при этом он будет направлен на домашнюю страницу и увидит сообщение об успешном добавлении комментария. В случае если он решил его не добавлять он может нажать на кнопку Cancel. При нажатии на кнопку ОК если пользователь не вел комментарий он увидит соответствующее сообщение об ошибке.

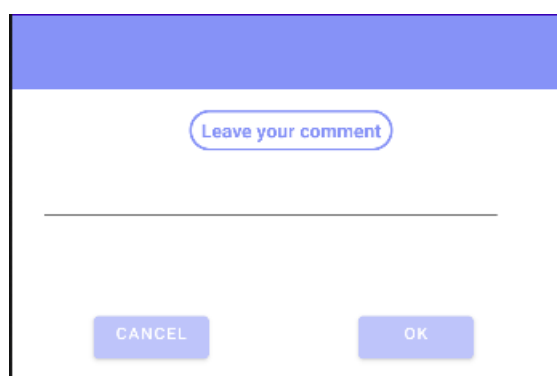


Рисунок 5.10 – Страница добавления комментария

Для изменения комментария, как и в случае с постом пользователю нужно нажать на значок меню в правом верхнем углу комментария, который он хочет изменить, и он увидит всплывающее окно с подтверждением его действий как представлено на рисунке 5.11 при нажатии на No окно закроется и никакие действия не будут предприняты, при нажатии на Yes и при условии что комментарий принадлежит этому пользователю будет направлен на страницу для его редактирования, в противном случае он увидит сообщение о том что он не может изменить чужой комментарий и никакие действия не будут выполнены.

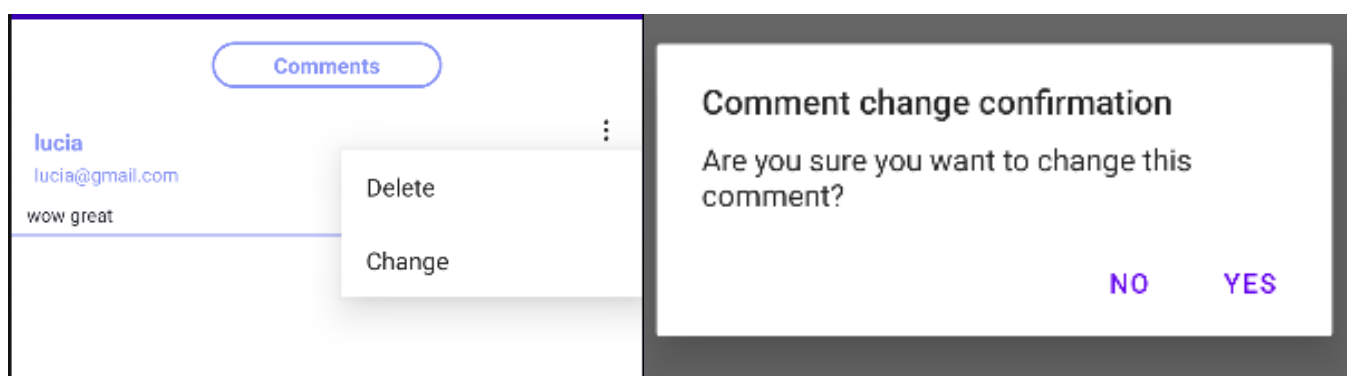


Рисунок 5.11 – Изменение комментария

На странице для изменения комментария пользователь увидит окно для редактирования с текущим комментарием, как представлено на рисунке 5.12, который он может изменить, при нажатии на кнопку Cancel изменения не будут сохранены, и пользователь будет возвращен на предыдущую страницу, при нажатии на ОК и условии что поле с комментарием не пустое, комментарий будет изменен, и пользователь получит соответствующее сообщение и будет направлен на домашнюю страницу, в противном случае, получит предупреждение с требованием ввести данные.



Рисунок 5.12 – Страница редактирования комментария

Для удаления комментария пользователь должен нажать на боковое меню соответствующего комментария в правом верхнем углу и выбрать опцию Delete как показано на рисунке 5.13, нажав на которую он увидит всплывающее окно с предупреждением об удалении комментария. При нажатии на No, окно с предупреждением будет закрыто и никакие действия не будут предприниматься, если же пользователь нажмет на Yes, есть два варианта действий, если он не является пользователем создавшим комментарий, увидит сообщение с предупреждением о том что он не может удалить чужой комментарий и никакие действия не будут предприняты, если же это его комментарий он увидит сообщение с подтверждением удаления комментария и будет направлен на домашнюю страницу.

