Министерство образования и науки Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего образования
АМУРСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
(ФГБОУ ВО «АмГУ»)

Факультет математики и информатики
Кафедра информационных и управляющих систем
Направление подготовки 09.03.02 – Информационные системы и технологии
Направленность (профиль) образовательной программы: Безопасность информационных систем

ДОПУСТИТЬ К ЗАЩИТЕ
Зав. кафедрой
_____ А.В. Бушманов
«_____»_____201_ г.

БАКАЛАВРСКАЯ РАБОТА

на тему: Разработка Android приложения мессенджер

Исполнитель
студент группы 455об _____ В.А. Саяпин
(подпись, дата)

Руководитель
доцент, канд. техн. наук _____ А.В. Бушманов
(подпись, дата)

Консультант
по части безопасности
и экологичности
доцент, канд. техн. наук _____ А.Б. Булгаков
(подпись, дата)

Нормоконтроль
инженер кафедры _____ В.В. Романико
(подпись, дата)

Благовещенск 2018

Министерство образования и науки Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего образования
АМУРСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
(ФГБОУ ВО «АмГУ»)

Факультет математики и информатики
Кафедра информационных и управляющих систем

УТВЕРЖДАЮ
Зав.кафедрой
_____А.В.Бушманов
«_____»_____2018 г.

З А Д А Н И Е

К бакалаврской работе студента Саяпина Владислава Алексеевича.

1. Тема бакалаврской работы: Разработка Android приложения мессенджер.

(утверждено приказом от 23.04.2018 № 914-уч)

2. Срок сдачи студентом законченной работы 21.06.2018 г.

3. Исходные данные к бакалаврской работе: отчет по преддипломной практике.

4. Содержание бакалаврской работы: анализ операционной системы Android, проектирование клиент-серверного приложения Android, разработка клиент-серверного приложения Android, исследование вопросов информационной безопасности приложения, рассмотрение аспектов безопасности жизнедеятельности.

5. Перечень материалов приложения: приложения.

6. Консультант по бакалаврской работе Булгаков Андрей Борисович – раздел рассмотрение аспектов безопасности и экологичности.

7. Дата выдачи задания 09.05.2018 г.

Руководитель бакалаврской работы Бушманов Александр Вениаминович, доцент, канд. техн. наук.

Задание принял к исполнению (дата): _____В.А. Саяпин

РЕФЕРАТ

Бакалаврская работа содержит 90 с., 36 рисунков, 15 таблиц, 8 приложений, 17 источников.

ПРОЕКТИРОВАНИЕ, РАЗРАБОТКА, БАЗА ДАННЫХ, ИНФОРМАЦИОННАЯ БЕЗОПАСНОСТЬ, ANDROID, БЕЗОПАСНОСТЬ ЖИЗНЕДЕЯТЕЛЬНОСТИ, МЕССЕНДЖЕР, ДАННЫЕ ПРИЛОЖЕНИЯ

Для данной бакалаврской работы объектом исследования была выбрана разработка Android приложения.

Целью работы является разработка приложения Android мгновенного обмена сообщениями «Открытки».

Работа выполнялась последовательно в соответствие со следующими этапами: изучение системы Android, проектирование клиент-серверного приложения, разработка приложения, рассмотрение вопросов информационной безопасности приложения, а также исследование аспектов безопасности жизнедеятельности.

Разработанное приложение предоставляет пользователям Android получить новый способ общения в интернете, отвечая всем современным стандартам приложений мгновенного обмена сообщений.

Приложение рассчитано для использования пользователей всех возрастов и возможностей.

СОДЕРЖАНИЕ

НОРМАТИВНЫЕ ССЫЛКИ

В настоящей бакалаврской работе использованы ссылки на следующие стандарты и нормативные документы:

ГОСТ Р 50739-95 Средства вычислительной техники. Защита от несанкционированного доступа к информации. Общие технические требования.

ГОСТ Р ИСО 6385-2007 Эргономика. Применение эргономических принципов при проектировании производственных систем

СП 12.13130.2009 Определение категорий помещений, зданий и наружных установок по взрывопожарной и пожарной опасности РД от 30.03.1992 СВТ защита от НСД. Показатели защищенности от НСД к информации

РД от 30.03.1992 АС защита от НСД. Классификация АС и требования по ЗИ

СанПиН 2.2.2/2.4.1340-03 Гигиенические требования к персональным электронно-вычислительным машинам и организации работы

Федеральный закон "Об отходах производства и потребления" от 24.06.1998 N 89-ФЗ

Федеральный закон "О внесении изменения в статью 1 Федерального закона "О минимальном размере оплаты труда" от 07.03.2018 N 41-ФЗ

ОБОЗНАЧЕНИЯ И СОКРАЩЕНИЯ

БД – база данных;

БЖД – безопасность жизнедеятельности;

ВКР – выпускная квалификационная работа;

НПА – нормативно-правовой акт;

НСД – несанкционированный доступ;

НФ – нормальная форма;

ОС – операционная система;

ПК – персональный компьютер;

СП – свод привил;

ПО – программное обеспечение;

СВТ – средство вычислительной техники;

СУБД – система управления базами данных;

ЧС – чрезвычайная ситуация;

XML – расширяемый язык разметки.

ВВЕДЕНИЕ

На сегодняшний день мобильные приложения приносят значительные революционные изменения в информационно-технологический мир. Практический каждый человек применяет их ежедневно для выполнения различных задач и действий, таких как: социальное общение, гео-маршрутизация, поиск информации и т.д. Под управлением ОС Android работают сотни миллионов мобильных устройств более в 190 странах по всему миру. Около 70% процентов всех пользователей смартфонов и около 75% всех мобильных устройств используют операционную систему Android. Она является самой устанавливаемой мобильной ОС и с каждым днем миллионы новых пользователей начинающие свое знакомство с ней ищут приложения, игры, и другой необходимый цифровой контент.

Это означает, что разработка Android приложений становится важным аспектом развития мобильных информационных технологий и разных вариантов использования возможностей и удобств мобильной электроники в целях упрощения решения определенных задач. ОС Android позволяет разрабатывать приложения простой и сложной структуры, оснащать их красивым плавно-анимированным дизайном, и одновременно мощным и продуктивным функционалом, который автоматический будет оптимизироваться под конкретное устройство: смартфон, планшет, умные часы и т.д.

Также ОС Android дает простую модель распространения новых приложений среди различных устройств - Google Play. Google Play – это открытая интернет площадка для распространения Android приложений. Каждый месяц с помощью нее совершают более 1.5 миллиарда скачиваний. По желанию разработчика площадка может публиковать любую информацию о приложении всем пользователям, либо конкретной группе пользователей, отбирая их по специфичным сегментам, таким как тип устройства, версия ОС Android, технические возможности устройства. Более того помимо свободного бесплатного распространения приложения, существует возможность его мо-

нетизации и осуществления продаж. Также есть возможность совершения продажи внутренних продуктов самого приложения. Google Play позволяет полностью контролировать все денежные транзакции, которые могут осуществляться в любое время и с помощью любой валюты.

На данных момент с большим количеством пользователей и постоянно растущим спросом ОС Android является не только платформой для поддержания развития и улучшения работы уже полностью сформированного и готового бизнеса. Она также стала площадкой для «молодых» стартап-проектов, которые вполне могут заработать внимание и интерес публики, наработать базу пользователей и стать успешной функционирующей организацией в своей области, например, интернет торговля, социальное общение, поиск информации, заказ услуг и т.д. В этом им также будет помогать сама площадка Google Play, которая с ростом популярности приложения будет поднимать его выше в списке топ приложений и давать лучшие слоты расположения в определенных категориях для дальнейшего продвижения.

В ходе данной работы предлагается разработка Android приложения мессенджера, которое позволит пользователю мгновенно обмениваться сообщениями. Приложение не предполагает получение прибили на прямую, так как является бесплатным, но, в долгосрочной перспективе, подразумевается приток пользователей, а, следовательно, прибыли, покрывающее расходы на создание и обслуживание.

Соответственно целью данной выпускной квалификационной работы является создание Android приложения - мессенджер «Открытки».

Данное название исходит из основной концепции приложения, она представляет уникальную форму общения, основанную не на обмене текстовой информации, а графической, то есть пользователи могут рисовать друг другу открытки, и в реальном времени видеть все изменения, производимые друг другом на холсте открытки, в процессе рисования. Каких либо аналогов подобного рода приложений нет. Существует множество основных концепций современных мессенджеров, ряд их которых будет реализован в прило-

жении, но идея взаимного рисования, в форме мессенджера, на Android рынке примеров не имеет, что придает актуальность и новизну решения поставляемых задач этой бакалаврской работы.

Идея разрабатываемого мессенджера предоставить широкому кругу пользователю новый быстрый способ общения в интернете.

Для создания мессенджера были определены следующие задачи:

- изучение особенностей системы Android;
- обоснование выбора используемых технологий для разработки;
- проектирование серверной части приложения;
- проектирование клиентской части приложения;
- проектирование БД;
- разработка клиент-серверного приложения;
- рассмотрение аспектов ИБ приложения;
- рассмотрение аспектов БЖД приложения.

1 ОПИСАНИЕ СИСТЕМЫ ANDROID

Android – это мобильная операционная система, разрабатываемая компанией Google, на основе модифицированного версии ядра Linux и другого открытого программного обеспечения, предназнбаченная преимущественно для управления устройств с сенсорным экраном: смартфоны и планшеты. Так же компания Google разрабатывает версии ОС Android для телевизоров Android TV, для автомобилей Android Auto, для умных часов Wear OS. Каждая из этих версий операционных систем обладает специализированным пользовательским интерфейсом и рядом особенностей, заточенных на возможности устройства. Существуют варианты Android системы для игровых консолей, цифровых камер, персональных компьютеров и другой электроники.

Становление системы Android начинается с 2005 года, когда Google покупает стартап-компанию Android Inc., сделав ее полностью дочерней. В 2007 году система была впервые показана публике, а в сентябре 2008 впервые запущена на коммерческом Android устройстве. С тех пор Android ОС претерпела огромное количество релизов и обновлений (таблица 1), последние из которых было сделано в декабре 2017 года нумеруемое 8.1 «Oreo» (рисунок 1). Исходный код системы Android Open Source Project(AOSP) полностью находится под лицензией Apache, которая поддерживает сохранение авторских прав и отказ об ответственности. Позволяет пользователям использовать программное обеспечение в любых целях, распространять модифицированные версии ПО под статусом лицензии без каких-либо денежных отчисление правообладателю.

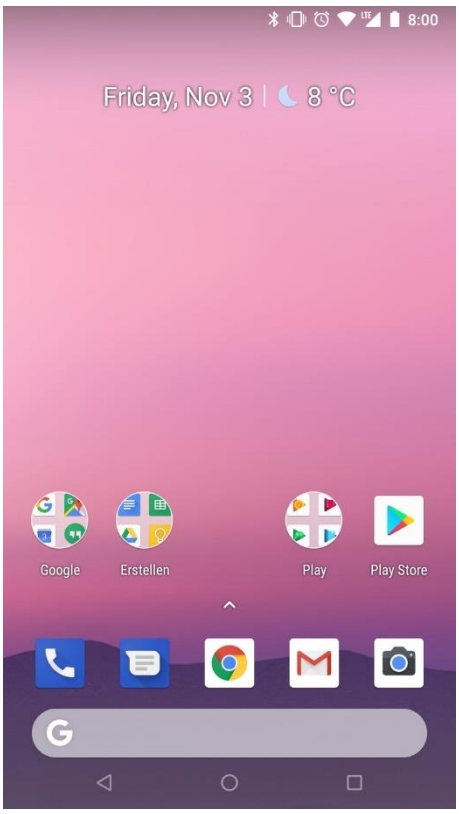


Рисунок 1 – Домашний экран Android Oreo 8.1

Таблица 1 – Распределение различных установленных версий Android ОС среди мобильных устройств по данным Google

Версия	Название	Дата релиза	Доля распределения
1	2	3	4
8.1	Oreo	5 декабря 2017	0.5%
8.0	Oreo	21 августа 2017	4.1%
7.1	Nougat	4 октября 2016	7.8%
7.0	Nougat	22 августа 2016	23%
6.0	Marshmallow	5 октября 2015	26%
5.1	Lollipop	9 марта 2015	18%
5.0	Lollipop	3 ноября 2014	4.9%
4.4	Kitkat	31 октября 2013	10.5%
4.3	Jelly Bean	24 июля 2013	0.6%
4.2	Jelly Bean	13 ноября 2012	2.2%
4.1	Jelly Bean	9 июля 2012	1.7%
4.0	Ice Cream Sandwich	19 октября 2011	0.4%

Продолжение таблицы 1

1	2	3	4
2.3	Gingerbread	9 февраля 2011	0.4%

Android система поставляется со встроенным программным обеспечением и сервисами компании Google, таких как почтовый клиент Gmail, поиск Google Search, магазином приложений и цифровым дистрибьютором Google Play и многие другие. Приложения полностью лицензированы производителями Android устройств и сертифицированы стандартами Google.

Android ОС является системой с самым большим пользовательским спросом в мире. В соответствии с данными на май 2017 года под ее управлением работает более 2 миллиардов активных мобильных устройств (рисунок 2), а Google Play насчитывает более 3,5 миллионов приложений.

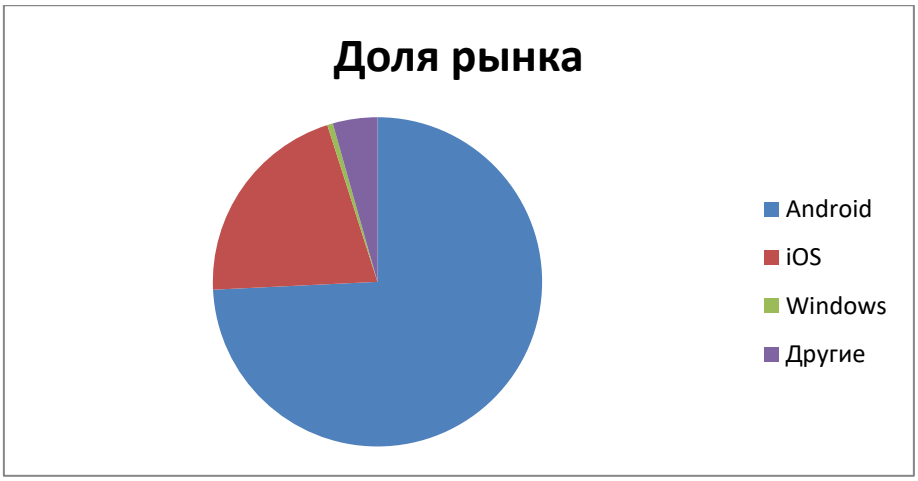


Рисунок 2 – Распределение операционных систем на мобильных устройствах на март 2018 года по данным gs.statcounter.com.

1.1 Особенности системы Android

Android обладает рядом преимуществ и достоинств в пользовательском интерфейсе, управлением приложениями и данными, поддержкой различного программного обеспечения.

Пользовательский интерфейс по умолчание полностью основан на применение прямых манипуляций с сенсорным экраном, используя жесты, соответствующие к похожим реальным действиям, таких как пролистывание,

нажатие, щелчок, обратное нажатие, а также использование виртуальной клавиатуры. Более того различные игровые контроллеры и полноразмерные физические клавиатуры поддерживаются системой с помощью Bluetooth или USB соединения. Ответом на манипуляции пользователя является экранный отклик программного интерфейса и использования вибрация устройства. Встроенные в мобильное устройство гироскоп, акселерометр, датчики приближения могут применяться различными приложениями для дополнительных взаимодействий с пользователем, например, смена ориентации экрана с горизонтальной на вертикальную и наоборот, симуляция управления транспортного средства вращением устройства, как рулевого колеса.

Первое что показывает Android устройство после включения — это домашний экран (home screen), он является первостепенной навигацией и информационным «портом» системы, некий аналог рабочего стола для персональных компьютеров. Домашний экран содержит иконки приложения и виджеты. Иконки запускают соответствующие приложения, виджеты отображают автоматически обновляемую информацию в реальном времени, например, погода, почтовые письма, новостная свода и т.д. Домашний экран может состоять из нескольких экранов, между которыми пользователь может переключаться путем пролистывания назад и вперед. Третья часть приложения Google Play и других магазинов могут кардинально изменять вид домашнего экрана, и даже делать их схожими с другими операционными системами, как Windows Phone и другие. Большинство производителей мобильных устройств изменяют внешний вид Android ОС для различия с конкурентами.

Вверху экрана мобильного устройства располагается статус бар (status bar), он предназначит для вывода информации о состоянии устройства и его подключениях. Статус бар может быть развернут вниз пользователем для показа релевантной информации о приложениях через уведомления. Уведомления – это короткие временные данные и перечень возможных действий выводимые приложением в статус бар для оповещения пользователя об изменение состояния данного приложения или системы в общем.

С домашнего экрана можно перейти в экран списка установленных приложений, откуда возможно запустить приложение или переместить ее иконку на домашний экран.

Все приложения, расширяющие функционал девайса, написаны используя набор средств для разработки Android Software Development Kit (SDK) на языке Java, который может быть скомбинирован вместе с C/C++. Также есть поддержка языков Go и Kotlin programming language, но функционал Application Programming Interfaces (API) для них ограничен и отстает от API Java.

SDK включает в себя комплексный набор инструментов разработчика: отладчик, библиотеки, эмулятор мобильных устройств, примеры кодов, обучающие уроки. На данный момент Google поддерживает интегрированную среду разработки (IDE) Android Studio.

Android приложения могут быть скачаны и установлены с помощью файлов Android Application Package (APK), а также через магазины приложений. Одним их них является встроенный магазин Google Play Store, он распространяет приложения совместимыми с требованиями лицензии Google Mobile Services Software. Магазин Play Store дает возможность пользователям искать, устанавливать, обновлять приложения. Также существуют альтернативные магазины, как Amazon Appstore, GetJar, SlideMe, F-Droid и другие, в которых могут содержаться приложения, которые не прошли проверку для публикации в Google Play Store или в связи с нарушением политики лицензирования или по другим причинам.

Поскольку Android устройства работают от питания аккумулятора, то управление процессами спроектировано для поддержания минимального расхода заряда. Если приложение не используется, то система может приостановить его выполнение, но не закрывая, а оставляя доступным для быстрого восстановления. Управление памятью происходит автоматически, если объем свободной памяти низкий, Android начнет скрыто закрывать неактив-

ные процессы, начиная с того, что дольше всех находится не в активном состоянии.

Основные возможности и особенности Android ОС:

- поддержка SMS и MMS сообщений, включающий потоковую отправку (C2DM) и Android Google Cloud Messaging (GCM);

- авто корректировка правописания слов и введение словаря, в который автоматически добавляются слова, используемые пользователем. В дальнейшем они будут предлагаться при письме в качестве смыслового продолжения предложения или словосочетания. Слова можно изменять и удалять;

встроенный веб браузер Google Chrome работающий на движке Chrome's V8 JavaScript;

- голосовое управление поиском Google, также совершение звонков, отправки сообщений, навигации;

- множественные касания экрана (multi-touch);

- многозадачность;

- захват экрана, возможность сделать снимок или запись видео с рабочего экрана устройства;

- захват и перевоспроизведение видео с Android TV;

- видео звонки через Google Talk, Skype, Google+;

- мультиязычность;

- встроенный голосовой помощник для людей с ограниченными возможностями зрения;

- поддержка следующих технологий связи: GSM/EDGE, Bluetooth, LTE, CDMA, EV-DO, UMTS, NFC, IDEN and WiMAX;

- Bluetooth поддерживает голосовые звонки, отправку файлов (OPP), контактов между телефонами, доступ к телефонной книге (PBAP), подключение дополнительной мышки, джойстика, клавиатуры и других устройств;

- возможность создания точки доступа Wi-Fi hotspot;

- поддержка потоковых технологий загрузки: RTP/RTSP streaming 3GPP PSS, ISMA, HTML5, Adobe Flash Streaming (RTMP) , HTTP Dynamic Streaming с помощью Flash Plagin, Apple HTTP Live Streaming;

- поддержка огромного количества аудио и видео медиа форматов файлов: WebM, H.263, H.264, AAC, HE-AAC (3GP или MP4), MPEG-4 SP, AMR, AMR-WB (3GP), MP3, MIDI, Ogg Vorbis, FLAC, WAV, JPEG, PNG, GIF, BMP, and WebP;

- поддержка устройств внешнего хранилища карт microSD следующих файловых систем FAT32, Ext3, Ext4, NTFS, HFS Plus, exFAT, а также возможность использования устройств через USB;

- поддержка камер, сенсорных экранов, GPS, акселерометров, гироскопов, барометров, магнитометров, игровых контроллеров, датчиков близости и давления, термометров, 2D и 3D графические ускорители;

- поддержка Java и использование Android Runtime вместо Java Virtual Machine;

- поддержка различных расширений экранов, вертикальная и горизонтальная ориентация, а также возможность подключения внешнего экрана через HDMI или Miracast, 2D и 3D графические библиотеки основанные на OpenGL ES 2.0;

- поддержка легкой базы данных SQLite;

- написание приложений в формате HTML;

- автоматическая открытие приложения вместо загрузки определенного сайта, если таковое приложение установлено на устройстве.

