Министерство науки и высшего образования Российской Федерации
Федеральное государственное автономное образовательное учреждение
высшего образования
«Южно-Уральский государственный университет
(национальный исследовательский университет)»
Высшая школа электроники и компьютерных наук
Кафедра «Электронные вычислительные машины»

РАБОТА ПРОВЕРЕНА
Рецензент
_____
_____2019 г.

ДОПУСТИТЬ К ЗАЩИТЕ
Заведующий кафедрой
_____Г.И. Радченко
_____2019 г.

Разработка мобильного приложения «Биржа работ» на базе Android

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА
К ВЫПУСКНОЙ КВАЛИФИКАЦИОННОЙ РАБОТЕ
ЮУрГУ-090301.2019.123.ПЗ ВКР

Руководитель работы,
к.п.н., доцент каф ЭВМ
_____Ю.Г. Плаксина
«___» _____2019 г.
Автор работы,
студент группы КЭ-452
_____Е.Д. Ваховский
«___» _____2019 г.
Нормоконтролёр, ст. преп. каф. ЭВМ
_____В.В. Лурье
«___» _____2019 г.

Челябинск 2019

Министерство науки и высшего образования Российской Федерации
Федеральное государственное автономное образовательное учреждение
высшего образования
«Южно-Уральский государственный университет
(национальный исследовательский университет)»
Высшая школа электроники и компьютерных наук
Кафедра «Электронные вычислительные машины»

УТВЕРЖДАЮ
Заведующий кафедрой
_____Г.И. Радченко
«___» _____2019
г.

ЗАДАНИЕ
на выпускную квалификационную работу бакалавра
студенту группы КЭ-452
Ваховский Евгений Дмитриевич
обучающемуся по направлению
09.03.01 «Информатика и вычислительная техника»

1. Тема работы: «Разработка мобильного приложения «Биржа работ» на базе Android» утверждена приказом по университету от 25 апреля 2019 г. №899

2. Срок сдачи студентом законченной работы: 1 июня 2019 г.

3. Исходные данные к работе.

3.1 Мартин Р. Чистый код: создание, анализ и рефакторинг. – Питер СПб, 2018. – 464 с.

3.2 Влиссидес Д., Джонсон Р., Хелм Ричард., Гамма Э. Приемы объектно-ориентированного проектирования. Паттерны проектирования. – Питер, 2016. – 366 с.

3.3 Гриффитс Д., Гриффитс Д. Head First. Программирование для Android – Питер, 2016. – 704 с.

4. Перечень подлежащих разработке вопросов.

4.1 Провести анализ предметной области.

4.2 Спроектировать мобильное приложение на базе ОС Android.

4.3 Разработать мобильное приложение на базе ОС Android.

4.4 Выпустить мобильное приложение на рынок.

5. Дата выдачи задания: 1 декабря 2018

Руководитель работы _____/Ю.Г. Плаксина/

Студент _____/Е.Д. Ваховский/

КАЛЕНДАРНЫЙ ПЛАН

Этап	Срок сдачи	Подпись руководителя
Введение и обзор литературы	15.01.2019	
Разработка модели, проектирование	01.04.2019	
Реализация системы	01.05.2019	
Тестирование, отладка, эксперименты	15.05.2019	
Компоновка текста работы и сдача на нормоконтроль	24.05.2019	
Подготовка презентации и доклада	30.05.2019	

Руководитель работы _____/Ю.Г. Плаксина/

Студент _____/Е.Д. Ваховский/

АННОТАЦИЯ

Ваховский Е.Д. Разработка мобильного приложения «Биржа работ» на базе Android – Челябинск: ФГАОУ ВО «ЮУрГУ (НИУ)», ВШЭКН; 2019, 192 с., 19 ил., библиографический список – 23 наим.

Выпускная квалификационная работа была выполнена с целью разработки мобильного приложения для поиска краткосрочной работы на базе ОС Android.

В данной работе были изучены существующие методы разработки мобильных приложений для ОС Android, выполнен анализ требований, выполнено проектирование мобильного приложения, протестировано разработанное приложение.

Результатом выполненной работы является полностью функционирующее и выпущенное на рынок мобильное приложение на базе Android.

Заказчиком приложения является компания ООО «Елена Плюс». Приложение получило название «Banomo» и имеет более 100 установок на мобильные устройства на базе Android.

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ

В настоящее время каждому человеку постоянно требуется иметь под рукой большое количество информации, развлечений и инструментов для работы. Одним из самых эргономичных решений, которые способны справиться с этими задачами, является смартфон. Смартфоны стали неотъемлемой частью жизни практически любого человека. Согласно исследованиям ассоциации GSMA (торговая организация, представляющая интересы операторов мобильной связи) число пользователей смартфонов в 2018 году превысило 3 миллиарда [1].

С 2008 по 2018 года количество используемых смартфонов, по сравнению с другими электронными устройствами выросло с 17% до 78% [2]. Количество загрузок мобильных приложений в 2018 году увеличилось на 35%, по сравнению с 2016 годом. 101 миллиард долларов пользователи потратили в магазине приложений. Рост на 75% по сравнению с 2016 годом. В самих мобильных устройствах, пользователи проводят в среднем по 3 часа в день. Рост на 50% [3].

Смартфоны сами по себе достаточно гибкие и умные устройства, но расширить функционал смартфона возможно с помощью мобильных приложений.

Мобильные приложения – это программные продукты, разработанные специально для мобильных устройств, смартфонов, планшетных компьютеров, умных телевизоров, умных часов и т.д. Мобильные приложения распространяются через магазины мобильных приложений, такие как: Apple AppStore (для приложения на базе IOS), Google Play Market (для приложений на базе Android), Microsoft Store (для приложений на базе Windows Phone).

Мобильные приложения помогают решать различные прикладные задачи: от звонков и приема e-mail сообщений до узкоспециализированных функций. Они призваны организовать жизнь пользователей мобильных устройств, а также ее разнообразить. Было создано огромное количество различных приложений, которые ориентированы на разные области.

1) Работа с различными типами файлов.
2) Карты и навигация.

3) Организация быта.

4) Социальные сети.

5) Браузеры.

6) Дистанционное управление техникой.

7) Обучение и саморазвитие.

В связи с тем, что человеку для жизни необходимы деньги, которые можно получить с помощью работы или же подработки, немалым спросом пользуются приложения для поиска работы.

На сегодняшний день ассортимент приложения для поиска работ широк и разнообразен, но большинство из них помогают найти постоянную работу, и только немногие из них, помогают найти подработку для того, чтобы быстро и в кратчайший срок подзаработать денег. В связи с этим, мобильное приложение для поиска краткосрочной работы будет актуальным как никогда ранее.

Целью работы является разработка мобильного приложения для поиска краткосрочной работы на базе ОС Android.

Для достижения поставленной цели необходимо решить поставлены следующие задачи.

1. Изучить существующие методы разработки мобильных приложения для ОС Android.
2. Определить и проанализировать функциональные и нефункциональные требований к мобильному приложению.
3. Выполнить проектирование мобильного приложения.
4. Реализовать мобильное приложение для ОС Android.
5. Протестировать разработанное приложение.
6. Выпустить мобильное приложение на рынок.

1 АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ

1.1 Обзор аналогов

На сегодняшний день уже существуют мобильные приложения для поиска краткосрочных работ. Но не все их них удовлетворяют потребностям пользователей. Более того некоторые из них создают больше трудностей, чем предоставляют возможностей. Например, в некоторых приложениях, пользователь может потратить свои собственные средства, так и не успев заработать.

Для выявления преимуществ и недостатков, было решено проанализировать два подобных приложения размещенных в магазине приложений Google Play, а именно: «Profi.Ru», «YouDo».

Profi.Ru.

«Profi.Ru» – бесплатный российский сервис поиска частных специалистов. Образован в итоге объединения семи проектов по подбору специалистов для решения различных задач [4].

Разработчики «Profi.Ru» утверждают, что сервис насчитывает около 4 миллионов клиентов и 400 тысяч исполнителей, но данная информация ставится под сомнение при анализе отзывов приложения в Google Play. Пользователи утверждают, что заказы составляет один и тот же человек (или робот) для создания видимости присутствия огромного количества клиентов. В добавок ко всему, пользователи постоянно жалуются, что вынуждены вкладывать свои деньги для просмотра контактов исполнителей или же для отклика на заявку, при это заработать и вернуть хотя бы вложенные деньги им не удаётся [6]. Во время анализа отзывов пользователей о приложении были замечены лживые отзывы, большинство из которых содержали одинаковый текст, расхваливающий приложение, и одинаковую оценку, что несомненно ставит под вопрос реальную популярность данного сервиса.

В добавок ко всему, сервис использует так называемые «холодные звонки», обзванивая своих потенциальных клиентов, чем ни раз вызвали недовольство пользователей [5].

Также стоит отметить, что сервис разделил приложение для заказчика и исполнителя на два, что создает свои неудобства (например, необходимость освобождать место для установки второго приложения).

YouDo

YouDo – российский сервис, который позволяет находить с помощью веб-версии и мобильного приложения исполнителей для решения бытовых и бизнес-задач [7].

Сервис YouDo имеет как схожие проблемы, так и отличные от предыдущего. Рассмотрим отличные от предыдущего сервиса проблемы.

Мобильное приложение сервиса имеет огромное количество так называемых накрученных отзывов, не соответствующие с реальным положением дел. Накрученные отзывы появляются каждый день и имеют практически идентичный текст, что отпугивает пользователей от приложения. Исходя из отзывов пользователей можно сделать вывод, что пользователи недовольны тем, что для того, чтобы отозваться на работу, необходимо заплатить деньги. В том случае, если пользователя не выберут исполнителем, деньги не возвращаются

Сервис имеет еще одну не маловажную проблему, для регистрации в роли исполнителя, необходимо прибыть в пункт регистрации лично. Эти пункты в свою очередь, находятся лишь в двух крупных городах, а именно Москве и Санкт-Петербурге, которые в свою очередь находятся практически на границе этих городов.

С учетом проанализированных аналогов, были определены требования для разрабатываемого приложения.

1.2 Анализ основных технологических решений

Разработка мобильных приложений – это процесс, в результате которого разрабатывается и создается приложение, способное работать на мобильных

устройствах, смартфонах, планшетах. На сегодняшний день существует несколько решений для создания мобильных приложений.

1. Нативная разработка – разработка, при которой разработчики используют оригинальные языки программирования и инструменты, интегрированные среды разработки, заявленные компанией-разработчиком платформы.

2. Кроссплатформенная разработка – разработка, при которой разработчики используют специальные инструменты, позволяющие разрабатывать программный продукт сразу на несколько операционных систем.

К интегрированным средам разработки для ОС Android относится Android Studio, разработанная компанией Google.

Android Studio – основанная на программном обеспечении IntelliJ IDEA от компании JetBrains интегрированная среда разработки [8]. Android Studio в свою очередь является официальным средством разработки приложений под ОС Android от компании Google [9]. Так как Android Studio основана на IntelliJ IDEA, то стоит упомянуть, что она переняла все плюс данной среды разработки и в связи с тем, что среда разработки IntelliJ IDEA в первую очередь направлена на ускорение разработки ПО, Android Studio обладаем теми же качествами.

Разработка приложений поддерживается на следующих языках: Java, C++, Kotlin.

Android Studio имеет огромное количество инструментов для отладки, анализа и рефакторинга приложений, а также является бесплатным в использовании. Перечислим доступные и используемые во время разработки приложения инструменты [8]:

1) Гибкая система автоматической сборки основанная Gradle.
2) Быстрый и многофункциональный эмулятор.
3) Мгновенное внесение изменений в работающее приложение без создания нового APK.
4) Шаблоны кода и интеграция с GitHub.
5) Инструменты тестирования.

6) Инструменты для анализа производительности, совместимости версий и других проблем.

7) Встроенный ProGuard и утилита для подписи приложений.

8) Шаблоны основных макетов и компонентов.

9) Встроенная поддержка Google Cloud Messaging.

К кроссплатформенным инструментам разработки можно отнести фреймворк Xamarin.

Xamarin – это фреймворк для кроссплатформенной разработки мобильных приложений (IOS, Android, Windows Phone) с использованием языка C# [10]. При этом фреймворк предоставляет полный доступ ко всем возможностям SDK платформы и стандартному механизму создания UI. Разработанное на выходе приложение, по неофициальным заверениям разработчиками фреймворка, ничем не уступает по производительности от нативных приложений. Тем не менее Android Studio имеет ряд преимуществ перед Xamarin, основные из которых:

1) Скорость работы IDE.

2) Стабильность работы приложений и IDE.

3) Скорость обновления до новых версий.

4) Количество и качество различных инструментов для отладки и разработки.

Для реализации взаимодействия мобильного приложения с сервером по HTTP протоколу существует два наиболее популярных решения.

1) REST – архитектурный стиль взаимодействия компонентов распределённого приложения в сети [11].

2) GraphQL – стандарт декларирования структуры данных и способов получения данных, который выступает дополнительным слоем между клиентом и сервером [12].

По своей сути REST и GraphQL очень похожи. Они выполняют одинаковую задачу на одном протоколе, а также имеют следующие сходства:

1) Есть понятие ресурса, есть возможность назначать идентификаторы ресурсам.

2) Ресурсы извлекаются с помощью GET- запроса.

3) Ответ на запрос возвращается в формате JSON.

4) Возможность различать запросы к API на чтение и запись.

5) Конечные точки в REST и поля в GraphQL в конечном итоге вызывают функции на сервере.

Но, они также имеют различия, которые в свою очередь являются преимуществом GraphQL, например:

1) В REST структура и объем ресурса определяются сервером. В GraphQL сервер определяет набор доступных ресурсов, а клиент указывает необходимые ему данные прямо в запросе.

2) В REST каждый запрос обычно вызывает ровно одну функцию для обработки маршрута, в GraphQL один запрос может вызвать множество функций для построения сложного ответа с множеством вложенных ресурсов. Что позволяет в свою очередь ускорить разработку проекта в несколько раз и сэкономить тысячи строк.

Из представленных решений для организации взаимодействия клиента мобильного приложения и сервера был выбран GraphQL, так как в сравнении с REST он является более удобным инструментом, а также позволяет ускорить разработку проекта.

Для поддержки PUSH уведомлений компания Google предлагает использовать библиотеку Firebase Cloud Messaging (FCM).

Firebase Cloud Messaging (FCM) – кроссплатформенное решение, позволяющее добавить в мобильное приложение поддержку PUSH уведомлений [13].

Принцип работы FCM представлен на рисунке 1.2.1.

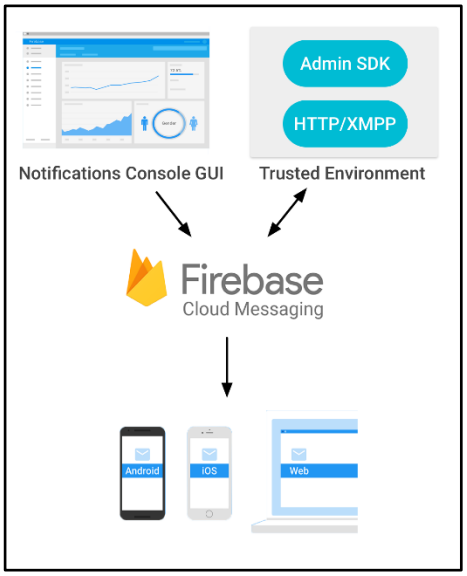

Рисунок 1.2.1 – Принцип работы FCM

FCM позволяет отправлять PUSH уведомления на все устройства разных платформ с использованием API или же панели инструментов. Вид панели инструментов для отправки PUSH уведомлений представлен на рисунках 1.2.2 и 1.2.3.

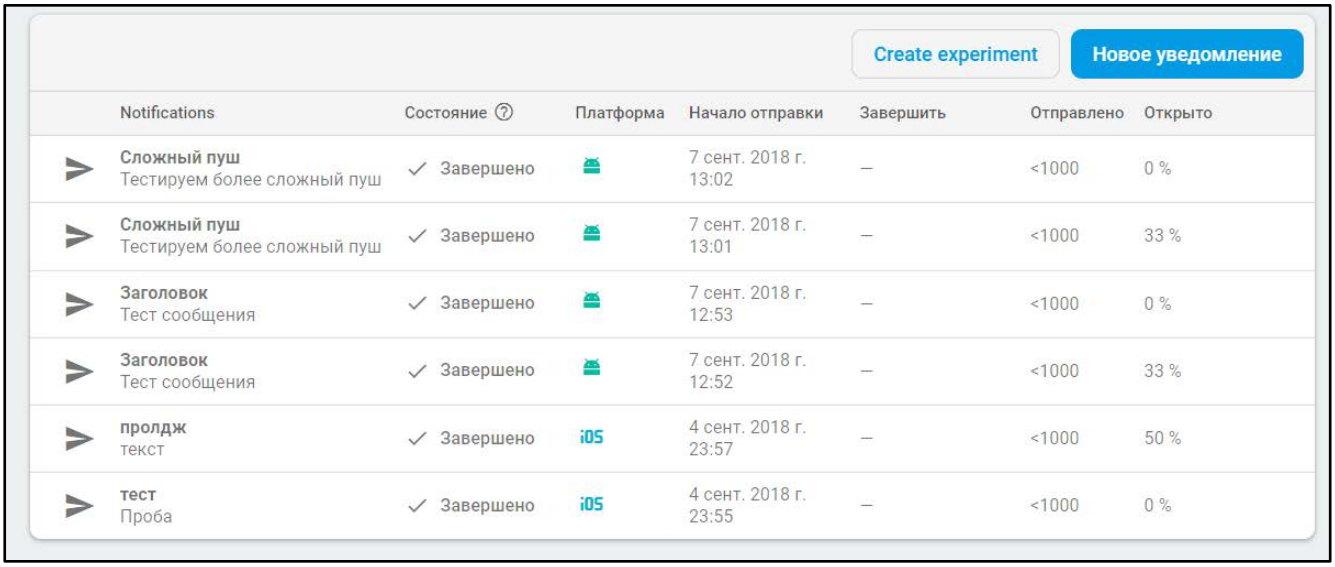

Рисунок 1.2.2 – Панель управления для отправки PUSH уведомлений

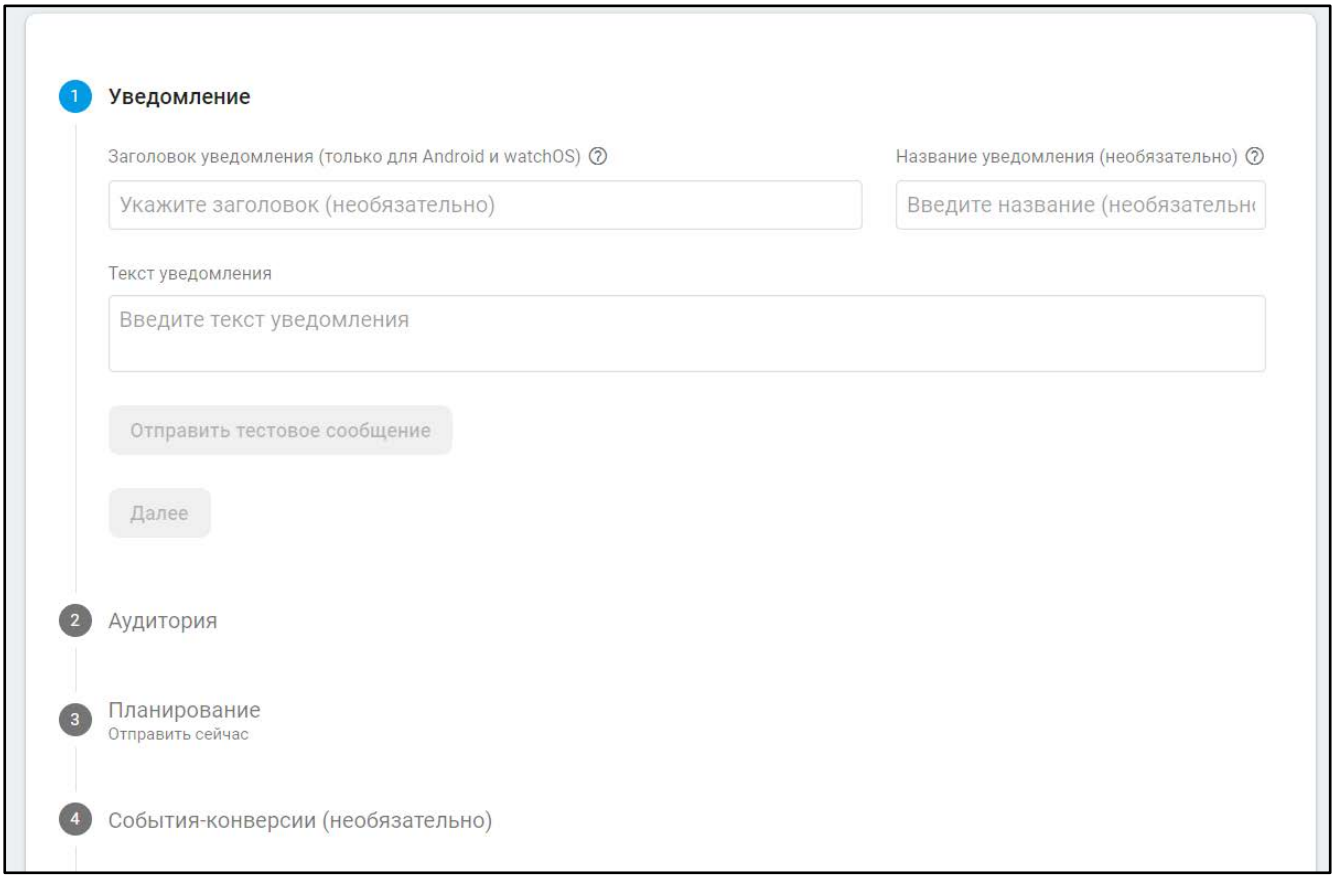

Рисунок 1.2.3 – Создание PUSH уведомления с использованием панели управления

FCM в добавок ко всему имеет огромное количество инструментов для отладки приложения и аналитики, например FCM позволяет отследить, на каком экране пользователи проводят больше всего времени или же из какой страны, города и даже улицы пользователи запускают приложение. Пример представлен на рисунках 1.2.4, 1.2.5 и 1.2.6. Данные, представленные на рисунках, являются реальными данными собранные во время работы мобильного приложения.

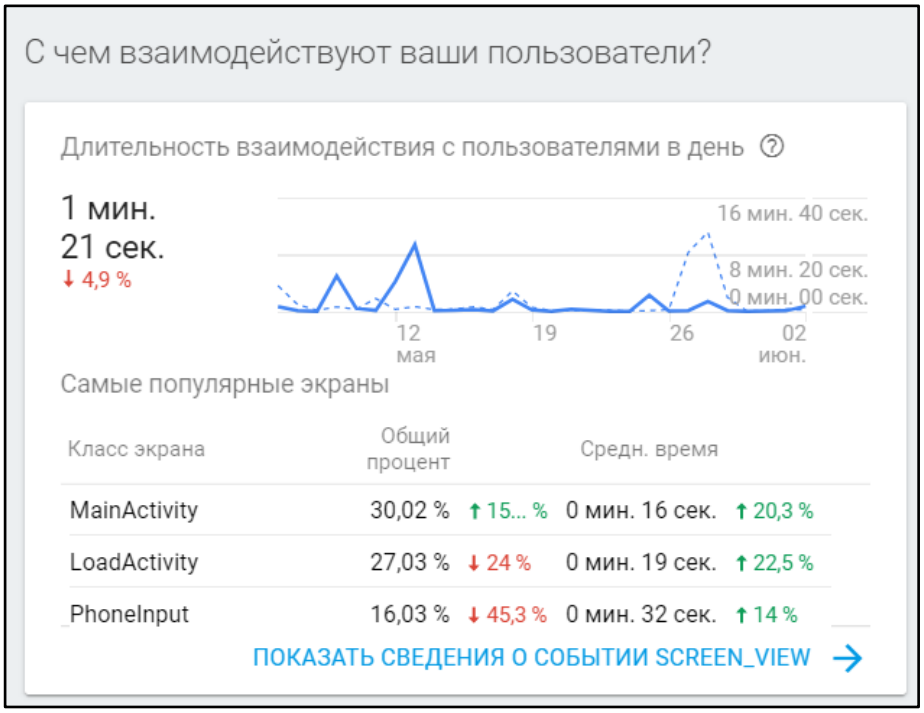

Рисунок 1.2.4 – Время взаимодействия пользователей с различными экранами приложения

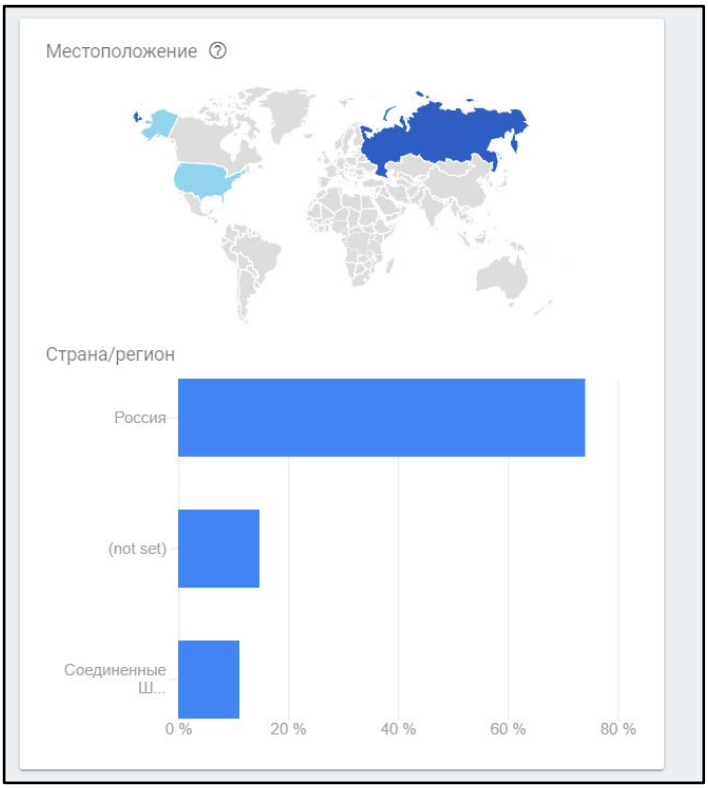

Рисунок 1.2.5 – Местоположение пользователей

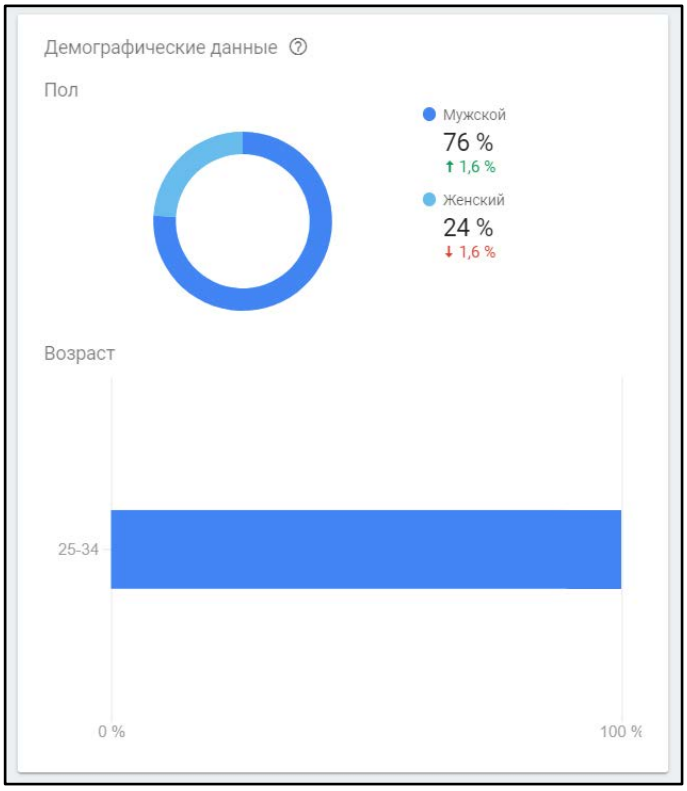

Рисунок 1.2.6 – Демографические данные пользователей

Вывод

Исходя из анализа существующих решений в области разработки мобильных приложений, под операционную систему Android была выбрана интегрированная среда разработки Android Studio. Ее инструменты позволят быстро и качественно реализовать мобильное приложение, протестировать его нагрузку и подготовить к запуску на рынок.

В качестве инструмента взаимодействия с сервером был выбран GraphQL, решающим аргументом в его пользу является скорость разработки проекта при использовании этого инструмента.