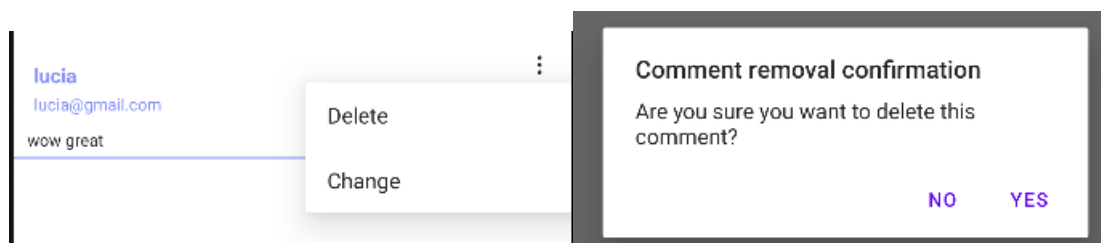


Рисунок 5.13 – Удаление комментария

Отзыв пользователю может добавить только другой пользователь, для этого он должен перейти на страницу интересующего его пользователя нажав на его данные в одном из постов. И затем на открывшейся странице нажать на опцию отзыва. На открывшейся странице он увидит отзывы, оставленные ранее и внизу страницы кнопку с плюсом для добавления отзыва как показано на рисунке 5.14, нажав на которую он будет направлен на страницу для ввода отзыва. При нажатии на эту кнопку все что требуется от пользователя это ввести отзыв и для его публикации нажать на ОК, при этом он будет направлен на домашнюю страницу и увидит сообщение об успешном добавлении отзыва. В случае если он решил его не добавлять он может нажать на кнопку Cancel. При нажатии на кнопку ОК если пользователь не вел отзыв он увидит соответствующее сообщение об ошибке.

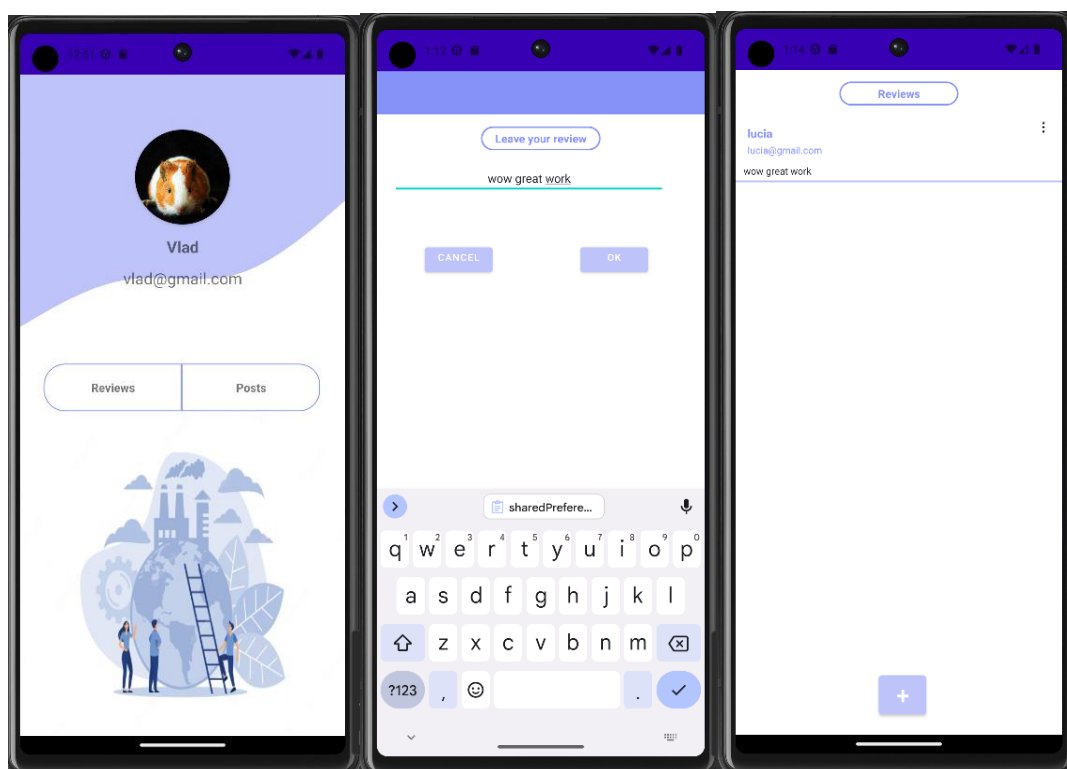


Рисунок 5.14 – Добавление отзыва

Для изменения отзыва, как и в случае с постом пользователю нужно нажать на значок меню в правом верхнем углу отзыва, который он хочет изменить, и он увидит всплывающее окно с подтверждением его действий при нажатии на No окно закроется и никакие действия не будут предприняты, при нажатии на Yes и при условии что отзыв принадлежит этому пользователю будет направлен на страницу для его редактирования, в противном случае он увидит сообщение о том что он не может изменить чужой отзыв и никакие действия не будут выполнены. На странице для изменения отзыва пользователь увидит окно для редактирования с текущим отзывом, который он может изменить, при нажатии на кнопку Cancel изменения не будут сохранены, и пользователь