1.2 Файловая структура Android приложения

Любой проект Android приложения состоит из одного или нескольких модулей. Модуль - это определённая структурированная коллекция исходных файлов приложения, которая позволяет логически разделять функционал приложения друг от друга. Существуют несколько предопределенных модулей, таких как Phone and Tablet Module, Wear OS Module, Android TV Module,

Glass Module, также есть возможность создания новых модулей, для определения своих библиотек и структуры файлов кода.

Любой модуль Android приложения поддерживает общую схему расположения файлов по директориям (рисунок 3):

- manifests, содержит файл AndroidManifest.xml, который описывает существенную информацию о приложение для инструментов сборки Android приложений, Android ОС и Google Play. Он требует содержания названия пакета приложения (package name), перечень компонентов приложения (activities, services, broadcast receivers, and content providers), список запрашиваемых разрешения для приложения (permissions), требования к программной и технической составляющей устройства;

- java, директория содержит файлы исходного кода Java, разделенные по пакетам и включающие тестовый код JUnit;

- res, содержит все исходные файлы, являющиеся не кодом, такие как XML, шаблоны, UI строки, bitmap изображения, распределённое по соответствующим поддиректориям (таблица 2).

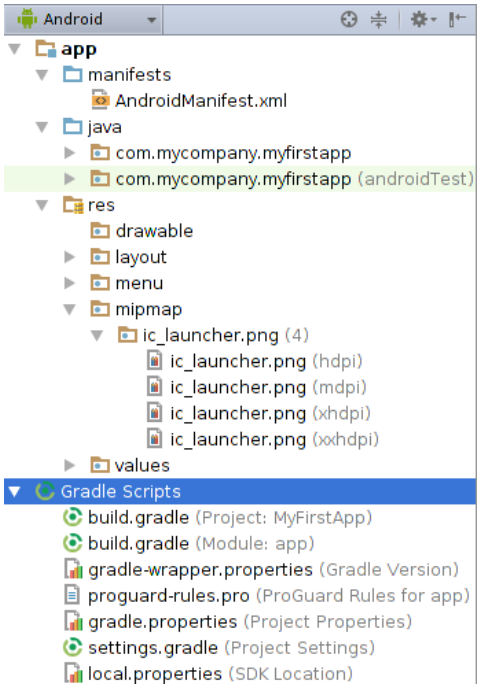


Рисунок 3 – Расположение файлов по директориям в Android Studio

Таблица 2 – Назначение поддиректорий res/

Название директории	Назначение
animator/	XML файлы определяющие свойства анимации
anim/	XML файлы определяющие виды анимации
color/	XML файлы определяющие используемые цвета в приложение
drawable/	Bitmap файлы (.png,.9.png, .jpg, .gif) или XML определяющие различные типы графики
mipmap/	Файлы иконок различных разрешений запуска приложения
layout/	XML файлы шаблонов интерфейса
menu/	XML файлы шаблонов меню и подменю приложения
raw/	Произвольные файлы для хранения в исходном виде
values/	XML файлы содержащие примитивы, такие как строки, числа, цвета
xml/	Произвольные XML файлы в исходном виде
font/	Файлы шрифтов .ttf, .otf,. ttc или XML, включающий элемент <font-family>

Нельзя не упомянуть, что существует возможность создания альтернативных версий некоторых перечисленных в таблице 2 директорий, которые будут подключаться для определенных подходящих разрешений экранов и языковых настроек Android ОС.

1.3 Схема работы жизненного цикла Android приложения

Android приложение включает в себя несколько составляющих:

- Activity – окно, класс пользовательского интерфейса и выполнения задач вывода информации;

- Service – подпрограмма-демон, класс для выполнения задач в фоне без интерфейса.

В то время как пользователь переключается между разными окнами Activity, все Activity проходят через разные состояния своего жизненного цикла. Класс Activity обеспечивает набор вызовов позволяющих отследить в каком состоянии находится Activity: остановлено, только создано, возобновлено, уничтожено. С помощью этих вызовов есть возможность задания своего поведения для Activity, когда пользователь покидает и возвращается в Activity.

Для навигации между состояниями жизненного цикла Activity ядро Android ОС содержать 6 callback методов: onCreate(), onStart(), onResume(), onPause(), onStop(), onDestroy(); которые будут мгновенно вызваться после перехода Activity в новое состояние (рисунок 4).

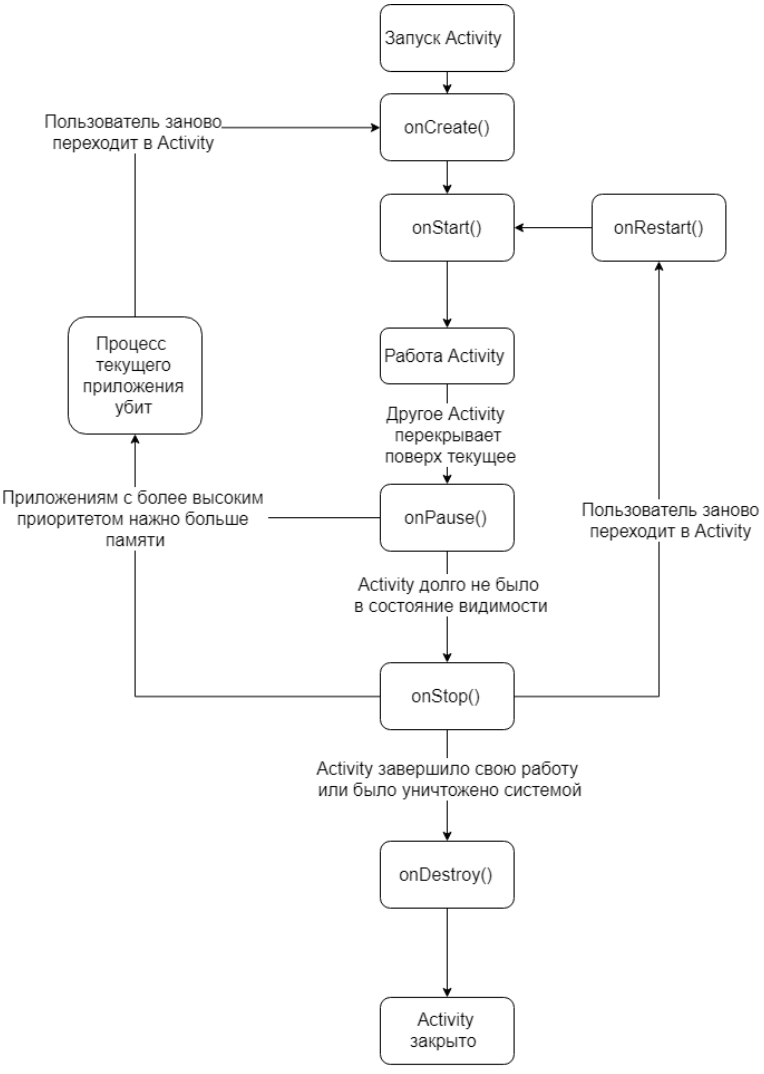


Рисунок 4 – Жизненный цикл Activity

Метод onCreate() вызывается при наступление стадии Created. Вызов происходит один раз только при первом создание Activity. Обычно в нем определяются все логические составляющие и компоненты приложения, для работы с ними в последующих методах (пользовательский интерфейс, переменные и т.д.).

Метод onStart() вызывается при наступлении стадии Started. Метод делает Activity видимым для пользователя, происходит очень быстро и сразу переходит к стадии Resumed.

Метод onResume() вызывается при наступление стадии Resumed. Это состояние взаимодействия приложения с пользователем. Приложение будет находится в этом состоянии пока фокус не перейдет на другое Activity или выключится экран. Метод будет вызываться каждый раз при развертывании Activity.

Метод onPaused() вызывается при наступлении стадии Paused. Это первый признак того, что пользователь покинул Activity. Он часто используется, для перехода работы Activity в фоновый режим. Вызов метода происходит довольно быстро поэтому используется для выполнения тяжелых задач, таких как сохранение состояния приложения или данных пользователя. Это состояние будет поддерживаться, пока Activity окончательно не станет невидимым пользователю.

Метод onStop() вызывается при наступлении стадии Stopped. Метод выполняется, когда Activity находится долгое время в фоне. В этом методе выполняются все тяжелые и нагруженные операции приложения при его закрытии.

Метод onDestroy() вызывается до того как Activity полностью уничтожится. Система выполняет этот метод при вызове метода finish () в коде приложения.

1.4 Способы хранения данных в Android приложениях

Операционная система Android поддерживает несколько различных возможностей сохранения данных. Выбор того или иного типа хранилища

зависит от объема пространства, которое необходимо сохраняемым данным, а также их типа, настроек доступа, то есть требуется ли доступ к ним из других приложений или же они должны быть закрытыми(private).

Существует несколько возможных видов хранилища:

- внутренние хранилище, содержит файлы с закрытым доступом в системном разделе;

- внешнее хранилище, содержит файлы доступ к которым могут получить различные пользователи и приложения;

- Shared Preferences, хранилище примитивных данных с закрытыми доступом вида ключ-значение;

- база данных, хранилище структурированных данных с закрытым закрытом доступом.

1.4.1 Shared Preferences

Если есть необходимость хранения большого количества неструктурированных данных, то можно воспользоваться Shared Preferences. Shared Preferences API позволят считывать и записывать постоянные пары ключ-значения примитивных данных, как booleans, floats, ints, longs, strings.

Ключ-значения пары записываются в XML файлы постоянного хранения, если даже приложение убито. Shared Preferences часто используют для сохранения различных настроек и статусов приложения.

Чтобы создать новый shred preferences или получить новый доступ к ним необходимо вызвать один из методов, который вернет объект Shared Preferences:

- getSharedPreferences() – используется если нужно задать особое имя файлу;

- getPreferences() – используется для создания и получения доступа к одному файлу preferences по умолчанию для каждого Activity.

Чтобы считать информацию из файла preferences, используются методы геттеры полученного объекта Shared Preferences, например для извлечения строки getString(name, def), где name - ключ, значение которого хотят по-

лучить, a def – значение, которое будет возвращено, если значения по ключу не существует (пример:sharedPref.getInt(getString(R.string.key),5)).

Для записи данных используется объект SharedPreferences.Editor, который можно получить, вызвав метод edit () объекта SharedPreferences. Далее путем вызова методов объекта SharedPreferences.Editor putInt(ключ,значение), putString(ключ,значение) и других, возможно приготовить данные для записи и подтвердить их запись вызовом метода commit() (пример использования находится в Приложение А).

1.4.2 Внутренне и внешнее хранилище

По умолчанию файлы, сохраненные на внутреннем хранилище, имеют закрытый доступ для других приложений, но доступ могут получить пользователи, которые обладают root правами). Это делает внутренне хранилище хорошим местом для внутренних данных приложений, к которым пользователю не нужен прямой доступ. Система обеспечивает закрытое хранилище в файловой системе для каждого приложения, где возможно структурировать и организовывать свой порядок данных.

Когда пользователь удаляет приложение данные автоматически стираются из памяти. Из-за этого не следует хранить на внутреннем сохранять данные, которые должны храниться постоянно независимо от приложения.

Для временного, непостоянного хранения внутренних данных используется специальная кэш директория приложения. Каждое приложение имеет закрытую кэш директорию, специализированную под такой тип данных. Когда устройство страдает о нехватки памяти, Android ОС может удалить эти файлы для восстановления свободного пространства. Однако полагаться на систему не следует, а стоит отчищать кэш директории программно, следует постоянно поддерживать лимит кэш пространства, например 1 Мб. После удаления приложения данные кэша автоматически стираются из памяти.

Для управления данными на устройстве использует File API. Объект File хорошо работает с чтением и записью огромного количества данных. С помощью различных методов этого объекта можно получить доступ к чте-

нию и записи данных как на внутреннем, так и на внешнем хранилище (пример использования находится в Приложение Б).

Каждое устройство поддерживает внешнее хранилище, которое может представлено в виде SD карты или другого устройства. Файлы, сохраненные на внешнем хранилище, являются общедоступными и могут быть прочитаны или модифицированы пользователем при подключении через USB или передачи данных на компьютер.

Прежде чем пытаться получить доступ к внешнему хранилищу, приложение должно проверить его наличие, а также наличие желаемых файлов. Также обязательно нужно запросить и перечислить разрешения прав (permissions) на чтение и запись на внешнем хранилище (android.permission.WRITE_EXTE-RNAL_STORAGE, android.permission. READ_EXTER NAL_STORAGE) в файле манифеста.

Если пользователь удаляет приложение, то его данные на внешнем хранилище сохраняются, обычно это фотографии и скаченные файлы из интернета. Система поддерживает стандартные общие директории для этих типов данных, так что пользователь может разместить их в одной из них: photos, music, ringtons, и другие.

Данные на внешнем хранилище могут быть стерты системой после удаления приложения и быть доступны только ему, если они были сохранены в специальных частных директориях приложения. Но таким файлам не гарантируется защищенность, так как есть возможность изъятия внешнего хранилища и прочтения его другим устройством.

В таблице 3 показаны основные различи между использованием внутреннего и внешнего хранилища.

Таблица 3 – Различия внутреннего и внешнего хранилища

Внутренние хранилище	Внешнее хранилище
1	2
Всегда доступно.	Не всегда доступно, так как пользователь может отключать такие хранилища

Продолжение таблицы 3

1	2
Доступ к файлам имеет только соответствующие приложение.	Доступ к файлам может быть осуществлен без ведома пользователя.
При удалении приложения удаляются все данные.	При удалении приложения удаляются данные только из специальной закрытой директории.

Как показано в таблице 3, внутренние хранилище является лучшим местом для сохранения информацией с закрытым доступом. Внешнее хранилище больше подходит для общего доступа к файлам всеми приложениями и просмотра их через другие устройства.

1.4.3 SQLite

Android ОС обеспечивает полную поддержку SQLite. Любая созданная база данных доступна только соответствующему приложению. SQLite идеально подходит для сохранения структурированных данных, такие как контакты и другие. Система поддерживает пакет API для работы с SQLite android.database.sqlite package. SQLite это встроенная библиотека реализующая автономный, без серверный, сконфигурированный, транзакционный SQL движок. Код SQLite находится в общей доступности и может использоваться в любых целях. SQLite хранит свои данные на специальном выделенном мете прямо диске. SQLite, как и множество других SQL баз данных включает в себя множественные таблицы, индексы, триггеры, представления, хранящиеся на одном файле. Файл имеет кроссплатформенный форма и может использоваться на 32 битных и 64-итных системах. Взаимодействие Android приложения и SQLite происходит посредством объектов класса SQLiteOpenHelper и SQLiteDatabase. С помощью SQLiteOpenHelper возможно определить общую схему и структуру БД, описать индексы, внешние ключи и т.д. С помощью SQLiteDatabase можно взаимодействовать с описанной БД: извлекать, удалять, добавлять, обновлять таблицы и строки данных (пример использования находится в Приложение В).

Основные особенности SQLite:

- таблицы, индексы, триггеры, представления в неограниченном количестве;
- до 32000 столбцов и неограниченного количества строк в таблицах;
- индексы для нескольких столбцов;
- CHECK, UNIQUE, NOT NULL, и FOREIGN KEY конструкции;
- транзакции BEGIN, COMMIT, и ROLLBACK конструкции;
- подзапросы;
- полнотекстовый поиск;
- поддержка JSON;
- продвинутый планировщик запросов;
- UPDATE, DELETE, INSERT, SELECT конструкции;
- исключения;
- конструкции JOIN;
- DISTINCT, ORDER BY, GROUP BY, HAVING, LIMIT, и OFFSET;
- UNION, UNION ALL, INTERSECT, и EXCEPT;
- функции.

1.5 Разработка требований

Для выполнения данной ВКР необходимо учитывать современные требования пользователей к скорости, удобству, безопасности мессенджеров, в соответствии с которыми можно выделить основные следующие:

- беспарольная авторизация и регистрация по номеру телефону, с доставкой кода подтверждения через SMS;
- удобный, интуитивно понятный интерфейс с плавной и быстрой анимацией, поддерживающий различные разрешения экранов устройств;
- поддержка нескольких распространённых версий Android;
- проверка вводимых пользователем данных и предотвращение непреднамеренных и злонамеренных действий, с помощью валидаций форм;
- удаленное хранение истории не полученных сообщений с возможностью последующей подзагрузкой;

- возможность отправки открыток всем пользователям, имеющиеся в контактной книге устройства;

- при возникновении сбоев и ошибок, предоставлять пользователю и администратору информационные сообщения с возможными способами решения проблемы;

- сбор информации о работе самого приложения, ведение логов;

- передача всех данных по защищенному каналу связи;

- мгновенная отправка и получение сообщений;

- мультиязычность;

- хранение открыток и истории сообщений на устройстве.

Постановка задачи четко определяет последовательность и направленность действий по дальнейшему проектированию и разработки клиент-серверного приложения.

На этом завершается раздел описания системы Android. Требования к приложению и особенности функционирования системы Android, произведенные на данном этапе, позволяют сформировать видение предметной области, определить основные аспекты проектирования будущего приложения.

2 ПРОЕКТИРОВАНИЕ КЛИЕНТ-СЕРВЕРНОГО ПРИЛОЖЕНИЯ

Мессенджер – это программа для обмена сообщениями в реальном времени через Интернет, состоящая из двух частей: клиент и сервер.

Проектирование клиент-серверного приложения является наиважнейшем этапом, так как от объема приложенных усилий и уровня проработки всех действующих компонентов данного аспекта напрямую зависит успешность и работоспособность будущего приложения. На рисунке 5 представлена общая архитектура клиент-серверного мобильного приложения.

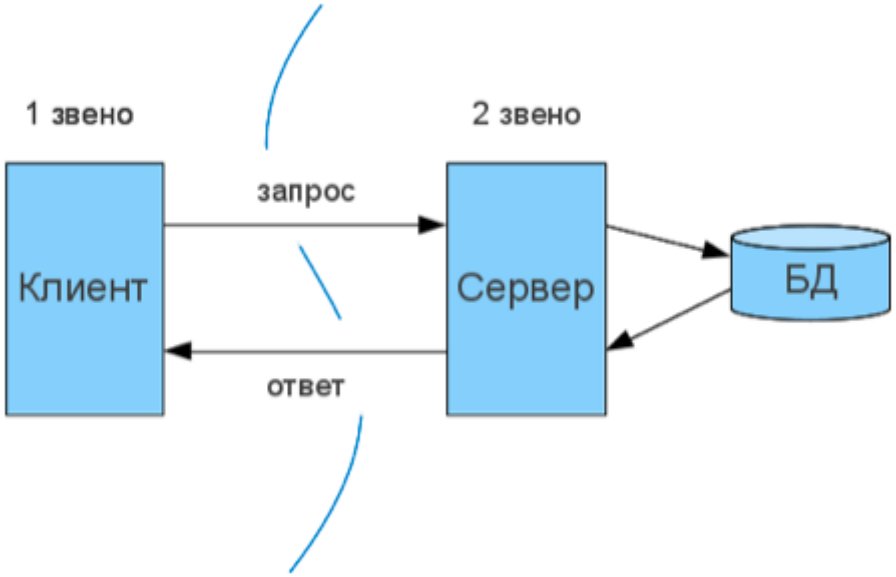


Рисунок 5 – Общая архитектура клиент-серверного приложения

2.1 Анализ требований к системе

Непосредственно перед проектированием необходимо произвести анализ требований, определяющих характеристики приложения, подобрать подходящие решения для выполнения всех условий, ставящихся перед процессом разработки.

Первым требованием является беспарольная авторизация и регистрация по номеру телефону. То есть для идентификации, пользователю необходимо

предоставить только свой номер телефона, куда ему для прохождения авторизации будет выслан SMS код подтверждения.

Согласно [2, с. 315], Идентификация – это процедура распознавания пользователя по его идентификатору, присвоенному данному пользователю ранее и занесенному в базу данных в момент его регистрации в качестве легального пользователя системы.

Аутентификация – процедура проверки подлинности входящего в систему объекта (пользователя, процесса или устройства), предъявившего свой идентификатор.