Инструментом для создания PUSH уведомлений, сбора данных о пользователях и стабильности работы приложения была выбрана библиотека Firebase Cloud Messaging. Данная библиотека обладает обширным функционалом для мобильной разработки и является отличным инструментом для сбора данных и отладки приложения.

2 ОПРЕДЕЛЕНИЕ ТРЕБОВАНИЙ

2.1 Функциональные требования

В ходе проектирования, разработки и анализа технического задания, с учетом требований, составленных заказчиком, были определены и проанализированы следующие функциональные требования.

1) Экран создания работы с возможностью указать: город выполнения, дату выполнения, тип работы, вариант оплаты (почасовая, за работу), число работников, описание работы (максимум 500 символов).

2) Экран с полной информацией о работе со стороны исполнителя с возможностью откликнутся на работу с указанием количества работников в зависимости от типа оплаты и желаемую сумму за выполнения работы.

3) Экран с полной информацией о работе со стороны заказчика с возможностью перехода к экрану просмотра откликнувшихся исполнителей и возможностью повтора работы.

4) Экран просмотра откликнувшихся исполнителей с возможностью выбора одного или нескольких из них в зависимости от типа оплаты и отображение номера телефона исполнителей после выбора.

5) Экран оценки исполнителей и заказчиков после выполнения работы.

6) Экран с указанием работ и откликов пользователя (кейс пользователя) с основной информацией по работе: дата начала, описание, тип работы, количество предложений или информация об отклике (количество человек и желаемая оплата), статус.

7) Экран с историей баланса и возможностью перехода к полной информации о работе.

8) Экран ввода промокода (промокоды распространяются вне приложения. В приложении должен быть осуществлен только ввод промокода и вывод результата его применения) для получения бонусных баллов.

9) Экран просмотра рейтинга пользователя и количество завершенных работ в роли как исполнителя, так и заказчика.

10) Экран настройки push и e-mail уведомлений.

11) Экран отправки сообщения технической поддержке.

12) Экран с лицензиями и соглашениями приложения.

13) Экран авторизации пользователя.

14) Экран регистрации пользователя.

15) Экран подтверждения номера телефона пользователя.

16) Экран изменения фильтров для поиска работы (фильтры работы: дата, тип работы, количество исполнителей, вариант оплаты, город).

17) Экран изменения фильтров в кейсе пользователя (фильтры в кейсе пользователей: статус работы, сортировка).

18) Загрузка данных для работы приложения с сервера.

18.1. Загрузка статусов работ.

18.2. Загрузка типов (сфер деятельности) работ.

18.3. Загрузка городов и областей.

18.4. Загрузка истории баланса пользователя.

По требованию заказчика были выделены требования к пользователям.

Каждый пользователь, работающий в системе, обязан:

1. Зарегистрироваться в системе и заполнить следующие обязательные поля в регистрационной форме:

a. Имя и фамилию пользователя.

b. Город.

c. Номер телефона.

1. Подтвердить свою личность через смс, отправленное на указанный пользователем номер телефона, чтобы подтвердить, что он является реальным человеком и владельцем указанного номера телефона.

2. Ознакомиться и принять пользовательское соглашение об условиях использования мобильного приложения.

3. Ознакомиться и принять правила и политику конфиденциальности персональных данных.

В системе должны быть выделены следующие типы пользователей:

– авторизованный пользователь;

– неавторизированный пользователь;

Авторизированный пользователь должен иметь возможность воспользоваться любым функционалом системы.

Неавторизированный пользователь должен иметь возможность воспользоваться функционалом поиска работ, фильтром работ и дальнейшей авторизацией или регистрации.

2.2 Нефункциональные требования

Заказчиком были определены нефункциональные требования к разрабатываемому приложению.

1) Мобильное приложение должно быть написано на языке Java.

2) Мобильное приложение должно функционировать на устройствах с операционной системой Android не ниже 4.0.3.

3) Мобильное приложение должно быть доступно только в портретной ориентации.

2.3 Описание бизнес-процессов системы

На рисунках 2.1 и 2.2 представлено описание работы основных бизнес-процессов системы. Для проектирования основных бизнес-процессов системы был использован язык графического описания объектного моделирования UML. Построена модель взаимодействия внешнего актера с мобильным приложением.

Авторизация пользователя

Неавторизированный пользователь может авторизоваться для работы в приложении под своим профилем. Однако данное действие является необязательным. В том случае если пользователь не авторизуется, ему будет доступен только поиск работ. Все остальные же его действия или их попытки, будут приводить к необходимости авторизации. Авторизовавшись, пользователь попадает на экран поиска работы по умолчанию. На рисунке 2.1 представлен процесс авторизации пользователя.

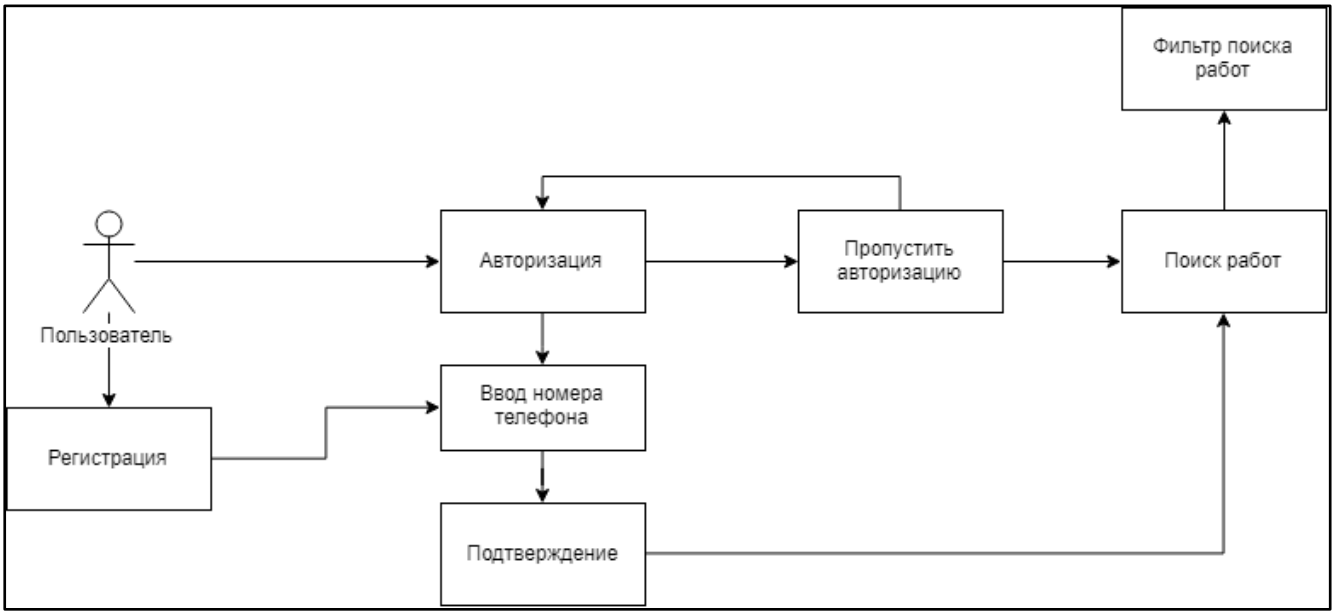

Рисунок 2.1 – Авторизация пользователя

Работа пользователя в роли заказчика

На рисунке 2.2 представлено описание бизнес-процесса и доступные действия по работе с заказом в роли заказчика.

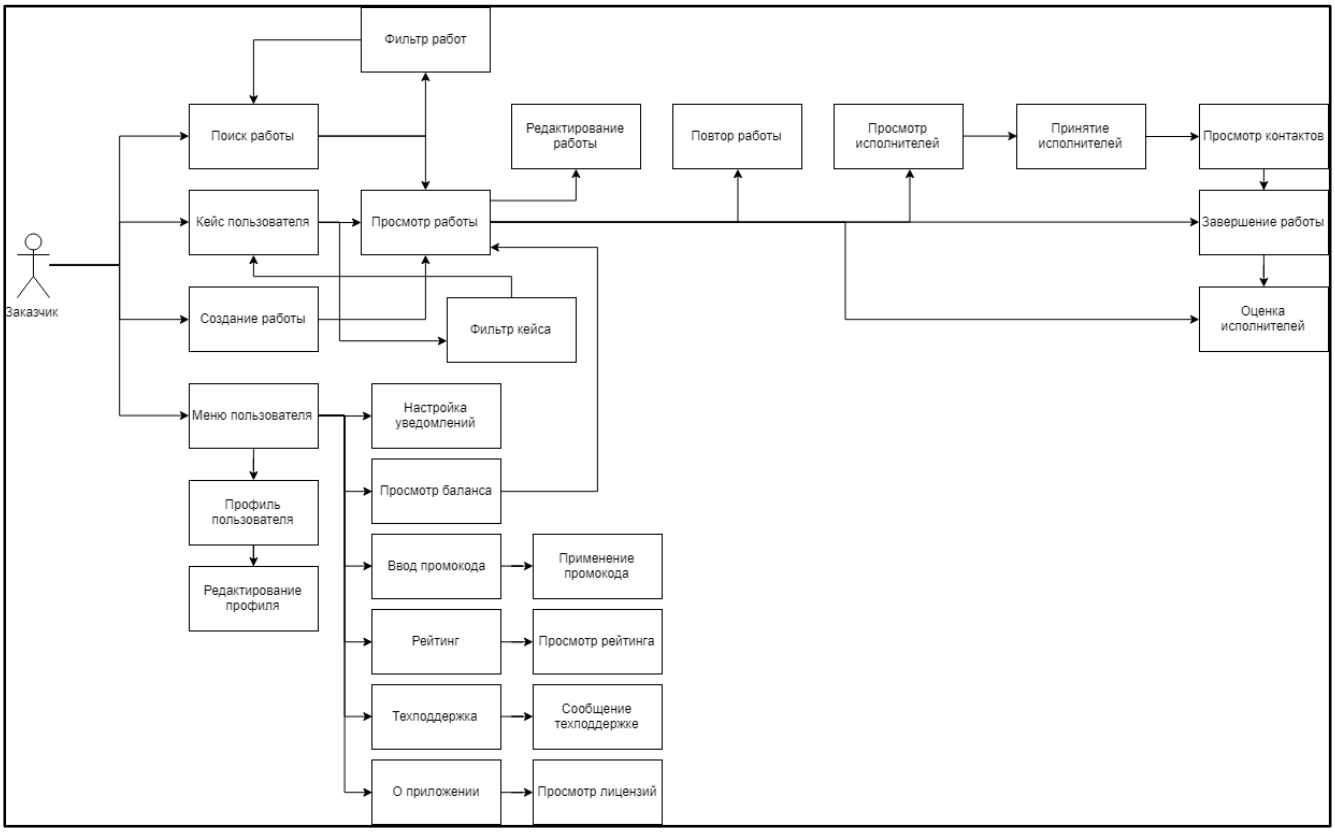

Рисунок 2.2 – Работа пользователя в роли заказчика

Пользователь, выступающий в роли заказчика, может реализовать следующие варианты использования приложения при работе с непосредственно со своим заказом:

1) Редактирование работы – отредактировать существующую работу.

2) Повторение работы – повторить существующую работу с возможность отредактировать необходимые параметры.

3) Просмотр исполнителей – просмотр исполнителей, откликнувшихся на работу.

4) Принятие исполнителей – принятие необходимое количество исполнителей на работу.

5) Просмотр контактов исполнителей – просмотр контактов исполнителей после принятия их на работу.

6) Завершение работы – завершение работы вручную раньше срока.

7) Оценка исполнителей – оценка исполнителей с отзывом если необходимо.

Работа пользователя в роли исполнителя

На рисунке 2.3 представлено описание бизнес-процесса и доступные действия по работе с заказом в роли исполнителя.

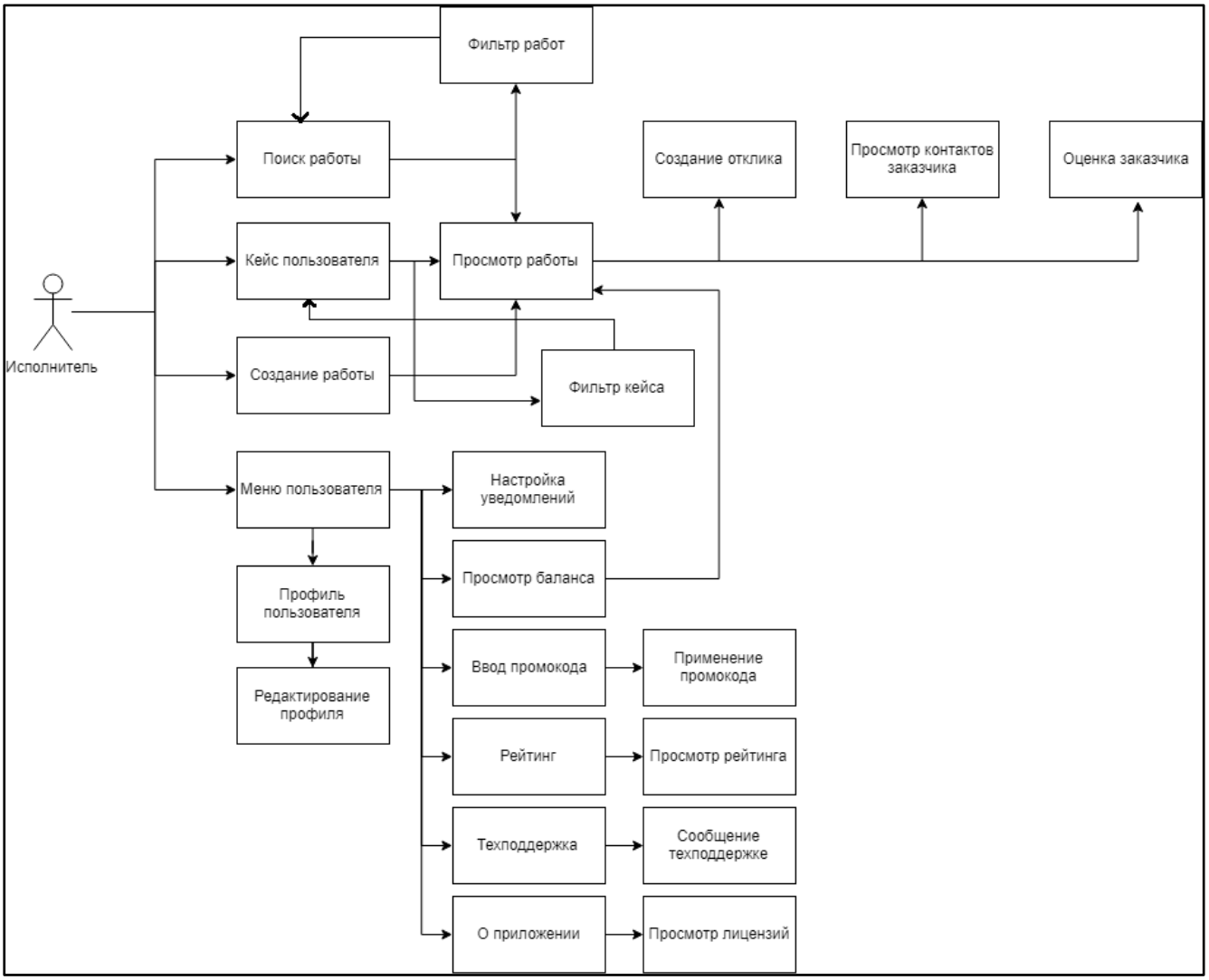

Рисунок 2.3 – Работа пользователя в роли исполнителя

Пользователь, выступающий в роли исполнителя, может реализовать следующие варианты использования приложения при работе с непосредственно со своим заказом:

1) Создание отклика – откликнуться на работу с указанием желаемой суммы и количеством человек.

2) Просмотр контактов заказчика – просмотр контактов заказчика при принятии на работу.

3) Оценка заказчика – оценка заказчика после окончания работы с отзывом если необходимо.

3 ПРОЕКТИРОВАНИЕ

3.1 Архитектура предлагаемого решения

На рисунке 3.1 представлена диаграмма компонентов мобильного приложения «Биржа работ» на базе ОС Android. В процессе проектирования приложения использовался архитектурный паттерн MVP.

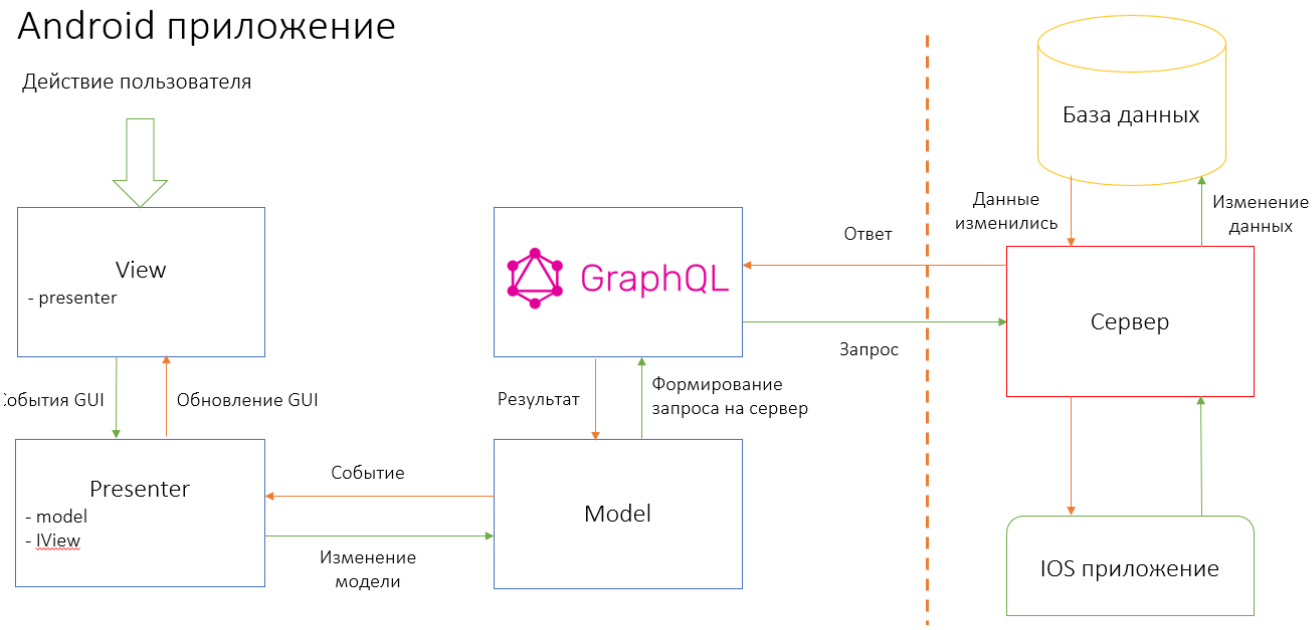

Рисунок 3.1 – Диаграмма компонентов системы

MVP – шаблон проектирования, производный от MVC, используемый в основном для проектирования пользовательского интерфейса [14,15,16]. Шаблон разработан для улучшения разделения ответственности в презентационной логике. MVP состоит из следующих компонентов:

1) Model (модель) – компонент приложения, отвечающий за формирование структуры данных внутри приложения, за изменения и чтения этих данных, а также место их хранения.

2) View (вид) – компонент приложения, реализующий интерфейс приложения. То, что пользователь видит и с чем может взаимодействовать: кнопки, поля ввода, изображения, текст и т.д.

3) Presenter (презентер) – компонент приложения отвечающий за обработку действий пользователя. Осуществляет взаимодействие модели и его приложения.

Главным достоинством данного паттерна является разделение ответственности между компонентами. Роль связующего звена между компонентами на себя берет Presenter, именно он руководит процессом, в нем хранятся экземпляры View и Model. При выполнении пользователем какого-либо действия View сообщает об этом Presenter (например, сообщает о том, что пользователь нажал кнопку), Presenter решает, как необходимо отреагировать на данное действие (например, загрузить данные из сети или попросить Model обновить данные о себе), результат возвращает назад во View для отображения информации пользователю.

Далее перечислен список некоторых презентеров мобильного приложения и их задачи:

1) PhoneVerificationPresenter – презентер, отвечающий за верификацию номера телефона пользователя.

2) PhoneInputPresenter – презентер, отвечающий за правильность введённого номера телефона, проверку на существование пользователя.

3) MainActivityPresenter – презентер, отвечающий за основную навигацию приложения и подключение к Push сервису.

4) JobSearchPresenter – презентер, отвечающий за поиск работ по заданным параметрам.

5) CreateJobPresenter – презентер, отвечающий за создание, редактирование и повтор работ.

6) ComplaintPresenter – презентер, отвечающий за создание запроса техподдержке.

7) LoadPresenter – презентер, отвечающий за загрузку приложения, его данных и их обновление.

8) UserJobsPresenter – презентер, отвечающий за загрузку и фильтрацию созданных пользователем работ.

9) UserOffersPresenter – презентер, отвечающий за загрузку и фильтрацию созданных пользователем откликов.

10) UsetJobPresenter – презентер, отвечающий за управление работой, созданной пользователем (просмотр откликов, завершение работы и т.д.).

11) ViewJobPresenter – презентер, отвечающий за просмотр работы исполнителем.

12) ProfilePresenter – презентер, отвечающий за редактирование профиля пользователя (смена имени и фамилии, почты, фотографии пользователя).

GraphQL – формирует запрос на запись или чтение данных с сервера. Запрос может быть как одиночным (запросить одну модель данных), так и вложенным (несколько моделей данных за раз в одном запросе), при ответе возвращает модель данных в единственном экземпляре.

3.2 Алгоритмы решения задач

Основной задачей приложения является организация взаимодействия между заказчиком и исполнителем работ. Необходимо в кратчайшие сроки связать эти две роли через приложение, для того чтобы заказчик получил результат вовремя, а исполнитель мог распределить свое время и избежать различных возможных конфликтов. Следовательно требуется:

1) вовремя уведомить заказчика о том, что на его работу был совершен отклик;

2) вовремя уведомить исполнителя о том, что его приняли на работу или отказали;

3) предоставить контакты обеим сторонам, для взаимообратной связи;

4) предоставить возможность оценки исполнителя и заказчика для оценки проведения встречи и результата выполненной работы.

На рисунке 3.2 представлен алгоритм ожидания откликов исполнителей со стороны заказчика работы

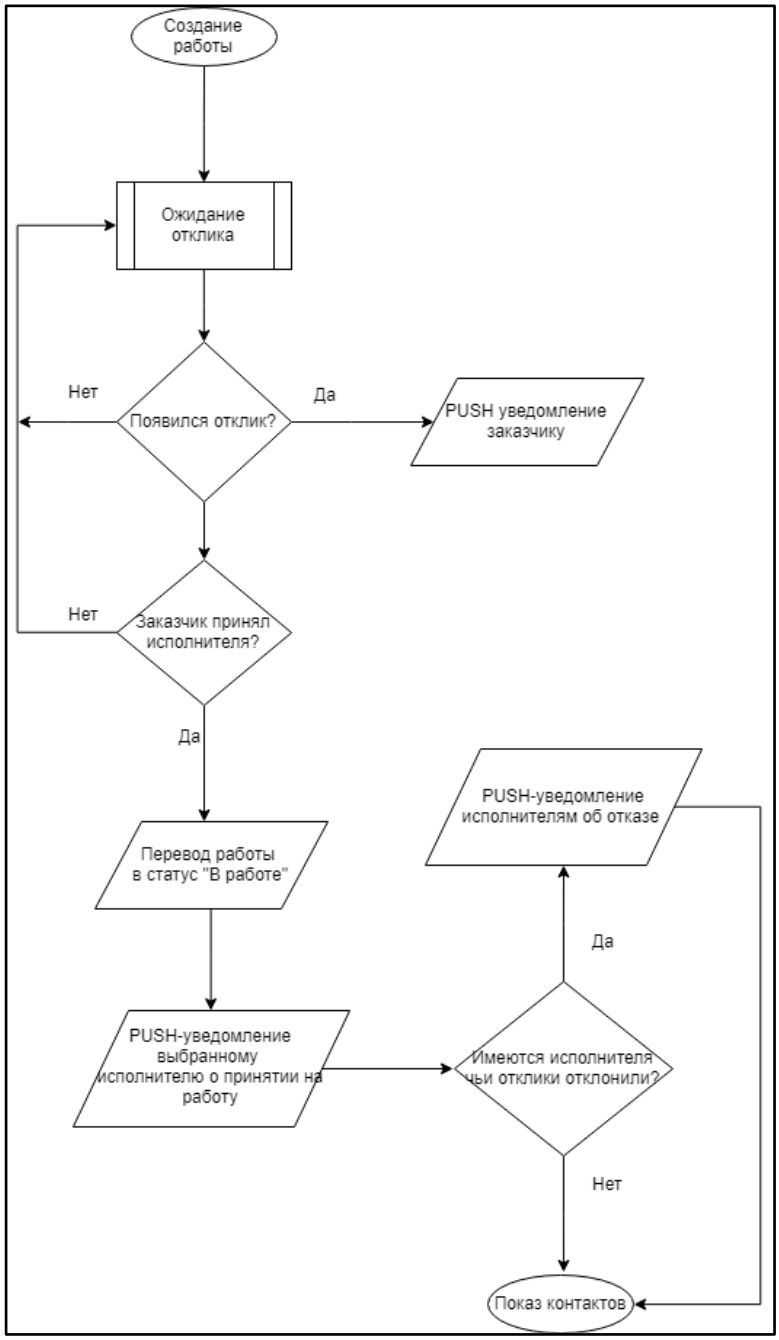

Рисунок 3.2 – Ожидание откликов исполнителей со стороны заказчика

3.3. Описание данных

Входные данные

В качестве входных данных от пользователя в момент регистрации используются:

1) Фамилия.
2) Имя.
3) Город.

4) Фотография (опционально).

В качестве входных данных от пользователя в момент авторизации используется номер телефона.

В качестве входных данных от пользователя в момент подтверждения номера телефона используется СМС код.

В качестве входных данных от пользователя в момент создания, редактирования, повтора работы используется:

1) Город.
2) Дата.
3) Тип работы.
4) Вариант оплаты.
5) Число работников.
6) Описание работы.
7) Фотографии работы.

В качестве входных данных от пользователя в момент редактирования фильтра поиска работы используется:

1) Город.
2) Дата.
3) Тип работы.
4) Вариант оплаты.
5) Число работников.
6) Значение переключателя «Не показывать мои работы в ленте».

В качестве входных данных от пользователя в момент редактирования профиля используется:

1) Фамилия.
2) Имя.
3) Город.
4) Фотография.

В качестве входных данных от пользователя в момент редактирования фильтров кейса пользователя используется:

1) Статус работы.

2) Тип сортировки.

В качестве входных данных от пользователя в момент редактирования в момент отклика на работу используется:

1) Сумма оплаты.

2) Количество человек.

В качестве входных данных от пользователя в момент редактирования почты используется почта.

В качестве входных данных от пользователя в момент редактирования настроек уведомлений используется:

1) Значение переключателя «Push-уведомлений».

2) Значение переключателя «Push новостей».

3) Значение переключателя «Отклики на мои работы» в подразделе push-уведомлений.

4) Значение переключателя «Статус моих откликов» в подразделе push-уведомлений.

5) Значение переключателя «E-mail новости».

6) Значение переключателя «Отклики на мои работы» в подразделе e-mail уведомлений.

7) Значение переключателя «Статус моих откликов» в подразделе e-mail уведомлений.

В качестве входных данных от пользователя в момент ввода промокода используется промокод в виде набора букв и цифр.

В качестве входных данных от пользователя в момент написания запроса в техподдержку используется: описание проблемы в виде набора букв и цифр.

В качестве входных данных от пользователя в момент принятия исполнителей используется: значение чек боксов карточек с откликами.

Выходные данные

В качестве выходных данных в мобильном приложении используются XML файлы, формирующие общий вид страницы:

1) экран создания работы с возможностью указать: город выполнения, дату выполнения, тип работы, вариант оплаты (почасовая, за работу), число работников, описание работы (максимум 500 символов);

2) экран с полной информацией о работе со стороны исполнителя с возможностью откликнутся на работу с указанием количества работников в зависимости от типа оплаты и желаемую сумму за выполнения работы;

3) экран с полной информацией о работе со стороны заказчика с возможностью перехода к экрану просмотра откликнувшихся исполнителей и возможностью повтора работы;

4) экран просмотра откликнувшихся исполнителей с возможностью выбора одного или нескольких из них в зависимости от типа оплаты и отображение номера телефона исполнителей после выбора;

5) экран оценки исполнителей и заказчиков после выполнения работы;

6) экран с указанием работ и откликов пользователя (кейс пользователя) с основной информацией по работе: дата начала, описание, тип работы, количество предложений или информация об отклике (количество человек и желаемая оплата), статус;

7) экран с историей баланса и возможностью перехода к полной информации о работе;

8) экран ввода промокода (промокоды распространяются вне приложения. В приложении должен быть осуществлен только ввод промокода и вывод результата его применения) для получения бонусных баллов;

9) экран просмотра рейтинга пользователя и количество завершенных работ в роли как исполнителя, так и заказчика;

10) экран настройки push и e-mail уведомлений;

11) экран отправки сообщения технической поддержке;

12) экран с лицензиями и соглашениями приложения;

13) экран авторизации пользователя;

14) экран регистрации пользователя;

15) экран подтверждения номера телефона пользователя;

16) экран изменения фильтров для поиска работы (фильтры работы: дата, тип работы, количество исполнителей, вариант оплаты, город);

17) экран изменения фильтров в кейсе пользователя (фильтры в кейсе пользователей: статус работы, сортировка).