будет возвращен на предыдущую страницу, при нажатии на ОК и условии что поле с отзывом не пустое, отзыв будет изменен, и пользователь получит соответствующее сообщение и будет направлен на домашнюю страницу, в противном случае, получит предупреждение с требованием ввести данные.

Для удаления отзыва пользователь должен нажать на боковое меню соответствующего комментария в правом верхнем углу и выбрать опцию Delete как показано на рисунке 5.15, нажав на которую он увидит всплывающее окно с предупреждением об удалении отзыва. При нажатии на No, окно с предупреждением будет закрыто и никакие действия не будут предприниматься, если же пользователь нажмет на Yes, есть два варианта действий, если он не является пользователем создавшим отзыв, увидит сообщение с предупреждением о том что он не может удалить чужой отзыв и никакие действия не будут предприняты, если же это его отзыв он увидит сообщение с подтверждением удаления отзыва и будет направлен на домашнюю страницу.

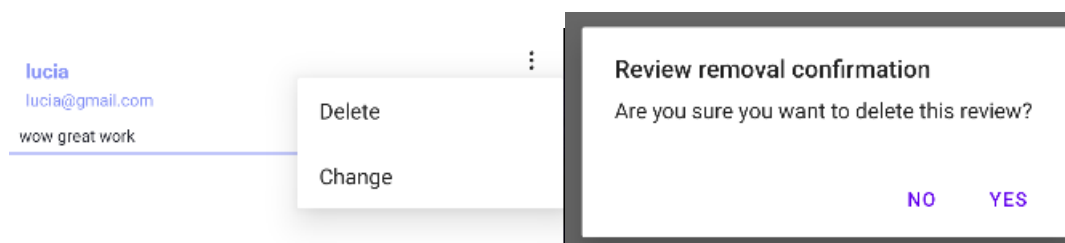


Рисунок 5.15- Удаление отзыва

Для изменения информации о пользователе пользователь должен сначала зайти на свою страницу профиля используя боковое меню или меню внизу экрана, и далее выбрав опцию Profile info, нажав на которую он сможет перейти на страницу для редактирования информации. В зависимости от того какую именно информацию желает изменить пользователь он может нажать на кнопку change рядом с полем с текущей информацией, которую он хочет изменить как представлено на рисунке 5.16. Пользователь может изменить имя, почту и изображение.

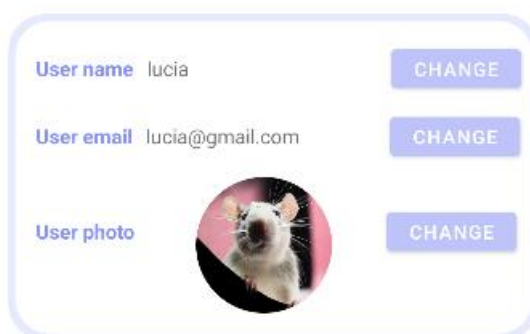


Рисунок 5.16 – Персональная информация пользователя

При нажатии на кнопку change возле имени пользователь увидит всплывающее окно с подтверждением его действий при нажатии на No окно закроется и никакие действия не будут предприняты, при нажатии на Yes он будет направлен на страницу для его редактирования, поле для редактирования с его текущим именем, которое он может отредактировать, как показано на рисунке 5.17. Нажав на кнопку Cancel, он может отменить редактирование и вернуться на прежнюю страницу. Нажав на OK при условии, что поле заполнено пользователь получит сообщение с подтверждением изменения имени и будет возвращен на предыдущую страницу, где отображаются уже обновленные данные.