Авторизация – процедура предоставления пользователю (процессу или устройству) определенных прав доступа к ресурсам системы после успешного прохождения им процедуры аутентификации, иными словами, авторизация устанавливает сферу действия пользователя и доступные ему ресурсы.

Для реализации этой возможности необходимо привлечь сторонний сервис рассылки SMS сообщений, с которому сервер будет предоставлять случайно сгенерированные числовые последовательности, используемые в качестве SMS кода подтверждения.

Одним из важнейших требований является поддержка нескольких распространенных версий Android, существенную роль здесь играет версия самая устаревшей поддерживаемая версия ОС, так как именно на ней будет вестись разработка приложения и она определяет уровень охвата аудитории устройств потенциальных пользователей. Также версия ОС, на которой будет разрабатываться приложение, напрямую влияет на возможности функционирования интерфейса и поддерживаемых библиотек и фреймворков. Согласно [1], фреймворк – программное обеспечение, облегчающее разработку и объединение разных компонентов большого программного проекта.

Для разработки мессенджера «Открытки», выберем минимальную версию Android 5.0, что соответствует SDK 21. Согласно [4] и таблице 1, на 7 мая 2018 года приложения для версии Android 5.0 Lollipop могут поддерживать 84.7% устройств работающих под управлением ОС Android.

Еще одним существенным требованием к приложению является мгновенное и быстрое получение сообщений. Важным аспектом здесь является выбор протокола обмена информацией.

2.2 Обоснование выбора протокола обмена WebSocket

Отличным решением для мгновенного обмена сообщениями является использование протокола WebSocket. [Согласно 11] , WebSocket — протокол полнодуплексной связи (может передавать и принимать одновременно) поверх TCP-соединения, предназначенный для обмена сообщениями между браузером и веб-сервером в режиме реального времени. В отличии от стратегии long-polling (длинные запросы), реализующиеся на протоколе HTTP, WebSocket поддерживает постоянно только одно соединение для обмена, также не пересылает огромное количество заголовков и куки при каждом запросе, а делает это только при подключении. Еще одной особенностью является то, что транзакции могут быть инициализироваться, как и клиентом, так и серверов, в то время как long-polling дает это делать только клиенту соединения. Эти преимущества позволяют WebSocket значительно уменьшить время задержки и нагрузку на сеть. WebSocket обменивается данными любого объема с помощью сообщений, являющимися структурирующим контейнером для пересылаемой информации участников транзакций.

2.3 Особенности представления пересылаемых данных

Так как основанная концепция приложения — это обмен рисунками в реальном времени, то возникает вопрос преобразования и фиксирования передаваемых данных из графического вида в текстовый. На рисунке 6 показан процесс преобразования графической информации в текстовую.

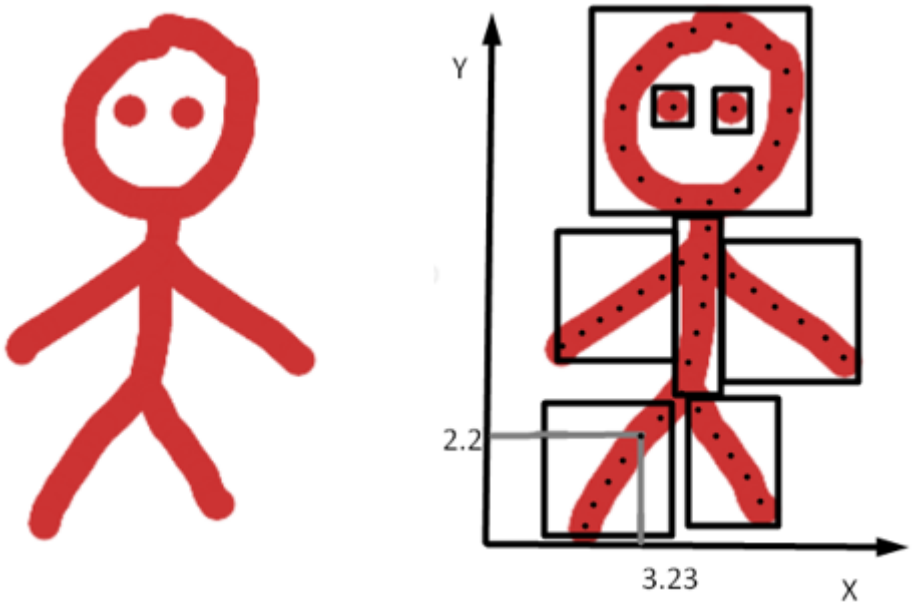


Рисунок 6 – Преобразование графической информации в текстовую

Как мы можем видеть на рисунке 6 вся композиция должна быть разделена на структурные части (пути), выделенные черными квадратными областями. Путем будем считать траекторию движения кисти, от момента касания пользователем экрана до момента отпускания. Пути должны фиксироваться по мере их поступления от пользователей, то есть, как только пользователь начинает рисовать. Каждый путь должен содержать координаты точек относительно окна приложения, по которым он рисуется, а также информацию о цвете кисти, толщине кисти, пользователе.

Чтобы обеспечить эффект взаимного рисования в реальном времени за каждое новое пересылаемое сообщение будем принимать информацию о поступающей точке. Также в сообщение должно входить информация о параметрах пути, которому эта точка принадлежит, чтобы при обратном преобразовании из текстовой информации в графическое изображение полностью соответствовало тому, что задумывал переслать автор.

2.4 Проектирование серверной части приложения

Правильность выбора компонентов и конфигурации сервера является главных аспектом функционирования приложения, влияющая на его быстроту работы. Серверная сторона должна полностью настроена под требующие-

ся задачи и ожидаемые нагрузки, а также быть готова к возможным перегрузкам и отказам системы.

2.4.1 Обоснование выбора двухуровневой конфигурации серверной части приложения

Наилучшим решением построения серверной части является двухуровневой конфигурации, на что есть ряд преимуществ использования дополнительного Front-end сервера в помощь к основному Back-end серверу.

Согласно [8], Front-End – публичная часть проекта, обеспечивающая прием запросов от пользователей, трансляцию запросов к Back-End и выдачу непосредственного содержимого пользователю. Back-End – исполнительная часть системы, которая обеспечивает выполнение PHP-скриптов, формирование контентных страниц и работу бизнес-логики приложений.

На рисунке 7 представлена общая структура двухуровневой конфигурации сервера.

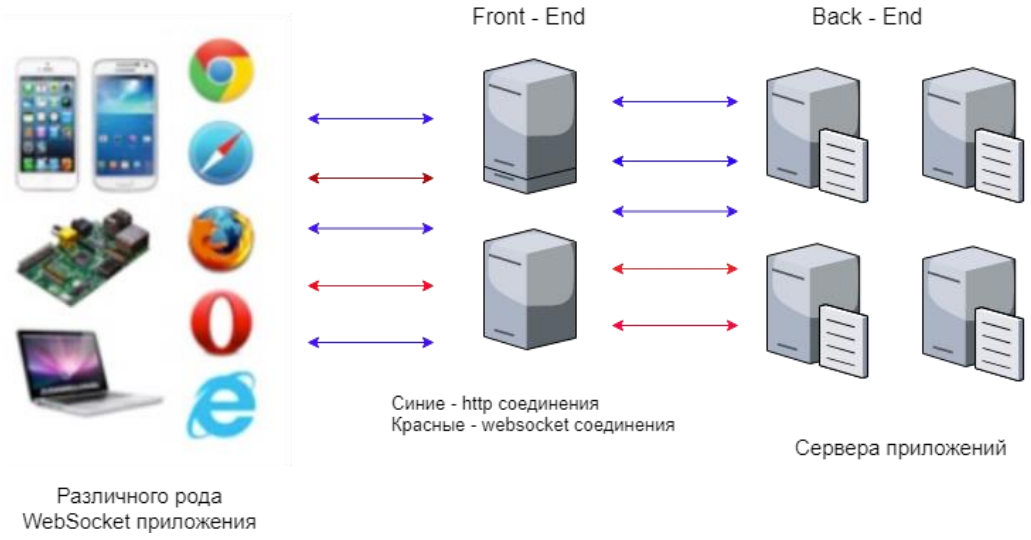


Рисунок 7 – Двухуровневая конфигурация сервер

Внедряя двухуровневую архитектуру, мы выставляем перед пользователем Front-end систему – не позволяющую общаться с сервером приложения на прямую. Front-end сервер исполняет все запросы от пользователя, какие может отработать самостоятельно, так же если одни и те же данные запраши-

ваются постоянно от Back-end сервера, то Front-end может их кешировать для уменьшения нагрузки на сервер приложений.

Основными преимуществами и выгодами от использования двухуровневой конфигурации сервера являются:

- минимизация числа запросов, поступающих к Back-end серверу;

- уменьшение нагрузки на Back-end сервере, во время передачи огромного количества данных. Back-end отдает эти данные Front-end серверу, высвобождая свои ресурсы, а тот уже отвечает за передачу их клиенту;

- использование кеширования часто запрашиваемых данных;

- защита Back-end сервера от большого числа запросов за счет механизма ожидания высвобождения ресурсов Back-end сервера Front-end сервером.

Как мы можем видеть Front-end сервер позволяет нам снять большую категорию рисков работы сервера приложения.

2.4.2 Обоснование выбора Node.js в качестве Back-End сервера

В первую очередь при реализации двухуровневой конфигурации сервера необходимо определить с Back-end частью, то есть сервером приложения, так как именно от него будет зависеть выбор других компонентов.

Лучшим решением для использования WebSocket – является сервер Node.js. Согласно [9], Node.js — программная платформа, основанная на движке V8 (транслирующем JavaScript в машинный код), превращающая JavaScript из узкоспециализированного языка в язык общего назначения. Node.js добавляет возможность JavaScript взаимодействовать с устройствами ввода-вывода через свой API (написанный на C++), подключать другие внешние библиотеки, написанные на разных языках, обеспечивая вызовы к ним из JavaScript-кода. Node.js применяется преимущественно на сервере, выполняя роль веб-сервера. Также Node.js отличное и легкое решение для работы с WebSocket, позволяющие поддерживать более 600000 одновременных подключений

2.4.3 Обоснование выбора Nginx в качестве Front-end сервера

Node.js имеет ряд проблем с безопасностью, которые, как показала практика использования, легко может решить Nginx. Сервер Nginx был создан для поддержания по меньшей мере 10000 одновременных подключений. Он использует асинхронную событийную архитектуру, схожую с Node.js, для управления этими подключениями. Он может быть использован, как веб сервер, обратны прокси сервер, распределитель нагрузки и HTTP кэш. И что для нас важное, так это то, что Nginx может быть использован как обратных прокси сервер для WebSocket в качестве Front-end сервера Node.js. Более того он позволяет распределять нагрузку обработки WebSocket соединений на сервера приложений, на давая одним простаивать, а других перегружаться.

Основные преимущества использования Nginx в связке с Node.js:

- упрощение настройки и обслуживания Node.js сервера;

- намного быстрее выполняется работа со статическими файлами;

- распределение нагрузки на сервера приложений;

- защита от DoS атак и гарантированный фильтрованный трафик на Node.js.

2.4.4 Обоснование выбора MongoDB в качестве серверной БД

MongoDB – одна из самых лучших неряляционных БД, хорошо зарекомендовавшееся в использование с Node.js. Согласно [5], MongoDB — документоориентированная система управления базами данных (СУБД) с открытым исходным кодом, не требующая описания схемы таблиц. Классифицирована как NoSQL, использует JSON-подобные документы, часто применяемые в JavaScript, и схему базы данных. Можно выделить ряд особенностей, влияющих на использование этой СУБД в работе с Node.js:

- отсутствие схемы и необходимости создания таблиц, миграций, определения типов данных;

- объектно-ориентированная структура хранения данных;

- богатая функция агрегации данных;

- нет необходимости в нормализации, данные принято хранить в виде, в котором их удобно использовать;

- простой формат использования индексов.

2.4.5 Обоснование выбора Twilio в качестве сервиса SMS рассылки

Для реализации возможности рассылки SMS сообщений необходимо использовать сторонние сервисы, предоставляющие эту услугу. Бесплатные сервисы не отвечают требованиям разрабатываемого приложения, а именно предоставления API для работы с сервисом и возможность регистрации одного конкретного номера отправителя, поэтому мы прибегнем к использованию платного сервиса. Одним из лидеров рынка коммуникационных сервисов является компания Twilio. Она предоставляет возможность рассылки SMS сообщений, видео и аудио звонков. Отправка и получение SMS по всему миру возможно с заранее зарегистрированного или на выбор свободного номера. Цена за 1 SMS зависит от страны получателя. Также существует возможность создания программируемых SMS, для разработки SMS ботов. Более того Twilio обеспечивает простое API для работы с сервисом с различных платформ, включая Node.js.

2.4.5 Проектирование архитектуры серверной части приложения

Теперь, когда определенны тип конфигурации и основные компоненты серверной части приложения, можно спроектировать его полную архитектуру. Как мы можем видеть на рисунке 8, общение клиента с сервером происходит только с помощью WebSocket. Nginx служит защитным барьером для главного сервера приложения Node.js, фильтрует трафик и предотвращает прямые атаки. Более того, в случае увеличение количества серверов приложения, для повышения скорости работы, Nginx распределяет нагрузку между ними. Node.js держит WebSocket соединения с Nginx, по которым отправляет требуемые данные клиентам, также осуществляет работу по хранению данных с СУБД MongoDB и связывается с внешним SMS сервисом для отправки случайно сгенерированных кодов подтверждения клиентам.

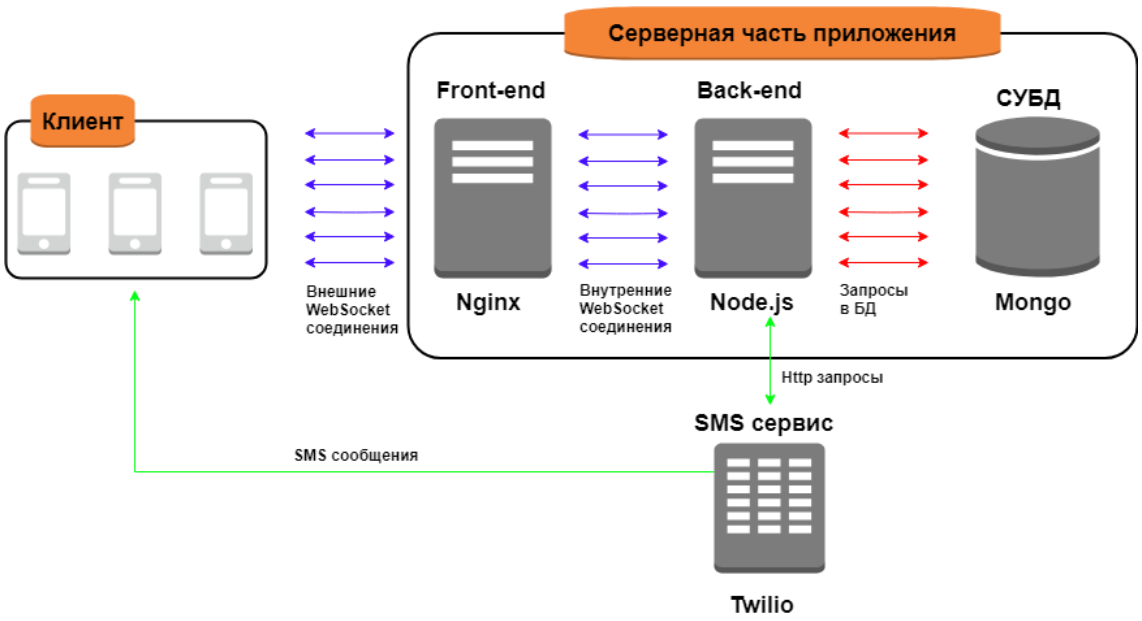


Рисунок 8 – Общая архитектура серверной части приложения

2.5 Проектирование клиентской части приложения

Клиентская часть играет самую весомую часть приложения, так как является его визитной карточкой, определяя внешний вид, способы взаимодействия с интерфейсом, и что самое главное отвечает за опыт и впечатления от использования приложения. Кроме интерактивности, цветовой схемы, общего дизайнерского решения, немаловажную роль играет адаптивность под различные разрешения экранов устройств.

Также для приложения Android существенную часть его оптимизация и производительности определяет грамотное использование многопоточности Java, то есть выделение тяжелых ресурсоемких задач в отдельные потоки или даже процессы, распараллеливание необходимых действий, создание специальных отдельных сервисов (Service) для работы с БД, интернет-соединением, внешним и внутренним хранилищем. Все это необходимо для снижения нагрузки на главный поток приложения (Main thread), в целях предотвращения его «заморозки» и последующей остановки системой.

2.5.1 Основные компоненты

На рисунке 9 показана общая архитектура клиентского приложения

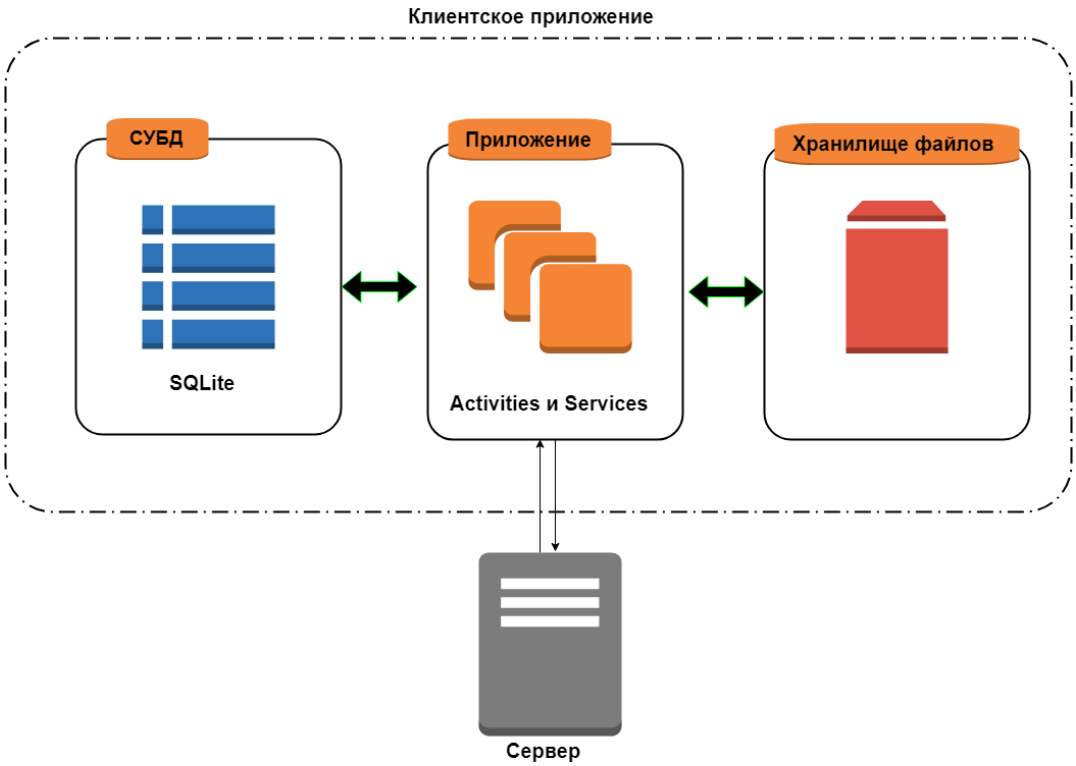


Рисунок 9 – Общая архитектура клиентского приложения Android

Из перечисленных выше требований к приложению и архитектуры выделяем следующие основные составляющие приложения:

1) HomeActivity – главное Activity являющееся начальной точкой работы с приложением для авторизованных пользователей. Должно содержать список отправленных открыток и доступ к их редактированию, возможность выхода из аккаунта, возможность добавления новых открыток, доступ к редактированию личной информации;

2) LoginActivity – Activity предназначенное для регистрации и авторизации, должно содержать формы ввода номера телефона и кода подтверждения;

3) PersonalActivity – Activity для редактирования личной информации пользователя;

4) ContactsActivity – Activity содержащие список контактов пользователя для добавления новой открытки;

5) PaintActivity – главное интерактивное Activity приложения осуществляющие процесс рисования открыток, обеспечивающееся удобным интерфейсом управления инструментами и процессом рисования;

6) NotificationService – Service для работы с интернетом, именно он держит WebSocket соединения. NotificationService необходимо держать в работе постоянно, для проверки и получения новых сообщений с сервером. Из-за этого его необходимо запускать при старте системы Android и в случае его остановки системой;

7) DBService – Service для работы с БД SQLite, берет на себя всю вычислительную работу на взаимодействия с БД. Держать его постоянно в работе не нужно и запускать только при необходимости записи или чтения БД SQLite.