4 РЕАЛИЗАЦИЯ

4.1 Реализация интерфейсов

При старте приложения пользователя приветствует загрузочное окно, представленное на рисунке 4.1

Рисунок 4.1 – Загрузочный экран

Для авторизации или регистрации пользователю необходимо указать свой номер телефона. Экран ввода представлен на рисунке 4.2.

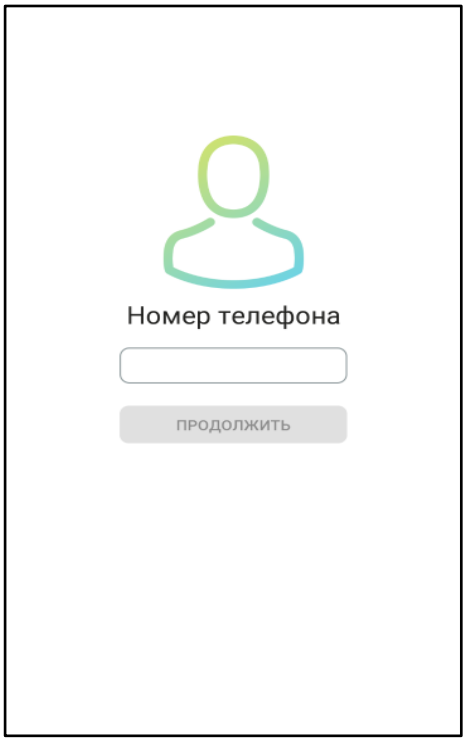

Рисунок 4.2 – Экран ввода номера телефона

Для подтверждения номера телефона, пользователю предлагается ввести СМС код. Экран ввода СМС кода представлен на рисунке 4.3.

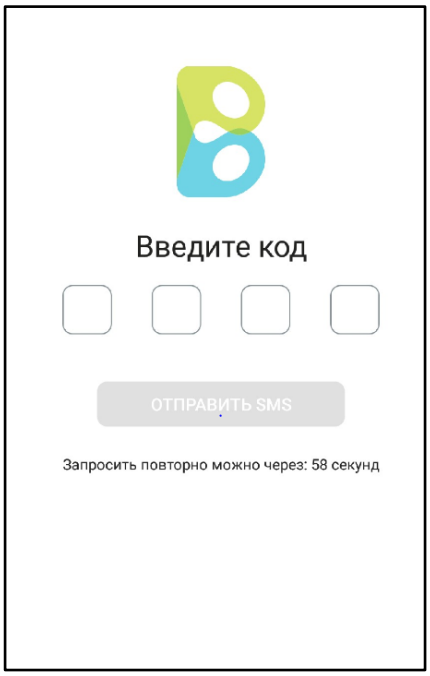

Рисунок 4.3 – Экран ввода СМС кода

Для редактирования профиля, пользователю предлагается заполнить поля, указанные на рисунке 4.4.

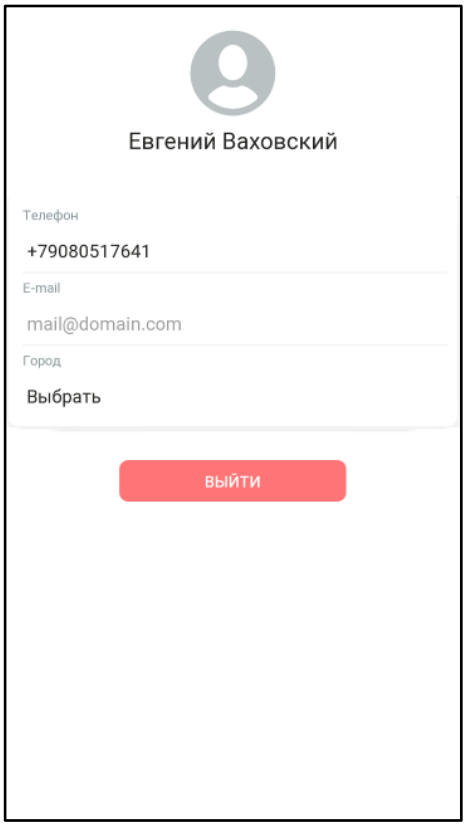

Рисунок 4.4 – Редактирование профиля

Для создания работы, пользователю предлагается заполнить следующие поля, указанные на рисунке 4.5.

Рисунок 4.5 – Создание работы

Для регистрации, пользователю предлагается заполнить поля, указанные на рисунке 4.6.

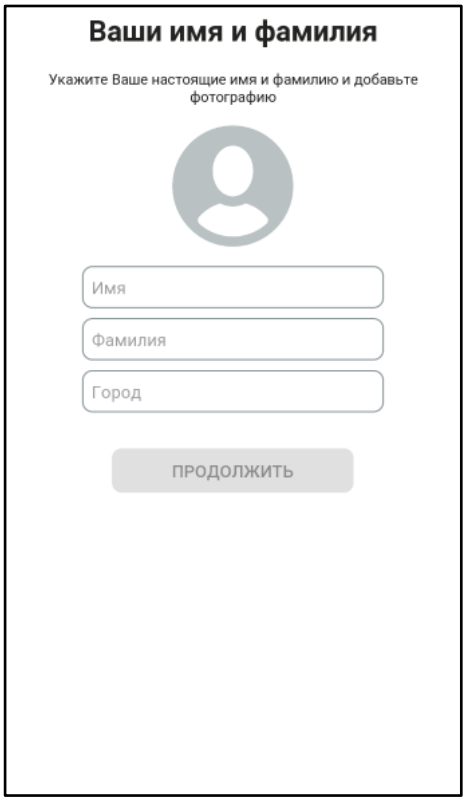

Рисунок 4.6 – Регистрация пользователя

Для добавления почты, пользователю предлагается заполнить поле, указанное на рисунке 4.7.

Рисунок 4.7 – добавление почты

Для просмотра или редактирования данных профиля, пользователю предлагается выбрать интересующий его раздел из меню, представленного на рисунке 4.8.

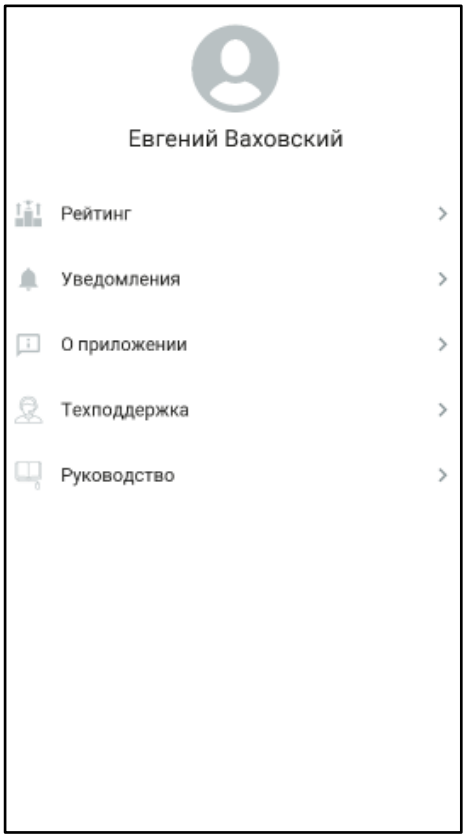

Рисунок 4.8 – Выбор разделов для просмотра или редактирования данных профиля.

5 ТЕСТИРОВАНИЕ

Для тестирование мобильного приложения было выбрано приемочное пользовательское тестирование (UAT), функциональное тестирование, интеграционное тестирование.

5.1 Методология тестирования

Приемочное пользовательское тестирование (UAT)

Приемочное пользовательское тестирование (UAT) – тестирование, проводимое конечными пользователями [17].

Ключевыми преимуществами UAT тестирования являются: обнаружение системных нарушений, обнаружение дефектов, связанных с удобством и простотой использования, короткий срок. Направления UAT тестирования:

1) операционное тестирование – проверка системы на способность выполнять свою роль в среде эксплуатации согласно бизнес-модели.

2) пользовательское тестирование – проверка пригодности системы для внедрения конечными пользователями.

3) альфа-тестирование – проверка независимой командой тестирования.

4) бета-тестирование – тестирование внешними пользователями, потенциальными клиентами.

Функциональное тестирование

Функциональное тестирование – это тестирование, в целях проверки программного продукта на соответствие установленным функциональным требованиям [18]. Функциональные требования определяют, что именно должен делать программный продукт и какие задачи он должен решать.

Интеграционное тестирование

Интеграционное тестирование – полная проверка программного обеспечения после его сборки с целью выявления ошибок, возникающих в процессе интеграции программных модулей или компонентов.

5.2 Проведение процедуры тестирования

Приемочное тестирование

В качестве направления приемочного тестирования было выбрано бета-тестирование. Бета-тестирование проводилось компанией «Radar Advertisement». В ходе тестирования были выявлены и устранены дефекты связанный с неудобством интерфейса приложения. Интерфейс был изменен согласно предложенным правкам.

Интеграционное тестирование

Для проведения интеграционного тестирования после окончательной сборки приложения были приобретены, взяты в аренду или эмулированы на эмуляторе интеграционной среды разработки Android Studio следующие смартфоны: Honor 8, Honor 8X, Pixel 2, Pixel, Honor 4c, Honor 9 Lite. На устройствах были проведены тестирования расположения элементов интерфейса на различных по размеру экранах, были протестированы все основные функции приложения. В результате тестирования ошибок не выявлено.

Кроме того, было проведено дополнительно тестирование с помощью эмулятора Android Studio на различных версиях ОС Android, начиная с 4.0.3 заканчивая 8.1.0. В результате тестирования ошибок не выявлено.

Функциональное тестирование

Набор тестов для проведения функционального тестирования и его результаты, представлен в таблице 1.

Таблица 1 – Набор тестов для проведения функционального тестирования и его результаты.

№	Описание	Шаги	Ожидаемый результат	Пройден
1	Авторизация пользователя в приложении	1. Запустить приложение 2. Пройти на экран авторизации 3. Ввести номер телефона 4. Ввести полученный СМС-код	На экране отобразится поиск работ.	Да
2	Создание работы	1. Нажать кнопку создания работы 2. Заполнить необходимые поля	На экране отобразится созданная работа	Да
3	Загрузка фотографии пользователя	1. Перейти в профиль пользователя 2. Нажать на существующую фотографию пользователя. 3. Выбрать картинку из галереи устройства 4. Нажать кнопку «Сохранить»	На экране отобразится меню настроек пользователя и изменится его фотография	Да

Продолжение таблицы 1

4	Отклик на работы	1. Перейти в поиск работ 2. Выбрать необходимую работу 3. Нажать кнопку «Откликнуться» 4. Указать сумму и количество человек(опционально)	На экране должно произойти обновление. Кнопка «Откликнуться» должна исчезнуть, на ее месте появится информация об отклике (сумма и количество человек), иконка удаления и редактирования	Да
5	Удаление отклика	1. Перейти в поиск работ или в список работ, на которые пользователь откликнулся 2. Выбрать необходимую работу 3. Нажать иконку удаления отклика	На экране должно произойти обновление, информация об отклике и иконки редактирования и удаления должны удалиться, на ее месте появиться кнопка «Откликнуться»	Да

Продолжение таблицы 1

6	Редактирование отклика	1. Перейти в поиск работ 2. Выбрать необходимую работу 3. Нажать на иконку редактирования отклика 4. Ввести новую сумму и количество человек (опционально)	На экране должно произойти обновление, информация об отклике должна обновиться на актуальную	Да
7	Редактирование работы	1. Выбрать необходимую работу 2. Нажать кнопку редактирования 3. Отредактировать работу 4. Нажать кнопку «Сохранить»	На экране должна отобразить работа с актуальной информацией	Да
8	Удаление работы	1.Выбрать необходимую работу 2. Нажать кнопку «Удалить»	На экране должно произойти обновление, удаленная работа должна удалиться из списка	Да

Продолжение таблицы 1

9	Просмотр фотографий, прикрепленных к работе	1. Выбрать необходимую работу 2. При наличии фотографии выбрать любую из них	На экране должно показаться выбранное изображение с возможность свайпа влево, вправо для перехода к следующему	Да
10	Выбор исполнителя из списка откликнувшихся	1. Выбрать необходимую работу 2. При наличии откликнувшихся исполнителей, нажать на чекбокс на карточке отклика 3. Нажать кнопку «Принять исполнителей» 4. В случае если выбранные исполнители не подойдут по заданным к работе критериям, согласиться с появившимся предупреждением о несоответствии критерий	На экране должно произойти обновление, невыбранные исполнители удалятся, у выбранных исполнителей появятся номера телефонов	Да

Окончание таблицы 1

11	Завершение работы до наступления даты начала работы	1. Выбрать необходимую работу 2. Нажать кнопку «Завершить работу»	На экране должно произойти обновление, вместо кнопки «Завершить работу» должна появиться кнопка «Оценить исполнителей»	Да

ЗАКЛЮЧЕНИЕ

В рамках выпускной квалификационной работы было спроектировано, реализовано и выпущено на рынок мобильное приложение на базе ОС Android для «Биржа работ». Объем кода приблизительно составил 15000 строк кода.

Для достижения поставленной цели были решены следующие задачи:

1) Выполнен анализ предметной области.

2) Выполнено проектирование мобильного приложение с использованием паттерна архитектурного проектирования MVP.инструментом взаимодействия с сервером GraphQL, библиотеки для работы PUSH уведомлений Firebase Cloud Messaging.

3) Реализовано мобильное приложение с учетом необходимых функциональных и нефункциональных требований.

4) Проведено тестирование мобильного приложения с использованием UAT тестирования, функционального тестирования и интеграционного тестирования.

Разработанное мобильное приложение было сдано заказчику в опытную эксплуатацию и выпущено на рынок с помощью магазина мобильных приложений Google Play. Приложение получило название «Banomo» и имеет более 100 установок.

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. The Mobile Economy. – https://clck.ru/GLQHb. Дата обращения: 22.02.2019.

2. Hymas C. A fifth of 16-24-year olds spend more than seven hours a day online every day of the week, exclusive Ofcom figures reveal. – https://www.telegraph.co.uk/news/2018/08/11/fifth-16-24-year-olds-spend-seven-hours-day-online-every-day/. Дата обращения: 22.02.2019.

3. Отчет «Состояние рынка мобильных приложений 2019». – https://apptractor.ru/measure/app-store-analytics/otchet-sostoyanie-ryinka-mobilnyih-prilozheniy-2019-ot-app-annie.html. Дата обращения: 22.02.2019.

4. Profi.ru – Википедия. – https://ru.wikipedia.org/wiki/Profi.ru. Дата обращения: 22.02.2019.

5. Максим Б., Отзыв на приложение Profi.Ru – https://clck.ru/GLQVw. Дата обращения: 22.02.2019.

6. Николай Б., Отзыв на приложение Profi.Ru – https://clck.ru/GLQWp. Дата обращения: 22.02.2019.

7. YouDo – Википедия. – https://ru.wikipedia.org/wiki/Youdo. Дата обращения: 23.02.2019.

8. Android Studio – Википедия. – https://ru.wikipedia.org/wiki/Android_Studio. Дата обращения: 23.01.2019.

9. Android – Википедия. – https://ru.wikipedia.org/wiki/Android. Дата обращения: 23.01.2019.

10. Xamarin – Википедия. – https://ru.wikipedia.org/wiki/Xamarin. Дата обращения: 23.01.2019.

11. REST. – https://clck.ru/GLQus. Дата обращения: 05.09.2018.

12. GraphQL – Википедия. – https://en.wikipedia.org/wiki/GraphQL. Дата обращения: 05.09.2018.

13. Firebase Cloud Messaging. – https://firebase.google.com/docs/cloud-messaging. Дата обращения: 11.08.2018.

14. Тепляков С. В. Паттерны проектирования на платформе .NET. – Питер, 2016. – 320 с.

15. Fowler M. GUI Architectures. – http://www.rusdoc.ru/articles/18358/. Дата обращения: 09.08.2018.

16. Fowler M. GUI Architectures. – https://www.martinfowler.com/eaaDev/uiArchs.html. Дата обращения: 09.08.2018.

17. Пользовательское тестирование (UAT). – https://www.performance-lab.ru/uat-polzovatelskoe-testirovanie. Дата обращения: 12.12.2018.

18. Функциональное тестирование – Википедия. – https://clck.ru/GLQty. Дата обращения: 12.12.2018.

19. Мартин Р. Чистый код: создание, анализ и рефакторинг. – Питер СПб, 2018. – 464 с.

20. Влиссидес Д. Приемы объектно-ориентированного проектирования. Паттерны проектирования / Р. Джонсон, Р. Хелм, Э. Гамма. – Питер, 2016. – 366 с.

21. Гриффитс Д. Head First. Программирование для Android – Питер, 2016. – 704 с.

22. Жемеров Д. Kotlin в действии / Д. Жемеров, С. Исакова. – ДМК Пресс, 2018. – 402 с.

23. Ваховский Е.Д. Исходный код мобильного приложения Banomo. - https://github.com/darkeneez/BanomoDiplom

Приложение А

Исходный код контроллеров

Листинг 1 - BasePresenter

```
package net.darkeneez.banomo.ui.base;

import net.darkeneez.banomo.ui.base.base_ui_interfaces.IBasePresenter;
import net.darkeneez.banomo.ui.base.base_ui_interfaces.IBaseView;

public class BasePresenter<T extends IBaseView> implements IBasePresenter<T> {

    private T mView;

    @Override
    public void attach(T view) {
        mView = view;
    }

    @Override
    public void detach() {
        mView = null;
    }

    @Override
    public T getAttachView() {
        return mView;
    }
}
```

Листинг 2 – ComplaintPresenter

```
package net.darkeneez.banomo.ui.complaint

import anet.darkeneez.banomo.CreateNewComplaintMutation
import com.apollographql.apollo.ApolloCall
import com.apollographql.apollo.api.Response
import com.apollographql.apollo.exception.ApolloException
import net.darkeneez.banomo.R
import net.darkeneez.banomo.application.apollo.BanomoApolloClient
import net.darkeneez.banomo.application.models.user.User
import net.darkeneez.banomo.ui.base.BasePresenter

class ComplaintPresenter<V : IComplainActivity> : BasePresenter<V>(), IComplaintPresenter<V>
{

    private var offerId: Long = -1
    private var jobId: Long = -1

    override fun setUpParams(offerId: Long, jobId: Long) {
        this.offerId = offerId
        this.jobId = jobId
    }

    override fun onSendClicked(description: String, point: String) {
        val user = User().instance

        BanomoApolloClient.getApolloClient().mutate(CreateNewComplaintMutation.builder()
                .phone(user.getsPhone())
                .token(user.getsToken())
```

```
                .jobId(jobId)
                .offerId(offerId)
                .description(description)
                .theme(point)
                .build())
                .enqueue(object : ApolloCall.Callback<CreateNewComplaintMutation.Data>() {
                    override fun onResponse(response:
Response<CreateNewComplaintMutation.Data>) {
                        if (response.hasErrors())
                            return

                        attachView.runOnUiThread { attachView.onComplainSendSuccessful() }
                    }

                    override fun onFailure(e: ApolloException) {
                        attachView.ShowSnackBar(e.message, R.color.colorGreen)
                    }
                })
    }
}
```

Листинг 3 – CreateJobPresenter

```
package net.darkeneez.banomo.ui.create_job;


import com.esafirm.imagepicker.model.Image;


import net.darkeneez.banomo.R;

import net.darkeneez.banomo.application.models.job.typesJob.TypesJob;

import net.darkeneez.banomo.application.models.regions.Locations;

import net.darkeneez.banomo.application.models.user.User;

import net.darkeneez.banomo.application.util.UPLoadUtil.UploadImageToServer;

import net.darkeneez.banomo.ui.base.BasePresenter;

import net.darkeneez.banomo.ui.create_job.job_interfaces.IJobCreated;

import net.darkeneez.banomo.ui.create_job.job_interfaces.IJobUpdated;

import net.darkeneez.banomo.ui.create_job.job_interfaces.IJobUploaded;

import net.darkeneez.banomo.ui.create_job.mutations.CreateJobMutation;

import net.darkeneez.banomo.ui.create_job.mutations.LoadJobMutation;

import net.darkeneez.banomo.ui.create_job.mutations.UpdateJobMutation;


import java.text.SimpleDateFormat;

import java.util.ArrayList;

import java.util.List;

import java.util.Locale;


import anet.darkeneez.banomo.fragment.Job;
```

```
public class CreateJobPresenter<V extends ICreateJob> extends BasePresenter<V> implements
ICreateJobPresenter<V> {

    private List<Image> mImagesList;
    private List<String> mImagesUrlList;

    private long mJobId = -1;
    private long mJobLocationId = -1;
    private long mJobTypeId = -1;
    private long mJobTypePayment = -1;
    private long mJobExecutorCount = -1;

    private boolean mIsNeedClear = false;
    private boolean mIsEdit = false;

    private String mJobDescription = null, mJobDate = null;

    @Override
    public void setEditMode(long isEdit) {
        mIsEdit = true;
        mJobId = isEdit;
    }

    @Override
    public void setImagesList(List<Image> imagesList) {
        mImagesList = imagesList;
        mImagesUrlList = new ArrayList<>();
    }

    @Override
    public void setJobLocationIdByUser() {
        mJobLocationId = new User().getInstance().getLocation();
        setAttachViewFields();
    }
```

```
@Override
public void setJobLocationId(long jobLocationId) {
    mJobLocationId = jobLocationId;
}

@Override
public void setJobTypeId(long jobTypeId) {
    mJobTypeId = jobTypeId;
}

@Override
public void setJobTypePayment(long typePayment) {
    mJobTypePayment = typePayment;

    if (typePayment == 2)
        setJobExecutorCount(0);

    getAttachView().enableExecutorCount(mJobTypePayment == 1 || mJobTypePayment == -1);
}

@Override
public void setIsNeedClear(boolean isNeedClear) {
    mIsNeedClear = isNeedClear;
}

@Override
public boolean isNeedClear() {
    return mIsNeedClear;
}

@Override
public void setJobDescription(String jobDescription) {
    mJobDescription = jobDescription;
}
```

```
@Override
public void setJobDate(String jobDate) {
    mJobDate = jobDate;
}

@Override
public void setJobExecutorCount(long executorCount) {
    mJobExecutorCount = executorCount;
}

@Override
public void setJobExecutorCount(String count) {
    if (mJobTypePayment == 2)
        mJobExecutorCount = 0;
    else if (mJobTypePayment == 1) {
        mJobExecutorCount = Long.parseLong(count);
    }
}

@Override
public void loadJob(long id) {
    LoadJobMutation.loadJob(id, new IJobUploaded() {
        @Override
        public void onJobUploadError(String error) {
            getAttachView().runOnUiThread(() -> getAttachView().onJobLoadError(error));
        }

        @Override
        public void onJobUploadSuccess(Job jobFragment) {
            getAttachView().runOnUiThread(() -> {
                readJobFragment(jobFragment);
                getAttachView().onJobLoadSuccess(jobFragment.id());
            });
        }
    });
```

```
    }

    private void readJobFragment(Job jobFragment) {
        setJobTypePayment(jobFragment.typePayment());
        setJobTypeId(jobFragment.typeJobs());
        setJobLocationId(jobFragment.locationId());
        setImagesList(createImageList(jobFragment.images()));
        if (mIsEdit)
            setJobDate(convertToClientDate(jobFragment.date()));
        setJobDescription(jobFragment.description());
        setJobExecutorCount(jobFragment.persons());

        setAttachViewFields();
    }

    private String convertToClientDate(String oldDate) {
        String clientDate = null;

        SimpleDateFormat sdf = new SimpleDateFormat("yyyy-MM-dd'T'HH:mm:ss'Z'", new
Locale("ru"));
        SimpleDateFormat outSDF = new SimpleDateFormat("dd.MM.yyyy HH:mm", new Locale("ru"));
        try {
            clientDate = outSDF.format(sdf.parse(oldDate));
        } catch (Exception ex) {
            ex.printStackTrace();
        }
        return clientDate;
    }


    private List<Image> createImageList(List<Job.Image> images) {
        List<Image> imagesModel = new ArrayList<>();

        long count = 0;

        for (Job.Image image : images) {
```

```
            if (!image.photo().equals("")) {
                imagesModel.add(new Image(count, "JobsImages", image.photo()));
                count++;
            }
        }

        return imagesModel;
    }

    private void setAttachViewFields() {
        V attachView = getAttachView();

        attachView.setDate(getDateByValue());
        attachView.setDescription(getDescriptionByValue());
        attachView.setJobType(getJobTypeById());
        attachView.setLocation(getLocationById());
        attachView.setJobTypePayment(getJobTypePaymentById());
        attachView.setExecutorCount(mJobExecutorCount == -1 ? "" :
(String.format(Locale.getDefault(), "%d", mJobExecutorCount)));
        attachView.setImagesList(mImagesList);
    }

    @Override
    public void clearButtonClicked() {
        mJobTypeId = -1;
        mJobLocationId = -1;
        mJobExecutorCount = -1;
        mJobTypePayment = -1;
        mJobDate = "Выбрать";
        mJobDescription = "";

        setAttachViewFields();
    }

    @Override
    public String getJobTypePaymentById() {
```

```
        switch ((int) mJobTypePayment) {
            case 1: {
                return "Почасовая";
            }
            case 2: {
                return "За работу";
            }
            default: {
                return "Выбрать";
            }
        }
    }

    @Override
    public String getJobTypeById() {
        TypesJob typesJob = new TypesJob();
        typesJob.loadCacheTypesJob();

        return mJobTypeId == -1 ? "Выбрать" : typesJob.getTypeName(mJobTypeId);
    }

    @Override
    public String getLocationById() {
        return mJobLocationId == -1 ? "Выбрать" : new
Locations().getInstance().getLocationById(mJobLocationId).getName();
    }

    @Override
    public String getDateByValue() {
        return mJobDate == null ? "Выбрать" : mJobDate;
    }

    @Override
    public String getDescriptionByValue() {
        return mJobDescription == null ? "" : mJobDescription;
    }
```

```
private String convertDateToServer(String oldDate) {
    String serverDate = "";

    SimpleDateFormat sdf = new SimpleDateFormat("dd.MM.yyyy HH:mm", new Locale("ru"));
    SimpleDateFormat outSDF = new SimpleDateFormat("MM.dd.yyyy HH:mm", new Locale("ru"));
    try {
        serverDate = outSDF.format(sdf.parse(oldDate));
    } catch (Exception ex) {
        ex.printStackTrace();
    }
    return serverDate;
}

@Override
public void createJob() {
    if (!checkFields())
        return;

    if (mIsEdit) {
        editJob();
        return;
    }

    if (mImagesList != null && mImagesList.size() > 0) {
        List<Image> tempImagesListUpload = new ArrayList<>();

        for (Image image : mImagesList) {
            if (image.getName().contains("JobsImages"))
                mImagesUrlList.add(image.getPath() + "|");
            else
                tempImagesListUpload.add(image);
        }

        mImagesList = new ArrayList<>(tempImagesListUpload);
```

```
            if (mImagesList.size() > 0)
                uploadImages();
            else
                createJob();
        } else {

            User user = new User().getInstance();

            CreateJobMutation.createJob(user.getsPhone(), user.getsToken(), mJobLocationId,
mJobTypeId, mJobTypePayment, mJobExecutorCount, convertDateToServer(mJobDate),
mJobDescription, mImagesUrlList, new IJobCreated() {
                @Override
                public void onCreateJobError(String error) {
                    getAttachView().runOnUiThread(() ->
getAttachView().onCreateJobError(error));
                }

                @Override
                public void onCreateJobSuccess(long jobId) {
                    getAttachView().runOnUiThread(() ->
getAttachView().onCreateJobSuccess(jobId));
                }
            });
        }
    }

    private boolean checkFields() {
        return checkJobLocation() && checkJobDate() && checkJobType() &&
checkJobTypePayment() && checkJobExecutorCount() && checkJobDescription();
    }

    private boolean checkJobDescription() {
        if (mJobDescription.length() < 15) {
            getAttachView().onCreateJobError("Опишите работу подробнее. Минимум 15
символов");
            return false;
        }
        return true;
```

```
}

private boolean checkJobExecutorCount() {
    if (mJobTypePayment == 1 && mJobExecutorCount == -1) {
        getAttachView().onCreateJobError("Укажите количество работников");
        return false;
    }

    return true;
}

private boolean checkJobTypePayment() {
    if (mJobTypePayment == -1) {
        getAttachView().onCreateJobError("Выберите тип оплаты");
        return false;
    }
    return true;
}

private boolean checkJobType() {
    if (mJobTypeId == -1) {
        getAttachView().onCreateJobError("Выберите тип работы");
        return false;
    }

    return true;
}

private boolean checkJobDate() {
    if (mJobDate.equals("Выбрать")) {
        getAttachView().onCreateJobError("Выберите дату начала работы");
        return false;
    }

    return true;
```

```
    }

    private boolean checkJobLocation() {
        if (mJobLocationId == -1) {
            getAttachView().onCreateJobError("Выберите город");
            return false;
        }
        return true;
    }

    @Override
    public void editJob() {
        if (mImagesList != null && mImagesList.size() > 0) {
            List<Image> tempImagesListUpload = new ArrayList<>();

            for (Image image : mImagesList) {
                if (image.getName().contains("JobsImages"))
                    mImagesUrlList.add(image.getPath() + "|");
                else
                    tempImagesListUpload.add(image);
            }

            mImagesList = new ArrayList<>(tempImagesListUpload);

            if (mImagesList.size() > 0)
                uploadImages();
            else
                editJob();
        } else {

            User user = new User().getInstance();

            UpdateJobMutation.updateJob(user.getsPhone(), user.getsToken(), mJobId,
mJobLocationId, mJobTypeId, mJobTypePayment, mJobExecutorCount,
convertDateToServer(mJobDate), mJobDescription, mImagesUrlList, new IJobUpdated() {
```

```
                @Override
                public void onJobUpdated(String error) {
                    getAttachView().runOnUiThread(() ->
getAttachView().onCreateJobError(error));
                }

                @Override
                public void onJobUpdated(long id) {
                    getAttachView().runOnUiThread(() ->
getAttachView().onJobUpdated(mJobId));
                }
            });
        }
    }

    private void uploadImages() {

        UploadImageToServer uploadImageToServer = new UploadImageToServer();

        uploadImageToServer.setUploadListener(new UploadImageToServer.UploadListener() {
            @Override
            public void onSuccess(String result) {
                mImagesUrlList.add(result.substring(result.lastIndexOf(':') + 2,
result.lastIndexOf('"')) + "|");
                createJob();
            }

            @Override
            public void onError(String error) {
                getAttachView().runOnUiThread(() -> getAttachView().ShowSnackBar(error,
R.color.colorRed));
            }
        });
        uploadImageToServer.execute(mImagesList.remove(0).getPath());
    }
}
```