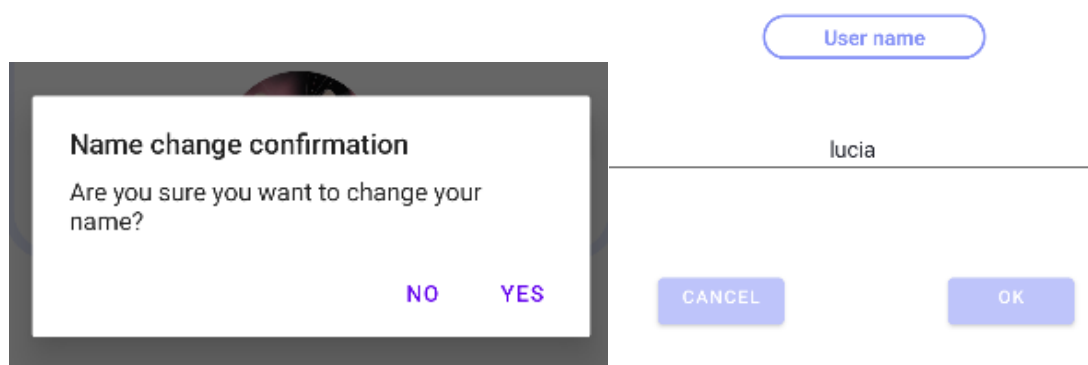


Рисунок 5.17 – Изменение имени пользователя

Изменение имени почты будут выполнены аналогичные действия, как показано на рисунке 5.18 с единственным отличием введенная почта будет проверяться на соответствие и, если пользователь с такой почтой уже существует пользователю будет предложено ввести другую. Изображение изменяется схожим образом только вместо поля для ввода текста пользователь должен нажать на текущее изображение и выбрать другое из изображений, хранящихся в памяти телефона.

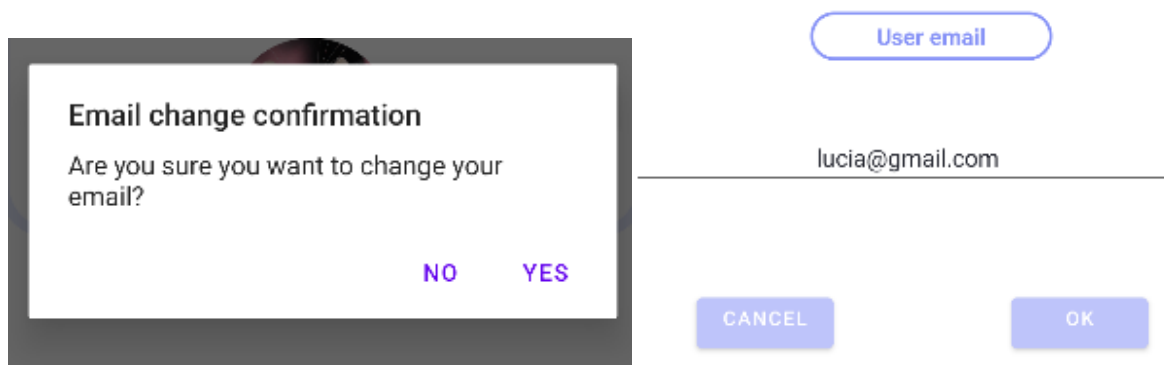


Рисунок 5.18 – Изменение почты пользователя

Пользователь может осуществить выход из приложения нажав на соответствующую опцию в боковом меню, а также на кнопку на странице пользователя. В обоих случаях при нажатии он

увидит всплывающее сообщение с подтверждением выхода как показано на рисунке 5.19. Нажав на No, никаких действий не будет предпринято, и пользователь останется на текущей странице. При нажатии на Yes будет выполнен выход из приложения, и пользователь будет перенаправлен на страницу для входа в приложение.