2.5.2 Логика взаимодействия компонентов

На рисунке 10 представлена декомпозиция архитектуры рисунка 8, с указанием всех вышеперечисленных компонентов системы.

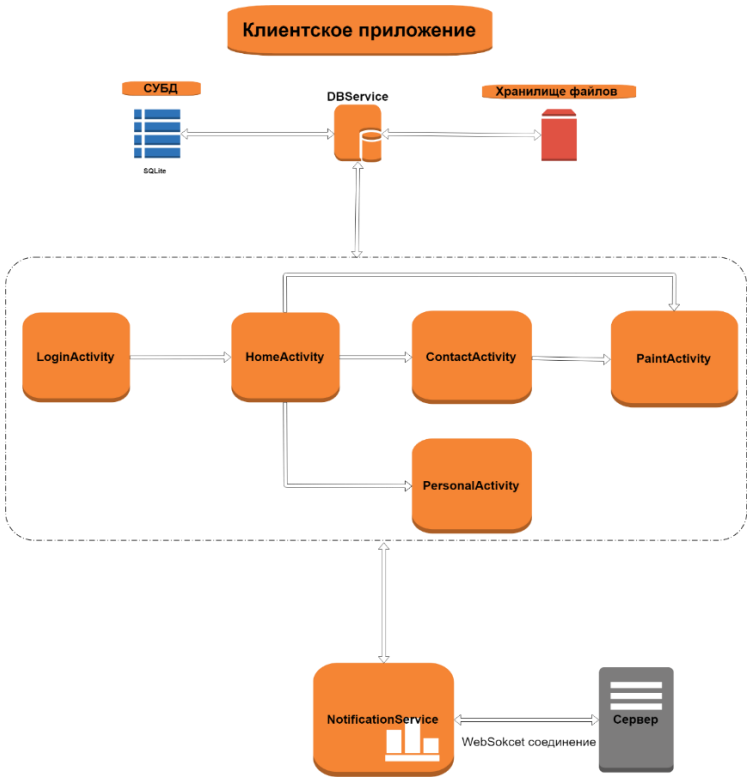


Рисунок 10 – Архитектура клиентского приложения
с указание всех действующих компонентов

Каждое Activity может запрашивать DBService или NotificationService выполнять операции по работе с СУБД, интернетом. DBService также служит для работы с файловым хранилищем устройства. Навигация по Activity также показана на рисунке 9, чтобы попасть в PaintActivity, необходимо начать движение с LoginActivity, то есть пройти авторизацию/регистрацию. Далее пользователь попадает на главное HomeActivity и если уже имеются ранее созданные открытки, то он может сразу перейти к онлайн редактированию в PaintActivities. Если же открыток нет, то необходимо их создать, перейдя в ContactActivity для выбора собеседника, и после этого уже в PaintActivity для онлайн общения. Также HomeActivity имеет возможность перехода в PersonalActivity, которое позволяет редактировать личную информацию пользователя.

Для проектирования дизайна всех Activity, необходимо определится с общей темой и направлением интерфейса. Отличным удобным решением является Material Design. Согласно [7] Material Design - дизайн программного обеспечения и приложений операционных систем Android и Chrome OS от компании Google. Главная идея дизайна заключается в приложениях, которые открываются и сворачиваются как карточки, используя эффекты теней. У приложений не должно быть острых углов, карточки должны переключаться между собой плавно и практически незаметно. Material Design представляет собой концепт минимизации дизайна приложений и на данный момент это самый распространённая модель на рынке, знакомая всем пользователям Android. Использование Material Design позволит дать пользователю привычный дизайн и как следствие быстрое время адаптации. Основной цветовой схемы примем светло-морскую расцветку, с использованием затемненных прозрачных теней и контрактного белого текста или иконок на фоне голубого цвета.

Все классы с отношениями, используемые для реализации работы Activity и Services, представлены на UML-диаграмме классов в приложениях А и Б. На схеме они обозначены словами на английском языке, поскольку реа-

лизованы на языке Java, также подробно расписаны их поля и методы, определены отношения наследования и реализации интерфейсов, необходимых для реализации спроектированных компонентов.

2.5.3 Проектирование HomeActivity

Первое что должно быть доступно пользователю, работающему в HomeActivity, это список уже созданных взаимных открыток. Возможность добавления новых открыток необходимо расположить поверх снизу в виде кнопки, назначение которой будет интуитивно понятно из-за соседства со списком открыток. На рисунке 10 представлен шаблон дизайна HomeActivity.

Доступ к редактированию личной информации PersonalActivity обычно находится в выпадающем списке в правой части бара Activity. Но в последнее время многие приложения используют выдвижное меню слева, оно намного удобнее, так как представляет собой почти отдельное Activity, в котором можно расположить необходимые пункты меню, не тратя пространство в основном Activity. На рисунке 11 представлен шаблон дизайна выдвижного меню для HomeActivity, содержащие пункт выхода их аккаунта и доступ к редактированию информации PersonalActivity через нажатие на текст содержащий краткую персональную информацию о пользователе. Программная структура классов HomeActivity представлена на рисунке А.1.

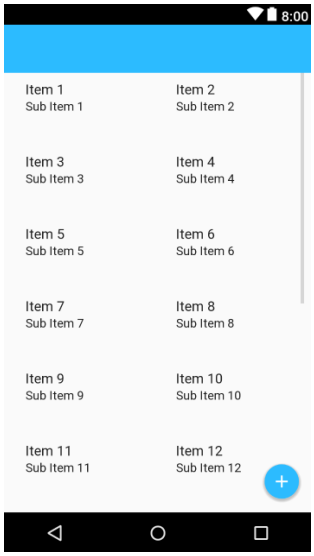


Рисунок 11 – Шаблон дизайна HomeActivity

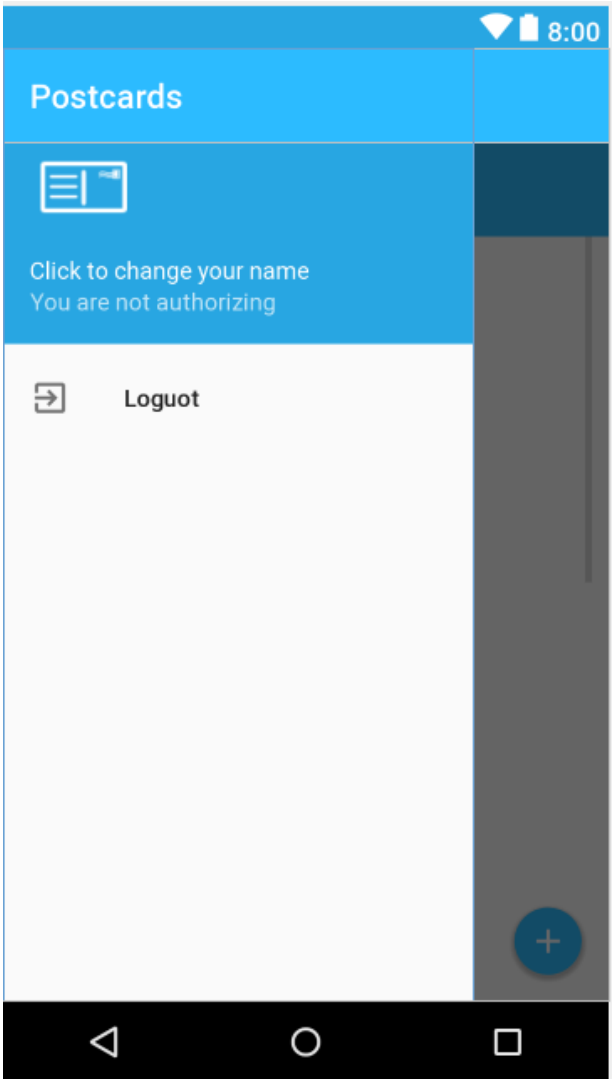


Рисунок 12 – Шаблон дизайна выдвижного меню HomeActivity

2.5.4 Проектирование LoginActivity

LoginActivity должно предоставлять пользователю возможность авторизации или регистрации через формы вводов номера телефона и кода подтверждения. Также проверять введенные данные на валидность и показывать возникающие ошибки вводе. На рисунке 12 представлен шаблон дизайна LoginActivity. Программная структура классов LoginActivity представлена на рисунке А.4.

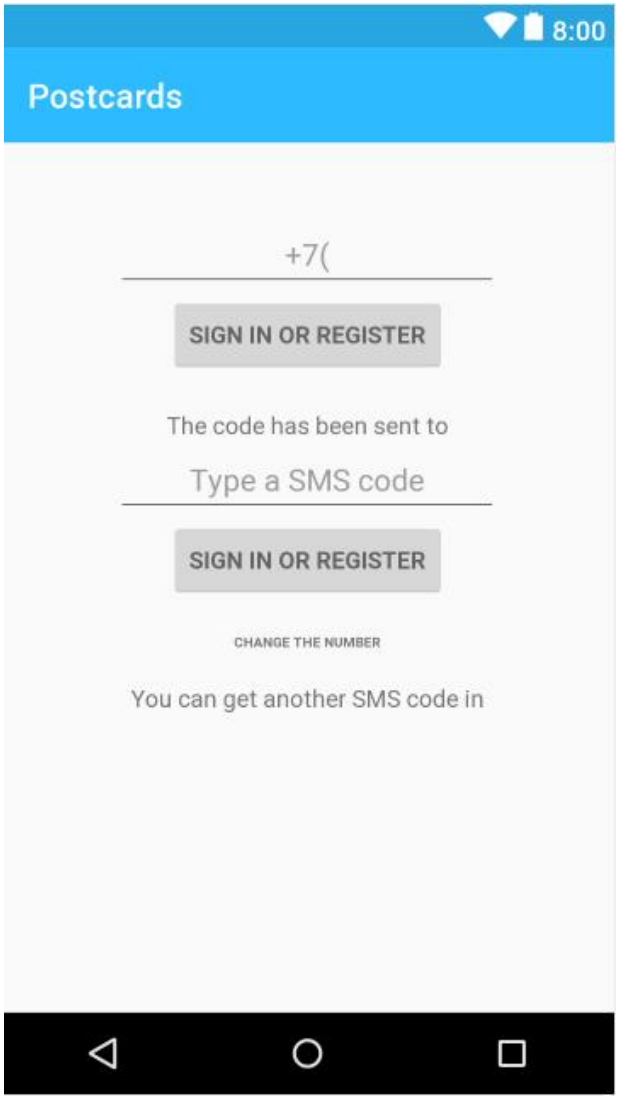


Рисунок 13 – Шаблон дизайна выдвижного меню HomeActivity

2.5.5 Проектирование ContactsActivity

ContactsActivity первостепенно должно запрашивать список контактов у ОС, приводить номера телефонов контактов в валидную форму, и выводить проверенный список пользователю с возможностью выбора конкретного номера у контакта, если же к него их несколько. Также для быстрой навигации в баре Activity расположим поиск по контактам. На рисунку 13 представлен шаблон дизайна ContactsActivity. Программная структура классов ContactsActivity представлена на рисунке А.2.

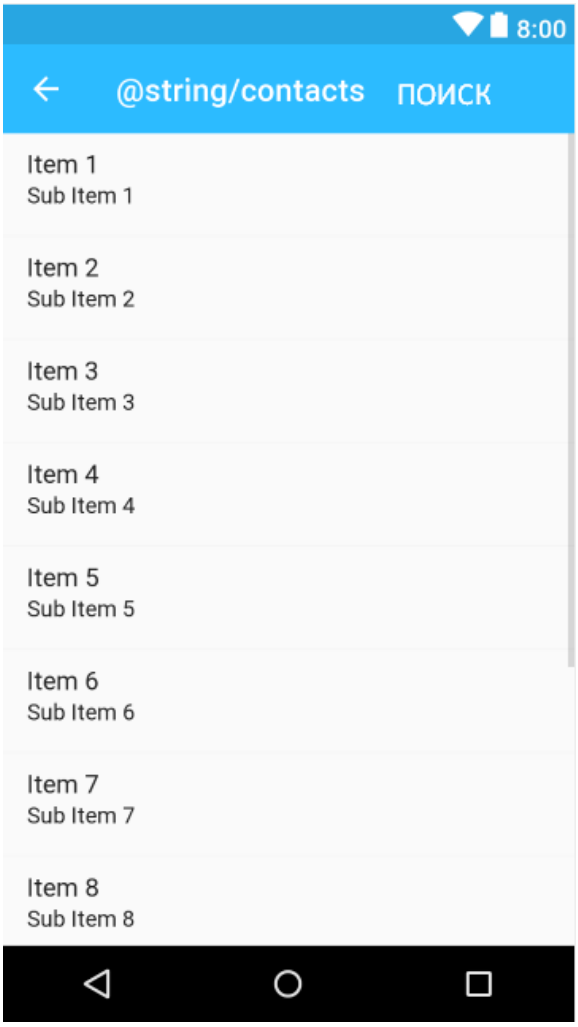


Рисунок 14 – Шаблон дизайна ContactsActivity

2.5.6 Проектирование PersonalActivity

PersonalActivity обеспечивает пользователя возможностью редактирования личной информации, соответственно оно должно содержать формы редактирования личных данных, а также кнопку подтверждения, для удобства расположенную в правой крайней части бара Activity. На рисунке 14 представлен шаблон дизайна PersonalActivity. Программная структура классов PersonalActivity представлена на рисунке А.3.

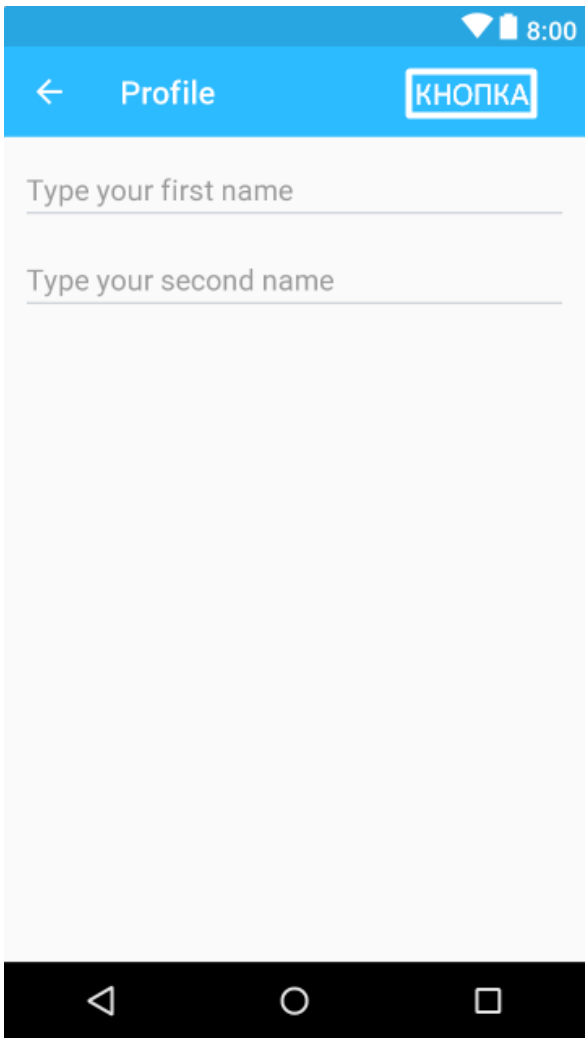


Рисунок 15 – Шаблон дизайна PersonalActivity

2.5.7 Проектирование PaintActivity

PaintActivity должно полностью должно быть занято рабочей областью PaintView для работы с открыткой. На рисунке 15 представлен шаблон дизайна PaintActivity. Инструменты рисования располагаются верхнем правом углу рабочей области (изменение цвета и размера кисти). История изменений (сообщений) находится в нижней части. Это делается для не загромождения рабочей области и предоставления пользователю максимум пространства для рисования. Программная структура классов PaintActivity представлена на рисунке А.5.

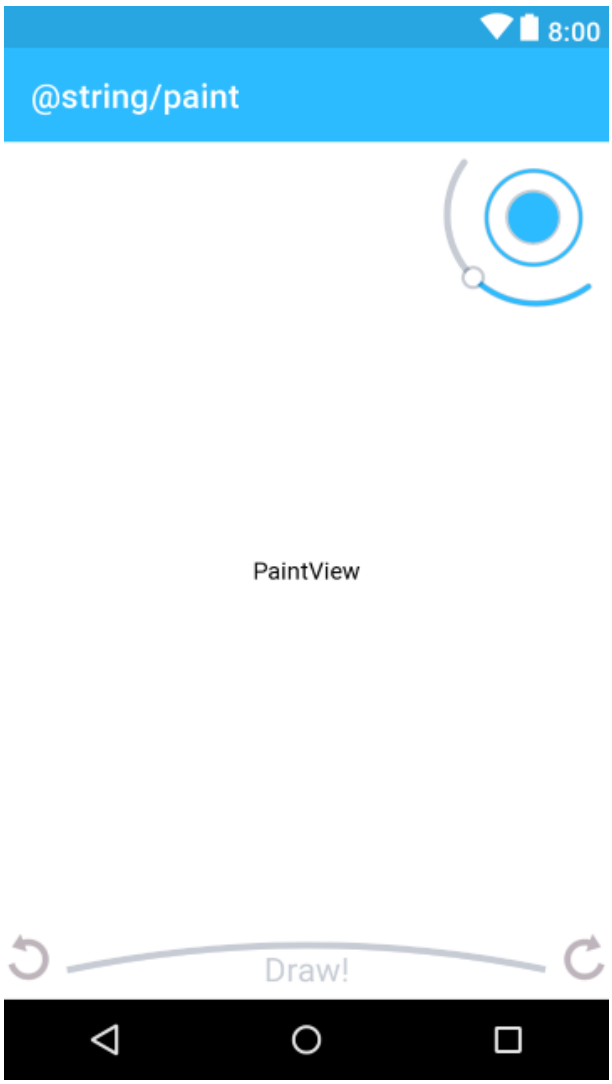


Рисунок 16 – Шаблон дизайна PaintActivity

2.5.8 Проектирование NotificationService

NotificationService должен обеспечивать бесперебойную работу с интернет соединением, получать и отправлять данные на сервер. Все операции выполнимые NotificationService необходимо вынести в отдельный поток, который будет работать без остановки циклически, и запускаться со стартом ОС. NotificationService берет на себя задачу по показу уведомлений о новых сообщениях, уведомление должно содержать информацию о количестве сообщений, в нашем случае путей нанесенных на открытку собеседником, также идентификаторы собеседника (имя или номер телефона), время получения сообщений. При нажатии на уведомление должен происходить переход в PaintActivity, где будут отображены новые изменения на соответствующей

открытке. На рисунке 16 представлен шаблон дизайна уведомления. Программная структура классов NotificationService представлена на рисунке А.7.

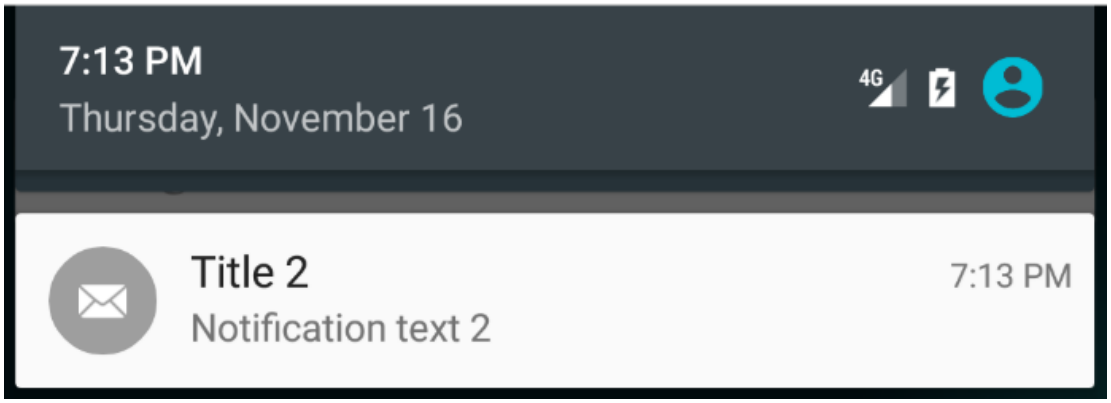


Рисунок 17 – Шаблон дизайна уведомлений

2.5.9 Проектирование DBService

Держать DBService постоянно в рабочем состоянии нет необходимости, но важно вынести всю его работу в отдельный поток, так как работа с БД очень ресурсоемкий процесс. Запускается DBService только при потребности обратится к БД и соответственно завершает свою работу после выполнения всех операции с БД. DBService должен брать на себя работу по преобразованию данных в нужных форма SQLite, генерации запросов и их выполнение, подключение к БД, и предоставлять Activity и Service, связывающееся с ним, удобное API для взаимодействия. Программная структура классов DBService представлена на рисунке А.6.