Листинг 4 – JobSearchPresenter

```
package net.darkeneez.banomo.ui.jobs_search;



import android.content.Intent;


import com.apollographql.apollo.ApolloCall;

import com.apollographql.apollo.api.Response;

import com.apollographql.apollo.exception.ApolloException;


import net.darkeneez.banomo.R;

import net.darkeneez.banomo.application.apollo.BanomoApolloClient;

import net.darkeneez.banomo.application.models.user.User;

import net.darkeneez.banomo.application.util.JobSearchFilter;

import net.darkeneez.banomo.ui.base.BasePresenter;

import net.darkeneez.banomo.ui.filter_activity.FilterActivity;


import java.util.ArrayList;

import java.util.List;


import javax.annotation.Nonnull;


import anet.darkeneez.banomo.GetJobsQuery;

import anet.darkeneez.banomo.fragment.Job;


public class JobSearchPresenter<V extends IJobSearch> extends BasePresenter<V> implements
IJobSearchPresenter<V> {


    private static JobSearchFilter mJobSearchFilter = new JobSearchFilter();


    public void applyFilter(JobSearchFilter jobSearchFilter) {

        mJobSearchFilter = jobSearchFilter;

        loadJobs(jobSearchFilter);

    }
```

```
    @Override
    public void detach() {
        super.detach();
    }

    @Override
    public Intent getFilterParams() {
        Intent intent = new Intent(getAttachView().getActivity(), FilterActivity.class);
        intent.putExtra("JOB_SEARCH_FILTER", mJobSearchFilter);
        return intent;
    }

    @Override
    public void loadJobs(Long jobType, Long executorCount, Long typePayment, Long regionID,
String date, boolean showUserJob) {
        if (regionID != null && regionID == -2) {
            regionID = new User().getInstance().getLocation();
            mJobSearchFilter.setmRegionID(regionID);
        }

        User user = new User().getInstance();

        boolean userJobShow;

        if (user.getsPhone().equals("") || user.getsToken().equals(""))
            userJobShow = false;
        else userJobShow = showUserJob;

        BanomoApolloClient.getApolloClient().query(
                GetJobsQuery.builder()
                        .typeJobs(jobType)
                        .persons(executorCount)
                        .typePayment(typePayment)
                        .date(date)
                        .locationId(regionID)
                        .phone(userJobShow ? user.getsPhone() : null)
```

```
                        .token(userJobShow ? user.getsToken() : null)
                        .notShowMyWork(userJobShow)
                        .build())
                .enqueue(new ApolloCall.Callback<GetJobsQuery.Data>() {
                    @Override
                    public void onResponse(@Nonnull final Response<GetJobsQuery.Data>
response) {
                        if (response.hasErrors()) {
                            getAttachView().runOnUiThread(() -> {

getAttachView().ShowSnackBar(response.errors().get(0).message(), R.color.colorRed);
                                getAttachView().stopRefresh();
                            });

                            return;
                        }

                        final List<Job> jobs = new ArrayList<>();

                        for (GetJobsQuery.JobsSearch job : response.data().JobsSearch())
                            jobs.add(job.fragments().job());

                        getAttachView().runOnUiThread(() ->
                                getAttachView().updateRecycler(jobs));
                    }

                    @Override
                    public void onFailure(@Nonnull final ApolloException e) {
                        e.printStackTrace();
                        getAttachView().runOnUiThread(() -> {
                            getAttachView().ShowSnackBar(e.getMessage(), R.color.colorRed);
                            getAttachView().stopRefresh();
                        });

                    }
                });
```

```
    }


    @Override
    public void loadJobs(JobSearchFilter jobSearchFilter) {
        Long jobType = jobSearchFilter.getmJobType() == -1 ? null :
jobSearchFilter.getmJobType();
        Long executorCount = jobSearchFilter.getmExecutorCount() == -1 ? null :
jobSearchFilter.getmExecutorCount();
        Long typePayment = jobSearchFilter.getmTypePayment() == -1 ? null :
jobSearchFilter.getmTypePayment();
        Long region = jobSearchFilter.getmRegionID() == -1 ? null :
jobSearchFilter.getmRegionID();


        loadJobs(jobType, executorCount, typePayment, region, jobSearchFilter.getmDate(),
jobSearchFilter.ismNotShowUserJob());
    }


    @Override
    public void loadJobs() {
        loadJobs(mJobSearchFilter);
    }
}
```

Листинг 5 – LoadPresenter

```
package net.darkeneez.banomo.ui.load;


import com.google.gson.Gson;
import com.google.gson.reflect.TypeToken;


import net.darkeneez.banomo.R;
import net.darkeneez.banomo.application.apollo.BanomoErrors;
import net.darkeneez.banomo.application.models.job.typesJob.OnUpdatedListener;
import net.darkeneez.banomo.application.models.regions.Location;
import net.darkeneez.banomo.application.models.regions.Locations;
import net.darkeneez.banomo.application.models.user.User;
import net.darkeneez.banomo.application.models.user.UserInterfaces.IUserUpdated;
import net.darkeneez.banomo.application.util.JSONResourceReader;
import net.darkeneez.banomo.ui.base.BasePresenter;
```

```
import org.jetbrains.annotations.NotNull;


import java.lang.reflect.Type;
import java.util.List;


public class LoadPresenter<V extends ILoad> extends BasePresenter<V> implements
ILoadPresenter<V> {


    private boolean mTypesJobs = false;
    private boolean mUserJobs = false;
    private boolean mLocations = false;


    private boolean mUserIsAuth = false;


    @Override
    public void loadApp() {
        loadLocations();
        loadTypesJob();
        loadUser();
    }


    private void checkLoad() {
        if (mTypesJobs && mUserJobs && mLocations) {
            if (!mUserIsAuth)
                getAttachView().runOnUiThread(() -> getAttachView().onLoadingEnd());
            else
                getAttachView().runOnUiThread(() -> getAttachView().startMainActivity());
        }
    }


    private void loadLocations() {
        Gson gson = new Gson();
        Type type = new TypeToken<List<Location>>() {
        }.getType();
```

```
        LoadActivity loadActivity = (LoadActivity) getAttachView();

        JSONResourceReader jsonResourceReader = new JSONResourceReader();
        jsonResourceReader.readJsonString(loadActivity.getResources(), R.raw.locations);

        List<Location> loadedLocations = gson.fromJson(jsonResourceReader.getJsonString(),
type);

        Locations locations = new Locations().getInstance();
        locations.setLocations(loadedLocations);

        mLocations = true;

        checkLoad();
    }

    private void loadTypesJob() {
        net.darkeneez.banomo.application.models.job.typesJob.TypesJob typesJob = new
net.darkeneez.banomo.application.models.job.typesJob.TypesJob();

        typesJob.setMITypesJob(new OnUpdatedListener() {
            @Override
            public void onUpdateError(@NotNull String error) {
                getAttachView().runOnUiThread(() -> getAttachView().ShowSnackBar(error,
R.color.colorRed));
                mTypesJobs = true;
                checkLoad();
            }

            @Override
            public void onUpdateSuccess() {
                mTypesJobs = true;
                checkLoad();
            }
        });
        typesJob.loadTypesJob();
```

```
    }

    private void loadUser() {
        final User user = new User().getInstance();
        user.setOnUserUpdated(new IUserUpdated() {
            @Override
            public void onError(final String error) {
                switch (error) {
                    case BanomoErrors.FAILED_TO_EXECUTE_HTTP_CALL: {
                        getAttachView().ShowSnackBar("Отсутствует интернет подключение",
R.color.colorGray);
                        mUserIsAuth = true;
                        mUserJobs = true;
                        break;
                    }
                    case BanomoErrors.HTTP_502_BAD_GATEWAY: {
                        getAttachView().ShowSnackBar("Сервер не доступен",
R.color.colorGray);
                        mUserIsAuth = true;
                        mUserJobs = true;
                        break;
                    }
                    case "USER_NOT_EXECUTE": {
                        mUserJobs = true;
                        mUserIsAuth = false;
                        break;
                    }
                    default: {
                        getAttachView().ShowSnackBar(error, R.color.colorRed);
                        mUserIsAuth = false;
                        mUserJobs = true;
                        break;
                    }
                }

                checkLoad();
```

```
            }

            @Override
            public void onSuccess() {
                mUserJobs = true;
                mUserIsAuth = true;
                checkLoad();
            }
        });

        user.loadUser();
    }
}
```

Листинг 6 – PhoneInputPresenter

```
package net.darkeneez.banomo.ui.login.phone_input;

import com.apollographql.apollo.ApolloCall;
import com.apollographql.apollo.api.Response;
import com.apollographql.apollo.exception.ApolloException;

import net.darkeneez.banomo.R;
import net.darkeneez.banomo.application.apollo.BanomoApolloClient;
import net.darkeneez.banomo.application.models.user.User;
import net.darkeneez.banomo.application.models.user.UserInterfaces.IUserCreated;
import net.darkeneez.banomo.application.models.user.UserInterfaces.IUserUpdated;
import net.darkeneez.banomo.ui.base.BasePresenter;

import javax.annotation.Nonnull;

import anet.darkeneez.banomo.CheckPhoneQuery;

public class PhoneInputPresenter<V extends IPhoneInput> extends BasePresenter<V> implements
IPhoneInputPresenter<V> {

    private String mPhone;
```

```
    private void onCreateUserError(String error) {
        switch (error) {
            case "Ошибка работы сервера: HTTP 502 Bad Gateway": {
                getAttachView().ShowSnackBar("Сервер не доступен, повторите попытку позже",
R.color.colorRed);
                break;
            }
            default: {
                getAttachView().ShowSnackBar(error, R.color.colorRed);
                break;
            }
        }
    }

    @Override
    public boolean checkPhone() {
        return mPhone != null && mPhone.length() == 11;
    }

    @Override
    public void setPhone(String phone) {
        StringBuilder buffer = new StringBuilder(phone);
        buffer.insert(0, "7");

        mPhone = buffer.toString();
    }

    @Override
    public void checkPhoneIfUsed() {
        BanomoApolloClient.getApolloClient().query(CheckPhoneQuery.builder()
                .phone(mPhone)
                .build())
                .enqueue(new ApolloCall.Callback<CheckPhoneQuery.Data>() {
                    @Override
```

```
                    public void onResponse(@Nonnull Response<CheckPhoneQuery.Data> response)
{
                        if (response.hasErrors()) {
                            getAttachView().runOnUiThread(() ->
getAttachView().ShowSnackBar(response.errors().get(0).message(), R.color.colorRed));
                            return;
                        }


                        if (response.data().CheckPhone().ok())
                            createNewUser();
                        else
                            getAttachView().runOnUiThread(() ->
getAttachView().newUserPhoneIsUsed());
                    }


                    @Override
                    public void onFailure(@Nonnull ApolloException e) {
                        getAttachView().runOnUiThread(() ->
getAttachView().ShowSnackBar(e.getLocalizedMessage(), R.color.colorRed));
                    }
                });


    }


    @Override
    public void loadUser() {
        User user = new User().getInstance();
        user.setOnUserUpdated(new IUserUpdated() {
            @Override
            public void onError(String error) {
                switch (error) {
                    case "sql: no rows in result set": {
                        getAttachView().runOnUiThread(() ->
                                getAttachView().userIsNotRegistered());
                        break;
                    }
                    default: {
```

```
                        getAttachView().runOnUiThread(() ->
getAttachView().ShowSnackBar(error, R.color.colorRed));
                        break;
                    }
                }
            }

            @Override
            public void onSuccess() {
                getAttachView().runOnUiThread(() -> getAttachView().startPhoneValidate(0));
            }
        });
        user.loadUser(mPhone);
    }

    @Override
    public void createNewUser() {
        User user = new User().getInstance();
        user.setOnUserCreated(new IUserCreated() {
            @Override
            public void onError(String error) {
                onCreateUserError(error);
            }

            @Override
            public void onSuccess(long sms) {
                getAttachView().startPhoneValidate(sms);
            }
        });
        user.createUser(mPhone, getAttachView().getRegistrationUser());
    }
}
```

Листинг 7 – PhoneVerificationPresenter

```
package net.darkeneez.banomo.ui.login.phone_verification;

import com.apollographql.apollo.ApolloCall;
```

```
import com.apollographql.apollo.api.Response;
import com.apollographql.apollo.exception.ApolloException;

import net.darkeneez.banomo.R;
import net.darkeneez.banomo.application.apollo.BanomoApolloClient;
import net.darkeneez.banomo.application.models.user.RegistrationUser;
import net.darkeneez.banomo.application.models.user.User;
import net.darkeneez.banomo.application.models.user.UserInterfaces.IUserPhotoUploaded;
import net.darkeneez.banomo.application.models.user.UserInterfaces.IUserUpdated;
import net.darkeneez.banomo.application.util.UPLoadUtil.UploadImageToServer;
import net.darkeneez.banomo.ui.base.BasePresenter;

import javax.annotation.Nonnull;

import anet.darkeneez.banomo.CreateNewSMSMutation;

public class PhoneVerificationPresenter<V extends IPhoneVerification> extends
BasePresenter<V> implements IPhoneVerificationPresenter<V> {

    private int mCounter = 0;
    private Long mSMS;

    private RegistrationUser mRegistrationUser;

    private boolean isUserUpdate, isPhotoUploaded;

    @Override
    public void setRegistrationUser(RegistrationUser registrationUser) {
        mRegistrationUser = registrationUser;
    }

    @Override
    public void onTextChanged(String text) {

        if (text.equals("")) {
            mCounter--;
```

```
            getAttachView().focusNext(mCounter);
        }

        if (text.length() == 1) {
            if (++mCounter > 3) {
                checkCode();
                return;
            }
            getAttachView().focusNext(mCounter);
        }
    }

    @Override
    public void readSMS(Long sms) {
        if (sms == 0) {
            requestCode();
            return;
        }
        mSMS = sms;
    }

    @Override
    public void checkCode() {
        getAttachView().hideKeyBoard();

        if (getAttachView().getUserInputSMS().equals(mSMS)) {
            final User user = new User().getInstance();
            user.setOnUserUpdated(new IUserUpdated() {
                @Override
                public void onError(String error) {
                    getAttachView().ShowSnackBar(error, R.color.colorRed);
                }


                @Override
```

```
                public void onSuccess() {
                    getAttachView().runOnUiThread(() -> {
                        if (mRegistrationUser == null)
                            getAttachView().startMainActivity();
                        else
                            uploadNewUser();
                    });
                }
            });
            user.verificationUserPhone(mSMS);
            return;
        } else {
            mCounter = 0;
            getAttachView().onCodeError();
        }

    }

    @Override
    public void uploadNewUser() {
        User user = new User().getInstance();

        if (mRegistrationUser.getmAvatar() == null) {
            isPhotoUploaded = true;
        } else {

            UploadImageToServer uploadImageToServer = new UploadImageToServer();
            uploadImageToServer.setUploadListener(new UploadImageToServer.UploadListener() {
                @Override
                public void onSuccess(String result) {

                    String avatar = result.substring(result.lastIndexOf(':') + 2,
result.lastIndexOf('"'));

                    user.setOnUserPhotoUploaded(new IUserPhotoUploaded() {
                        @Override
```

```
                        public void onError(String error) {
                            isPhotoUploaded = true;
                            getAttachView().runOnUiThread(() ->
getAttachView().ShowSnackBar(error, R.color.colorRed)
                            );
                            checkNewUserCreated();
                        }

                        @Override
                        public void onSuccess(String uri) {
                            user.setUserAvatar(uri);
                            isPhotoUploaded = true;
                            checkNewUserCreated();
                        }
                    });
                    user.uploadPhoto(avatar);
                }

                @Override
                public void onError(String error) {
                    isPhotoUploaded = true;
                    getAttachView().runOnUiThread(() -> getAttachView().ShowSnackBar(error,
R.color.colorRed)
                    );
                    checkNewUserCreated();
                }
            });
            uploadImageToServer.execute(mRegistrationUser.getmAvatar());
        }

        user.setOnUserUpdated(new IUserUpdated() {
            @Override
            public void onError(String error) {
                getAttachView().ShowSnackBar(error, R.color.colorRed);
            }
```

```
            @Override
            public void onSuccess() {
                isUserUpdate = true;
                checkNewUserCreated();
            }
        });
        user.updateUser(mRegistrationUser.getmName(), mRegistrationUser.getmSecondName(),
mRegistrationUser.getmRegionId());
    }

    private void checkNewUserCreated() {
        if (isUserUpdate && isPhotoUploaded) {
            getAttachView().runOnUiThread(() -> getAttachView().startMainActivity());
        }
    }

    @Override
    public void requestCode() {
        mCounter = 0;
        getAttachView().clearAll();

        BanomoApolloClient.getApolloClient().mutate(CreateNewSMSMutation.builder()
                .phone(new User().getInstance().getsPhone())
                .build())
                .enqueue(new ApolloCall.Callback<CreateNewSMSMutation.Data>() {
                    @Override
                    public void onResponse(@Nonnull final Response<CreateNewSMSMutation.Data>
response) {
                        if (response.hasErrors()) {
                            getAttachView().ShowSnackBar(response.errors().get(0).message(),
R.color.colorRed);
                            return;
                        }

                        mSMS = response.data().newSMS().sms();
                    }
```

```
                    @Override
                    public void onFailure(@Nonnull ApolloException e) {
                        e.printStackTrace();
                    }
                });
    }
}
```

Листинг 8 – MainActivityPresenter

```
package net.darkeneez.banomo.ui.main;

import android.view.MenuItem;

import androidx.fragment.app.Fragment;
import androidx.fragment.app.FragmentManager;
import androidx.fragment.app.FragmentTransaction;

import com.apollographql.apollo.ApolloCall;
import com.apollographql.apollo.api.Response;
import com.apollographql.apollo.exception.ApolloException;
import com.google.android.gms.common.ConnectionResult;
import com.google.android.gms.common.GoogleApiAvailability;
import com.google.firebase.iid.FirebaseInstanceId;

import net.darkeneez.banomo.R;
import net.darkeneez.banomo.application.apollo.BanomoApolloClient;
import net.darkeneez.banomo.application.models.user.User;
import net.darkeneez.banomo.ui.AuthFragment;
import net.darkeneez.banomo.ui.base.BasePresenter;
import net.darkeneez.banomo.ui.create_job.CreateJob;
import net.darkeneez.banomo.ui.jobs_search.JobSearch;
import net.darkeneez.banomo.ui.user_case.UserCase;
import net.darkeneez.banomo.ui.user_manager.settings.SettingsFragment;
```

```
import javax.annotation.Nonnull;


import anet.darkeneez.banomo.RegisterDeviceForPushMutation;



public class MainActivityPresenter<V extends IMainActivity> extends BasePresenter<V>
implements IMainPresenter<V> {



    public boolean NavigationMenuSelected(MenuItem item) {


        User user = new User().getInstance();
        if (!user.isUserAuthorized() && item.getItemId() != R.id.action_search) {
            setMenuFragment(new AuthFragment(), false, "Banomo");
            return false;
        }


        switch (item.getItemId()) {
            case R.id.action_search: {
                setMenuFragment(new JobSearch(), true, "Поиск работы", "JobSearch");
                break;
            }
            case R.id.action_add_offer: {
                setMenuFragment(new CreateJob(), false, "Новая работа");
                break;
            }
            case R.id.action_orders: {
                setMenuFragment(new UserCase(), true, "Ваши работы", "UserCase");
                break;
            }
            case R.id.action_settings: {
                setMenuFragment(new SettingsFragment(), false, "Настройки");
                break;
            }
```

```
            default: {
                return false;
            }
        }
        return true;
    }

    public void setMenuFragment(Fragment fragment, boolean showIcon, String title) {
        MainActivity mainActivity = (MainActivity) getAttachView();

        FragmentManager fragmentManager = mainActivity.getSupportFragmentManager();

        if (fragmentManager != null) {

            FragmentTransaction fragmentTransaction = fragmentManager.beginTransaction();

            fragmentTransaction.replace(R.id.main_fragment_container, fragment);

            fragmentTransaction.commitAllowingStateLoss();

            getAttachView().setToolbarIconVisible(showIcon);
            getAttachView().setToolbarTitle(title);
        }
    }

    public void setMenuFragment(Fragment fragment, boolean showIcon, String title, String
tag) {
        MainActivity mainActivity = (MainActivity) getAttachView();

        FragmentManager fragmentManager = mainActivity.getSupportFragmentManager();

        if (fragmentManager != null) {

            FragmentTransaction fragmentTransaction = fragmentManager.beginTransaction();
```

```
            fragmentTransaction.replace(R.id.main_fragment_container, fragment, tag);

            fragmentTransaction.commit();

            getAttachView().setToolbarIconVisible(showIcon);
            getAttachView().setToolbarTitle(title);
        }
    }

    @Override
    public void startPushService() {
        User user = new User().getInstance();

        if (!user.isUserAuthorized())
            return;

        int resultCode =
GoogleApiAvailability.getInstance().isGooglePlayServicesAvailable((MainActivity)
getAttachView());

        if (resultCode != ConnectionResult.SUCCESS) {
            if (GoogleApiAvailability.getInstance().isUserResolvableError(resultCode)) {
                getAttachView().ShowSnackBar("CODE: " + resultCode, R.color.colorRed);
            } else {
                getAttachView().ShowSnackBar("(DEVICE NOT SUPPORTED?) CODE: " + resultCode,
R.color.colorRed);
            }

            return;
        }

        FirebaseInstanceId.getInstance().getInstanceId().addOnCompleteListener(task -> {
            if (!task.isSuccessful()) {
                return;
            }
```

```
BanomoApolloClient.getApolloClient().mutate(RegisterDeviceForPushMutation.builder()
                    .deviceId(task.getResult().getToken())
                    .deviceType((long) 1)
                    .phone(user.getsPhone())
                    .token(user.getsToken())
                    .build())
                    .enqueue(new ApolloCall.Callback<RegisterDeviceForPushMutation.Data>() {
                        @Override
                        public void onResponse(@Nonnull
Response<RegisterDeviceForPushMutation.Data> response) {
                            if (response.hasErrors()) {
                                String error = response.errors().get(0).message();


                                if (error == null || !error.equals("Данное устройство уже
зарегистрированно"))
                                    getAttachView().runOnUiThread(() ->
getAttachView().ShowSnackBar(
                                            "Ошибка регистрации push-сервиса. Сообщите
разработчику \n" +
                                                    response.errors().get(0).message(),
R.color.colorRed));
                            }
                        }


                        @Override
                        public void onFailure(@Nonnull ApolloException e) {
                            getAttachView().runOnUiThread(() ->
getAttachView().ShowSnackBar("Ошибка регистрации push-сервиса. Сообщите разработчику \n" +
e.getMessage(), R.color.colorRed));
                        }
                    });
        });
    }
}
```

Листинг 9 – UserOffersPresenter

```
package net.darkeneez.banomo.ui.user_case.UCOffers;

import android.content.Context;
import android.content.SharedPreferences;

import com.apollographql.apollo.ApolloCall;
import com.apollographql.apollo.api.Response;
import com.apollographql.apollo.exception.ApolloException;
import com.google.gson.Gson;

import net.darkeneez.banomo.application.apollo.BanomoApolloClient;
import net.darkeneez.banomo.application.models.user.User;
import net.darkeneez.banomo.ui.base.BasePresenter;
import net.darkeneez.banomo.ui.user_case.UCElementsInterfaces.IUCElements;

import java.util.ArrayList;
import java.util.List;
import java.util.Map;

import javax.annotation.Nonnull;

import anet.darkeneez.banomo.GetJobByIdQuery;
import anet.darkeneez.banomo.GetOfferByUserQuery;
import anet.darkeneez.banomo.fragment.Job;

public class UserOffersPresenter<V extends IUCElements> extends BasePresenter<V> implements
IUserOffersPresenter<V> {

    private List<Job> mJobs;
    private long[] mJobsID;
    private int mCount;

    @Override
    public void loadUserJobOffers() {
        mJobs = new ArrayList<>();
```

```
        BanomoApolloClient.getApolloClient().query(GetOfferByUserQuery.builder()
                .phone(new User().getInstance().getsPhone())
                .build())
                .enqueue(new ApolloCall.Callback<GetOfferByUserQuery.Data>() {
                    @Override
                    public void onResponse(@Nonnull Response<GetOfferByUserQuery.Data>
response) {
                        if (response.data() == null)
                            return;

                        List<GetOfferByUserQuery.GetOffersByUser> offersList =
response.data().getOffersByUser();
                        mCount = offersList.size();

                        if (mCount == 0) {
                            getAttachView().runOnUiThread(() ->
getAttachView().updateRecycler(mJobs));

                            return;
                        }

                        mJobsID = new long[mCount];

                        for (int i = 0; i < mCount; i++) {
                            mJobsID[i] = offersList.get(i).fragments().offer().jobId();
                        }
                        mCount--;
                        loadJobs();
                    }

                    @Override
                    public void onFailure(@Nonnull ApolloException e) {
                        loadCacheJobs();
                    }
                });
    }
```

```
    private void loadJobs() {
        if (mCount == -1) {
            clearCache();
            cacheJobs();
            getAttachView().runOnUiThread(() -> getAttachView().updateRecycler(mJobs));
            return;
        } else {
            BanomoApolloClient.getApolloClient().query(GetJobByIdQuery.builder()
                    .id(mJobsID[mCount])
                    .build())
                    .enqueue(new ApolloCall.Callback<GetJobByIdQuery.Data>() {
                        @Override
                        public void onResponse(@Nonnull Response<GetJobByIdQuery.Data>
response) {

                            if (response.data() == null) {
                                return;
                            }

                            if (response.data().JobById() != null)
                                mJobs.add(response.data().JobById().fragments().job());
                            mCount--;
                            loadJobs();
                        }

                        @Override
                        public void onFailure(@Nonnull ApolloException e) {
                            loadCacheJobs();
                        }
                    });
        }
    }

    private void loadCacheJobs() {
```

```
        SharedPreferences sharedPreferences =
getAttachView().getActivity().getSharedPreferences("OffersCache", Context.MODE_PRIVATE);


        Map<String, ?> jobSets = sharedPreferences.getAll();

        final List<Job> jobs = new ArrayList<>();


        for (Map.Entry<String, ?> entry : jobSets.entrySet()) {

            Gson gson = new Gson();

            String json = entry.getValue().toString();

            Job job = gson.fromJson(json, Job.class);


            jobs.add(job);

        }


        getAttachView().runOnUiThread(() -> getAttachView().updateRecycler(jobs));

    }


    private void clearCache() {

        getAttachView().getActivity().getSharedPreferences("OffersCache",
Context.MODE_PRIVATE)

                .edit()

                .clear()

                .apply();

    }




    private void cacheJobs() {

        SharedPreferences.Editor jobsCache =
getAttachView().getActivity().getSharedPreferences("OffersCache",
Context.MODE_PRIVATE).edit();


        for (Job job : mJobs) {

            boolean skip = false;


            for (Job.Offer offer : job.offers()) {

                if (!offer.fragments().offer().owner().phone().equals(new User().getsPhone())
|| offer.fragments().offer().status() != 2) {
```

```
                    skip = true;
                    break;
                }
            }

            if (skip)
                continue;

            Gson json = new Gson();
            String Json = json.toJson(job);
            jobsCache.putString(job.id().toString(), Json);
        }

        jobsCache.apply();
    }
}
```