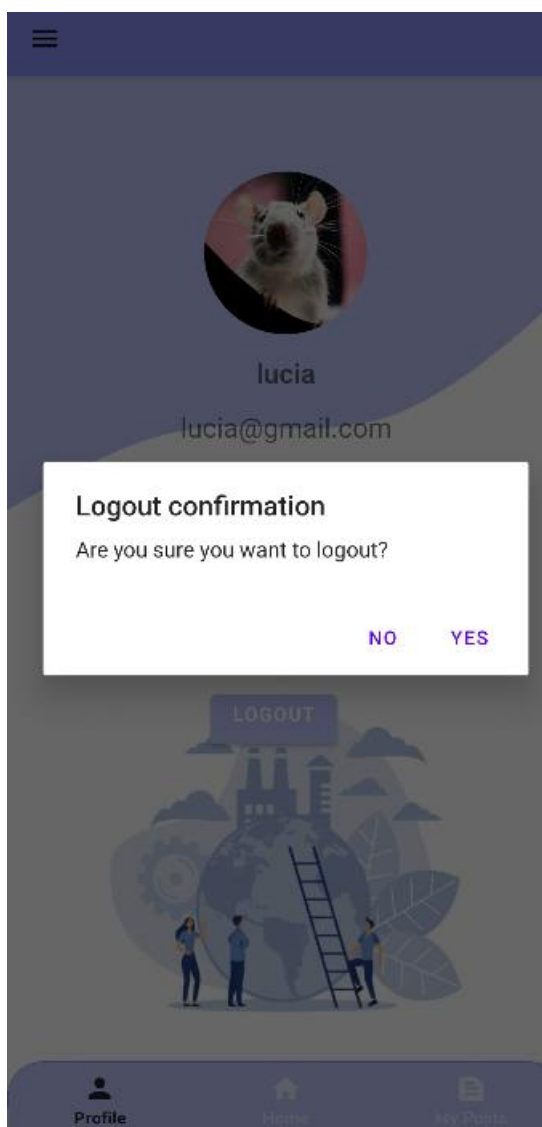


Рисунок 5.19- Выход из приложения

ЗАКЛЮЧЕНИЕ

В данном проекте была рассмотрена важность выбранной темы и ее актуальность. Был проведен анализ существующих на рынке аналогов и были рассмотрены их особенности, положительные и отрицательные стороны, это позволило получить более точное представление о разрабатываемом продукте и о функционале, который будет реализован в дальнейшем. Также был проведен анализ целей и требований к разрабатываемой информационной системе, для определения используемого типа архитектуры, технологий и программных решений.

Другой важной частью данной работы стало моделирование и проектирование системы. Данные этапы являются неотъемлемой частью жизненного цикла системы и позволяет представить все свойства и характеристики системы в пригодном виде, для последующего изучения и реализации системы. Моделирование системы позволяет систематизировать все требования к системе, а также к ее функциональности с учетом заданных ограничений. Информационная система для организации работы волонтеров была смоделирована, используя язык UML.

Последним этапом стало начало работы над непосредственной реализацией приложения, а также тестирование его корректной работы. Были реализованы следующие процессы, регистрация и аутентификация пользователя, добавление удаление и изменение постов, добавление, удаление и изменение комментариев к постам, добавление удаление и изменение отзывов о пользователе, распределение постов по категориям, изменение информация пользователя, а также выход из системы.

Также была проведена оценка стоимости проекта и создана документация, что стало завершающим этапом работы над проектом.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Steamhot: Волонтерское движение в социальной сфере. Деятельность социального работника в волонтерском движении. Министерство образования и науки РФ [электронный ресурс], [цитирован 15.01.2023]. - Режим доступа: <https://steamhot.ru/bez-vlozhenijj/volonterskoe-dvizhenie-v-socialnoi-sfere-deyatelnost.html>
2. VOLUNTEER FDIP: Countries with Highest Numbers of Volunteers: USA, Canada, Australia, UK, France, and Many More. [электронный ресурс], [цитирован 15.01.2023]. - Режим доступа: <https://www.volunteerfdip.org/countries-with-highest-numbers-of-volunteers-usa-canada-australia-uk-france>
3. Google Play: ДОБРО.РФ. [электронный ресурс], [цитирован 15.01.2023]. - Режим доступа: <https://play.google.com/store/apps/details?id=ru.dobro&hl=ru&gl=US>
4. Pinterest: About BuzzFeed. [электронный ресурс], [цитирован 15.01.2023]. - Режим доступа: <https://www.pinterest.com/pin/volunteer-match-is-a-locationbased-app-that-shows-you-nearby-volunteer-opportunities--311803974174546340/>
5. MUO: 7 Apps That Let You Help Someone and Do Good Today. [электронный ресурс], [цитирован 15.01.2023]. - Режим доступа: <https://www.makeuseof.com/tag/apps-help-someone-do-good/>
6. Образовательный портал: Виды диаграмм UML кратко [цитирован 02.03.2023]. - Режим доступа: <https://obrazovanie-gid.ru/pereskazy1/vidy-diagramm-uml-kratko.html>
7. Lucidchart: Разновидности диаграмм UML. [электронный ресурс], [цитирован 02.03.2023] Режим доступа: <https://www.lucidchart.com/blog/ru/types-of-UML-diagrams>
8. HEXLET: Клиент-серверная архитектура. [электронный ресурс], [цитирован 16.01.2023]. - Режим доступа: https://ru.hexlet.io/courses/internet-fundamentals/lessons/client-server/theory_unit
9. ТЕСТИНГ.РУ: Тестирование программного обеспечения - основные понятия и определения [электронный ресурс], [цитирован 16.01.2023]. - Режим доступа: <http://www.protesting.ru/testing/>
10. ХАБР: Оценка стоимости разработки программного продукта, информационной системы, сервиса или задачи [электронный ресурс], [цитирован 16.01.2023]. - Режим доступа: <https://habr.com/ru/articles/713998/>