2.6 Проектирование клиентской БД

Так как клиентская часть представлена СУБД SQLite, то разработка таких типов БД осуществляется в три этапа, на выходе которого получается схема БД с SQL описанием для дальнейшего импорта в СУБД.

2.6.1 Инфологическое проектирование

Данный этап характеризуется спецификацией сущностей, их атрибутов и связи между ними. Результатом является схема сущностей и их связей.

С учетом раздела 2.3 были выделены и определены следующие сущности, представленные в таблицу 4. Пустые ячейки количества экземпляров для сущности означают, что их заполнение будет выполнено по ходу работы и приложения и нельзя заранее предсказать их точное количество.

Таблица 4 – Определение сущностей

Название	Описание	Количество экземпляров
1	2	3
Пользователь	данная сущность содержит персональные данные пользователей, для которых есть открытки	>100
Открытка	информация о нарисованных открытках	>100
Путь	информация о траектории рисования и параметров линии, наносимой пользователем	
Точка	информация о точке на экране определенного пути, наносимого пользователем	

Далее выявим атрибуты сущностей, которые описаны в таблицах 5 – 8.

Таблица 5 – Атрибуты сущности Пользователь

Название	Описание	Диапазон значений	Единица измерения	Пример
Код пользователя	уникальный идентификатор пользователя	1, 2, …	-	4
Фамилия	фамилия пользователя	-	-	Иванов
Имя	имя пользователя	-	-	Иван
Телефон	телефон пользователя	0000000000 - 9999999999	-	+7(999)9999999
Главный	флаг для определения авторизованного пользователя от остальных	true,false	-	false

Таблица 6 – Атрибуты сущности Открытка

Название	Описание	Диапазон значений	Единица измерения	Пример
1	2	3	4	5
Код открытки	уникальный идентификатор открытки	1, 2, …	-	4
Тип	тип открытки	0,1, 2, …	-	3

Продолжение таблицы 6

1	2	3	4	5
Название	название открытки	-	-	Иван
Хэш	уникальный строковый идентификатор открытки	-	-	41c3324b945852f9a72d9d6f77c45941
Дата создания	дата создания записи пользователя	-	число, месяц, год	2017-04-23 10:42:41
Обновление	флаг для определения состояния просмотра изменений открытки	true,false	-	true

Таблица 7 – Атрибуты сущности Путь

Название	Описание	Диапазон значений	Единица измерения	Пример
Код пути	уникальный идентификатор пути	1, 2, …	-	4
Размер	тип открытки	0,1, 2, …	-	50
Цвет	название открытки	…-1,0,1, 2.	-	-13845505
Дата создания	дата создания записи пользователя	-	число, месяц, год	2017-04-23 10:42:41

Таблица 8 – Атрибуты сущности Точка

Название	Описание	Диапазон значений	Единица измерения	Пример
Код точки	уникальный идентификатор точки	1, 2, …	-	4
x	x координата точки	…-1,0,1, 2.	-	50
y	y координата точки	…-1,0,1, 2.	-	-105
Дата создания	дата создания записи пользователя	-	число, месяц, год	2017-04-23 10:42:41

Все выявленные взаимосвязи между сущностями описаны и представлены в таблице 9.

Таблица 9 – Распределение связей между сущностями

Название первой сущности, участвующей в связи	Название второй сущности, участвующей в связи	Название связи	Тип связи	Обоснование выбора типа связи
1	2	3	4	5
Пользователь	Открытка	относится	многие-ко-многим	Множеству пользователей может относится множество открыток и наоборот нескольким записям открыток может относится несколько записей пользователей. Несколько человек могут работать в нескольких открытках.
Путь	Открытка	соответствует	один-ко-многим	Одной записи сущности Путь соответствует одна запись сущности Открытка, каждой записи сущности Открытка соответствует множество записей сущности Путь. Путь может относится только к одной открытке, в то время как Открытка может иметь несколько путей
Точка	Путь	относится	один-ко-многим	Одной записи сущности Точка соответствует одна запись сущности Путь, каждой записи сущности Путь соответствует множество записей сущности Точка. Путь состоит из множества точек, в то время как точка относится к одному конкретному пути

.

Продолжение таблицы 9

1	2	3	4	5
Путь	Пользователь	относится	один-ко-многим	Одной записи сущности Путь соответствует одна запись сущности Пользователь, каждой записи сущности Пользователь соответствует множество записей сущности Путь. Путь может быть нарисован одним пользователем, в то время как один пользователь может нарисовать множество путей

В результате выделения сущностей, их атрибутов и связей была составлена схема «сущность-связь», которая представлена на рисунке В.1.

2.6.2 Логическое проектирование

На данном этапе проектирования БД необходимо преобразовать данные, полученные на предыдущем этапе, к реляционной модели. Реляционная логическая модель представляет собой совокупность нормализованных отношений, в которых проработаны связи между объектами предметной области и выполнены все преобразования, необходимые для ее эффективной реализации в среде конкретной СУБД. Выполним отображение сущностей инфологической модели на отношения реляционной модели. Для этого определим мигрирующие ключи, которые преобразуются в первичные или внешние, в соответствие с типами идентифицирующих или не идентифицирующих связей.

Связь «Пользователь - Открытка» является связью типа многие–ко–многим. При отображении такой связи на реляционную модель использую третью вспомогательную таблицу, содержащую ключи исходных сущностей. Связь показана на рисунке 18, на рисунке 19 приведены итоговые отношения.

Сущность – Пользователь

Код пользователя	Фамилия	Имя
Телефон	Главный	

Сущность – Открытка

Код открытки	Название	Хэш	Дата создания
Обновление			

Рисунок 18 – Связь Пользователь – Открытка

Сущность – Пользователь

Код пользователя	Фамилия	Имя
Телефон	Главный	

Код пользователя
Код открытки

Сущность – Открытка

Код открытки	Название	Хэш	Дата создания
Обновление			

Рисунок 19 – Отображение связи Пользователь – Открытка

Связь «Путь – Открытка» является связью типа один–ко–многим. При отображении ключ порожденной сущности добавляется в исходную сущность. Исходной сущностью является сущность Путь, порожденной – Открытка. Связь показана на рисунке 20, на рисунке 21 приведены итоговые отношения.

Сущность – Путь

Код пути	Размер	Цвет	Дата создания

Сущность – Открытка

Код открытки	Название	Хэш	Дата создания
Обновление			

Рисунок 20 – Связь Путь – Открытка

Сущность – Путь

Код пути	Размер	Цвет	Дата создания	**Код открытки**

Сущность – Открытка

Код открытки	Название	Хэш	Дата создания
Обновление			

Рисунок 21 – Отображение связи Путь - Открытка

Связь «Точка – Путь» является связью типа один–ко–многим. При отображении ключ порожденной сущности добавляется в исходную сущность. Исходной сущностью является сущность Точка, порожденной – Путь. Связь показана на рисунке 22, на рисунке 23 приведены итоговые отношения.

Сущность – Точка

Код точки	x	y	Дата создания

Сущность – Путь

Код пути	Размер	Цвет	Дата создания

Рисунок 22 – Связь Точка – Путь

Сущность – Точка

Код точки	x	y	Дата создания	**Код пути**

Сущность – Путь

Код пути	Размер	Цвет	Дата создания

Рисунок 23 – Отображение связи Точка – Путь

Связь «Путь – Пользователь» является связью типа один–ко–многим. При отображении ключ порожденной сущности добавляется в исходную сущность. Исходной сущностью является сущность Путь, порожденной – Пользователь. Связь показана на рисунке 24, на рисунке 25 приведены итоговые отношения.

Сущность – Путь

Код пути	Размер	Цвет	Дата создания

Сущность – Пользователь

Код пользователя	Фамилия	Имя
Телефон	Главный	

Рисунок 24 – Связь Путь – Пользователь

Сущность – Путь

Код пути	Размер	Цвет	Дата создания	**Код пользователь**

Сущность – Пользователь

Код пользователя	Фамилия	Имя
Телефон	Главный	

Рисунок 25 – Отображение связи Путь – Пользователь

Следующим шагом является нормализация полученных отношений, для исключения будущих возможных проблем по работе с БД.

Отношения называют приведенными к первой нормальной форме, если значения всех его атрибутов атомарные. Все атрибуты согласно таблицам 5-8 являются простыми, соответственно все отношения находятся в 1НФ.

Отношения приведены ко 2НФ, если все отношения находятся в 1НФ, и каждый не ключевой атрибут функционально полно зависит от первичного ключа. Диаграммы функциональных зависимостей, которые изображены на рисунках Г.1 – Г.4.

Проанализировав представленные отношения, можно сделать вывод, что они находятся в 3НФ, так как они находятся во второй нормальной форме и все атрибуты, которые не являются ключевыми, не имеют транзитивной зависимости от ключевых атрибутов.

Результатом этапа логического проектирования является реляционная модель, представленная в приложение Д, а также логическая модель, изображенная на рисунке Е.1.

2.6.3 Физическое проектирование

Данных этап характеризуются приведением отношений реляционной модели к форме таблиц, которые затем будут реализованы в СУБД. Согласно пункту 1.4 данного документа, в качестве СУБД клиентской части приложе-

ния используется SQLite, данные в 10-14 представлены в соответствующей форме SQLite.

Таблица 10 – Пользователь

Название поля	Тип данных	Ограничения	Значения по умолчанию	NULL	Индексация
Код пользователя	INTEGER	PRIMARY KEY	-	нет	нет
Фамилия	TEXT	-	-	нет	нет
Имя	TEXT	-	-	нет	нет
Телефон	TEXT	UNIQUE	-	нет	да
Главный	BOOLEAN	-	0	нет	нет

Таблица 11 – Путь

Название поля	Тип данных	Ограничения	Значения по умолчанию	NULL	Индексация
Код пути	INTEGER	PRIMARY KEY	-	нет	нет
Размер	INTEGER	-	-	нет	нет
Цвет	TEXT	-	-	нет	нет
Дата создания	TIMESTAMP	-	-	нет	нет
Код пользователя	INTEGER	FK	-	нет	нет
Код открытки	INTEGER	FK	-	нет	нет

Таблица 12 – Открытка

Название поля	Тип данных	Ограничения	Значения по умолчанию	NULL	Индексация
Код открытки	INTEGER	PRIMARY KEY	-	нет	нет
Тип	INTEGER	-	-	нет	нет
Название	TEXT	-	-	нет	нет
Хэш	TEXT	UNIQUE	-	нет	да
Дата создания	TIMESTAMP	-	CURRENT_TIMESTAMP	нет	нет

Таблица 13 – Точка

Название поля	Тип данных	Ограничения	Значения по умолчанию	NULL	Индексация
Код точки	INTEGER	PRIMARY KEY	-	нет	нет
x	REAL	-	-	нет	нет
y	REAL	-	-	нет	нет
Дата создания	TIMESTAMP	-	CURRENT_TIMESTAMP	нет	нет
Код пути	INTEGER	FK	-	нет	нет

Таблица 14 – Пользователь – Открытка

Название поля	Тип данных	Ограничения	Значения по умолчанию	NULL	Индексация
Код	INTEGER	PRIMARY KEY	-	нет	нет
Код открытки	INTEGER	FK	-	нет	нет
Код пользователя	INTEGER	FK	-	нет	нет

SQLite предусматривает инструменты проверки ссылочной целостности, таблица 15 содержит настройки ограничений внешних ключей.

Таблица 15 – Ссылочной целостность

Название таблицы	Внешний ключ	Требование ссылочной целостности
Пользователь-открытка	Код открытки	Каскадное удаление: при удалении открытки удаляется и его запись в таблице Пользователь-открытка.
Пользователь-открытка	Код пользователя	Каскадное удаление: при удалении пользователя удаляется и его запись в таблице Пользователь-открытка.
Точка	Код пути	Каскадное удаление: при удалении пути удаляется и соответствующая записи в таблице точка.
Путь	Код открытки	Каскадное удаление: при удалении открытки удаляется и его запись в таблице Путь.
Путь	Код пользователя	Каскадное удаление: при удалении пользователя удаляется и его соответствующая запись в таблице Путь.

В результате завершения данного этапа и полного проектирования клиентской БД была сформирована схема физического проекта, приведенная в приложение рисунок Ж.1.

2.7 Проектирование серверной БД

Согласно пункту 2.4.4 MongoDB является не реляционной, а объектно-ориентированной. Данные в ней представляется в виде объектов. Отношения между ними реализуются, так же, как и в ООП с помощью ссылок, либо включения объекта.

Серверной БД достаточно хранить информацию о зарегистрированных пользователях, авторизированных устройствах и пересылаемых сообщениях,

для требуемого обеспечения функционирования приложения, остальную дополнительная информация хранится в вышеописанной клиентской БД на устройстве пользователя. На рисунке И.1 показаны объекты серверной БД и связи между ними.

2.8 Проектирование логики функционирования клиент-серверного приложения

Заключительным этапом проектирования является создание логики взаимодействия клиентской части с серверной. На данном этапе необходимо выделить и четко прописать порядок действий всех процессов обмена информации.

2.8.1 Беспарольная авторизация

Доступ к личному аккаунту всех современных мессенджеров осуществляется по номеру телефона. Потребность в создании пароля полностью отпадает при возможности авторизации через SMS код подтверждения. К тому же использование этого метода значительно повышает уровень защищенности доступа к данных, так как код подтверждения представляет собой случайно сгенерированную численную последовательность доступную, только владельцу номера телефона. Время действия кода подтверждения примем за 60 минут, соответственно запрос на обновление кода подтверждения тоже равняется 60. Для удобства пользования приложением процесс регистрации можно упразднить, сделав его пассивным. Это значит, что пользователям для входа в приложение будет дана только одна форма ввода номера, если не существует аккаунта по данному номеру телефона, то он будет автоматически создан, при вводе кода подтверждении. Регистрация будет скрыта под авторизацией, что позволит пользователям получить быстрый доступ к приложению.

На основе этой вышеописанной информации выстраивается следующий порядок действий для выполнения авторизации:

1) отправка клиентом номера телефона на сервер;

2) проверка сервером корректности номера и отправка ошибки клиенту в случае неправильности набора, с целью повторного ввода, то есть возвращение к действию 1;

3) проверка на существование пользователя в системе, и в случае его отсутствия занесение его в БД;

4) генерация случайной числовой последовательности и отправка ее в виде SMS кода клиенту;

5) выставления таймера для конкретного номера телефона в течении 60 минут действия кода, в течение этого времени невозможно получить и запросить сервера нового кода подтверждения;

6) получение клиентом кода и отправка его серверу через форму подтверждения;

7) проверка кода подтверждения, в случае несоответствия отправка ошибки с просьбой повторного ввода, возвращение к 6;

8) в случае правильности кода, генерация идентифицирующей строки для авторизованного устройства - токена, выставление даты окончания действия токена, занесение данных об авторизованном устройстве в объект БД пользователя User;

9) перенаправление клиента на главное HomeActivity;

10) отправка токена клиентом серверу при каждом запросе, для подтверждения доступа к приложению.

2.8.2 Обмен сообщениями

Обмен сообщениями происходит между 2 пользователями, обозначим их Клиент А и Клиент Б. Сообщения представляют собой точки с координатами нанесения путей на открытку. Существует два варианта случая проведения процесса общения, первый, когда оба клиента находятся в PaintActivity и могут видеть все изменения и сообщения в реальном времени. Второй, когда только один из участников рисования находится в PaintActivity, а другой участник не участвует в совместном рисовании. В таком случае сообщения необходимо сохранять на сервере, для последующей загрузки, клиентом, не

участвующего в рисовании в данный момент, он должен быть своевременно уведомлен, о том, что для него есть новое сообщение.

Опишем алгоритм взаимодействия с сервером, когда Клиент А и Клиент Б оба находится в PaintActivty;

1) клиент А отправляет сообщение на сервер о том, что находится в PaintActivity и готов для принятия данных;

2) клиент Б отправляет сообщение на сервер о том, что находится в PaintActivity и готов для принятия данных;

3) сервер, видя, что Клиент А и Б готовы для получения данных и отрисовки, начинает проверять объекты Point БД, на наличие не загруженных сообщений для клиентов;

4) при наличии оставленных ранее и неотправленных сообщений для клиентов, сервер пересылает их и создает специальный канал (комнату) через который пользователи будут пересылать сообщения друг другу;

5) пользователи получают идентификатор комнаты общения от сервера, и могут приступать к совместному рисованию.

Рассмотрим алгоритм действий приложения второго случая, когда Клиент А находится в PaintActivity, а Клиент Б нет:

1) клиент А отправляет сообщение на сервер о том, что находится в PaintActivity и готов для принятия данных;

2) сервер зная, что Клиент Б не готов для получения данных и не находится в PaintActivity, говорит Клиенту А, что готов получить данные для Клиента Б и сохранить их для дальнейшей переправки;

3) клиент А, начинает передавать данные на сервер;

4) сервер начинает принимать и сохранять данные, одновременно отравляя сообщение Клиенту Б, что для него есть новые сообщение;

5) клиент Б, получив сообщение о наличии новых сообщений, создает уведомление для пользователя;

6) пользователь Клиента Б нажимая на уведомление переходит в PaintActivity и уведомляет сервер о готовности к загрузке сообщений;

7) сервер отправляет оставленные сообщения Клиенту Б.

Переключение между этими режимами функционирования происходят быстро и незаметно для пользователя. Клиент может работать с несколькими открытками одновременно, получая различные уведомления и просматривать их.

Этап проектирования взаимодействия клиентской и серверных частей приложения завершает раздел проектирования. Здесь мы разработали проекты общего видения архитектуры, клиентской и серверной БД, функционала и дизайна приложения. На основе этой существенной информации в дальнейшем будет выполнена реализация проекта.

3 ОПИСАНИЕ РАЗРАБОТАННОГО ПРИЛОЖЕНИЯ

Для описания проделанного объема работы по написанию приложения, рассмотрим пример тестового использования. Именно, навигацию через Activity, авторизацию, процесс рисования и создания открыток, работу с БД.

3.1 Описание БД

Применив физическое описание, спроектированное для клиентской SQLite, получим каркасы будущих таблиц. Используя диаграмму объектов, описанную для серверной БД, получим структуру для записи документов в MongoDB. На рисунке 26 представлен результат разработки БД SQLite. На рисунке 27 представлен результат разработки для MongoDB. Таблицы готовы для заполнения данными.

paths		CREATE TABLE paths (id INTEGER PRIMARY KEY,size INTEGER,color I
id	INTEGER	`id` INTEGER
size	INTEGER	`size` INTEGER
color	INTEGER	`color` INTEGER
date	TIMESTAMP	`date` TIMESTAMP DEFAULT CURRENT_TIMESTAMP
user_id	INTEGER	`user_id` INTEGER
post_id	INTEGER	`post_id` INTEGER
point		CREATE TABLE point (id INTEGER PRIMARY KEY,x REAL,y REAL,date
id	INTEGER	`id` INTEGER
x	REAL	`x` REAL
y	REAL	`y` REAL
date	TIMESTAMP	`date` TIMESTAMP DEFAULT CURRENT_TIMESTAMP
path_id	INTEGER	`path_id` INTEGER
post_users		CREATE TABLE post_users (id INTEGER PRIMARY KEY,user_id INTEGE
id	INTEGER	`id` INTEGER
user_id	INTEGER	`user_id` INTEGER
post_id	INTEGER	`post_id` INTEGER
postcards		CREATE TABLE postcards (id INTEGER PRIMARY KEY,type INTEGER,n
id	INTEGER	`id` INTEGER
type	INTEGER	`type` INTEGER
name	STRING	`name` STRING
hash	TEXT	`hash` TEXT UNIQUE
date	TIMESTAMP	`date` TIMESTAMP DEFAULT CURRENT_TIMESTAMP
updated	BOOLEAN	`updated` BOOLEAN DEFAULT 0
users		CREATE TABLE users (id INTEGER PRIMARY KEY,name TEXT,admin B
id	INTEGER	`id` INTEGER
name	TEXT	`name` TEXT
admin	BOOLEAN	`admin` BOOLEAN DEFAULT 0
number	TEXT	`number` TEXT UNIQUE
Indices (2)		
index_hash		CREATE UNIQUE INDEX index_hash ON postcards(hash)
hash		`hash`
index_number		CREATE UNIQUE INDEX index_number ON users(number)
number		`number`

Рисунок 26 – Итоговые таблицы SQLite

```
var PointSchema = new mongoose.Schema({
    x: {
        type: Number,
        required: true
    },
    y: {
        type: Number,
        required: true
    },
    type: {
        type: Number,
        required: true
    },
    sended: {
        type: Boolean,
        default: false
    },
    received: {
        type: Boolean,
        default: false
    },
    postcard_type: {
        type: Number,
        required: true
    },
    color: {
        type: Number,
        required: true
    },
    size: {
        type: Number,
        required: true
    },
    mode: {
        type: Number,
    },
    to: [{type: Schema.Types.ObjectId, ref: 'User'}],
    from: [{type: Schema.Types.ObjectId, ref: 'User'}]
});
```

```
var UserSchema = new mongoose.Schema({
    fullName: {
        type: String,
        trim: true
    },
    countryCode: {
        type: String
    },
    phone: {
        type: String,
        index: true,
        required: true,
        unique: true,
        lowercase: true,
        trim: true
    },
    email: {
        type: String,
        lowercase: true,
        trim: true
    },
    devices: {
        type: [{
            name: String,
            did: String,
            ip: String,
            token: { type: [String], index: true },
            token_time: Number
        }]
    },
});
```

Рисунок 27 – Итоговые таблицы MongoDB

3.2 Результат работы

Рассмотрим тестовый пример работы приложения. Нового пользователя всегда будет встречать LoginActivity требующее вести номер мобильного телефона. Форма ввода используют маску для валидации и корректности номера, также это помогает пользователю понять какой формат номера телефона необходим. На рисунке 28 можно увидеть форму ввода мобильного телефона.