Листинг 10 – UserJobsPresenter

```
package net.darkeneez.banomo.ui.user_case.UCJobs;

import android.content.Context;
import android.content.SharedPreferences;

import com.apollographql.apollo.ApolloCall;
import com.apollographql.apollo.api.Response;
import com.apollographql.apollo.exception.ApolloException;
import com.google.gson.Gson;

import net.darkeneez.banomo.R;
import net.darkeneez.banomo.application.apollo.BanomoApolloClient;
import net.darkeneez.banomo.application.models.user.User;
import net.darkeneez.banomo.ui.base.BasePresenter;
import net.darkeneez.banomo.ui.user_case.UCElementsInterfaces.IUCElements;

import java.util.ArrayList;
import java.util.List;
```

```
import java.util.Map;


import javax.annotation.Nonnull;


import anet.darkeneez.banomo.DeleteJobMutation;

import anet.darkeneez.banomo.GetJobsByUserQuery;

import anet.darkeneez.banomo.fragment.Job;


public class UserJobsPresenter<V extends IUCElements> extends BasePresenter<V> implements
IUserJobsPresenter<V> {

    @Override

    public void loadUserJobs() {

        User user = new User().getInstance();


        BanomoApolloClient.getApolloClient().query(GetJobsByUserQuery.builder()

                .phone(user.getsPhone())

                .build())

                .enqueue(new ApolloCall.Callback<GetJobsByUserQuery.Data>() {

                    @Override

                    public void onResponse(@Nonnull final Response<GetJobsByUserQuery.Data>
response) {


                        if (response.hasErrors() || response.data() == null)

                            return;


                        final List<Job> jobs = new ArrayList<>();



                        for (GetJobsByUserQuery.GetJobsByUser job :
response.data().getJobsByUser()) {

                            jobs.add(job.fragments().job());

                        }


                        clearCache();

                        cacheJobs(jobs);
```

```
                        getAttachView().runOnUiThread(() ->
getAttachView().updateRecycler(jobs));
                    }


                    @Override
                    public void onFailure(@Nonnull ApolloException e) {
                        e.printStackTrace();
                        loadCacheJobs();
                    }
                });
    }


    private void clearCache() {
        getAttachView().getActivity().getSharedPreferences("JobCache", Context.MODE_PRIVATE)
                .edit()
                .clear()
                .apply();
    }


    @Override
    public void deleteJob(Long id) {
        User user = new User().getInstance();


        BanomoApolloClient.getApolloClient().mutate(DeleteJobMutation.builder()
                .jobId(id)
                .phone(user.getsPhone())
                .token(user.getsToken())
                .build())
                .enqueue(new ApolloCall.Callback<DeleteJobMutation.Data>() {
                    @Override
                    public void onResponse(@Nonnull Response<DeleteJobMutation.Data>
response) {
                        if (response.hasErrors()) {
                            return;
                        }
```

```
                        getAttachView().ShowSnackBar("Ваше объявление удалено",
R.color.colorGreen);
                    }


                    @Override
                    public void onFailure(@Nonnull ApolloException e) {
                        e.printStackTrace();
                    }
                });
    }


    private void loadCacheJobs() {
        SharedPreferences sharedPreferences =
getAttachView().getActivity().getSharedPreferences("JobCache", Context.MODE_PRIVATE);


        Map<String, ?> jobSets = sharedPreferences.getAll();
        final List<Job> jobs = new ArrayList<>();


        for (Map.Entry<String, ?> entry : jobSets.entrySet()) {
            Gson gson = new Gson();
            String json = entry.getValue().toString();
            Job job = gson.fromJson(json, Job.class);


            jobs.add(job);
        }


        getAttachView().runOnUiThread(() -> getAttachView().updateRecycler(jobs));
    }


    private void cacheJobs(List<Job> jobs) {
        SharedPreferences.Editor jobsCache =
getAttachView().getActivity().getSharedPreferences("JobCache", Context.MODE_PRIVATE).edit();


        for (Job job : jobs) {
            if (job.status() != 6)
                continue;
```

```
            Gson json = new Gson();
            String Json = json.toJson(job);
            jobsCache.putString(job.id().toString(), Json);
        }

        jobsCache.apply();
    }
}
```

Листинг 11 – UserJobOffersPresenter

```
package net.darkeneez.banomo.ui.user_job_offers

import android.annotation.SuppressLint
import anet.darkeneez.banomo.CompleteJobMutation
import anet.darkeneez.banomo.GetJobByIdQuery
import anet.darkeneez.banomo.UpdateJobStatusMutation
import anet.darkeneez.banomo.UpdateOfferStatusMutation
import anet.darkeneez.banomo.fragment.Job
import com.apollographql.apollo.ApolloCall
import com.apollographql.apollo.api.Response
import com.apollographql.apollo.exception.ApolloException
import net.darkeneez.banomo.application.apollo.BanomoApolloClient
import net.darkeneez.banomo.application.models.user.User
import net.darkeneez.banomo.ui.base.BasePresenter
import java.util.*

class UserJobOffersPresenter<V : IUserJobOffers> : BasePresenter<V>(),
IUserJobOffersPresenter<V> {

    private var mJobID: Long = 0
    private var mNeedPersons: Long? = null

    @SuppressLint("UseSparseArrays")
    private val mSelectedOffers = HashMap<Long, Long>()

    private val mIsOffersAccepted = false
```

```
    val personsSum: Int
        get() {
            var sum = 0

            for ((_, value) in mSelectedOffers) {
                sum = sum + value.toInt()
            }

            return sum
        }

    override fun loadOffers() {
        BanomoApolloClient.getApolloClient().query(GetJobByIdQuery.builder()
                .id(mJobID)
                .build())
                .enqueue(object : ApolloCall.Callback<GetJobByIdQuery.Data>() {
                    override fun onResponse(response: Response<GetJobByIdQuery.Data>) {
                        if (response.hasErrors())
                            return

                        val offers: MutableList<Job.Offer>?

                        mNeedPersons =
response.data()!!.JobById()!!.fragments().job().persons()

                        if (response.data()?.JobById()?.fragments()?.job()?.status()?.toInt()
== 6) {
                            offers = ArrayList()

                            for (offer in
response.data()!!.JobById()!!.fragments().job().offers()!!) {
                                if (offer.fragments().offer().status()?.toInt() != 2)
                                    continue

                                offers.add(offer)
                            }
```

```
                            attachView.runOnUiThread { attachView.startAcceptedMode(offers) }
                        } else {
                            offers = response.data()!!.JobById()!!.fragments().job().offers()

                            attachView.runOnUiThread {
attachView.startNotAcceptedMode(offers) }
                        }
                    }

                    override fun onFailure(e: ApolloException) {

                    }
                })
    }

    override fun onAccepted() {
        for ((key) in mSelectedOffers) {
            BanomoApolloClient.getApolloClient().mutate(UpdateOfferStatusMutation.builder()
                    .jobId(mJobID)
                    .offerId(key)
                    .phone(User().instance.getsPhone())
                    .token(User().instance.getsToken())
                    .status(2)
                    .build())
                    .enqueue(object : ApolloCall.Callback<UpdateOfferStatusMutation.Data>() {
                        override fun onResponse(response:
Response<UpdateOfferStatusMutation.Data>) {
                            if (response.hasErrors())
                                return

                            val user = User().instance

BanomoApolloClient.getApolloClient().mutate(UpdateJobStatusMutation.builder()
                                    .jobID(mJobID)
                                    .phone(user.getsPhone())
```

```
                                    .token(user.getsToken())
                                    .status(6)
                                    .build())
                                    .enqueue(object :
ApolloCall.Callback<UpdateJobStatusMutation.Data>() {
                                        override fun onResponse(response:
Response<UpdateJobStatusMutation.Data>) {
                                            if (response.hasErrors())
                                                return


BanomoApolloClient.getApolloClient().mutate(UpdateOfferStatusMutation.builder()
                                                    .jobId(mJobID)
                                                    .offerId(key)
                                                    .phone(user.getsPhone())
                                                    .token(user.getsToken())
                                                    .status(2)
                                                    .build())
                                                    .enqueue(object :
ApolloCall.Callback<UpdateOfferStatusMutation.Data>() {
                                                        override fun onResponse(response:
Response<UpdateOfferStatusMutation.Data>) {
                                                            if (response.hasErrors())
                                                                return


                                                            attachView.runOnUiThread {
attachView.onAcceptedSuccess() }
                                                        }


                                                        override fun onFailure(e:
ApolloException) {


                                                        }
                                                    })


                                        }

                                        override fun onFailure(e: ApolloException) {
```

```
                                    }
                                })
                    }

                    override fun onFailure(e: ApolloException) {
                        return
                    }
                })
    }
}

//*OK*//
override fun setJobID(JobID: Long) {
    mJobID = JobID
}

override fun completeJob() {
    val user = User().instance
    BanomoApolloClient.getApolloClient().mutate(CompleteJobMutation.builder()
            .jobId(mJobID)
            .phone(user.getsPhone())
            .token(user.getsToken())
            .build())
            .enqueue(object : ApolloCall.Callback<CompleteJobMutation.Data>() {
                override fun onResponse(response: Response<CompleteJobMutation.Data>) {
                    if (response.hasErrors()) {
                        return
                    }

                    attachView.runOnUiThread {
                        attachView.jobFinished()
                    }
                }
```

```
                override fun onFailure(e: ApolloException) {

                }
            })
    }

    fun getJob(): Long {
        return mJobID
    }

    override fun addOffer(offerID: Long, persons: Long, workType: Boolean) {

        if (workType)
            mSelectedOffers.clear()

        mSelectedOffers[offerID] = persons
        attachView.updatePersonsCount(personsSum)
    }

    override fun deleteOffer(offerID: Long) {
        mSelectedOffers.remove(offerID)
        attachView.updatePersonsCount(personsSum)
    }

    override fun isOffersAccepted(): Boolean {
        return mIsOffersAccepted
    }

    override fun getMaxPersons(): Long {
        return mNeedPersons!!
    }
}
```

Листинг 12 – UserJobPresenter

```
package net.darkeneez.banomo.ui.user_job_view
```

```
import anet.darkeneez.banomo.CompleteJobMutation
import anet.darkeneez.banomo.GetJobByIdQuery
import anet.darkeneez.banomo.fragment.Job
import com.apollographql.apollo.ApolloCall
import com.apollographql.apollo.api.Response
import com.apollographql.apollo.exception.ApolloException
import net.darkeneez.banomo.R
import net.darkeneez.banomo.application.apollo.BanomoApolloClient
import net.darkeneez.banomo.application.models.user.User
import net.darkeneez.banomo.ui.base.BasePresenter

class UserJobPresenter<V : IUserJobView> : BasePresenter<V>(), IUserJobViewPresenter<V> {
    private var mJob: Job? = null

    override fun loadJob(id: Long?) {
        BanomoApolloClient.getApolloClient().query(GetJobByIdQuery.builder()
                .id(id!!)
                .build())
                .enqueue(object : ApolloCall.Callback<GetJobByIdQuery.Data>() {
                    override fun onResponse(response: Response<GetJobByIdQuery.Data>) {
                        if (response.hasErrors())
                            return

                        mJob = response.data()!!.JobById()!!.fragments().job()

                        if (mJob == null)
                            return

                        attachView.runOnUiThread {
                            attachView.readJob(mJob!!)
                            attachView.loadJobImages(response.data()?.JobById()?.images())
                        }
                    }
```

```
                    override fun onFailure(e: ApolloException) {
                        e.printStackTrace()
                        attachView.ShowSnackBar("Неудалось загрузить работу",
R.color.colorRed)
                    }
                })
    }

    override fun completeJob() {
        val user = User().instance
        BanomoApolloClient.getApolloClient().mutate(CompleteJobMutation.builder()
                .jobId(mJob!!.id()!!)
                .phone(user.getsPhone())
                .token(user.getsToken())
                .build())
                .enqueue(object : ApolloCall.Callback<CompleteJobMutation.Data>() {
                    override fun onResponse(response: Response<CompleteJobMutation.Data>) {
                        if (response.hasErrors()) {
                            return
                        }
                        loadJob(response.data()!!.completeJob()!!.id())
                    }

                    override fun onFailure(e: ApolloException) {

                    }
                })
    }

    fun getJobID(): Long {
        return mJob!!.id()!!
    }
}
```

Листинг 13 – ProfilePresenter

```
package net.darkeneez.banomo.ui.user_manager.profile;
```

```
import android.content.Context;

import com.apollographql.apollo.ApolloCall;
import com.apollographql.apollo.api.Response;
import com.apollographql.apollo.exception.ApolloException;
import com.google.firebase.iid.FirebaseInstanceId;

import net.darkeneez.banomo.R;
import net.darkeneez.banomo.application.apollo.BanomoApolloClient;
import net.darkeneez.banomo.application.models.user.User;
import net.darkeneez.banomo.application.models.user.UserInterfaces.IUserPhotoUploaded;
import net.darkeneez.banomo.application.models.user.UserInterfaces.IUserUpdated;
import net.darkeneez.banomo.application.util.UPLoadUtil.UploadImageToServer;
import net.darkeneez.banomo.ui.base.BasePresenter;

import java.io.IOException;

import javax.annotation.Nonnull;

import anet.darkeneez.banomo.DeleteDeviceIDMutation;

public class ProfilePresenter<V extends IProfile> extends BasePresenter<V> implements
IProfilePresenter<V> {
    private String mFilePath = "";

    @Override
    public void onSaveClicked(String name, String surname, long regionID, String email) {
        uploadImageToServer(name, surname, regionID, email);
    }

    @Override
    public int checkField(String field) {
        int status = 0;

        if (field.equals(""))
```

```
            status = 1;
        if (field.length() < 2)
            status = 2;

        return status;
    }

    @Override
    public void onExitClick() {
        deleteDeviceID();
    }

    private void exit() {
        new User().getInstance().clearUserCached();

        clearPreferences();
        getAttachView().startLoginActivity();
    }

    private void clearPreferences() {
        ProfileActivity profileActivity = (ProfileActivity) getAttachView();

        profileActivity.getSharedPreferences("temp_job", Context.MODE_PRIVATE)
                .edit()
                .clear()
                .apply();

        profileActivity.getSharedPreferences("OffersCache", Context.MODE_PRIVATE)
                .edit().clear().apply();
        profileActivity.getSharedPreferences("JobCache", Context.MODE_PRIVATE)
                .edit().clear().apply();
    }

    private void deleteDeviceID() {
        User user = new User().getInstance();
```

```
        FirebaseInstanceId.getInstance().getInstanceId().addOnCompleteListener(task -> {
            if (!task.isSuccessful()) {
                return;
            }

            BanomoApolloClient.getApolloClient().mutate(DeleteDeviceIDMutation
                    .builder()
                    .phone(user.getsPhone())
                    .token(user.getsToken())
                    .deviceId(task.getResult().getToken())
                    .build())
                    .enqueue(new ApolloCall.Callback<DeleteDeviceIDMutation.Data>() {
                        @Override
                        public void onResponse(@Nonnull Response<DeleteDeviceIDMutation.Data>
response) {
                            if (response.hasErrors())
                                getAttachView().runOnUiThread(() ->
getAttachView().ShowSnackBar(response.errors().get(0).message(), R.color.colorRed));
                            else
                                getAttachView().runOnUiThread(() -> {
                                    getAttachView().ShowSnackBar("Успешно удалили deviceID",
R.color.colorGreen);
                                    exit();
                                });
                        }

                        @Override
                        public void onFailure(@Nonnull ApolloException e) {
                            try {

FirebaseInstanceId.getInstance().deleteToken(task.getResult().getToken(),
task.getResult().getId());
                            } catch (IOException e1) {
                                e1.printStackTrace();
                            }
                        }
```

```
                });
        });
    }

    @Override
    public void onImageSelected(String image) {
        mFilePath = image;
    }


    private void uploadImageToServer(final String name, final String surname, final long
regionID, final String email) {

        final User user = new User().getInstance();

        user.setOnUserUpdated(new IUserUpdated() {
            @Override
            public void onError(String error) {
                getAttachView().ShowSnackBar(error, R.color.colorRed);
            }

            @Override
            public void onSuccess() {
                getAttachView().finish();
            }
        });

        if (mFilePath.equals("")) {
            user.updateUser(name, surname, regionID);
            return;
        }

        UploadImageToServer uploadImageToServer = new UploadImageToServer();

        uploadImageToServer.setUploadListener(new UploadImageToServer.UploadListener() {
            @Override
```

```
            public void onSuccess(String result) {


                String avatar = result.substring(result.lastIndexOf(':') + 2,
result.lastIndexOf('"'));


                user.setOnUserPhotoUploaded(new IUserPhotoUploaded() {
                    @Override
                    public void onError(String error) {
                        getAttachView().runOnUiThread(() -> {
                                    getAttachView().ShowSnackBar(error, R.color.colorRed);
                                    getAttachView().hideAvatarProgressBar();
                                }
                        );
                    }


                    @Override
                    public void onSuccess(String uri) {
                        user.updateUser(name, surname, regionID);
                        getAttachView().runOnUiThread(() -> {
                            getAttachView().ShowSnackBar("Ваш аватар загружен",
R.color.colorGreen);
                            getAttachView().hideAvatarProgressBar();
                        });
                    }
                });
                user.uploadPhoto(avatar);
            }


            @Override
            public void onError(String error) {
                getAttachView().runOnUiThread(() -> {
                            getAttachView().ShowSnackBar(error, R.color.colorRed);
                            getAttachView().hideAvatarProgressBar();
                        }
                );
            }
```

```
        });
        getAttachView().showAvatarProgressBar();
        uploadImageToServer.execute(mFilePath);
    }
}
```

Листинг 14 – SettingsFragmentPresenter

```
package net.darkeneez.banomo.ui.user_manager.settings;


import net.darkeneez.banomo.application.models.user.User;
import net.darkeneez.banomo.application.models.user.UserInterfaces.IUserUpdated;
import net.darkeneez.banomo.ui.base.BasePresenter;


public class SettingsFragmentPresenter<V extends ISettingsFragment> extends BasePresenter<V>
implements ISettingsPresenter<V> {


    public void loadUserInfo() {
        User user = new User().getInstance();


        getAttachView().SetUserInfo(user.getsName() + " " + user.getsSurname(),
user.getsPhone(), user.getsAvatar(), (int) user.getsBalance());
    }


    public void updateUser() {
        final User user = new User().getInstance();
        user.setOnUserUpdated(new IUserUpdated() {
            @Override
            public void onError(String error) {
                getAttachView().runOnUiThread(() ->
getAttachView().SetUserInfo(user.getsName() + " " + user.getsSurname(),
                        user.getsPhone(), user.getsAvatar(), (int) user.getsBalance()));
            }


            @Override
            public void onSuccess() {
                getAttachView().runOnUiThread(() ->
getAttachView().SetUserInfo(user.getsName() + " " + user.getsSurname(),
                        user.getsPhone(), user.getsAvatar(), (int) user.getsBalance()));
```

```
            }
        });
        user.loadUser();
    }
}
```

Листинг 15 – SupportPresenter

```
package net.darkeneez.banomo.ui.user_manager.support;

import com.apollographql.apollo.ApolloCall;
import com.apollographql.apollo.api.Response;
import com.apollographql.apollo.exception.ApolloException;

import net.darkeneez.banomo.R;
import net.darkeneez.banomo.application.apollo.BanomoApolloClient;
import net.darkeneez.banomo.application.models.user.User;
import net.darkeneez.banomo.ui.base.BasePresenter;

import java.util.Locale;

import javax.annotation.Nonnull;

import anet.darkeneez.banomo.CreateNewComplaintMutation;

public class SupportPresenter<V extends ISupport> extends BasePresenter<V> implements
ISupportPresenter<V> {

    @Override
    public void onSendButtonClicked(String message) {
        if (message.length() == 0) {
            getAttachView().ShowSnackBar("Опишите вашу проблему", R.color.colorRed);
            return;
        } else if (message.length() < 20) {
            getAttachView().ShowSnackBar("Опишите вашу проблему подробнее",
R.color.colorRed);
            return;
        }
```

```
        sendMessageToServer(message);
    }

    @Override
    public void sendMessageToServer(String message) {
        User user = new User().getInstance();
        BanomoApolloClient.getApolloClient().mutate(CreateNewComplaintMutation.builder()
                .description(message)
                .jobId((long) 0)
                .offerId((long) 0)
                .theme("Проблема с приложением")
                .phone(user.getsPhone())
                .token(user.getsToken())
                .build())
                .enqueue(new ApolloCall.Callback<CreateNewComplaintMutation.Data>() {
                    @Override
                    public void onResponse(@Nonnull Response<CreateNewComplaintMutation.Data>
response) {
                        if (response.hasErrors())
                            return;

                        int supportID = response.data().createComplaint().id().intValue();

                        if (supportID > 0) {
                            getAttachView().ShowSnackBar(String.format(Locale.getDefault(),
                                    "Номер вашего обращения: %d", supportID),
R.color.colorGreen);
                        }
                    }

                    @Override
                    public void onFailure(@Nonnull ApolloException e) {

                    }
                });
```

```
    }
}
```

Листинг 16 – ViewJobPresenter

```
package net.darkeneez.banomo.ui.view_job;

import com.apollographql.apollo.ApolloCall;
import com.apollographql.apollo.api.Response;
import com.apollographql.apollo.exception.ApolloException;

import net.darkeneez.banomo.R;
import net.darkeneez.banomo.application.apollo.BanomoApolloClient;
import net.darkeneez.banomo.application.models.user.User;
import net.darkeneez.banomo.ui.base.BasePresenter;

import javax.annotation.Nonnull;

import anet.darkeneez.banomo.CreateNewOfferMutation;
import anet.darkeneez.banomo.DeleteOfferMutation;
import anet.darkeneez.banomo.GetJobByIdQuery;
import anet.darkeneez.banomo.RateCustomerMutation;
import anet.darkeneez.banomo.UpdateOfferMutation;
import anet.darkeneez.banomo.UpdateOfferStatusMutation;



public class ViewJobPresenter<V extends IViewJob> extends BasePresenter<V> implements
IViewJobPresenter<V> {

    private User mUser;
    private Long JobID;

    @Override
    public void attach(V view) {
        super.attach(view);
        mUser = new User().getInstance();
    }
```

```
    @Override
    public void onOfferSendClicked(Long persons, Long payment) {

        BanomoApolloClient.getApolloClient().mutate(CreateNewOfferMutation.builder()
                .token(mUser.getsToken())
                .phone(mUser.getsPhone())
                .jobId(JobID)
                .payment(payment)
                .persons(persons)
                .build())
                .enqueue(new ApolloCall.Callback<CreateNewOfferMutation.Data>() {
                    @Override
                    public void onResponse(@Nonnull Response<CreateNewOfferMutation.Data>
response) {
                        if (response.hasErrors())
                            return;

                        loadJob();
                    }

                    @Override
                    public void onFailure(@Nonnull ApolloException e) {
                        e.printStackTrace();
                    }
                });
    }

    @Override
    public void onDeleteOfferClicked(Long JobID) {
        BanomoApolloClient.getApolloClient().mutate(DeleteOfferMutation.builder()
                .token(mUser.getsToken())
                .phone(mUser.getsPhone())
                .offerId(JobID)
                .build())
                .enqueue(new ApolloCall.Callback<DeleteOfferMutation.Data>() {
```

```
                    @Override
                    public void onResponse(@Nonnull final Response<DeleteOfferMutation.Data>
response) {
                        if (response.hasErrors())
                            return;


                        getAttachView().runOnUiThread(() -> {
                            getAttachView().ShowSnackBar("Ваше предложение удалено",
R.color.colorGreen);
                            getAttachView().showRespondElement();
                        });
                    }


                    @Override
                    public void onFailure(@Nonnull ApolloException e) {
                        e.printStackTrace();
                    }
                });
    }


    @Override
    public void afterOfferEdit(Long persons, Long payment, Long id) {
        BanomoApolloClient.getApolloClient().mutate(UpdateOfferMutation.builder()
                .token(mUser.getsToken())
                .phone(mUser.getsPhone())
                .offerId(id)
                .payment(payment)
                .persons(persons)
                .build())
                .enqueue(new ApolloCall.Callback<UpdateOfferMutation.Data>() {
                    @Override
                    public void onResponse(@Nonnull Response<UpdateOfferMutation.Data>
response) {
                        if (response.hasErrors())
                            return;


                        loadJob();
```

```
                    }

                    @Override
                    public void onFailure(@Nonnull ApolloException e) {
                        e.printStackTrace();
                    }
                });
    }

    @Override
    public void loadJob() {

        if (JobID == -1)
            return;

        BanomoApolloClient.getApolloClient().query(GetJobByIdQuery.builder()
                .id(JobID)
                .build())
                .enqueue(new ApolloCall.Callback<GetJobByIdQuery.Data>() {
                    @Override
                    public void onResponse(@Nonnull final Response<GetJobByIdQuery.Data>
response) {
                        if (response.hasErrors())
                            return;

                        getAttachView().runOnUiThread(() -> {

getAttachView().readJob(response.data().JobById().fragments().job());

getAttachView().loadJobImages(response.data().JobById().images());
                                }

                        );
                    }

                    @Override
                    public void onFailure(@Nonnull ApolloException e) {
```

```
                        e.printStackTrace();
                        getAttachView().ShowSnackBar("Неудалось загрузить работу",
R.color.colorRed);
                    }
                });
    }

    @Override
    public void customerRating(final boolean up, final boolean down, final long offerID) {
        final User user = new User().getInstance();

        BanomoApolloClient.getApolloClient().mutate(RateCustomerMutation
                .builder()
                .jobId(JobID)
                .phone(user.getsPhone())
                .token(user.getsToken())
                .ratingCustomerDown(down)
                .ratingCustomerUp(up)
                .build())
                .enqueue(new ApolloCall.Callback<RateCustomerMutation.Data>() {
                    @Override
                    public void onResponse(@Nonnull Response<RateCustomerMutation.Data>
response) {
                        if (response.hasErrors())
                            return;

BanomoApolloClient.getApolloClient().mutate(UpdateOfferStatusMutation.builder()
                                .status(5)
                                .token(user.getsToken())
                                .phone(user.getsPhone())
                                .jobId(JobID)
                                .offerId(offerID)
                                .build())
                                .enqueue(new
ApolloCall.Callback<UpdateOfferStatusMutation.Data>() {
                                    @Override
```

```
                                    public void onResponse(@Nonnull
Response<UpdateOfferStatusMutation.Data> response) {
                                        if (response.hasErrors())
                                            return;

                                        getAttachView().runOnUiThread(() ->
getAttachView().showMessageToExecutor("Работа успешно завершена.\nЕсли вам нужна помощь,
обратитесь в тех.поддержку", null));
                                    }

                                    @Override
                                    public void onFailure(@Nonnull ApolloException e) {

                                    }
                                });
                    }

                    @Override
                    public void onFailure(@Nonnull ApolloException e) {

                    }
                });
    }

    public Long getJobID() {
        return JobID;
    }

    public void setJobID(Long JobID) {
        this.JobID = JobID;
    }
}
```

Приложение Б
Исходный код видов

Листинг 17 – ActivityCaseFilter

```
package net.darkeneez.banomo.ui.case_sort;

import android.content.Intent;
import android.os.Bundle;
import android.widget.ArrayAdapter;
import android.widget.Spinner;

import androidx.recyclerview.widget.LinearLayoutManager;
import androidx.recyclerview.widget.RecyclerView;

import net.darkeneez.banomo.R;
import net.darkeneez.banomo.application.models.job.JobStatus;
import net.darkeneez.banomo.ui.base.BaseActivity;

import org.greenrobot.eventbus.EventBus;

import java.util.ArrayList;
import java.util.List;

import butterknife.BindView;
import butterknife.ButterKnife;
import butterknife.OnClick;

public class ActivityCaseFilter extends BaseActivity implements onCheckBoxClick {

    @BindView(R.id.status_recycler)
    RecyclerView mRecycleView;

    @BindView(R.id.case_sort_spinner)
    Spinner mSortSpinner;

    private ArrayList<Long> mStatuses = new ArrayList<>();
    private ChoiceStatusAdapter mAdapter = new ChoiceStatusAdapter(this, this);
```

Продолжение приложения Б

```
private boolean mIsJob;

@Override
public void onCreate(Bundle saveInstance) {
    super.onCreate(saveInstance);
    setUnbinder(ButterKnife.bind(this));

    initAdapter();
    initSpinner();
}

private void initAdapter() {
    LinearLayoutManager linearLayout = new LinearLayoutManager(this);

    mRecycleView.setLayoutManager(linearLayout);
    mRecycleView.setNestedScrollingEnabled(false);

    mAdapter = new ChoiceStatusAdapter(this, this);
    mRecycleView.setAdapter(mAdapter);

    List<JobStatus> idList = new ArrayList<>();

    Intent intent = getIntent();
    mIsJob = intent.getBooleanExtra("IS_JOB", true);

    if (mIsJob) {
        idList.add(new JobStatus("На рассмотрении", (long) 1));
        idList.add(new JobStatus("Работа не выполнена", (long) 2));
        idList.add(new JobStatus("Заблокирована модератором", (long) 3));
        idList.add(new JobStatus("Ожидает предложений", (long) 4));
        idList.add(new JobStatus("Выполняется", (long) 6));
        idList.add(new JobStatus("Оценить исполнителя", (long) 7));
        idList.add(new JobStatus("Завершена", (long) 8));
    } else {
        idList.add(new JobStatus("На рассмотрении", (long) 1));
        idList.add(new JobStatus("Выбран исполнителен", (long) 2));
```

Продолжение приложения Б

```
            idList.add(new JobStatus("Отказано", (long) 3));
            idList.add(new JobStatus("Требуется оценка", (long) 4));
            idList.add(new JobStatus("Завершено", (long) 5));
        }

        mStatuses = (ArrayList<Long>) intent.getSerializableExtra("STATUS_ID");
        mAdapter.setOldStatuses(mStatuses);
        mAdapter.addAll(idList);
    }

    private void initSpinner() {
        Intent intent = getIntent();

        int sort_id = intent.getIntExtra("SORT_ID", 0);

        ArrayAdapter<String> sortAdapter = new ArrayAdapter<>(this,
android.R.layout.simple_spinner_item, new String[]{"По умолчанию", "Дате создания работы",
"По дате начала работы"});
        sortAdapter.setDropDownViewResource(android.R.layout.simple_spinner_dropdown_item);

        mSortSpinner.setAdapter(sortAdapter);
        mSortSpinner.setSelection(sort_id);
    }

    @OnClick(R.id.case_filter_apply)
    protected void onFilterApply() {
        Intent filterIntent = new Intent();

        int sortID = mSortSpinner.getSelectedItemPosition();

        filterIntent.putExtra("STATUS_ID", mStatuses);
        filterIntent.putExtra("SORT_ID", sortID);

        setResult(1, filterIntent);

        EventBus.getDefault().post(filterIntent);
```

Продолжение приложения Б

```
    finish();
}

@OnClick(R.id.case_filter_clear)
protected void onFilterClear() {
    mAdapter.setCheckAll();

    mStatuses = new ArrayList<>();
    mStatuses.add((long) 1);
    mStatuses.add((long) 2);
    mStatuses.add((long) 3);
    mStatuses.add((long) 4);
    mStatuses.add((long) 5);

    if (mIsJob) {
        mStatuses.add((long) 6);
        mStatuses.add((long) 7);
        mStatuses.add((long) 8);
    }

    mSortSpinner.setSelection(0);
}

@Override
protected int getLayoutContentViewID() {
    return R.layout.activity_case_filter;
}

@Override
public void onDestroy() {
    super.onDestroy();
    EventBus.getDefault().unregister(this);
}

@Override
```

Продолжение приложения Б

```
    public void onCheckBoxClicked(JobStatus jobStatus, boolean add) {
        if (add)
            mStatuses.add(jobStatus.getId());
        else
            mStatuses.remove(jobStatus.getId());
    }
}
```

Листинг 18 – BaseChoiceElementsActivity

```
package net.darkeneez.banomo.ui.choice_element;

import android.view.View;

import androidx.recyclerview.widget.LinearLayoutManager;
import androidx.recyclerview.widget.RecyclerView;

import net.darkeneez.banomo.R;
import net.darkeneez.banomo.application.models.base.BaseLongElement;
import net.darkeneez.banomo.ui.base.BaseActivity;
import net.darkeneez.banomo.ui.base.PFRecyclerViewAdapter;

import org.greenrobot.eventbus.EventBus;

import java.util.ArrayList;
import java.util.List;

import butterknife.BindView;
import butterknife.ButterKnife;

public abstract class BaseChoiceElementsActivity<T> extends BaseActivity implements
PFRecyclerViewAdapter.OnViewHolderClick<BaseLongElement> {

    public ChoiceElementsAdapter mAdapter;

    @BindView(R.id.location_recycler)
    RecyclerView mRecycleView;
```

Продолжение приложения Б

```
    public void initView(String title) {
        setUnbinder(ButterKnife.bind(this));
        setTitle(title);
        LinearLayoutManager linearLayout = new LinearLayoutManager(this);

        mRecycleView.setLayoutManager(linearLayout);
        mRecycleView.setNestedScrollingEnabled(false);

        mAdapter = new ChoiceElementsAdapter(this, this);
        mRecycleView.setAdapter(mAdapter);
    }

    public List<BaseLongElement> castToBaseLocationElement(List<T> Elements) {
        List<BaseLongElement> baseLocationElements = new ArrayList<>();

        for (T element : Elements) {
            if (element instanceof BaseLongElement) {
                baseLocationElements.add((BaseLongElement) element);
            }
        }
        return baseLocationElements;
    }

    @Override
    public void onClick(View view, int position, BaseLongElement item) {
        EventBus.getDefault().post(item);
        finish();
    }

    @Override
    protected int getLayoutContentViewID() {
        return R.layout.activity_choice_location;
    }
}
```

Продолжение приложения Б

Листинг 19 – ChoiceTypePayment

```
package net.darkeneez.banomo.ui.choice_type_payment;


import android.os.Bundle;

import android.view.View;


import androidx.recyclerview.widget.LinearLayoutManager;

import androidx.recyclerview.widget.RecyclerView;


import net.darkeneez.banomo.R;

import net.darkeneez.banomo.application.models.job.typesJob.JobTypePayment;

import net.darkeneez.banomo.ui.base.BaseActivity;

import net.darkeneez.banomo.ui.base.PFRecyclerViewAdapter;


import org.greenrobot.eventbus.EventBus;


import java.util.ArrayList;

import java.util.List;


import butterknife.BindView;

import butterknife.ButterKnife;


public class ChoiceTypePayment extends BaseActivity implements
PFRecyclerViewAdapter.OnViewHolderClick {


    @BindView(R.id.location_recycler)

    RecyclerView mRecycleView;



    @Override

    protected void onCreate(Bundle saveInstanceState) {

        super.onCreate(saveInstanceState);

        initView();

    }
```

Продолжение приложения Б

```
protected void initView() {
    setTitle("Выберите вариант оплаты");
    setUnbinder(ButterKnife.bind(this));

    LinearLayoutManager linearLayout = new LinearLayoutManager(this);

    mRecycleView.setLayoutManager(linearLayout);
    mRecycleView.setNestedScrollingEnabled(false);

    ChoiceTypePaymentAdapter mAdapter = new ChoiceTypePaymentAdapter(this, this);
    mRecycleView.setAdapter(mAdapter);

    List<String> idList = new ArrayList<>();

    idList.add("За работу");
    idList.add("Почасовая");

    if (getIntent().getBooleanExtra("is_from_filter", false))
        idList.add(0, "Любой вариант оплаты");

    mAdapter.addAll(idList);
}

@Override
public void onClick(View view, int position, Object item) {
    Long id;
    switch ((String) item) {
        case "Почасовая": {
            id = (long) 1;
            break;
        }
        case "За работу": {
            id = (long) 2;
            break;
```

Продолжение приложения Б

```
            }
            default: {
                id = (long) -1;
                break;
            }
        }
        JobTypePayment jobTypePayment = new JobTypePayment(id);

        EventBus.getDefault().post(jobTypePayment);
        finish();
    }

    @Override
    protected int getLayoutContentViewID() {
        return R.layout.activity_choice_location;
    }
}
```

Листинг 20 – ComplaintActivity

```
package net.darkeneez.banomo.ui.complaint

import android.app.Activity
import android.os.Bundle
import kotlinx.android.synthetic.main.activity_complain.*
import net.darkeneez.banomo.R
import net.darkeneez.banomo.application.util.BanomoApplicationTags.JOB_ID
import net.darkeneez.banomo.ui.base.BaseActivity

class ComplaintActivity : BaseActivity(), IComplainActivity {

    private var mComplaintPresenter: ComplaintPresenter<ComplaintActivity>? = null

    override fun onCreate(saveInstanceState: Bundle?) {
        super.onCreate(saveInstanceState)
        initView()
    }
```

Продолжение приложения Б

```
    private fun initView() {
        title = "Отправить жалобу"
        mComplaintPresenter = ComplaintPresenter()
        mComplaintPresenter!!.attach(this)
        mComplaintPresenter?.setUpParams(intent.getLongExtra(JOB_ID, -1),
intent.getLongExtra("offerId", -1))
        activity_complaint_send_button.setOnClickListener {
            val message: String = activity_complaint_description.text.toString()

            if (message.length < 15)
                ShowSnackBar("Опишите проблему подробнее. Минимум 15 символов",
R.color.colorRed)
            else
                mComplaintPresenter!!.onSendClicked(message, "Жалоба на работу")
        }
    }

    override fun getLayoutContentViewID(): Int {
        return R.layout.activity_complain
    }

    override fun onComplainSendSuccessful() {
        setResult(Activity.RESULT_OK)
        finish()
    }

    override fun onComplainSendError(error: String?) {
        ShowSnackBar(error, R.color.colorRed)
    }
}
```

Листинг 21 – CreateJob

```
package net.darkeneez.banomo.ui.create_job;
```



```
import android.content.Intent;
```

```
import android.os.Bundle;
import android.text.Editable;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.EditText;
import android.widget.LinearLayout;
import android.widget.ProgressBar;
import android.widget.ScrollView;
import android.widget.TextView;

import androidx.recyclerview.widget.LinearLayoutManager;
import androidx.recyclerview.widget.RecyclerView;

import com.esafirm.imagepicker.model.Image;
import com.esafirm.imagepicker.view.GridSpacingItemDecoration;

import net.darkeneez.banomo.R;
import net.darkeneez.banomo.application.models.job.typesJob.JobType;
import net.darkeneez.banomo.application.models.job.typesJob.JobTypePayment;
import net.darkeneez.banomo.application.models.regions.Location;
import net.darkeneez.banomo.application.util.CompatUtils;
import net.darkeneez.banomo.ui.base.BaseFragment;
import net.darkeneez.banomo.ui.base.PFRecyclerViewAdapter;
import net.darkeneez.banomo.ui.choice_element.ChoiceJobType;
import net.darkeneez.banomo.ui.choice_type_payment.ChoiceTypePayment;
import net.darkeneez.banomo.ui.dialogs.DatePicker;
import net.darkeneez.banomo.ui.dialogs.TimePicker;
import net.darkeneez.banomo.ui.location_select.LocationSelectActivity;
import net.darkeneez.banomo.ui.photo_job.PhotoJob;
import net.darkeneez.banomo.ui.photo_thumbnail.PhotoThumbnailAdapter;
import net.darkeneez.banomo.ui.user_job_view.UserJobView;

import org.greenrobot.eventbus.EventBus;
```

Продолжение приложения Б

```
import org.greenrobot.eventbus.Subscribe;
```

```
import java.util.ArrayList;
import java.util.List;
import java.util.Locale;

import javax.annotation.Nonnull;

import butterknife.BindView;
import butterknife.ButterKnife;
import butterknife.OnClick;
import butterknife.OnTextChanged;

import static net.darkeneez.banomo.application.util.BanomoApplicationTags.JOB_ID;

public class CreateJob extends BaseFragment implements ICreateJob,
PFRecyclerViewAdapter.OnViewHolderClick {

    @BindView(R.id.create_new_job_scroll_view)
    ScrollView mScrollView;

    @BindView(R.id.job_edit_executor_count)
    EditText mEditExecutorCount;

    @BindView(R.id.executor_count_text_view)
    TextView mExecutorCountText;

    @BindView(R.id.job_edit_date)
    TextView mEditDate;

    @BindView(R.id.job_edit_type)
    TextView mEditJobType;

    @BindView(R.id.job_edit_locationId)
    TextView mEditLocationId;
```

Продолжение приложения Б

```
    @BindView(R.id.job_edit_type_payment)
    TextView mEditTypePayment;

    @BindView(R.id.job_edit_description)
    EditText mEditDescription;

    @BindView(R.id.create_new_job_action_layout)
    LinearLayout mActionLayout;

    @BindView(R.id.create_new_job_progress_bar)
    ProgressBar mProgressBar;

    @BindView(R.id.create_job_recycler_view)
    RecyclerView mPhotoRecycler;

    @BindView(R.id.create_job_photo_text)
    TextView mPhotoTextView;

    private PhotoThumbnailAdapter mPhotoAdapter;
    private CreateJobPresenter<CreateJob> mCreateJobPresenter;

    @Override
    public View onCreateView(@Nonnull LayoutInflater inflater, ViewGroup container, Bundle
saveInstance) {
        View rootView = inflater.inflate(R.layout.fragment_create_new_job, container, false);
        initView(rootView);
        return rootView;
    }


    public void initView(View rootView) {
        setUnbinder(ButterKnife.bind(this, rootView));

        mCreateJobPresenter = new CreateJobPresenter<>();
```

Продолжение приложения Б

```
        mCreateJobPresenter.attach(this);
```

```
        initRecyclerView();

        checkBundle();

        EventBus.getDefault().register(this);
    }

    private void initRecyclerView() {
        mPhotoAdapter = new PhotoThumbnailAdapter(getActivity(), this);

        mPhotoRecycler.setHasFixedSize(true);
        mPhotoRecycler.setNestedScrollingEnabled(false);

        mPhotoRecycler.setLayoutManager(new LinearLayoutManager(getActivity(),
LinearLayoutManager.HORIZONTAL, false));
        mPhotoRecycler.addItemDecoration(new GridSpacingItemDecoration(2, 8, true));
        mPhotoRecycler.setAdapter(mPhotoAdapter);
    }

    private void checkBundle() {
        Bundle bundle = getArguments();

        if (bundle == null) {
            checkIfContinue();
        } else if (bundle.getBoolean("is_edit", false)) {
            mCreateJobPresenter.setEditMode(bundle.getLong(JOB_ID));
            mCreateJobPresenter.loadJob(bundle.getLong(JOB_ID));
        } else if (bundle.getBoolean("is_repeat", false)) {
            mCreateJobPresenter.loadJob(bundle.getLong(JOB_ID));
        }
    }

    private void checkIfContinue() {
```

Продолжение приложения Б

```
        mCreateJobPresenter.setJobLocationIdByUser();
```

```
    }

    @OnTextChanged(value = R.id.job_edit_executor_count, callback =
OnTextChanged.Callback.TEXT_CHANGED)
    public void onExecutorChanged(Editable editable) {
        if (editable.toString().equals("") || editable.toString().equals("Любое"))
            return;

        int count = Integer.parseInt(editable.toString());

        if (count > 20) {
            mEditExecutorCount.setText("20");
            ShowSnackBar("Максимум 20 исполнителей!", R.color.colorRed);
        }
    }

    @Override
    public void onResume() {
        super.onResume();

        if (mCreateJobPresenter.isNeedClear()) {
            onClearButtonClicked();
            mCreateJobPresenter.setJobLocationIdByUser();
            mCreateJobPresenter.setIsNeedClear(false);
        }
    }

    @Override
    public void onDestroy() {
        super.onDestroy();
        EventBus.getDefault().unregister(this);
    }
```

Продолжение приложения Б

```
    //**Set Params**//
```

```
@Override
public void setImagesList(List<Image> images) {
    if (images != null && images.size() > 0) {
        mPhotoAdapter.reset();
        mPhotoAdapter.addAll(images);
        setPhotoTextVisible(false);
    } else {
        mPhotoAdapter.reset();
        setPhotoTextVisible(true);
    }
}

@Override
public void setExecutorCount(String count) {
    mEditExecutorCount.setText(count);
}

@Override
public void setJobType(String jobType) {
    mEditJobType.setText(jobType);
}

@Override
public void setLocation(String region) {
    mEditLocationId.setText(region);
}

@Override
public void setDate(String date) {
    mEditDate.setText(date);
}
```

Продолжение приложения Б

```
@Override
```

```
    public void setJobTypePayment(String name) {
        mEditTypePayment.setText(name);
    }

    @Override
    public void setDescription(String description) {
        mEditDescription.setText(description);
    }

    @Override
    public void enableExecutorCount(boolean enable) {
        if (enable) {
            CompatUtils.setEditTextMaxLength(mEditExecutorCount, 2);
            mEditExecutorCount.setText("");
mEditExecutorCount.setTextColor(getActivity().getResources().getColor(R.color.colorGray));
mExecutorCountText.setTextColor(getActivity().getResources().getColor(R.color.colorBlack));
        } else {
            CompatUtils.setEditTextMaxLength(mEditExecutorCount, 5);
            mEditExecutorCount.setText("Любое");
mEditExecutorCount.setTextColor(getActivity().getResources().getColor(R.color.colorGray));
mExecutorCountText.setTextColor(getActivity().getResources().getColor(R.color.colorGray));
        }

        mEditExecutorCount.setEnabled(enable);
    }

    //**Subscribes**//

    @Subscribe
    public void onTypePaymentChanged(JobTypePayment typePaymentID) {
        mCreateJobPresenter.setJobTypePayment(typePaymentID.getId());
        setJobTypePayment(mCreateJobPresenter.getJobTypePaymentById());
```

Продолжение приложения Б

```
    }
```

```
@Subscribe
public void onTypeJobChanged(JobType typeJob) {
    mCreateJobPresenter.setJobTypeId(typeJob.getId());
    setJobType(mCreateJobPresenter.getJobTypeById());
}

@Subscribe
public void onLocationChanged(Location location) {
    mCreateJobPresenter.setJobLocationId(location.getId());
    setLocation(location.getName());
}

@Subscribe
public void onImageChanges(List<Image> images) {
    setImagesList(images);
}


private void setPhotoTextVisible(boolean visible) {
    if (visible) {
        mPhotoTextView.setVisibility(View.VISIBLE);
        mPhotoRecycler.setVisibility(View.GONE);
    } else {
        mPhotoTextView.setVisibility(View.GONE);
        mPhotoRecycler.setVisibility(View.VISIBLE);
    }
}

///**JobCallbacks**///
@Override
public void onJobLoadError(String error) {
    ShowSnackBar(error, R.color.colorRed);
    hideProgressBar();
```

Продолжение приложения Б

```
}
```

```
@Override
public void onJobLoadSuccess(long jobId) {
    hideProgressBar();
}

@Override
public void onCreateJobError(String error) {
    ShowSnackBar(error, R.color.colorRed);
    hideProgressBar();
}

@Override
public void onCreateJobSuccess(long jobId) {
    hideProgressBar();
    mCreateJobPresenter.setIsNeedClear(true);
    startActivity(new Intent(getActivity(), UserJobView.class).putExtra(JOB_ID, jobId));
}

@Override
public void onJobUpdated(long jobId) {
    hideProgressBar();
    startActivity(new Intent(getActivity(), UserJobView.class).putExtra(JOB_ID, jobId));
}

///**OnClick**///

@OnClick(R.id.create_job_add_photo)
void onAddPhotoClicked() {
    startAddPhotoActivity();
}

@OnClick(R.id.create_new_job_button_create)
void onCreateJobClicked() {
```

Продолжение приложения Б

```
    showProgressBar();
```

```
        mCreateJobPresenter.setJobDescription(mEditDescription.getText().toString());
        mCreateJobPresenter.setJobDate(mEditDate.getText().toString());
        if (mEditExecutorCount.getText().toString().equals("")) {
            onCreateJobError("Укажите количество работников");
            return;
        }
        mCreateJobPresenter.setJobExecutorCount(mEditExecutorCount.getText().toString());
        mCreateJobPresenter.setImagesList(mPhotoAdapter.getList());
        mCreateJobPresenter.createJob();
    }

    @OnClick(R.id.job_edit_locationId)
    void onRegionClicked() {
        startActivity(new Intent(getActivity(), LocationSelectActivity.class));
    }

    @OnClick(R.id.job_edit_date)
    void onDateClicked() {
        DatePicker dateDialog = new DatePicker();
        dateDialog.setOnDateChangeListener(date -> {
            TimePicker timePicker = new TimePicker();
            timePicker.setOnTimeChangedListener(time ->
mEditDate.setText(String.format(Locale.getDefault(), "%s %s", date, time)));
            timePicker.show(getActivity().getSupportFragmentManager(), "TimePicker");
        });
        dateDialog.show(getActivity().getSupportFragmentManager(), "datePicker");
    }

    @OnClick(R.id.job_edit_type)
    void onTypeJobClicked() {
        startActivity(new Intent(getActivity(), ChoiceJobType.class));
    }
```

Продолжение приложения Б

```
    @OnClick(R.id.job_edit_type_payment)
    void onTypePaymentClicked() {
        startActivity(new Intent(getActivity(), ChoiceTypePayment.class));
    }

    @OnClick(R.id.create_new_job_button_clear)
    void onClearButtonClicked() {
        mScrollView.smoothScrollTo(0, 0);
        mCreateJobPresenter.clearButtonClicked();
    }

    ///**View Changes**///
    private void hideProgressBar() {
        mProgressBar.setVisibility(View.GONE);
        mActionLayout.setVisibility(View.VISIBLE);
    }

    private void showProgressBar() {
        mProgressBar.setVisibility(View.VISIBLE);
        mActionLayout.setVisibility(View.GONE);
    }

    @Override
    public void onClick(View view, int position, Object item) {
        startAddPhotoActivity();
    }

    private void startAddPhotoActivity() {
        startActivity(new Intent(getActivity(),
PhotoJob.class).putParcelableArrayListExtra("images", new
ArrayList<>(mPhotoAdapter.getList())));
    }
}
```

Продолжение приложения Б

Листинг 22 – FilterActivity

```
package net.darkeneez.banomo.ui.filter_activity;

import android.content.Intent;
import android.os.Bundle;
import android.view.View;
import android.widget.EditText;
import android.widget.Switch;
import android.widget.TextView;

import net.darkeneez.banomo.R;
import net.darkeneez.banomo.application.models.job.typesJob.JobType;
import net.darkeneez.banomo.application.models.job.typesJob.JobTypePayment;
import net.darkeneez.banomo.application.models.job.typesJob.TypesJob;
import net.darkeneez.banomo.application.models.regions.Location;
import net.darkeneez.banomo.application.models.regions.Locations;
import net.darkeneez.banomo.application.models.user.User;
import net.darkeneez.banomo.application.util.CompatUtils;
import net.darkeneez.banomo.application.util.JobSearchFilter;
import net.darkeneez.banomo.ui.base.BaseActivity;
import net.darkeneez.banomo.ui.choice_element.ChoiceJobType;
import net.darkeneez.banomo.ui.choice_type_payment.ChoiceTypePayment;
import net.darkeneez.banomo.ui.dialogs.DatePicker;
import net.darkeneez.banomo.ui.location_select.LocationSelectActivity;

import org.greenrobot.eventbus.EventBus;
import org.greenrobot.eventbus.Subscribe;

import java.text.ParseException;
import java.text.SimpleDateFormat;
import java.util.Date;
import java.util.Locale;

import butterknife.BindView;
import butterknife.ButterKnife;
import butterknife.OnClick;

import static android.view.View.GONE;
```

```
public class FilterActivity extends BaseActivity {

    private static final String mEmptyRegion = "Любой";
    private static final String mEmptyType = "Любой";
    private static final String mEmptyTypePayment = "Любой";
    private static final String mEmptyDate = "Любая";

    @BindView(R.id.executor_count_text_view)
    TextView mExecutorCountText;

    @BindView(R.id.job_edit_locationId)
    TextView mLocation;

    @BindView(R.id.job_edit_date)
    TextView mDate;

    @BindView(R.id.job_edit_type)
    TextView mJobType;

    @BindView(R.id.job_edit_type_payment)
    TextView mEditTypePayment;

    @BindView(R.id.job_edit_clear_date)
    TextView mClearDate;

    @BindView(R.id.job_edit_executor_count)
    EditText mEditExecutorCount;

    @BindView(R.id.filter_not_show_user_job)
    Switch mNotShowUserJob;

    private long mLocationId = -1, mJobTypeID = -1, mTypePaymentID = -1;
```

Продолжение приложения Б

```
    @Override
    protected void onCreate(Bundle saveInstance) {
        super.onCreate(saveInstance);
        initView();
    }

    private void initView() {
        setUnbinder(ButterKnife.bind(this));
        setTitle("Фильтр поиска");

        onFilterClearClicked();

        readUserFilter();


        EventBus.getDefault().register(this);
    }


    private void readUserFilter() {
        Intent filterIntent = getIntent();

        if (filterIntent == null)
            return;

        JobSearchFilter jobSearchFilter =
filterIntent.getParcelableExtra("JOB_SEARCH_FILTER");

        long bufferForID = jobSearchFilter.getmRegionID();

        if (bufferForID != -1) {
            Location location = new Locations().getInstance().getLocationById(bufferForID);

            if (location != null)
```

Продолжение приложения Б

```
                setRegion(location.getName(), location.getId());
```

```
    }

    bufferForID = jobSearchFilter.getmJobType();

    if (bufferForID != -1) {
        TypesJob typesJob = new TypesJob();
        typesJob.loadCacheTypesJob();

        setJobType(typesJob.getTypeName(bufferForID), bufferForID);
    }

    bufferForID = jobSearchFilter.getmTypePayment();

    if (bufferForID != -1)
        setJobTypePayment(bufferForID);

    bufferForID = jobSearchFilter.getmExecutorCount();
    if (bufferForID != -1)
        setExecutorCount(jobSearchFilter.getmExecutorCount());

    String date = jobSearchFilter.getmDate();

    if (date != null)
        setDate(jobSearchFilter.getmDate());

    if (new User().getInstance().isUserAuthorized())
        setShowUserJob(jobSearchFilter.ismNotShowUserJob());
    else
        mNotShowUserJob.setEnabled(false);
}

private void setShowUserJob(boolean show) {
    mNotShowUserJob.setChecked(show);
}
```

Продолжение приложения Б

```
private void setDate(String date) {
    if (date.equals(mEmptyDate))
        mClearDate.setVisibility(GONE);
    else {
        date = getRightDate(date, "yyyy-dd-mm", "mm.dd.yyyy");
        mClearDate.setVisibility(View.VISIBLE);
    }
    mDate.setText(date);
}

private void setRegion(String regionName, Long regionID) {
    mLocation.setText(regionName);
    mLocationId = regionID;
}

private void setJobType(String jobTypeName, Long jobTypeID) {
    mJobType.setText(jobTypeName);
    mJobTypeID = jobTypeID;
}

private void setExecutorCount(Long executorCount) {
    switch (executorCount.intValue()) {
        case 0: {
            setWorkType();
            break;
        }
        default: {
            setHourType(executorCount.toString());
            break;
        }
    }
}

public void setJobTypePayment(long typePayment) {
```

Продолжение приложения Б

```
    switch ((int) typePayment) {
```

```
        case 2: {
            setExecutorCountNeverMind();
            mEditTypePayment.setText("За работу");
            mTypePaymentID = typePayment;
            break;
        }
        case 1: {
            setExecutorCountHourType();
            mEditTypePayment.setText("Почасовая");
            mTypePaymentID = typePayment;
            break;
        }
        default: {
            setExecutorCountNeverMind();
            mEditTypePayment.setText(mEmptyTypePayment);
            mTypePaymentID = -1;
            break;
        }
    }

}

@Subscribe
public void onLocationChanged(Location location) {
    setRegion(location.getName(), location.getId());
}

@Subscribe
public void onTypePaymentChanged(JobTypePayment typePayment) {
    setJobTypePayment(typePayment.getId());
}

@Subscribe
public void onTypeJobChanged(JobType typeJob) {
```