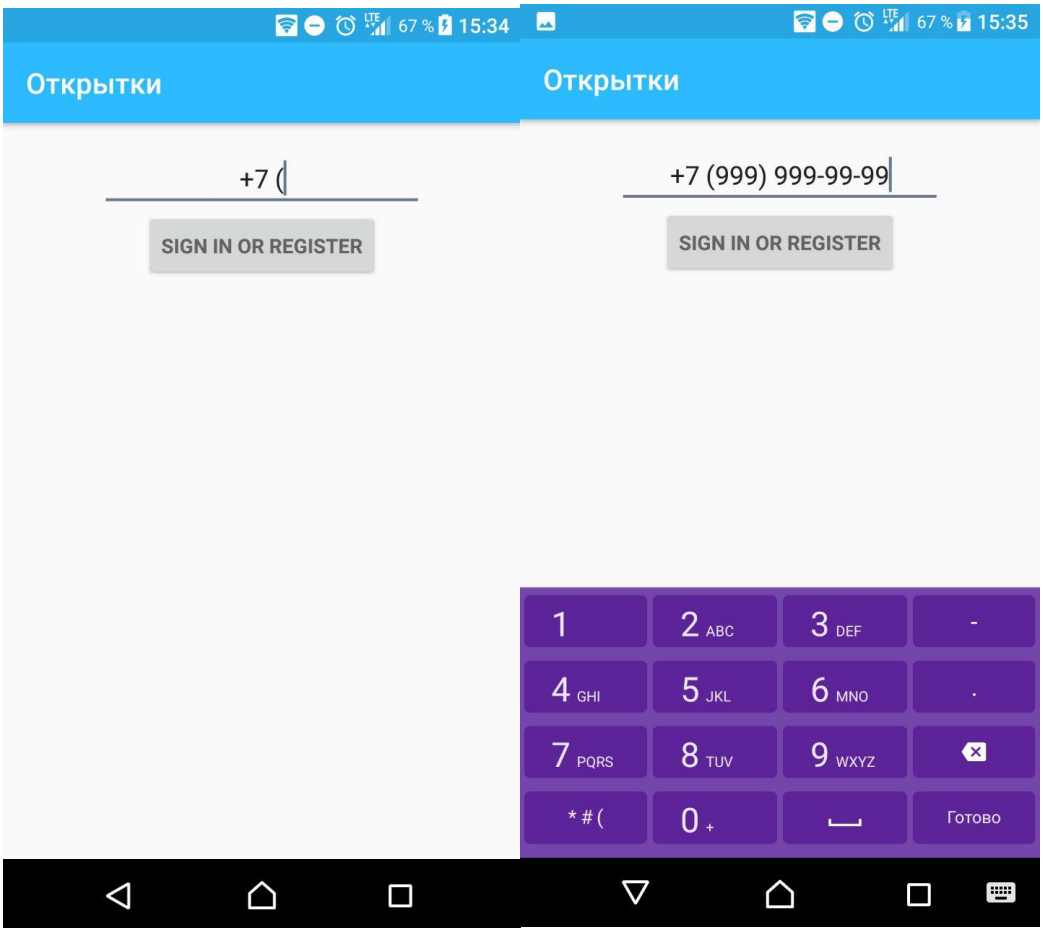
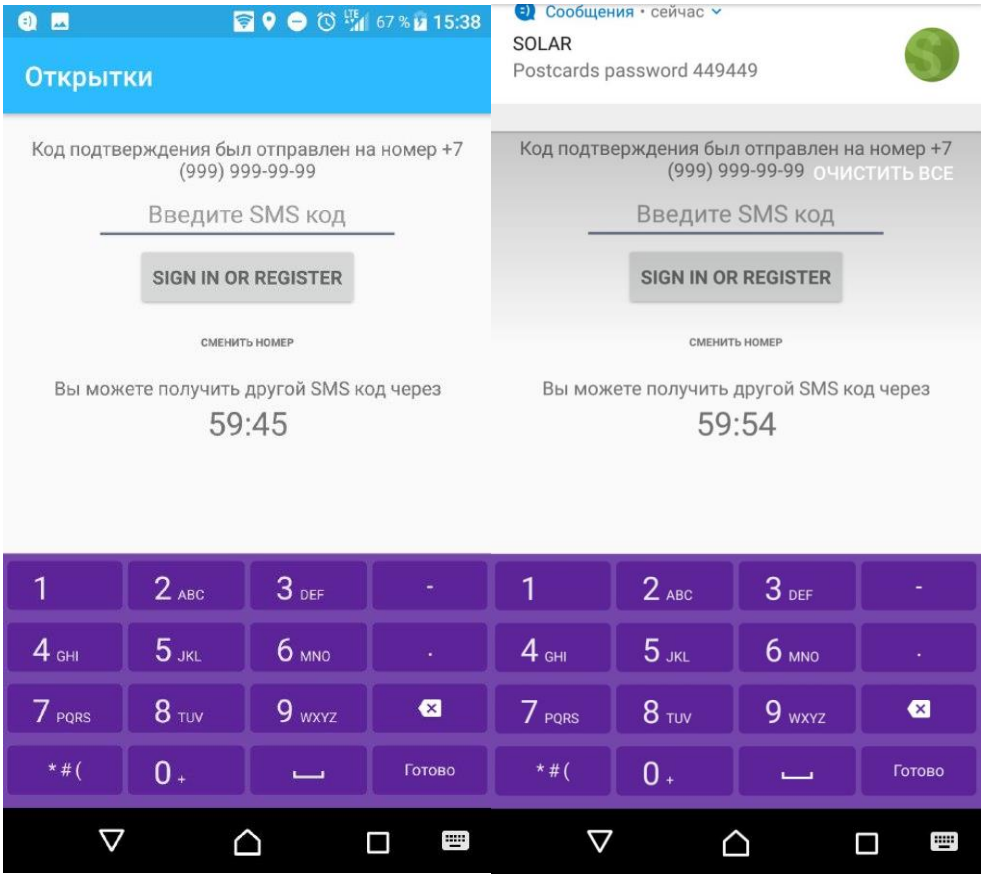


Рисунок 28 – Пример ввода номера телефона

После ввода номера телефона форма исчезнет и появится другая для ввода кода подтверждения. Она также содержит обратных отсчет о действительности этого кода и кнопку о смене номера телефона, по нажатию на которую вам покажут прошлую форму ввода. На сервере в это время произошло сохранение номера телефона в БД и запоминание SMS кода для него. По истечению примерно 10 секунд SMS приходит, и мы можем ввести шестизначный код подтверждения. При неправильном вводе кода, пользователю покажется ошибка. На рисунке 29 показан процесс получения и ввода SMS кода.



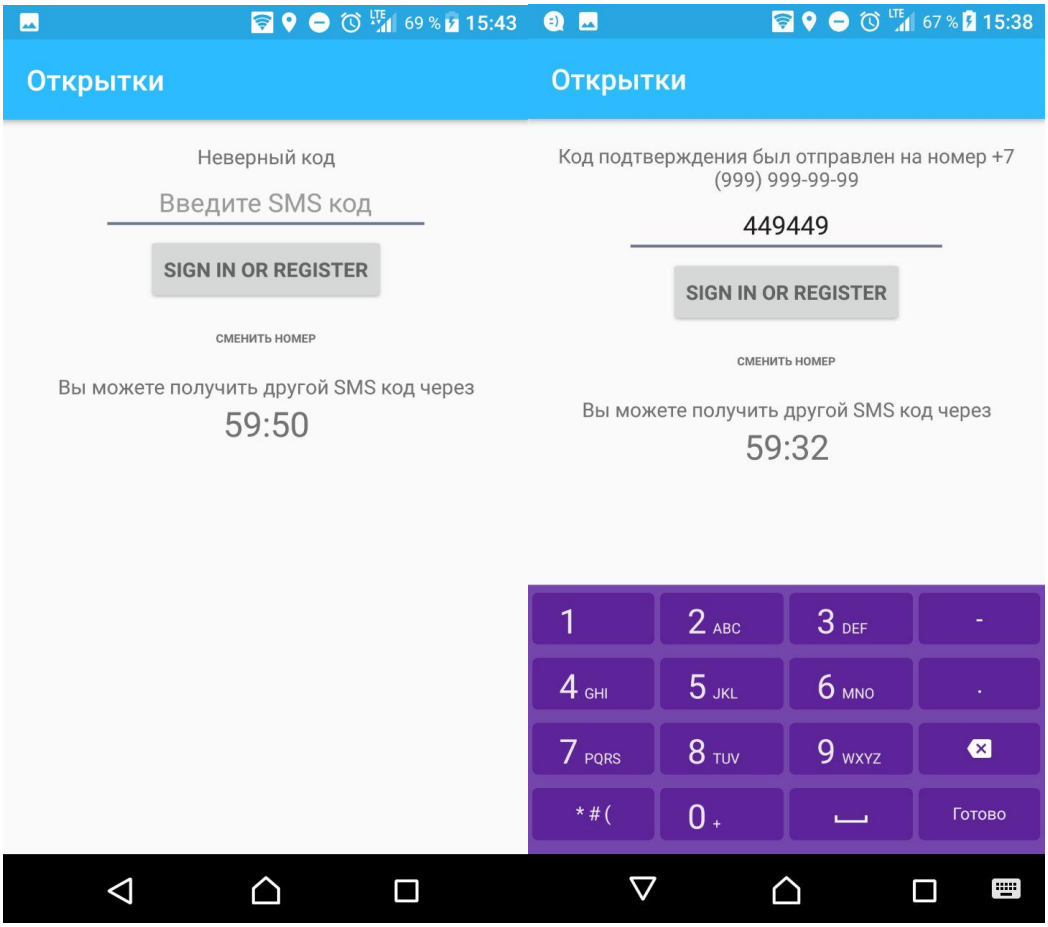


Рисунок 29 – Пример ввода, получения SMS кода и ошибки

После верного ввода кода сервер авторизует устройство под соответствующим аккаунтом и отправит клиенту токен. Клиент при получении токена получает полный доступ к функционалу приложения. В первую очередь после авторизации мы видим PersonalActivity предлагающие нам уточнить личные данные. На рисунке 30 представлен пример изменения личных данных в приложении.

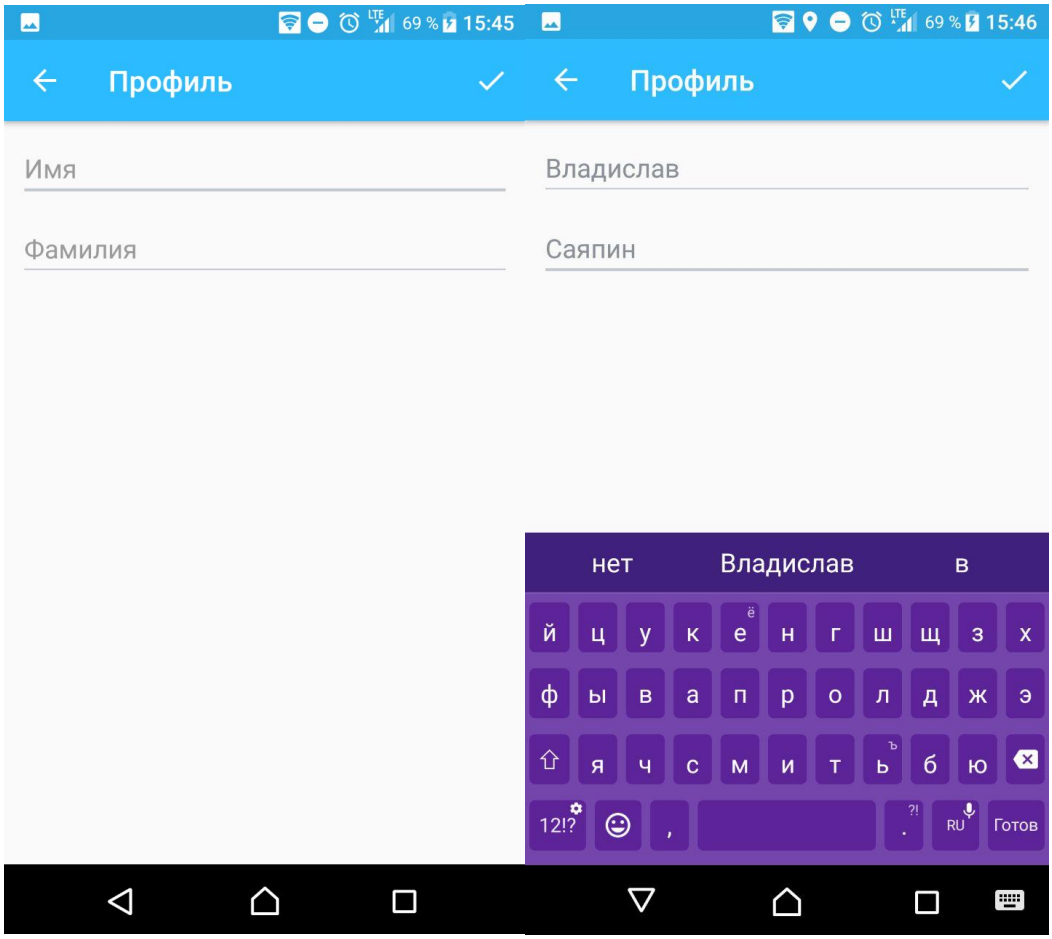


Рисунок 30 – Пример изменения личных данных

Далее мы попадаем на главное HomeActivity, на котором еще нет созданных открыток. С помощью верхней левой иконки, а также свайпа слева направо, мы можем вызвать левое выдвижное меню. В котором находится кнопка выхода из аккаунта, при нажатии на которую появится всплывающее окно подтверждения. А с помощью нажатие на краткую информацию о пользователе вызовется PersonalActivity. При нажатии на круглую голубую кноп-

ку в HomeActivity мы можем добавить новую открытку. На рисунке 31 показан функционал HomeActivity.

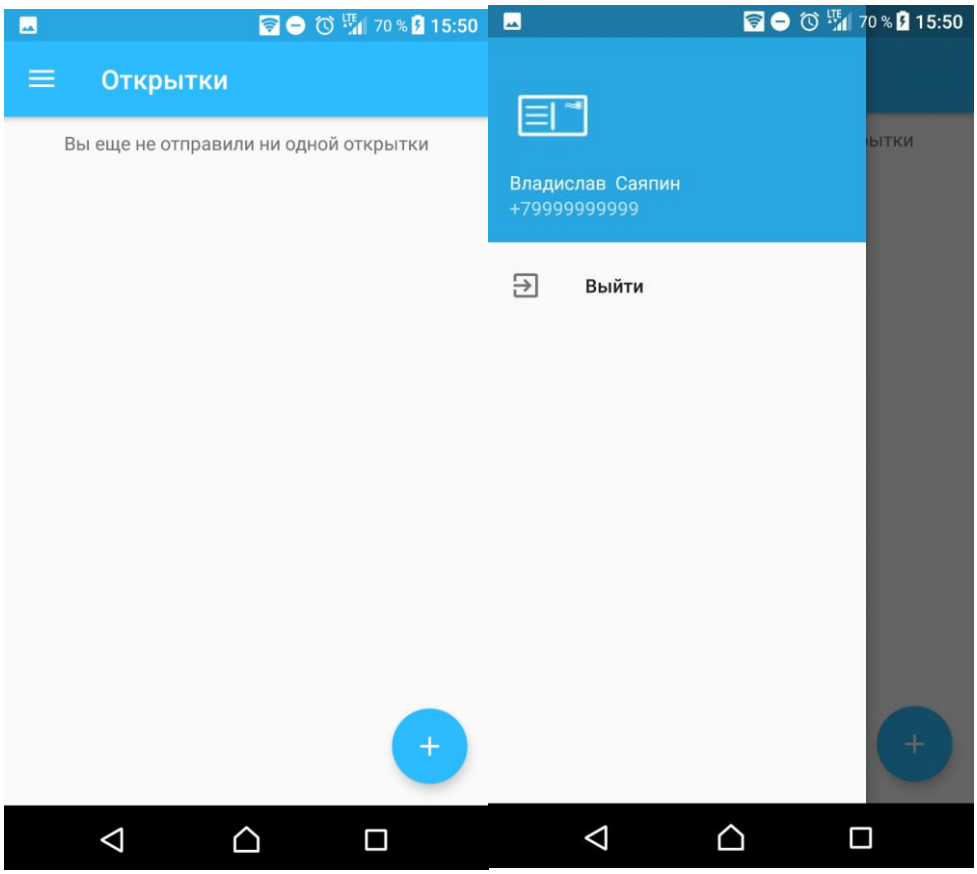


Рисунок 31 – Пример работы с HomeActivity

Сначала мы попадаем в ContactsActivity для выбора человека кому мы хотим отправить открытку, при выборе определенного номера появится всплывающее окно с перечнем доступных номеров контакта. Также для удобной навигации в баре Activity расположен поиск. На рисунке 32 можно посмотреть пример работы с ContactsActivity.