Продолжение приложения Б

```
    setJobType(typeJob.getName(), typeJob.getId());
```

```
    }

    private void setExecutorCountNeverMind() {
        CompatUtils.setEditTextMaxLength(mEditExecutorCount, 5);
        mEditExecutorCount.setText("Любое");
        mEditExecutorCount.setEnabled(false);
        mEditExecutorCount.setTextColor(getResources().getColor(R.color.colorGray));
        mExecutorCountText.setTextColor(getResources().getColor(R.color.colorGray));
    }

    private void setExecutorCountHourType() {
        CompatUtils.setEditTextMaxLength(mEditExecutorCount, 2);
        mEditExecutorCount.setText("");
        mEditExecutorCount.setEnabled(true);
        mEditExecutorCount.setTextColor(getResources().getColor(R.color.colorAccent));
        mExecutorCountText.setTextColor(getResources().getColor(R.color.colorBlack));
    }

    @OnClick(R.id.job_edit_locationId)
    protected void onRegionClicked() {
        startActivity(new Intent(this,
LocationSelectActivity.class).putExtra("is_from_filter", true));
    }

    @OnClick(R.id.job_edit_type)
    protected void onTypeJobClicked() {
        startActivity(new Intent(this, ChoiceJobType.class).putExtra("is_from_filter",
true));
    }

    @OnClick(R.id.job_edit_type_payment)
    protected void onTypePaymentClicked() {
        startActivity(new Intent(this, ChoiceTypePayment.class).putExtra("is_from_filter",
true));
    }
```

Продолжение приложения Б

```
@OnClick(R.id.job_edit_date)
protected void onDateClicked() {
    DatePicker dateDialog = new DatePicker();
    dateDialog.setOnDateChangeListener(this::setDate);
    dateDialog.show(getSupportFragmentManager(), "datePicker");
}

@OnClick(R.id.job_edit_clear_date)
protected void onDateClearClicked() {
    setDate(mEmptyDate);
}

@OnClick(R.id.filter_clear)
protected void onFilterClearClicked() {
    setJobTypePayment(-1);
    setRegion(mEmptyRegion, (long) -1);
    setJobType(mEmptyType, (long) -1);
    setDate(mEmptyDate);
    setShowUserJob(false);
}

@Override
protected int getLayoutContentViewID() {
    return R.layout.activity_filter;
}

@OnClick(R.id.filter_apply)
protected void onFilterApplyClicked() {
    long executorCount = -1;
    JobSearchFilter jobSearchFilter = new JobSearchFilter();

    jobSearchFilter.setmRegionID(mLocationId);
    jobSearchFilter.setmJobType(mJobTypeID);
```

Продолжение приложения Б

```
        if (mTypePaymentID == 2 || mTypePaymentID == -1) {
            executorCount = (long) 0;
        }
        jobSearchFilter.setmTypePayment(mTypePaymentID);

        if (!mDate.getText().equals(mEmptyDate))
            jobSearchFilter.setmDate(getRightDate(mDate.getText().toString(), "mm.dd.yyyy",
"yyyy-dd-mm"));

        if (executorCount != 0) {
            if (!mEditExecutorCount.getText().toString().equals("")) {
                executorCount = Long.parseLong(mEditExecutorCount.getText().toString());
            }
        }
        jobSearchFilter.setmExecutorCount(executorCount);
        jobSearchFilter.setmNotShowUserJob(mNotShowUserJob.isChecked());

        EventBus.getDefault().post(jobSearchFilter);

        finish();
    }

    private String getRightDate(String date, String patternStart, String patternEnd) {
        String outDate = "";

        SimpleDateFormat inSDF = new SimpleDateFormat(patternStart, new Locale("ru"));
        SimpleDateFormat outSDF = new SimpleDateFormat(patternEnd, new Locale("ru"));

        try {
            Date sDate = inSDF.parse(date);
            outDate = outSDF.format(sDate);
        } catch (ParseException e) {
            e.printStackTrace();
        }
```

Продолжение приложения Б

```
        if (outDate.equals(""))
```

```
            outDate = date;
        return outDate;
    }

    private void setWorkType() {
        CompatUtils.setEditTextMaxLength(mEditExecutorCount, 5);
        mEditExecutorCount.setText("Любое");
        mEditExecutorCount.setEnabled(false);

        int colorGray = getResources().getColor(R.color.colorGray);

        mEditExecutorCount.setTextColor(colorGray);
        mExecutorCountText.setTextColor(colorGray);
    }

    private void setHourType(String persons) {
        CompatUtils.setEditTextMaxLength(mEditExecutorCount, 2);
        mEditExecutorCount.setText(String.format(Locale.getDefault(), "%s", persons));
        mEditExecutorCount.setEnabled(true);
        mEditExecutorCount.setTextColor(getResources().getColor(R.color.colorAccent));
    }

    @Override
    public void onDestroy() {
        super.onDestroy();
        EventBus.getDefault().unregister(this);
    }
}
```

Листинг 23 – ImageFullScreen

```
package net.darkeneez.banomo.ui.image_full_screen;

import android.app.ActionBar;
import android.content.Intent;
import android.os.Build;
```

Продолжение приложения Б

```
import android.os.Bundle;
```

```
import android.view.View;
import android.view.WindowManager;
import android.view.animation.AlphaAnimation;
import android.view.animation.Animation;
import android.widget.RelativeLayout;
import android.widget.TextView;

import androidx.recyclerview.widget.LinearLayoutManager;
import androidx.recyclerview.widget.RecyclerView;
import androidx.viewpager.widget.ViewPager;

import com.esafirm.imagepicker.model.Image;

import net.darkeneez.banomo.R;
import net.darkeneez.banomo.ui.base.BaseActivity;
import net.darkeneez.banomo.ui.base.PFRecyclerViewAdapter;
import net.darkeneez.banomo.ui.photo_thumbnail.PhotoThumbnailAdapter;

import java.util.List;
import java.util.Locale;

import butterknife.BindView;
import butterknife.ButterKnife;

public class ImageFullScreen extends BaseActivity implements
PFRecyclerViewAdapter.OnViewHolderClick, ImageClickImp {

    private static final String ARG_CURRENT_POSITION = "position";
    private static final String ARG_ALL_IMAGES = "images";

    @BindView(R.id.image_full_screen_view_pager)
    ViewPager mViewPager;

    @BindView(R.id.image_full_screen_title)
```

Продолжение приложения Б

```
    TextView mTitle;
```

```
@BindView(R.id.image_full_screen_recycler_view)
RecyclerView mRecyclerView;

@BindView(R.id.image_full_screen_base_layout)
RelativeLayout mRelativeLayout;

private int mCurrentSelected;

private boolean isVisibleInformation = true;

private List<Image> mImages;

@Override
public void onCreate(Bundle saveInstance) {
    super.onCreate(saveInstance);
    initView();
}

private void initView() {
    hideStatusBar();
    setUnbinder(ButterKnife.bind(this));

    initImages();
    initPageAdapter();
    initRecyclerAdapter();
}

private void initImages() {
    Intent intent = getIntent();

    mImages = intent.getParcelableArrayListExtra(ARG_ALL_IMAGES);
    mCurrentSelected = intent.getIntExtra(ARG_CURRENT_POSITION, 0);
}
```

Продолжение приложения Б

```
    private void initRecyclerAdapter() {
        PhotoThumbnailAdapter photoThumbnailAdapter = new PhotoThumbnailAdapter(this, this);
        photoThumbnailAdapter.setList(mImages);
        mRecyclerView.setLayoutManager(new LinearLayoutManager(this,
LinearLayoutManager.HORIZONTAL, false));
        mRecyclerView.setAdapter(photoThumbnailAdapter);
    }

    private void initPageAdapter() {
        ImageFullScreenAdapter adapter = new ImageFullScreenAdapter(this, mImages, this);
        mViewPager.setAdapter(adapter);
        mViewPager.setCurrentItem(mCurrentSelected);
        showTitle(mCurrentSelected);
        mViewPager.addOnPageChangeListener(new ViewPager.OnPageChangeListener() {
            @Override
            public void onPageScrolled(int position, float positionOffset, int
positionOffsetPixels) {

            }

            @Override
            public void onPageSelected(int position) {
                showTitle(position);
            }

            @Override
            public void onPageScrollStateChanged(int state) {

            }
        });
    }

    private void showTitle(int position) {
        if (mTitle.getVisibility() == View.GONE)
            fadeInInformation(mTitle);
```

Продолжение приложения Б

```
        mTitle.setText(String.format(Locale.getDefault(), "%d из %d", position + 1,
mViewPager.getAdapter().getCount()));
    }

    private void hideStatusBar() {
        if (Build.VERSION.SDK_INT < 16) {
            getWindow().setFlags(WindowManager.LayoutParams.FLAG_FULLSCREEN,
WindowManager.LayoutParams.FLAG_FULLSCREEN);
        } else {
            View decorView = getWindow().getDecorView();
            int uiOptions = View.SYSTEM_UI_FLAG_FULLSCREEN;
            decorView.setSystemUiVisibility(uiOptions);
            ActionBar actionBar = getActionBar();
            if (actionBar != null)
                actionBar.hide();
        }
    }

    @Override
    protected int getLayoutContentViewID() {
        return R.layout.activity_image_full_screen;
    }



    private void fadeInInformation(View view) {
        if (view.getVisibility() == View.VISIBLE)
            return;

        AlphaAnimation fadeIn = new AlphaAnimation(0.0f, 1.0f);

        fadeIn.setDuration(500);
        fadeIn.setAnimationListener(new Animation.AnimationListener() {
            @Override
            public void onAnimationStart(Animation animation) {

            }
```

Продолжение приложения Б

```
        @Override
        public void onAnimationEnd(Animation animation) {
            view.setVisibility(View.VISIBLE);
        }

        @Override
        public void onAnimationRepeat(Animation animation) {

        }
    });
    view.startAnimation(fadeIn);
}

private void fadeOutInformation(View view) {

    if (view.getVisibility() == View.GONE)
        return;

    AlphaAnimation fadeOut = new AlphaAnimation(1.0f, 0.0f);

    fadeOut.setDuration(500);
    fadeOut.setAnimationListener(new Animation.AnimationListener() {
        @Override
        public void onAnimationStart(Animation animation) {

        }

        @Override
        public void onAnimationEnd(Animation animation) {
            view.setVisibility(View.GONE);
        }

        @Override
        public void onAnimationRepeat(Animation animation) {
```

Продолжение приложения Б

```
            }
        });
        view.startAnimation(fadeOut);
    }

    @Override
    public void onClick(View view, int position, Object item) {
        mViewPager.setCurrentItem(position);
    }

    @Override
    public void onImageClicked() {
        if (isVisibleInformation) {
            fadeOutInformation(mTitle);
            fadeOutInformation(mRecyclerView);

            isVisibleInformation = false;
        } else {
            fadeInInformation(mTitle);
            fadeInInformation(mRecyclerView);

            isVisibleInformation = true;
        }
    }
}
```

Листинг 24 – JobSearch

```
package net.darkeneez.banomo.ui.jobs_search;

import android.content.Intent;
import android.os.Bundle;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.TextView;
```

Продолжение приложения Б

```
import androidx.fragment.app.FragmentActivity;
import androidx.recyclerview.widget.LinearLayoutManager;
import androidx.recyclerview.widget.RecyclerView;
import androidx.swiperefreshlayout.widget.SwipeRefreshLayout;

import net.darkeneez.banomo.R;
import net.darkeneez.banomo.application.models.user.User;
import net.darkeneez.banomo.application.util.JobSearchFilter;
import net.darkeneez.banomo.application.util.views.VerticalSpaceItemDecoration;
import net.darkeneez.banomo.ui.base.BaseFragment;
import net.darkeneez.banomo.ui.base.PFRecyclerViewAdapter;
import net.darkeneez.banomo.ui.user_job_view.UserJobView;
import net.darkeneez.banomo.ui.view_job.ViewJob;

import org.greenrobot.eventbus.EventBus;
import org.greenrobot.eventbus.Subscribe;

import java.util.List;

import javax.annotation.Nonnull;

import anet.darkeneez.banomo.fragment.Job;
import butterknife.BindView;
import butterknife.ButterKnife;

import static net.darkeneez.banomo.application.util.BanomoApplicationTags.JOB_ID;

public class JobSearch extends BaseFragment implements IJobSearch,
PFRecyclerViewAdapter.OnViewHolderClick,
        SwipeRefreshLayout.OnRefreshListener {

    @BindView(R.id.job_search_recycler_view)
    RecyclerView mRecycleView;
```

Продолжение приложения Б

```
    @BindView(R.id.swipe_layout_with_result)
    SwipeRefreshLayout mSwipeLayoutResult;

    @BindView(R.id.swipe_layout_without_result)
    SwipeRefreshLayout mSwipeLayoutNoResult;

    @BindView(R.id.job_fragment_no_result)
    TextView mNoResult;

    private JobSearchPresenter<JobSearch> mSearchController;
    private JobSearchAdapter mAdapter;

    @Override
    public View onCreateView(@Nonnull LayoutInflater layoutInflater, ViewGroup container,
Bundle saveInstance) {
        View rootView = layoutInflater.inflate(R.layout.fragment_job_search, container,
false);
        initView(rootView);
        return rootView;
    }

    public void initView(View rootView) {
        setUnbinder(ButterKnife.bind(this, rootView));

        mSearchController = new JobSearchPresenter<>();
        mSearchController.attach(this);

        LinearLayoutManager linearLayout = new LinearLayoutManager(getActivity());

        mRecycleView.setLayoutManager(linearLayout);
        mRecycleView.setNestedScrollingEnabled(false);
        mRecycleView.addItemDecoration(new VerticalSpaceItemDecoration(16));

        mAdapter = new JobSearchAdapter(getActivity(), this);
```

Продолжение приложения Б

```
    mRecycleView.setAdapter(mAdapter);
    mSwipeLayoutResult.setOnRefreshListener(this);
    mSwipeLayoutNoResult.setOnRefreshListener(this);

    mSwipeLayoutResult.setRefreshing(true);
    mSearchController.loadJobs();
    EventBus.getDefault().register(this);
}

@Override
public void onDestroy() {
    super.onDestroy();
    EventBus.getDefault().unregister(this);
}

@Subscribe
public void filterChanged(JobSearchFilter jobSearchFilter) {
    mAdapter.reset();
    mSwipeLayoutResult.setRefreshing(true);
    mSearchController.applyFilter(jobSearchFilter);
}

@Override
public void onFilterClicked() {
    startActivity(mSearchController.getFilterParams());
}

public void updateRecycler(List<Job> jobs) {
    mSwipeLayoutResult.setRefreshing(false);
    mSwipeLayoutNoResult.setRefreshing(false);
    if (jobs.size() == 0)
        showNoResultMessage("Не найдено ни одной работы\nПопробуйте изменить фильтр");
    else {
        hideNoResultMessage();
```

Продолжение приложения Б

```
            mAdapter.addAll(jobs);
        }
    }

    @Override
    public void onClick(View view, int position, Object item) {
        Job jobParser = (Job) item;
        FragmentActivity activity = getActivity();

        Intent intent;

        if (new
User().getInstance().getsPhone().equals(jobParser.owner().fragments().owner().phone()))
            intent = new Intent(activity, UserJobView.class);
        else
            intent = new Intent(activity, ViewJob.class);



        intent.putExtra(JOB_ID, jobParser.id());
        activity.startActivity(intent);
    }

    @Override
    public void onRefresh() {
        mAdapter.reset();
        mSearchController.loadJobs();
    }

    public void stopRefresh() {
        mSwipeLayoutResult.setRefreshing(false);
        mSwipeLayoutNoResult.setRefreshing(false);
        showNoResultMessage("Произошла ошибка во время загрузки работ\nПовторите попытку");
    }

    private void hideNoResultMessage() {
```

Продолжение приложения Б

```
        mSwipeLayoutNoResult.setVisibility(View.GONE);
        mSwipeLayoutResult.setVisibility(View.VISIBLE);
    }

    private void showNoResultMessage(String message) {
        mSwipeLayoutNoResult.setVisibility(View.VISIBLE);
        mSwipeLayoutResult.setVisibility(View.GONE);
        mNoResult.setText(message);
    }
}
```

Листинг 25 – LoadActivity

```
package net.darkeneez.banomo.ui.load;

import android.app.AlertDialog;
import android.content.Context;
import android.content.Intent;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.ProgressBar;
import android.widget.TextView;

import net.darkeneez.banomo.R;
import net.darkeneez.banomo.ui.base.BaseActivity;
import net.darkeneez.banomo.ui.login.phone_input.PhoneInput;
import net.darkeneez.banomo.ui.main.MainActivity;
import net.darkeneez.banomo.ui.registration.RegistrationActivity;

import butterknife.BindView;
import butterknife.ButterKnife;
import butterknife.OnClick;

public class LoadActivity extends BaseActivity implements ILoad {

    @BindView(R.id.load_progress_bar)
```

Продолжение приложения Б

```
ProgressBar mLoadingBar;

@BindView(R.id.skip_auth_button)
TextView mSkipAuthButton;

@BindView(R.id.login_button)
Button mLoginButton;

@BindView(R.id.load_buttons_layout)
View mButtonsView;

@BindView(R.id.registration_button)
Button mRegistration;



private LoadPresenter<LoadActivity> mLoadPresenter;

@Override
protected void onCreate(Bundle saveInstanceState) {
    setTheme(R.style.BanomoTheme_NoToolBar);
    super.onCreate(saveInstanceState);

    setUnbinder(ButterKnife.bind(this));

    mLoadPresenter = new LoadPresenter<>();

    mLoadPresenter.attach(this);
    mLoadPresenter.loadApp();
}

@Override
protected int getLayoutContentViewID() {
    return R.layout.activity_load;
}
```

Продолжение приложения Б

```
@Override
public void startMainActivity() {
    final Context context = this;
    runOnUiThread(() -> {
        startActivity(new Intent(context, MainActivity.class));
        finish();
    });
}

@Override
public void onLoadingEnd() {
    runOnUiThread(() -> {
        mLoadingBar.setVisibility(View.GONE);
        mButtonsView.setVisibility(View.VISIBLE);
        mSkipAuthButton.setVisibility(View.VISIBLE);
    });
}

@OnClick(R.id.login_button)
public void onLoginButtonClicked() {
    startActivity(new Intent(this, PhoneInput.class));
}

@OnClick(R.id.registration_button)
public void onRegistrationButtonClicked() {
    startActivity(new Intent(this, RegistrationActivity.class));
}

@OnClick(R.id.skip_auth_button)
public void onSkipAuthButtonClicked() {
    AlertDialog.Builder mBuilder = new AlertDialog.Builder(this);

    final View mView = this.getLayoutInflater().inflate(R.layout.dialog_skip_auth, null);
```

Продолжение приложения Б

```
        mBuilder.setView(mView);

         final AlertDialog skipAlertDialog = mBuilder.create();
         skipAlertDialog.setCanceledOnTouchOutside(false);

         mView.findViewById(R.id.dialog_skip_auth_cancel_button).setOnClickListener(view ->
skipAlertDialog.cancel());

         mView.findViewById(R.id.dialog_skip_auth_button).setOnClickListener(view -> {
             skipAlertDialog.cancel();
             startActivity(new Intent(LoadActivity.this, MainActivity.class));
             finish();
         });

         skipAlertDialog.show();
    }
}
```

Листинг 26 – LocationSelectActivity

```
package net.darkeneez.banomo.ui.location_select;

import android.annotation.SuppressLint;
import android.os.Bundle;
import android.text.Editable;
import android.text.TextWatcher;
import android.widget.EditText;

import androidx.appcompat.app.AppCompatActivity;
import androidx.recyclerview.widget.DefaultItemAnimator;
import androidx.recyclerview.widget.LinearLayoutManager;
import androidx.recyclerview.widget.RecyclerView;

import net.darkeneez.banomo.R;
import net.darkeneez.banomo.application.models.regions.Location;
import net.darkeneez.banomo.application.models.regions.Locations;
```

Продолжение приложения Б

```
import org.greenrobot.eventbus.EventBus;


import java.util.ArrayList;
import java.util.List;


import butterknife.BindView;
import butterknife.ButterKnife;
import butterknife.OnClick;


public class LocationSelectActivity extends AppCompatActivity implements
LocationSelectAdapter.LocationSelectListener {


    @BindView(R.id.content_select_region_recycler)
    RecyclerView recyclerView;


    @BindView(R.id.location_select_searchView)
    EditText mSearchEditText;


    private List<Location> contactList;
    private LocationSelectAdapter mAdapter;



    @SuppressLint("ClickableViewAccessibility")
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_select_location);


        ButterKnife.bind(this);
        setTitle("Выбор региона");


        contactList = new ArrayList<>();
        mAdapter = new LocationSelectAdapter(contactList, this);
```

Продолжение приложения Б

```
        RecyclerView.LayoutManager mLayoutManager = new
LinearLayoutManager(getApplicationContext());


        recyclerView.setLayoutManager(mLayoutManager);
        recyclerView.setItemAnimator(new DefaultItemAnimator());
        recyclerView.setAdapter(mAdapter);


        contactList.clear();
        contactList.addAll(new Locations().getInstance().getLocations());


        if (getIntent().getBooleanExtra("is_from_filter", false))
            contactList.add(0, new Location("Любой", (long) -1));


        mAdapter.notifyDataSetChanged();


        mSearchEditText.addTextChangedListener(new TextWatcher() {
            @Override
            public void beforeTextChanged(CharSequence charSequence, int i, int i1, int i2) {


            }


            @Override
            public void onTextChanged(CharSequence charSequence, int i, int i1, int i2) {


            }


            @Override
            public void afterTextChanged(Editable editable) {
                searchLocations(editable.toString());
            }
        });
    }


    @OnClick(R.id.location_select_clear_action)
```

Продолжение приложения Б

```
    protected void onClearClicked() {
        mSearchEditText.setText("");
    }

    @OnClick(R.id.location_select_search_action)
    protected void onSearchClicked() {
        searchLocations(mSearchEditText.getText().toString());
    }

    private void searchLocations(String query) {
        mAdapter.getFilter().filter(query);
    }

    @Override
    public void onLocationSelect(Location location) {
        EventBus.getDefault().post(location);
        finish();
    }
}
```

Листинг 27 – PhoneInput

```
package net.darkeneez.banomo.ui.login.phone_input;



import android.app.AlertDialog;
import android.content.Intent;
import android.os.Bundle;
import android.text.Editable;
import android.text.TextWatcher;
import android.view.View;
import android.widget.Button;
import android.widget.CheckBox;
import android.widget.TextView;

import com.google.android.material.button.MaterialButton;
```

Продолжение приложения Б

```
import net.darkeneez.banomo.R;
import net.darkeneez.banomo.application.models.user.RegistrationUser;
import net.darkeneez.banomo.application.util.views.MaskedEditText.MaskedEditText;
import net.darkeneez.banomo.ui.base.BaseActivity;
import net.darkeneez.banomo.ui.choice_element.ChoiceWebView;
import net.darkeneez.banomo.ui.login.phone_verification.PhoneVerification;
import net.darkeneez.banomo.ui.registration.RegistrationActivity;

import butterknife.BindView;
import butterknife.ButterKnife;
import butterknife.OnClick;

public class PhoneInput extends BaseActivity implements IPhoneInput {
    @BindView(R.id.phone_input_edit_text)
    MaskedEditText mPhoneEditText;

    @BindView(R.id.phone_input_continue)
    Button mContinue;

    @BindView(R.id.phone_input_confidence)
    CheckBox mConfidence;

    @BindView(R.id.phone_input_confidence_layout)
    View mPolicyView;

    private PhoneInputPresenter<PhoneInput> mPhoneInputPresenter;

    private boolean isNewUser = false;

    private RegistrationUser mRegistrationUser = null;

    @Override
    protected void onCreate(Bundle saveInstanceState) {
        super.onCreate(saveInstanceState);
```

Продолжение приложения Б

```
        initView();
```

```
}

public RegistrationUser getRegistrationUser() {
    return mRegistrationUser;
}

protected void initView() {
    setUnbinder(ButterKnife.bind(this));

    mPhoneInputPresenter = new PhoneInputPresenter<>();
    mPhoneInputPresenter.attach(this);

    mPhoneEditText.addTextChangedListener(new TextWatcher() {
        @Override
        public void beforeTextChanged(CharSequence charSequence, int i, int i1, int i2) {

        }

        @Override
        public void onTextChanged(CharSequence charSequence, int i, int i1, int i2) {

        }

        @Override
        public void afterTextChanged(Editable editable) {
            if (editable.length() == 13) {
                hideKeyBoard();
            }
        }
    });

    Intent intent = getIntent();

    isNewUser = intent.getBooleanExtra("new_user", false);
```

Продолжение приложения Б

```
        if (isNewUser) {
            mRegistrationUser = intent.getParcelableExtra("registration_user");
            mPolicyView.setVisibility(View.VISIBLE);
        }

        setTitle("Введите телефон");
    }

    @Override
    protected int getLayoutContentViewID() {
        return R.layout.activity_phone_input;
    }

    @Override
    public void startPhoneValidate(long SMS) {
        Intent intent = new Intent(PhoneInput.this, PhoneVerification.class);
        intent.putExtra("SMS", SMS);

        if (isNewUser)
            intent.putExtra("registration_user", mRegistrationUser);

        startActivity(intent);
    }

    private boolean checkCanContinue() {
        if (!mPhoneInputPresenter.checkPhone()) {
            ShowSnackBar("Введите номер телефона", R.color.colorRed);
            return false;
        } else if (isNewUser && !mConfidence.isChecked()) {
            ShowSnackBar("Вы должны принять условия и правила политики конфиденциальности",
R.color.colorRed);
            return false;
        }
        return true;
```

Продолжение приложения Б

```
    }
```

```
@OnClick(R.id.phone_input_confidence_text)
protected void onPrivatePolicyClicked() {
    startActivity(new Intent(this, ChoiceWebView.class));
}

@OnClick(R.id.phone_input_continue)
protected void onContinueClicked() {
    hideKeyBoard();
    mPhoneInputPresenter.setPhone(mPhoneEditText.getRawText());

    if (checkCanContinue()) {
        if (isNewUser)
            mPhoneInputPresenter.checkPhoneIfUsed();
        else
            mPhoneInputPresenter.loadUser();
    }
}

@Override
public void newUserPhoneIsUsed() {
    ShowSnackBar("Данный номер телефона уже используется", R.color.colorRed);
}

@Override
public void userIsNotRegistered() {

    AlertDialog.Builder mBuilder = new AlertDialog.Builder(this);

    final View mView = this.getLayoutInflater().inflate(R.layout.dialog_yes_no, null);
    mBuilder.setView(mView);

    final AlertDialog skipAlertDialog = mBuilder.create();
    skipAlertDialog.setCanceledOnTouchOutside(false);
```

Продолжение приложения Б

```
        TextView textView = mView.findViewById(R.id.dialog_description);
        textView.setText("Данный пользователь не зарегистрирован");

        textView = mView.findViewById(R.id.dialog_title);
        textView.setText("Внимание!");

        mView.findViewById(R.id.dialog_no).setOnClickListener(view ->
skipAlertDialog.cancel());

        MaterialButton buttonYes = mView.findViewById(R.id.dialog_yes);
        buttonYes.setText("Регистрация");
        buttonYes.setOnClickListener(view -> {
            skipAlertDialog.cancel();
            startActivity(new Intent(this, RegistrationActivity.class));
        });
        skipAlertDialog.show();
    }
}
```

Листинг 28 – PhoneVerification

```
package net.darkeneez.banomo.ui.login.phone_verification;

import android.content.Intent;
import android.os.Bundle;
import android.os.CountDownTimer;
import android.text.Editable;
import android.view.View;
import android.widget.EditText;

import com.google.android.material.button.MaterialButton;

import net.darkeneez.banomo.R;
import net.darkeneez.banomo.ui.base.BaseActivity;
import net.darkeneez.banomo.ui.main.MainActivity;
```

Продолжение приложения Б

```
import java.util.Locale;


import butterknife.BindView;
import butterknife.BindViews;
import butterknife.ButterKnife;
import butterknife.OnClick;
import butterknife.OnTextChanged;


public class PhoneVerification extends BaseActivity implements IPhoneVerification {
    private static int mTime = 60;
    private static int mCurrentTime = -1;


    @BindViews({R.id.code_one_part, R.id.code_two_part, R.id.code_three_part,
R.id.code_four_part})
    EditText[] mCodes_Array;


    @BindView(R.id.phone_verification_root_layout)
    View view;


    @BindView(R.id.phone_verification_request_code)
    MaterialButton mRequestAgain;


    private PhoneVerificationPresenter<PhoneVerification> mPhoneVerificationPresenter;
    private boolean clear;
    private CountDownTimer mCountDownTimer;


    @Override
    protected void onCreate(Bundle saveInstanceState) {
        super.onCreate(saveInstanceState);
        initView();
    }


    protected void initView() {
        setUnbinder(ButterKnife.bind(this));
```

Продолжение приложения Б

```
        mPhoneVerificationPresenter = new PhoneVerificationPresenter<>();
        mPhoneVerificationPresenter.attach(this);



        mPhoneVerificationPresenter.readSMS(getIntent().getLongExtra("SMS", 0));

mPhoneVerificationPresenter.setRegistrationUser(getIntent().getParcelableExtra("registration_
user"));


        if (mCurrentTime != -1) {
            mTime = mCurrentTime;
            startCountDownTimer();
        }
    }


    @Override
    protected int getLayoutContentViewID() {
        return R.layout.activity_phone_verification;
    }


    @OnClick(R.id.phone_verification_request_code)
    void onRequestCodeAgain() {
        mPhoneVerificationPresenter.requestCode();
        startCountDownTimer();
    }


    @OnTextChanged(value = {R.id.code_one_part, R.id.code_two_part, R.id.code_three_part,
R.id.code_four_part},
            callback = OnTextChanged.Callback.AFTER_TEXT_CHANGED)
    public void OnTextChanged(Editable editable) {
        if (!clear)
            mPhoneVerificationPresenter.onTextChanged(editable.toString());
    }


    public void focusNext(int index) {
```

Продолжение приложения Б

```
        mCodes_Array[index].requestFocus();
```

```
}

public void clearAll() {
    clear = true;

    for (EditText editText : mCodes_Array) {
        editText.getText().clear();
    }
    mCodes_Array[3].clearFocus();

    clear = false;
}

@Override
public void onDestroy() {
    super.onDestroy();
    if (mCountDownTimer != null)
        mCountDownTimer.cancel();
}

@Override
public void startMainActivity() {
    Intent intent = new Intent(this, MainActivity.class);
    intent.addFlags(Intent.FLAG_ACTIVITY_NEW_TASK | Intent.FLAG_ACTIVITY_CLEAR_TASK);

    startActivity(intent);
    finish();
}

public Long getUserInputSMS() {
    StringBuilder sms = new StringBuilder();
    for (int i = 0; i < 4; i++) {
        sms.append(mCodes_Array[i].getText().toString());
    }
```

Продолжение приложения Б

```
        return Long.valueOf(sms.toString());
    }

    public void startCountDownTimer() {
        mRequestAgain.setEnabled(false);

mRequestAgain.setSupportBackgroundTintList(getResources().getColorStateList(R.color.colorGray
));

        mCountDownTimer = new CountDownTimer(mTime * 1000, 1000) {

            @Override
            public void onTick(long l) {
                mCurrentTime = Math.round(l / 1000);
                mRequestAgain.setText(String.format(Locale.getDefault(), "Запросить еще раз
через 0:%s", checkDigit(mCurrentTime)));
            }

            @Override
            public void onFinish() {

mRequestAgain.setText(getResources().getString(R.string.banomo_request_code_again));

mRequestAgain.setSupportBackgroundTintList(getResources().getColorStateList(R.color.colorAcce
nt));
                mRequestAgain.setEnabled(true);

                mCurrentTime = -1;
            }
        }.start();
    }

    public String checkDigit(int number) {
        return number <= 9 ? "0" + number : String.valueOf(number);
    }
```

Продолжение приложения Б

```
    @Override
```

```
    public void onCodeError() {
        clearAll();
        ShowSnackBar("Неправильный код", R.color.colorRed);
    }
}
```

Листинг 29 – MainActivity

```
package net.darkeneez.banomo.ui.main;

import android.app.AlertDialog;
import android.content.Intent;
import android.graphics.drawable.Drawable;
import android.os.Bundle;
import android.view.View;
import android.widget.ImageView;
import android.widget.TextView;

import androidx.annotation.Nullable;
import androidx.fragment.app.Fragment;
import androidx.fragment.app.FragmentManager;

import com.google.android.material.bottomnavigation.BottomNavigationView;

import net.darkeneez.banomo.R;
import net.darkeneez.banomo.ui.base.BaseActivity;
import net.darkeneez.banomo.ui.create_job.CreateJob;
import net.darkeneez.banomo.ui.jobs_search.JobSearch;
import net.darkeneez.banomo.ui.login.phone_input.PhoneInput;
import net.darkeneez.banomo.ui.user_case.UserCase;

import org.greenrobot.eventbus.EventBus;
import org.greenrobot.eventbus.Subscribe;

import butterknife.BindView;
```

Продолжение приложения Б

```
import butterknife.ButterKnife;
import butterknife.OnClick;
```

```
import static net.darkeneez.banomo.application.util.BanomoApplicationTags.JOB_ID;

public class MainActivity extends BaseActivity implements IMainActivity {

    @BindView(R.id.main_activity_toolbar)
    View mToolbarView;
    @BindView(R.id.navigation_bottom)
    BottomNavigationView mBottomNavigationView;

    private MainActivityPresenter<MainActivity> mMainActivityPresenter;
    private TextView mToolbarTitle;
    private ImageView mToolbarIcon;

    @Override
    protected void onCreate(Bundle saveInstanceState) {
        super.onCreate(saveInstanceState);
        initView();

        EventBus.getDefault().register(this);
    }

    @Subscribe
    public void onEditJob(Long jobID) {
        Bundle bundle = new Bundle();
        bundle.putBoolean("is_edit", true);
        bundle.putLong(JOB_ID, jobID);


        Fragment createJob = new CreateJob();
        createJob.setArguments(bundle);
        mMainActivityPresenter.setMenuFragment(createJob, false, "Редактирование работы");
        mBottomNavigationView.setOnNavigationItemSelectedListener(null);
```

Продолжение приложения Б

```
        mBottomNavigationView.setSelectedItemId(R.id.action_add_offer);
```

```
        mBottomNavigationView.setOnNavigationItemSelectedListener(item ->
mMainActivityPresenter.NavigationMenuSelected(item));
    }

    @Subscribe
    public void onRepeatJob(RepeatJob job) {
        Bundle bundle = new Bundle();
        bundle.putBoolean("is_repeat", true);
        bundle.putLong(JOB_ID, job.getId());


        Fragment createJob = new CreateJob();
        createJob.setArguments(bundle);
        mMainActivityPresenter.setMenuFragment(createJob, false, "Повтор работы");
        mBottomNavigationView.setOnNavigationItemSelectedListener(null);
        mBottomNavigationView.setSelectedItemId(R.id.action_add_offer);
        mBottomNavigationView.setOnNavigationItemSelectedListener(item ->
mMainActivityPresenter.NavigationMenuSelected(item));
    }

    public void initView() {
        setUnbinder(ButterKnife.bind(this));
        mMainActivityPresenter = new MainActivityPresenter<>();
        mMainActivityPresenter.attach(this);
        mMainActivityPresenter.startPushService();

        mToolbarTitle = mToolbarView.findViewById(R.id.toolbar_title);
        mToolbarIcon = mToolbarView.findViewById(R.id.toolbar_icon);

        BottomNavigationView mBottom = findViewById(R.id.navigation_bottom);

        mBottom.setOnNavigationItemSelectedListener(item ->
mMainActivityPresenter.NavigationMenuSelected(item));

        mMainActivityPresenter.setMenuFragment(new JobSearch(), true, "Поиск работы",
"JobSearch");
```

Продолжение приложения Б

```
    }
```

```
@Override
protected int getLayoutContentViewID() {
    return R.layout.activity_main;
}

@Override
public void onBackPressed() {
    openQuitDialog();
}

@Override
public void openQuitDialog() {


    AlertDialog.Builder builder = new AlertDialog.Builder(MainActivity.this);
    final View mView = this.getLayoutInflater().inflate(R.layout.dialog_yes_no, null);
    builder.setView(mView);

    final AlertDialog dialog_quit = builder.create();

    mView.findViewById(R.id.dialog_no).setOnClickListener(view -> dialog_quit.cancel());

    mView.findViewById(R.id.dialog_yes).setOnClickListener(
            v -> {
                dialog_quit.cancel();
                finish();
            });
    dialog_quit.show();
}

@Override
protected void onActivityResult(int requestCode, int resultCode, @Nullable Intent data) {
    super.onActivityResult(requestCode, resultCode, data);
```

Продолжение приложения Б

```
    if (requestCode == 107)
```

```
            mMainActivityPresenter.setMenuFragment(new JobSearch(), true, "Поиск работы",
"JobSearch");
    }

    @Override
    public void startPhoneInputActivity() {
        startActivity(new Intent(this, PhoneInput.class));
    }

    @Override
    public void setTitle(String title) {
        setToolbarTitle(title);
    }

    @Override
    public void setToolbarTitle(String title) {
        mToolbarTitle.setText(title);
    }

    @Override
    public void setToolbarIcon(Drawable icon) {
        mToolbarIcon.setImageDrawable(icon);
    }

    @Override
    public void setToolbarIconVisible(boolean isVisible) {
        mToolbarIcon.setVisibility(isVisible ? View.VISIBLE : View.GONE);
    }

    @OnClick(R.id.toolbar_icon)
    public void onToolbarIconClicked() {
        FragmentManager fm = getSupportFragmentManager();

        JobSearch jobSearch = (JobSearch) fm.findFragmentByTag("JobSearch");
```

Продолжение приложения Б

```
        if (jobSearch != null) {
            jobSearch.onFilterClicked();
            return;
        }
        UserCase userCase = (UserCase) fm.findFragmentByTag("UserCase");

        if (userCase != null) {
            userCase.onFilterClicked();
        }
    }

    @Override
    public void onDestroy() {
        super.onDestroy();
        EventBus.getDefault().unregister(this);
    }
}
```

Листинг 30 – PhotoJob

```
package net.darkeneez.banomo.ui.photo_job;

import android.content.Intent;
import android.os.Bundle;
import android.view.View;
import android.widget.ImageView;

import androidx.recyclerview.widget.GridLayoutManager;
import androidx.recyclerview.widget.RecyclerView;

import com.esafirm.imagepicker.features.ImagePicker;
import com.esafirm.imagepicker.model.Image;

import net.darkeneez.banomo.R;
import net.darkeneez.banomo.ui.base.BaseActivity;
```

Продолжение приложения Б

```
import org.greenrobot.eventbus.EventBus;

import java.util.ArrayList;
import java.util.List;

import butterknife.BindView;
import butterknife.ButterKnife;
import butterknife.OnClick;

public class PhotoJob extends BaseActivity {
    @BindView(R.id.activity_photo_job_recycler)
    RecyclerView mRecyclerView;

    @BindView(R.id.photo_job_toolbar_delete)
    ImageView mDeleteButton;

    @BindView(R.id.photo_job_toolbar_add)
    ImageView mAddButton;

    private PhotoAdapter mPhotoAdapter;

    private boolean mDeleteMode;



    private PhotoAdapter.onImageClickedCallback mDeleteImageClickCalLBack = (image,
viewHolder) -> {
        if (mDeleteMode)
            viewHolder.removeItem(image);
    };

    @Override
    public void onCreate(Bundle saveInstance) {
        super.onCreate(saveInstance);
        initView();
```

Продолжение приложения Б

```
    }
```

```
private void initView() {
    setUnbinder(ButterKnife.bind(this));

    mPhotoAdapter = new PhotoAdapter(this, null);
    mPhotoAdapter.setOnImageClickedCallback(mDeleteImageClickCalLBack);

    initRecyclerView();
    setEditMode(getIntent().getParcelableArrayListExtra("images"));
}

private void initRecyclerView() {
    mRecyclerView.setHasFixedSize(true);
    mRecyclerView.setNestedScrollingEnabled(false);
    mRecyclerView.setLayoutManager(new GridLayoutManager(this, 2));
    mRecyclerView.setAdapter(mPhotoAdapter);
}

private void setEditMode(ArrayList<Image> images) {
    if (images != null && images.size() > 0) {
        mPhotoAdapter.setList(images);
        mDeleteButton.setVisibility(View.VISIBLE);
    }
    mAddButton.setVisibility(View.VISIBLE);
}

@Override
public void onDestroy() {
    super.onDestroy();
    EventBus.getDefault().post(mPhotoAdapter.getList());
}

@Override
protected int getLayoutContentViewID() {
```

Продолжение приложения Б

```
    return R.layout.activity_photo_job;
```

```
    }


    @OnClick(R.id.photo_job_toolbar_add)
    protected void onAddButtonClicked() {
        ImagePicker.create(this)
                .folderMode(true)
                .toolbarFolderTitle("Папка")
                .toolbarImageTitle("Нажмите, чтобы выбрать")
                .theme(R.style.BanomoTheme_NoToolBar)
                .limit(10)
                .imageDirectory("Camera")
                .enableLog(true)
                .start();
    }

    @OnClick(R.id.photo_job_toolbar_delete)
    protected void onDeleteButtonClicked() {
        enableDeleteMode(true);
    }

    private void enableDeleteMode(boolean enable) {
        if (enable) {

mAddButton.setImageDrawable(getResources().getDrawable(R.drawable.ic_check_mark));
            mAddButton.setOnClickListener(v -> enableDeleteMode(false));
            mDeleteButton.setVisibility(View.GONE);
        } else {
            mAddButton.setImageDrawable(getResources().getDrawable(R.drawable.ic_add));
            mAddButton.setOnClickListener(v -> onAddButtonClicked());
            if (mPhotoAdapter.getItemCount() > 0)
                mDeleteButton.setVisibility(View.VISIBLE);
            else
                mDeleteButton.setVisibility(View.GONE);
```

Продолжение приложения Б

```
        }
```

```
        mDeleteMode = enable;
    }

    @Override
    protected void onActivityResult(int requestCode, int resultCode, Intent data) {
        if (ImagePicker.shouldHandle(requestCode, resultCode, data)) {
            List<Image> images = ImagePicker.getImages(data);
            if (images != null && images.size() > 0) {
                mPhotoAdapter.addAll(images);
                mDeleteButton.setVisibility(View.VISIBLE);
            } else {
                mDeleteButton.setVisibility(View.GONE);
            }
        }
        super.onActivityResult(requestCode, resultCode, data);
    }
}
```

Листинг 31 – RateExecutors

```
package net.darkeneez.banomo.ui.rate_executors;

import android.content.res.ColorStateList;
import android.os.Bundle;
import android.view.View;
import android.widget.LinearLayout;
import android.widget.ProgressBar;
import android.widget.TextView;

import androidx.recyclerview.widget.LinearLayoutManager;
import androidx.recyclerview.widget.RecyclerView;

import com.apollographql.apollo.ApolloCall;
import com.apollographql.apollo.api.Response;
import com.apollographql.apollo.exception.ApolloException;
```

Продолжение приложения Б

```
import com.google.android.material.button.MaterialButton;
```

```
import net.darkeneez.banomo.R;
import net.darkeneez.banomo.application.apollo.BanomoApolloClient;
import net.darkeneez.banomo.application.models.user.User;
import net.darkeneez.banomo.application.util.views.VerticalSpaceItemDecoration;
import net.darkeneez.banomo.ui.base.BaseActivity;

import java.util.ArrayList;
import java.util.List;

import javax.annotation.Nonnull;

import anet.darkeneez.banomo.GetJobByIdQuery;
import anet.darkeneez.banomo.RateExecutorsMutation;
import anet.darkeneez.banomo.fragment.Job;
import butterknife.BindView;
import butterknife.ButterKnife;
import butterknife.OnClick;

import static net.darkeneez.banomo.application.util.BanomoApplicationTags.JOB_ID;

public class RateExecutors extends BaseActivity implements
RateExecutorsAdapter.onRatingChanged {

    @BindView(R.id.rate_executors_progress)
    ProgressBar mProgressBar;

    @BindView(R.id.rate_executors_accept)
    MaterialButton mRateExecutorsAccept;

    @BindView(R.id.rate_executors_buttons_layout)
    LinearLayout mActionCardView;

    @BindView(R.id.rate_executors_recycler)
```

Продолжение приложения Б

```
    RecyclerView mRecycleView;
```

```
@BindView(R.id.rate_executors_rated)
TextView mAllRated;

private RateExecutorsAdapter mAdapter;

private ArrayList<Long> mOffersUp = new ArrayList<>();
private ArrayList<Long> mOffersDown = new ArrayList<>();

@Override
protected void onCreate(Bundle saveInstanceState) {
    super.onCreate(saveInstanceState);

    setUnbinder(ButterKnife.bind(this));

    LinearLayoutManager linearLayout = new LinearLayoutManager(this);

    mRecycleView.setLayoutManager(linearLayout);
    mRecycleView.setNestedScrollingEnabled(false);
    mRecycleView.addItemDecoration(new VerticalSpaceItemDecoration(16));

    mAdapter = new RateExecutorsAdapter(this, this);
    mRecycleView.setAdapter(mAdapter);

    loadOffers();
}

@Override
protected int getLayoutContentViewID() {
    return R.layout.activity_rate_executors;
}

private void loadOffers() {
    BanomoApolloClient.getApolloClient().query(GetJobByIdQuery.builder()
```

Продолжение приложения Б

```
            .id(getIntent().getLongExtra(JOB_ID, -1))
```

```
                .build())
                .enqueue(new ApolloCall.Callback<GetJobByIdQuery.Data>() {
                    @Override
                    public void onResponse(@Nonnull Response<GetJobByIdQuery.Data> response)
{
                        if (response.hasErrors())
                            return;

                        final List<Job.Offer> offers = new ArrayList<>();

                        for (Job.Offer offer :
response.data().JobById().fragments().job().offers()) {
                            if (!offer.fragments().offer().isLike())
                                offers.add(offer);
                        }

                        runOnUiThread(() -> updateRecycler(offers));
                    }

                    @Override
                    public void onFailure(@Nonnull ApolloException e) {

                    }
                });
    }

    private void updateRecycler(List offers) {
        if (offers.size() == 0)
            goneRateElements();
        else
            mAdapter.addAll(offers);
    }

    private void goneRateElements() {
        mActionCardView.setVisibility(View.GONE);
```

Продолжение приложения Б

```
        mRecycleView.setVisibility(View.GONE);
```

```
        mAllRated.setVisibility(View.VISIBLE);
    }



    @Override
    public void addRating(boolean up, final long offerID) {
        if (up)
            mOffersUp.add(offerID);
        else
            mOffersDown.add(offerID);


        checkButton();
    }


    @Override
    public void deleteRating(boolean up, long offerID) {
        if (up)
            mOffersUp.remove(offerID);
        else
            mOffersDown.remove(offerID);
        checkButton();
    }


    private void checkButton() {
        if (mOffersUp.size() == 0 && mOffersDown.size() == 0) {
            mRateExecutorsAccept.setEnabled(false);

mRateExecutorsAccept.setSupportBackgroundTintList(ColorStateList.valueOf(getResources().getCo
lor(R.color.colorGray)));
        } else {
            mRateExecutorsAccept.setEnabled(true);

mRateExecutorsAccept.setSupportBackgroundTintList(ColorStateList.valueOf(getResources().getCo
lor(R.color.colorAccent)));
        }
    }
```

Продолжение приложения Б

```
    @OnClick(R.id.rate_executors_accept)
    protected void onExecutorsRateAccept() {
        showProgress();

        User user = new User().getInstance();

        RateExecutorsAdapter.onRatingSuccess onRatingSuccess =
mAdapter.getmOnRatingSuccess();

        for (Long offerID : mOffersDown) {
            BanomoApolloClient.getApolloClient().mutate(RateExecutorsMutation.builder()
                    .offerId(offerID)
                    .phone(user.getsPhone())
                    .token(user.getsToken())
                    .ratingExecutorUp(false)
                    .ratingExecutorDown(true)
                    .build())
                    .enqueue(new ApolloCall.Callback<RateExecutorsMutation.Data>() {
                        @Override
                        public void onResponse(@Nonnull Response<RateExecutorsMutation.Data>
response) {
                            if (response.hasErrors()) {
                                runOnUiThread(() ->
ShowSnackBar(response.errors().get(0).message(), R.color.colorRed));
                            }

                            runOnUiThread(() -> onRatingSuccess.onRatingSuccess(offerID));
                        }

                        @Override
                        public void onFailure(@Nonnull ApolloException e) {
                            ShowSnackBar(e.getMessage(), R.color.colorRed);
                        }
                    });
        }
```

Продолжение приложения Б

```
        for (Long offerID : mOffersUp) {
            BanomoApolloClient.getApolloClient().mutate(RateExecutorsMutation.builder()
                    .offerId(offerID)
                    .phone(user.getsPhone())
                    .token(user.getsToken())
                    .ratingExecutorUp(true)
                    .ratingExecutorDown(false)
                    .build())
                    .enqueue(new ApolloCall.Callback<RateExecutorsMutation.Data>() {
                        @Override
                        public void onResponse(@Nonnull Response<RateExecutorsMutation.Data>
response) {
                            if (response.hasErrors()) {
                                runOnUiThread(() ->
ShowSnackBar(response.errors().get(0).message(), R.color.colorRed));
                            }

                            runOnUiThread(() -> onRatingSuccess.onRatingSuccess(offerID));
                        }

                        @Override
                        public void onFailure(@Nonnull ApolloException e) {
                            ShowSnackBar(e.getMessage(), R.color.colorRed);
                        }
                    });
        }

        hideProgress(mAdapter.offersCount() == 0);
    }

    private void showProgress() {
        mRateExecutorsAccept.setVisibility(View.GONE);
        mProgressBar.setVisibility(View.VISIBLE);
    }
```

Продолжение приложения Б

```
    private void hideProgress(boolean isButton) {
```

```
        if (isButton)
            mRateExecutorsAccept.setVisibility(View.VISIBLE);
        else
            mAllRated.setVisibility(View.VISIBLE);
        mProgressBar.setVisibility(View.GONE);
    }
}
```

Листинг 32 – RegistrationActivity

```
package net.darkeneez.banomo.ui.registration;

import android.Manifest;
import android.content.Intent;
import android.content.pm.PackageManager;
import android.graphics.Bitmap;
import android.net.Uri;
import android.os.Build;
import android.os.Bundle;
import android.widget.EditText;
import android.widget.TextView;

import androidx.annotation.NonNull;
import androidx.appcompat.app.AlertDialog;
import androidx.core.app.ActivityCompat;
import androidx.core.content.ContextCompat;

import net.darkeneez.banomo.R;
import net.darkeneez.banomo.application.models.regions.Location;
import net.darkeneez.banomo.application.models.user.RegistrationUser;
import net.darkeneez.banomo.application.util.CropImage;
import net.darkeneez.banomo.ui.base.BaseActivity;
import net.darkeneez.banomo.ui.location_select.LocationSelectActivity;
import net.darkeneez.banomo.ui.login.phone_input.PhoneInput;
```

Продолжение приложения Б

```
import org.greenrobot.eventbus.EventBus;
```

```
import org.greenrobot.eventbus.Subscribe;

import javax.annotation.Nullable;

import butterknife.BindView;
import butterknife.ButterKnife;
import butterknife.OnClick;
import de.hdodenhof.circleimageview.CircleImageView;

public class RegistrationActivity extends BaseActivity {

    private static final int GALLERY_CODE = 1;
    private static final int REQUEST_CROP_ICON = 2;

    @BindView(R.id.register_user_name)
    EditText mUserName;

    @BindView(R.id.register_user_family)
    EditText mUserFamily;

    @BindView(R.id.register_user_location)
    TextView mUserLocation;

    @BindView(R.id.register_user_avatar)
    CircleImageView mUserAvatar;

    private long mLocationId = -1;
    private String mName = null, mSecondName = null, mAvatar = null;

    @Override
    protected void onCreate(@Nullable Bundle saveInstanceState) {
        super.onCreate(saveInstanceState);
        setUnbinder(ButterKnife.bind(this));
```

Продолжение приложения Б

```
        EventBus.getDefault().register(this);
```

```
    setTitle("Заполните профиль");
}

@Subscribe
public void onLocationChanged(Location location) {
    mLocationId = location.getId();
    mUserLocation.setText(location.getName());
}


@Override
protected void onActivityResult(int requestCode, int resultCode, Intent data) {
    super.onActivityResult(requestCode, resultCode, data);
    if (requestCode == GALLERY_CODE) {
        if (resultCode == RESULT_OK) {
            Uri selectedImage = data.getData();

            CropImage cropImage = new CropImage(this);
            cropImage.setSource(selectedImage);

            try {
                startActivityForResult(cropImage.getIntentForCrop(), REQUEST_CROP_ICON);
            } catch (Exception ex) {
                mAvatar = cropImage.getBitmapPath();
                mUserAvatar.setImageBitmap(cropImage.getBitmap());
            }
        }
    } else if (requestCode == REQUEST_CROP_ICON) {
        if (data == null) {
            saveCropImageError();
            return;
        }
```

Продолжение приложения Б

```
        Bundle extras = data.getExtras();
```

```
        if (extras != null) {
            Bitmap photo = extras.getParcelable("data");

            CropImage cropImage = new CropImage(this);
            cropImage.setAfterCrop(photo);

            mAvatar = cropImage.getBitmapPath();
            mUserAvatar.setImageBitmap(cropImage.getBitmap());
        }
    }
}

private boolean isCanContinue() {

    mName = mUserName.getText().toString();
    mSecondName = mUserFamily.getText().toString();

    if (mName == null || mName.isEmpty()) {
        ShowSnackBar("Укажите имя", R.color.colorRed);
        return false;
    }

    if (mSecondName == null || mSecondName.isEmpty()) {
        ShowSnackBar("Укажите фамилию", R.color.colorRed);
        return false;
    }

    if (mLocationId == -1) {
        ShowSnackBar("Укажите домашний регион", R.color.colorRed);
        return false;
    }

    return true;
```

Продолжение приложения Б

```
}
```

```
    @OnClick(R.id.register_user_location)
    protected void onRegionClicked() {
        startActivity(new Intent(this, LocationSelectActivity.class));
    }

    @Override
    protected int getLayoutContentViewID() {
        return R.layout.activity_registration;
    }

    @OnClick(R.id.register_continue_button)
    protected void onContinueClicked() {
        if (isCanContinue()) {
            startActivity(new Intent(this, PhoneInput.class)
                    .putExtra("new_user", true)
                    .putExtra("registration_user", new RegistrationUser(mLocationId, mName,
mSecondName, mAvatar)));
        }
    }

    @OnClick(R.id.register_user_avatar)
    protected void onUserAvatarClicked() {
        askStoragePermissions();
    }

    @Override
    protected void onDestroy() {
        super.onDestroy();
        EventBus.getDefault().unregister(this);
    }

    private void askStoragePermissions() {
        if (Build.VERSION.SDK_INT <= Build.VERSION_CODES.JELLY_BEAN)
```

Продолжение приложения Б

```
            return;
```

```
        if (ContextCompat.checkSelfPermission(this, Manifest.permission_group.STORAGE) !=
PackageManager.PERMISSION_GRANTED) {

            if (ActivityCompat.shouldShowRequestPermissionRationale(this,
Manifest.permission_group.STORAGE))

                ActivityCompat.requestPermissions(this, new
String[]{Manifest.permission.WRITE_EXTERNAL_STORAGE,
Manifest.permission.READ_EXTERNAL_STORAGE}, 1);

            else

                ActivityCompat.requestPermissions(this, new
String[]{Manifest.permission.WRITE_EXTERNAL_STORAGE,
Manifest.permission.READ_EXTERNAL_STORAGE}, 1);

        } else

            getImage();

    }


    private void getImage() {

        startActivityForResult(new Intent(Intent.ACTION_PICK).setType("image/*"),
GALLERY_CODE);

    }


    private void permissionDeniedDialog() {

        AlertDialog.Builder builder = new AlertDialog.Builder(this);


        builder.setMessage("Разрешите доступ к памяти устройства, чтобы сохранять и загружать
фото и файлы");


        builder.setNegativeButton("Отмена", (dialogInterface, i) ->
dialogInterface.cancel());

        builder.setPositiveButton("Разрешить", ((dialogInterface, i) -> {

            dialogInterface.cancel();

            askStoragePermissions();

        }));


        builder.create().show();

    }
```

Окончание приложения Б

```
    @Override
```

```
    public void onRequestPermissionsResult(int requestCode, @NonNull String[] permissions,
@NonNull int[] grantResults) {
        if (grantResults.length <= 0 || grantResults[0] != PackageManager.PERMISSION_GRANTED)
            permissionDeniedDialog();
        else
            getImage();
    }


    private void saveCropImageError() {
        //TO-DO text
    }
}
```