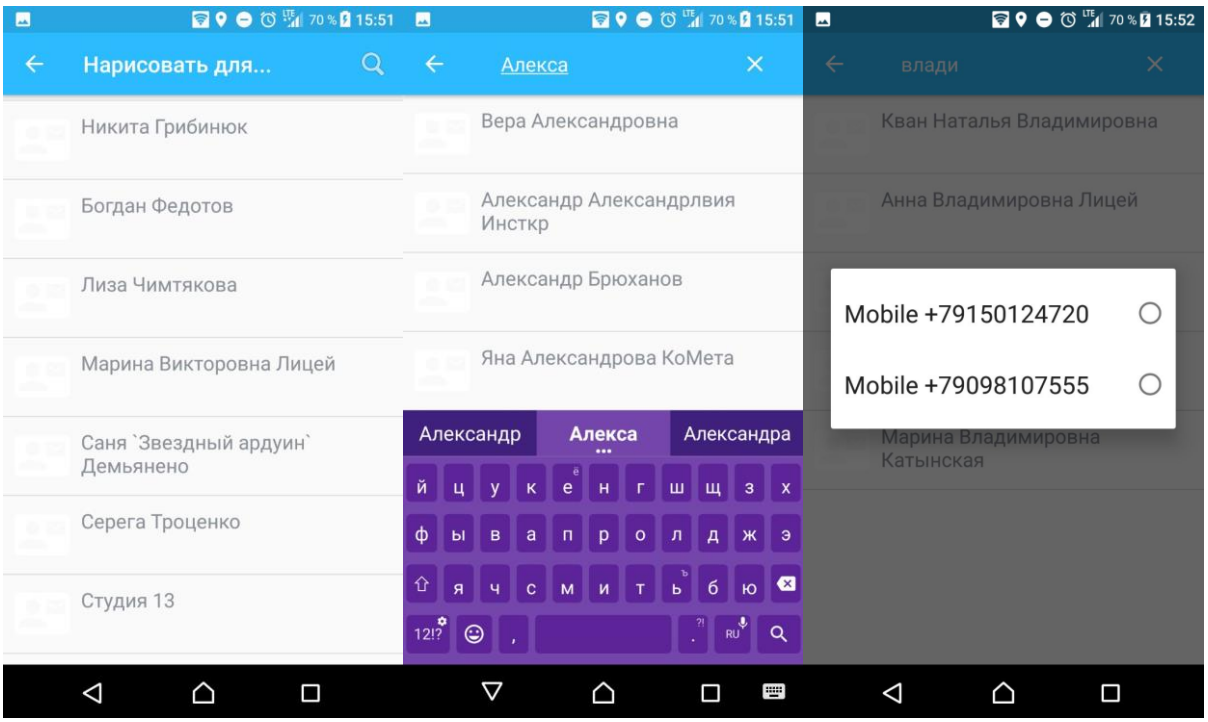


Рисунок 32 – Пример работы с ContactsActivity

После выбора номера телефона создается для него новая открытка, и данные сохраняются в БД. Открывается PaintActivity представляющее собой открытку и инструменты для рисования. С помощью одного зажатого касания возможно рисовать, с помощью двух передвигать открытку, менять масштаб. В верхнем правом углу PaintActivity находится инструменты изменения размера кисти и цвета, при нажатии на круглую кнопку выдвигается палитра для выбора цвета. В нижней части PaintActivity расположена инструмент управлением историей изменений. С помощью ползунка возможно отматывать действия назад и вперед, и продолжать рисование с выбранного момента истории, при этом действия, оставшиеся после будут стерты. В правом углу бара Activity расположено меню с дополнительными вспомогательными действиями по масштабизации и позиционированию холста открытки. Все линии, наносимые на открытку будут тут же сохранены в SQLite, а также отправлены на сервера для загрузки собеседником или его уведомления. После выхода из PaintActivity, HomeActivity отобразит новую открытку в своей сетке, через которую можно попасть напрямую в PaintActivity для онлайн

общения. На рисунке 33 представлен функционал PaintActivity. Обновленная сетка открыток HomeActivity представлена на рисунке 34

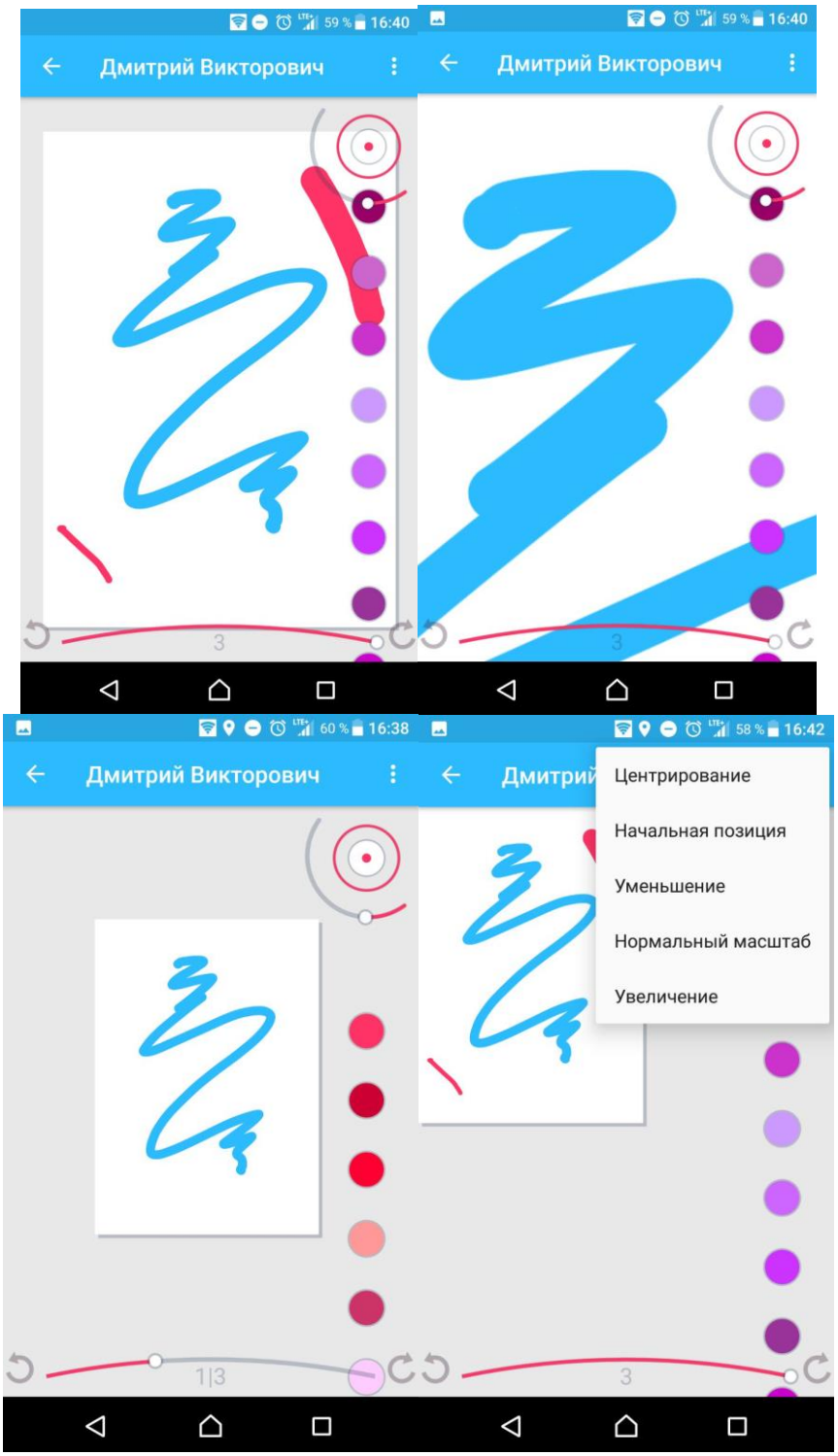


Рисунок 33 – Пример функционала PaintActivity

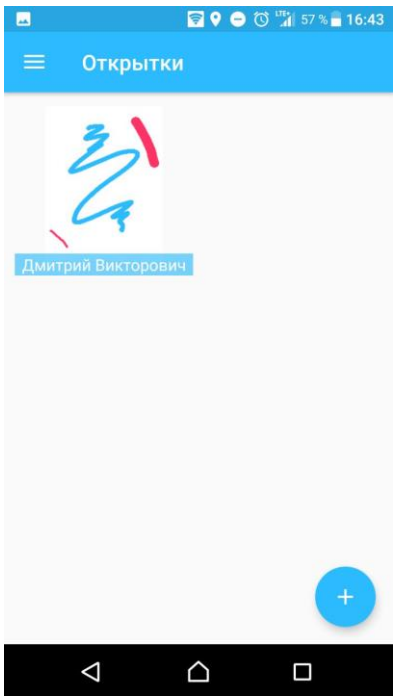


Рисунок 34 – Обновленное HomeActivity

На рисунке 35 можно увидеть уведомления, приходящие второму участнику рисования, говорящие о существовании новых сообщений.

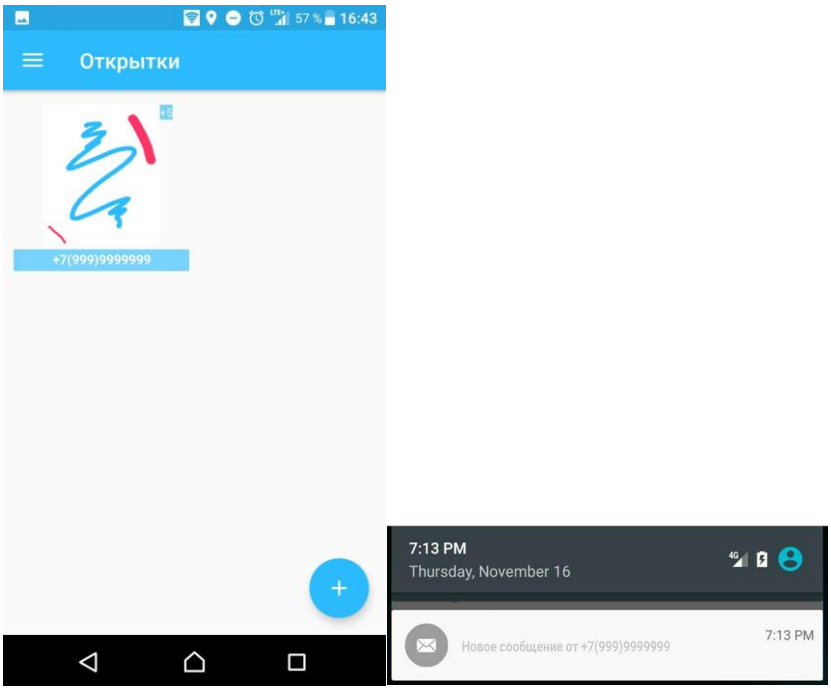


Рисунок 35 – Уведомления о новых сообщениях

На этом завершается этап ознакомления с разработанным приложением. Рассмотрены реализации клиентской и серверной БД, а также тестовый пример основных функциональных возможностей.

4 ОПИСАНИЕ СЕРВИСОВ ОБЕСПЕЧЕНИЯ ИБ ПРИЛОЖЕНИЯ

Защита персональных данных является одним из ключевых моментов разработки приложений для Android ОС. Система имеет встроенные особенности безопасности, что значительно уменьшает проблемы несанкционированного доступа и позволяет избежать сложностей в вопросах поиска решения обеспечения безопасности:

- приложение находится в «песочнице», которая изолирует данные и код от других приложений;

- общий функционал безопасности: криптография, разрешения, безопасная межпроцессорная коммуникация IPC;

- технологии для снижения риска возникновения ошибок управления памятью ASLR, NX, ProPolice, safe_iop, OpenBSD dlmalloc, OpenBSD calloc, и Linux mmap_min_addr;

- шифрование файловой системы, которое может быть активировано для защиты данных, украденных с устройства;

- разрешения, выдаваемые пользователем для доступа к системным возможностям и пользовательской информации;

- разрешения, выдаваемые приложениями для контроля данных приложения;

- блокировка устройства при показе важной информации, запрос на ввод пароля;

- отключение возможности делать скриншот;

- отключение показа скриншота в диспетчере задач;

- ввод пароля блокировки перед выполнением важных действий.

Проблема состоит в том, что при получении root прав, появляется возможность просмотра данных всех приложений, их изменения и просмотра. Также существует проблема защиты пропавших данных. Решением этих вопросов является то, что все данные приложения должны быть зашифрованы, но это нехорошо сказывается на скорости получения данных из-за добавле-

ния подзадачи расшифровки. Компромиссом может являться частичное шифрование файлов с важной информацией, потеря которой недопустима. Но возникает другой вопрос. Как хранить ключи шифрования? Если держать их на устройстве, то все попытки шифрования будут тщетны, так как злоумышленнику достаточно заполучить ключ и с помощью него начать расшифровывать файлы. Легким решение может выступить попытка доверится специальным онлайн инструментам. Но, к счастью, разработчики Android ОС подумали насчет этого и добавили в систему специализированный набор Android инструментов шифрования KeyStored. Он создан для обращения и хранения зашифрованных ключей с помощью самых известных и безопасных алгоритмов шифрования в мире. Инструмент встроен в систему и поддерживает лучшую производительность и скорость доступа к данным. Использование этого хранилища ключей делает шифрование данных действенным и совершенным.

4.1 Использование Android Keystone System

Android Keystone System – это система позволяющая хранить криптографические ключи в специальном «контейнере», доступ к которому получить крайне сложно. Ключи находящиеся в KeyStone могут использоваться для криптографических операций оставаясь не экспортируемыми. KeyStone предлагает возможность ограничивать доступ к ключам по времени и пользователям, например, ввод пароля пользователя при запросе ключа.

Android Keystone System не дает прямой доступ к ключам из процесса приложения и Android устройства в целом. Система заставляет каждому приложение авторизоваться для получения доступа к ключу.

Keystone использует следующие меры для предотвращения несанкционированного извлечения данных:

- ключи никогда не передаются в процесс приложения. Когда приложение выполняет криптографические операции, то открытый текст, зашифрованный текст, сообщения передаются в специальный системный процесс, который выполняет операции;

- ключи могу быть привязаны к специальному защитному аппаратному обеспечению на устройстве. Если у злоумышленника есть возможность доступа к внутреннему хранилищу, то он сможет только прочитать данные ключей, но не извлечь их;

- для доступа к ключу приложение должно пройти обязательную авторизацию, параметры которой задаются один раз при генерации или импорте ключа и не изменяются более. В качестве параметров авторизации могут быть виды криптографических операций (шифровка, расшифровка), которые могут применяться; типы алгоритмов; цели; режимы работы шифров; временной интервал, во время которого может быть осуществлён доступ; авторизация пользователя, которому разрешен доступ.

Использование системы KeyStone осуществляется с помощью классов KeyPairGenerator и KeyGenerator вместе с провайдером AndroidKeyStore.

4.2 Шифрование SharedPreferences

Так как работа SharedPreferences основана на значение ключа в файле XML, то полное шифрование файла недопустимо, так как это не позволит приложению парсить его. Единственный способ защиты SharedPreferences – это шифрование самого ключа и его значения. Поэтому использование SharedPreferences для хранения важных данных не рекомендуется, из-за неудобства их защиты.

Существует несколько удобных библиотек, которые обеспечивают лучшее шифрование SharedPreferences:

- Secure-preferences – оболочка для SharedPreferences, которая шифрует значения общих настроек с использованием AES 128, CBC, PKCS5 и с проверкой целостности в виде хэша SHA 256. Каждый ключ хранится как односторонний хэш SHA 256. Оба ключа и значения кодируются в base64 перед сохранением в файл XML. По умолчанию сгенерированный ключ сохраняется в файле настроек резервного копирования и поэтому может быть прочитан и извлечен пользователем root;

- Encrypted-userprefs – библиотека помогает шифровать SharedPreferences с помощью симметричного шифра AES. Имеет легкий API инструментария для использования;

- android-secure-preferences – библиотека шифрования SharedPreferences с помощью шифра AES. Требуется уровень API 8.

4.3 Шифрование SQLite

Шифрование данных в базе данных можно осуществлять 2 способами. Зашифровывать данные перед их сохранением и потом расшифровывать после выборки, аналогично, как и SharedPreferences в Приложение Д. Или использовать готовое решение библиотеки, обеспечивающие комплексное и быстрое шифрование БД. Самым распространенным таким решением на Android ОС является SQLCipher.

SQLCipher – это расширение SQLite, которое обеспечивает прозрачное 256-битное шифрование AES файлов базы данных. Библиотека постоянно отслеживает обновления SQLite и выпускает релевантные релизы для новых версий БД. Ключ шифрования генерируется на основе специального слова passphrase, которые хранится в БД и может быть задан, измене, сброшен с помощью директивы PRAGMA.

Особенности SQLCipher:

- быстрая производительность с наименьшими дополнительными расходами нагрузки около 5-10%;

- 100% шифрование всех данные в файле БД;

- используются лучшие технологии безопасности (CBC mode, HMAC, key derivation);

- нулевая начальная конфигурация;

- используемые алгоритмы представлены экспертной криптографический библиотекой OpenSSL;

- настраиваемые крипто ресурсы.

Внедрение шифрование БД в приложение не составит никакого труда, так как изменения, которым подвергнется код минимальны, необходимо пе-

реопределить библиотеки работы с БД на библиотеки из пакета SQLCipher и изменить некоторые аргументы функций (добавить пароль и т.д.).

4.4 Протокол HTTPs

Чтобы гарантировать конфиденциальность передаваемых данных между клиентом и сервер используем протокол передачи данных HTTPs. Согласно [6], HTTPS (HyperText Transfer Protocol Secure) — расширение протокола HTTP для поддержки шифрования в целях повышения безопасности. Данные в протоколе HTTPs передаются поверх криптографических протоколов SSL или TLS. HTTPs гарантирует:

- конфиденциальность данных, все данные передачи шифруются, скрывая от публичного доступа чувствительную информацию метаданных и заголовков запросов;

- подлинность сервера, клиенты твердо уверены, что общаются с истинным сервером, исключая возможность подмены или атак «человек по середине»;

- целостность данных, данные не могут быть изменены в процессе передачи.

HTTPs шифрует всю информации передаваемую между клиентом и сервером, в частности тело запроса, полный URL адрес, параметры запросы, HTTP заголовки.

Для внедрения передачи данных через HTTPs необходимо приобрести SSL сертификат, включающий открытые ключи для начала защищенной сессии. Он также является гарантом безопасного подключения и предоставляется клиенту для верификации соединения. Протокол необходимо установить на Front-end сервер Nginx. Для того чтобы получить SSL сертификат можно воспользоваться бесплатным сервисом Let’s Encrypt, который занимается бесплатной выдачей SSL сертификата с целью улучшить безопасность данных в интернете. На рисунке 36 можно увидеть серверную архитектуру, включающую HTTPs соединение.

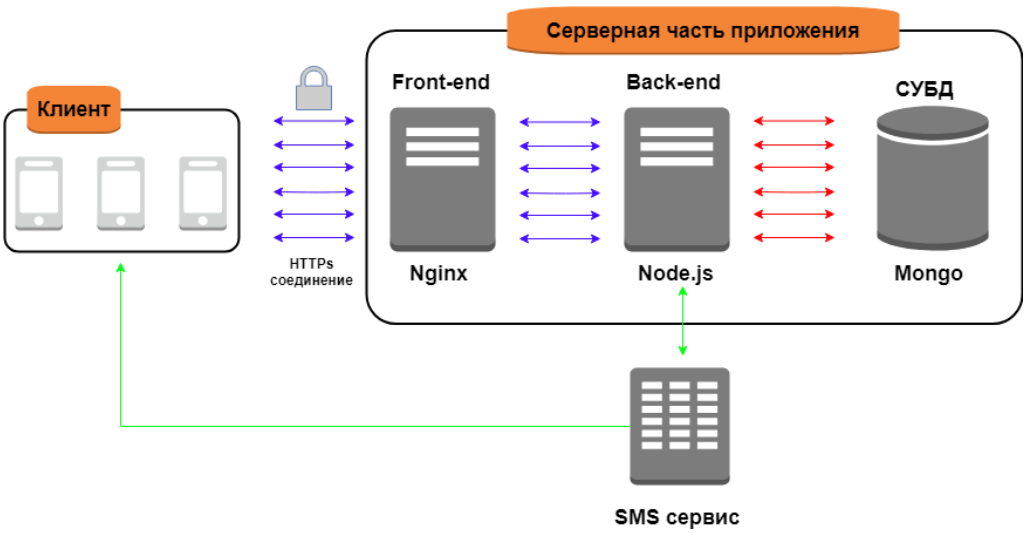


Рисунок 36 – Использование HTTPs соединения

На этом завершается рассмотрение данного этапа. Создать полностью защищённое приложение достаточно сложно и в большинстве случаев невозможно. Но внедрение базовых инструментов безопасности обязательно, для предотвращения основных потенциальных угроз.

5 РАССМОТРЕНИЕ АСПЕКТОВ БЕЗОПАСНОСТИ И ЭКОЛОГИЧНОСТИ

Данный раздел посвящен вопросам безопасности жизнедеятельности пользователей приложения. Необходимо определить правила работы за ПК и мобильным устройством, способы безопасной утилизации носителей информации и компонентов ИС, а также меры, позволяющие предотвратить чрезвычайные ситуации, форс-мажоры и их нежелательные последствия. Исследоваться будут 4 положения: безопасность, экологичность и защита от ЧС, комплекс физических упражнений.

На основе СанПиНа 2.2.2/2.4.1340-03 будет проведен анализ аспектов БЖД.

5.1 Безопасность

В данном подразделе рассмотрим правильные, с точки зрения БЖД, способы построения графических интерфейсов. Выделим 3 аспекта: зонирование или позиционирование элементов, цветовое оформление и типографические представления.

Построение экранных форм начинается с выбора структуры отображения ее элементов. Основное требование в данном вопросе – это интуитивно понимание функционально назначения всех компонентов. Для Android приложение характерно разделение окна на следующие блоки:

- бар, верхняя область, содержащая наименование приложение и доступу к меню;

- блок основного контента Activity.

На рисунке 12 можно увидеть озонные блоки окна Android приложения.

Хоть и существует множество приложений спроектированы не по этой схеме, но такой подход давно зарекомендовал себя как базовый, то при разработке данного приложения был использован именно он. В частности, стоит

отметить, что блок с основным контентом должен занимает весь экран, исключая небольшую полоску бара сверху.

Согласно ГОСТ Р ИСО, 6385-2007 учет эргономических аспектов при построении графических интерфейсов позволит добиться оптимизации производственной нагрузки, исключить эффекты расслабления, а также снизить вероятность появления производственного стресса. Потому рассмотрим некоторые правила проектирования эргономических интерфейсов.

Компоненты навигации являются наиболее используемыми в системе, поэтому должны быть в зоне досягаемости пользователя и легкодоступны. Для соответствия этому требованию бар приложения содержит доступ к выдвижному меню (рисунок 31), а также при переходе между Activity отображает кнопку возвращения в предыдущие Activity, ведя историю посещений (рисунок 14).

Информационные сообщения и сигналы должны акцентировать внимание пользователя, чтобы избежать вероятности ошибки при работе с приложением. Все сообщения располагаются в одной области ввода текста, соответствуют смысловому содержанию и цветовой схеме (рисунки 13,29).

Расположение управляющих элементов должно быть согласовано и последовательно. Так логично располагать кнопку отправки формы внизу под формами ввода, поскольку, заполнив их, пользователь окажется именно там. Кнопки управления инструментами рисования не должны пропадать из поля зрения при их динамическом изменении, должны оставаться на одном месте, чтобы пользователь интуитивно знал, куда обратить свое внимание (рисунки 13,15).

Выбор визуальных эффектов должен соответствовать их функциональному назначению, так кнопка при нажатии должна менять цвет, чтобы обратить на себя внимание и предупредить неосознанные и необратимые действия (рисунок 33).

Соблюдение подобных правил позволит пользователю приложения сконцентрироваться на непосредственной работе, чем повысит ее эффективность.

От цветового оформления проекта зачастую зависит визуальный опыт работы с приложением. Так как работа за ПК схожа с работой за мобильным девайсом, в плане нагрузки для глаз, то, согласно [3], она характеризуется тем, что экран устройства является источником света. Отсюда вытекают некоторые особенности.

Во-первых, важен правильный подбор яркости и контрастности элементов дизайна. Так тусклые цвета заставляют пользователя напрягать зрение, а яркие «режут» глаз.

Во-вторых, содержимое экран устройства периодически изменяется. Частая и резкая смена цветов мешает восприятию, развеивает внимание и негативно влияет на работу мозга.

В-третьих, работая за монитором, пользователь сильно концентрируется на содержимом экран устройства, что приводит к большим нагрузкам на зрительную систему пользователя.

Так же стоит отметить, что в теории цвета существует разделение на теплые (красный, желтый), холодные (синий, голубой) и нейтральные (серый) цвета.

В соответствие с этими принципами один из способов построения цветовой палитры интерфейса выглядит следующим образом. Для фоновых элементов и элементов, занимающих наибольшую площадь интерфейса, следует использовать нейтральные или неяркие холодные цвета. Для функциональных и управляющих элементов, с которыми взаимодействует пользователь, стоит использовать яркие холодные или теплые цвета. Для выделения важных компонентов или определения их текущего состояния, возможно использование ярких теплых оттенков или теней, но количество таких элементов должно быть минимально.

Тогда сформируется концепция, позволяющая пользователю не напрягать свое зрение, работая с основным содержимым, функциональные компоненты всегда будут выделены, а при необходимости внимание будет акцентировано на важном элементе.

Данной концепции придерживается разрабатываемое приложение, как упоминалось в пункте 2.4, в качестве основного концепта дизайне используется Material Design, который полностью соответствует требованиям, описанным выше.

Принципы, используемые в типографии, основаны на многовековом опыте верстки печатных изданий. Для интерфейсов приложений характерно важно заимствование этих принципов.

Слишком маленький шрифт перенапрягает зрение пользователя, поэтому от него лучше отказаться. Слишком большой распыляет внимание и заставляет пользователя читать его несколько раз, что снижает эффективность работы.

Определение размеров шрифтов, следует начинать с обычного блока длинного текста, далее для небольших блоков текста его стоит увеличить, для заголовков стоит использовать резко большее начертание.

Еще один аспект — это выбор шрифтов, так шрифты можно разделить на шрифты без засечек и с засечками. Первые характерны для вывода на экран, а вторые для печатных изданий. Шрифты без засечек стоит использовать, когда:

- монитор низкого разрешения;
- читатели – дети;
- используется большое количество неконтрастных цветов;
- текст маленький или узкий.

Это позволит избежать резких пиксельных граней, сливания текста с фоном или слипания букв. В данном приложении для каждого разрешения экрана используются динамически подбираемый подходящий размер стандартного шрифта Android.

5.2 Экологичность

Данный аспект рассмотрим с точки зрения сбора и утилизации отходов ЭВМ, составные части, вспомогательное оборудование и оргтехники. ФЗ № 89 от 24.06.1998 является основным в вопросах регулирования обращения с отходами производства и потребления с целью предотвращения вредного воздействия отходов на здоровье человека и окружающую среду.

В данном НПА определяется разделение отходов на классы опасности. Всего определенно 5 классов опасности. В Федеральным Классификационным Каталогом Отходов выделены отдельные технические средства и их комплектующие. Согласно [10], системный блок компьютера определяется, как изделие из нескольких материалов, и имеет класс опасности – IV (малоопасные отходы). Аккумуляторы ноутбуков имеют класс опасности – II (высокоопасные отходы) и т.д.

В целом утилизация ЭВМ комплексный и сложный процесс, поэтому его стоит рассмотреть с разных сторон.

Во-первых, наиболее простым этот процесс представляется для физических лиц. Им необходимо обратиться в организацию, занимающуюся непосредственно утилизацией отходов. Такие организации должны пройти обязательное лицензирование своей деятельности.

Для юридических лиц этот процесс намного сложнее. Он также основан на передаче вычислительных средств сторонней организации, но этому должен предшествовать этап списания аппаратного оборудования. Списание оборудования сопровождается оценкой их экологических свойств штатным или приглашенным экспертом, который составляет паспорт отходов оргтехники и компьютеров (вычислительной техники). Соответственно, организации выгоднее накапливать единицы непригодной в работе техники, а затем утилизировать ее в больших объемах. После этого она передается специализированной организации по утилизации.

Отдельно стоит отметить утилизацию информации на носителях и компонентах ЭВМ. Данным вопросом может заниматься как сторонняя орга-

низация, так и владелец техники. Способы и требования по уничтожению информации с носителей описываются в ГОСТ Р 50739-95, а также в РД от 30.03.1992 1 и 2, защита от НСД к информации. Согласно этим нормативам уничтожение может производиться, как с помощью блокирования доступа к информации на носителях, ее затиранию, а также дополнительным включением маскирующей информации.

Наличие в компонентах ЭВМ технического золота или других драгоценных металлов накладывает на организацию дополнительную ответственность. Эти аспекты регулируются законодательством в соответствие с ФЗ №41. Несоблюдение данных требований может повлечь административную ответственность.

5.3 Защита от ЧС

Помещения, в которых происходит работа с ЭВМ, относят к категории В – пожароопасные помещения, согласно СП 12.13130.2009. Проблема обеспечения противопожарной безопасности в них является одной из основополагающих при рассмотрении аспектов БЖД.

Специфика эксплуатации ЭВМ подразумевает наличие большого количества электрических приборов, токопроводящих кабелей и высоких нагрузок на электросеть. Поэтому их установка, эксплуатация, техническое обслуживание, проверка, замена и утилизация должны соответствовать принятым законодательным нормам и стандартам.

При расположении ЭВМ необходимо учитывать не только их расположение внутри помещения, но взаимодействия друг с другом, а также расположение смежных помещений. Так, например, площадь одного рабочего места с ПК для взрослого должна составлять не менее 6 $м^2$, а объем не менее 20 $м^3$. Для хранения носителей информации, расходных и комплектующих частей ЭВМ или оргтехники, необходимо оборудовать соответствующее помещение, оборудованных негорящими стеллажами и шкафами.

Хранение технических средств должно осуществляться в закрытых контейнерах для предотвращения накопления пыли в их составных частях.

При эксплуатации ЭВМ и оргтехники необходимо проверять целостность токопроводящих кабелей, вилки и розетки, отсутствие повреждений аппаратуры.

Компоненты ЭВМ должны иметь функцию самоотключения при повышении температуры входе неисправности систем охлаждения и кондиционирования. Для предотвращения перегрева.

При работе электроприбором возможно образование статических зарядом на корпусах ЭВМ, периферии и оргтехники. Такие разряды могут привести к выводу из техники строя. Для их предотвращения необходимо использовать антистатическое покрытие полов, увлажнители воздухов и т.д.

Так же в помещениях, оборудованных ЭВМ, необходима установка средств пожаротушения. К таким средствам относятся огнетушители различных конструкций: порошковые (ПСБ, ПФ, ОП), пенные (ОХП- 10), углекислотные (ОУ-2, ОУ-5). Так же распространение получили установки водяного, пенного и газового пожаротушения.

Для оповещения посетителей и работников помещения при возникновении пожар следует устанавливать средства пожарной сигнализации.

Технические средства должны проходить проверки и техническое обслуживание. Так необходимо проверять работоспособность, целостность и другие рабочие характеристики. Необходимо проводить уборку и очистку этих устройств. Так для удаления пыли и пятен должны применяться негорючие жидкости и материалы.

5.4 Комплексы физических управлений для сохранения и укрепления индивидуального здоровья и обеспечения полноценной профессиональной деятельности

Для предупреждения развития переутомления за работой ПЭВМ, согласно СанПиНу 2.2.2/2.4.1340-03, обязательными мероприятиями являются:

- проведение упражнений для глаз через каждые 20 - 25 мин работы за ПЭВМ;

- проведение упражнений физкультминутки в течение 1 - 2 мин для снятия локального утомления, которые выполняются индивидуально при появлении начальных признаков усталости.

Упражнения для глаз выполняются сидя или стоя, отвернувшись от экрана при ритмичном дыхании, с максимальной амплитудой движения глаз и выполнить следующие действия. Закрыть глаза, сильно напрягая глазные мышцы, на счет 1 - 4, затем раскрыть глаза, расслабив мышцы глаз, посмотреть вдаль на счет 1 - 6. Повторить эти действия 4 - 5 раз. Посмотреть на переносицу и задержать взор на счет 1 - 4. До усталости глаза не доводить. Затем открыть глаза, посмотреть вдаль на счет 1 - 6. Повторить эти действия 4 - 5 раз. Не поворачивая головы, посмотреть направо и зафиксировать взгляд на счет 1 - 4, затем посмотреть вдаль прямо на счет 1 - 6. Аналогичным образом проводятся упражнения, но с фиксацией взгляда влево, вверх и вниз. Повторить эти действия 3 - 4 раза. Перенести взгляд быстро по диагонали: направо вверх - налево вниз, потом прямо вдаль на счет 1 - 6; затем налево вверх направо вниз и посмотреть вдаль на счет 1 - 6. Повторить эти действия 4 - 5 раз.

Физкультминутка способствует снятию локального утомления. Для проведения эффективного восстановления необходимо выполнить следующие упражнения. Наклоны и повороты головы оказывают механическое воздействие на стенки шейных кровеносных сосудов, повышают их эластичность; раздражение вестибулярного аппарата вызывают расширение кровеносных сосудов головного мозга. Дыхательные упражнения, особенно дыхание через нос, изменяют их кровенаполнение. Все это усиливает мозговое кровообращение, повышает его интенсивность и облегчает умственную деятельность. Динамические упражнения с чередованием напряжения и расслабления отдельных мышечных групп плечевого пояса и рук, улучшают кровоснабжение, снижают напряжение.

Таким образом, в данном разделе мы рассмотрели основные вопросы, связанные обеспечением БЖД при использовании ЭВМ. Подробны рассмот-

рели темы эргономичного проектирования интерфейсов взаимодействия с пользователем, проблемы утилизации ЭВМ, ее компонентов и вспомогательной техники, а также комплексы физических упражнений, вопросы обеспечения защиты от ЧС.

ЗАКЛЮЧЕНИЕ

Произведенные работы в ходе выполнения выпускной квалификационной работы позволили создать новый сервис для общения в интернете, состоящий из клиента Android приложения и серверной части.

Этапы по анализу предметной области, проектированию клиентской и серверной части, разработке программного приложения, исследования вопросов информационной безопасности приложения, канала связи и рассмотрении аспектов безопасности жизнедеятельности определили следующие результаты:

- исследована предметная область;
- проанализированы, дополнены и оформлены требования к будущему приложению мессенджеру;
- рассмотрены технологии разработки и выбраны наиболее подходящие из них;
- разработан проект клиентской БД;
- разработан проект серверной БД;
- спроектированы серверная и клиентская составляющие приложения;
- выполнена реализация БД в СУБД;
- согласно проектному описанию разработано приложение на языках программирования;
- описан тестовый пример работы системы;
- исследованы угрозы ИБ, а также меры противодействия им;
- рассмотрены вопросы проектирования эргономичного графического интерфейса пользователя, утилизации ПК, их компонентов, оргтехники и комплектующих, а также вопросы противопожарной безопасности при работе с ЭВМ.

В совокупности были выполнены все поставленные для данной выпускной квалификационной работы задачи.

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1 Википедия [Электронный ресурс] : офиц. сайт. – 15.01.2001 – Режим доступа: https://ru.wikipedia.org/wiki/Фреймворк. – 10.05.2018.

2 Шаньгин В.Ф. Информационная безопасность и защита информации [Электронный ресурс] / В.Ф. Шаньгин – Электрон. текстовые данные. – Саратов: Профобразование, 2017. – 702 с. – Режим доступа: http://www.iprbookshop.ru/63594.html. – ЭБС «IPRbooks». – 6.05.2018.

3 Симакова, Н.Н. Организация рабочих мест с персональными электронно-вычислительными машинами (ПЭВМ) [Электронный ресурс]: учебное пособие / Н.Н. Симакова – Электрон. текстовые данные. – Новосибирск: Сибирский государственный университет телекоммуникаций и информатики, 2013. – 78 с. – Режим доступа: http://www.iprbookshop.ru/55486.html. – ЭБС «IPRbooks». – 12.05.2018.

4 Distribution dashboard [Электронный ресурс]. – Режим доступа https://developer.android.com/about/dashboards/, свободный. – Загл. с экрана. – 12.05.2018.

5 Википедия MongoDB [Электронный ресурс]. – Режим доступа: https://ru.wikipedia.org/wiki/MongoDB, свободный. – Загл. с экрана. – 13.05.2018.

6 HTTP Over TLS [Электронный ресурс]. – Режим доступа: https://tools.ietf.org/html/rfc2818#section-2.3, свободный. – Загл. с экрана. – 05.05.2018.

7 Material Design [Электронный ресурс]. – Режим доступа: https://www.engadget.com/2014/06/25/googles-new-design-language-is-called-material-design/],, свободный. – Загл. с экрана. – 29.04.2018.

8 Двухуровневая конфигурация [Электронный ресурс]. – Режим доступа: https://dev.1c-bitrix.ru/learning/course/index.php? COURSE_ID =32&CHAPTER_ID=04900&LESSON_PATH=3903.4897.4900, свободный. – Загл. с экрана. – 09.05.2018.

9 Википедия Node.js [Электронный ресурс]. – Режим доступа: https://ru.wikipedia.org/wiki/Node.js, свободный. – Загл. с экрана. – 29.04.2018.

10 Федеральный Классификационный Каталог Отходов [Электронный ресурс]. – Режим доступа: http://eco-c.ru/guides/fkko, свободный. – Загл. с экрана. – 29.04.2018.

11 WebSocket [Электронный ресурс]. – Режим доступа : https://ru.wikipedia.org/wiki/WebSocket, свободный. – Загл. с экрана. – 29.04.2018.

12 Качановский, Ю.П. Основные технические, программные и организационные меры защиты информации при работе с компьютерными системами [Электронный ресурс]: методические указания к проведению лабораторной работы по курсу «Информатика»/ Ю.П. Качановский, А.С. Широков – Электрон. текстовые данные. – Липецк: Липецкий государственный технический университет, ЭБС АСВ, 2014. – 24 с. – Режим доступа: http://www.iprbookshop.ru/55120.html. – ЭБС «IPRbooks». – 29.04.2018.

13 Information Security/ Информационная безопасность [Электронный ресурс]: офиц. сайт. – 2003 – Режим доступа: http://www.itsec.ru/articles2/Oborandteh/bezopasnost-web-prilozhenij/. – 14.05.2018.

14 Android Security [Электронный ресурс]: офиц. Сайт. –Режим доступа: https://source.android.com/security. – 20.05.2018.

15 Android Developer [Электронный ресурс]: офиц. Сайт. –Режим доступа: https://developer.android.com / index.html. – 20.04.18.

16 Википедия [Электронный ресурс]: офиц. сайт. – 15.01.2001 –Режим доступа: https://en.wikipedia.org/wiki/Android_(operating_system). - 10.04.18.

17 Documentation SQLite [Электронный ресурс]. – Режим доступа: https://www.sqlite.org/docs.html, свободный. – Загл. с экрана. – 10.04.18.

ПРИЛОЖЕНИЕ А

Диаграммы классов Activities и Services

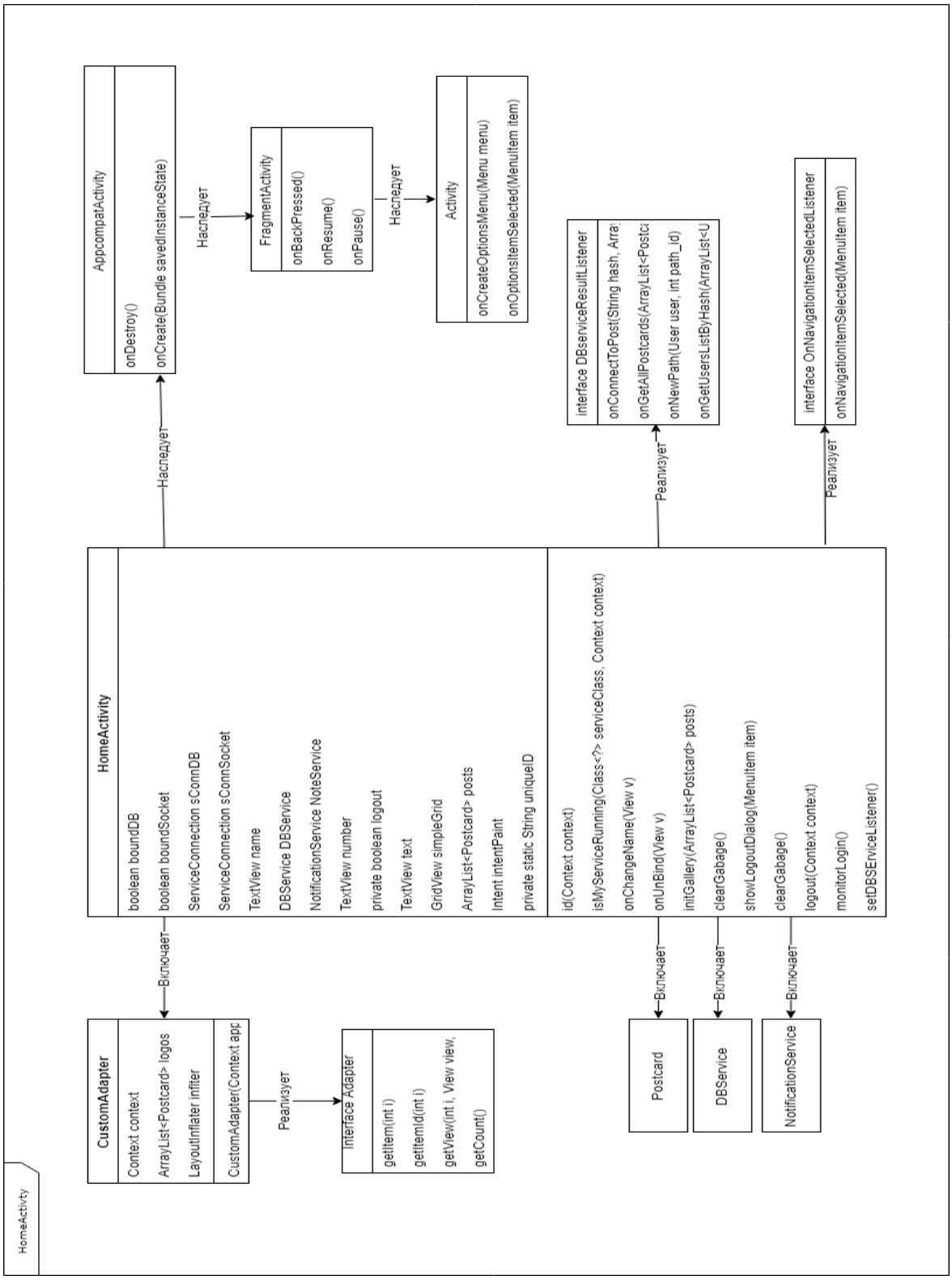


Рисунок А.1 – Диаграмма классов HomeActivity

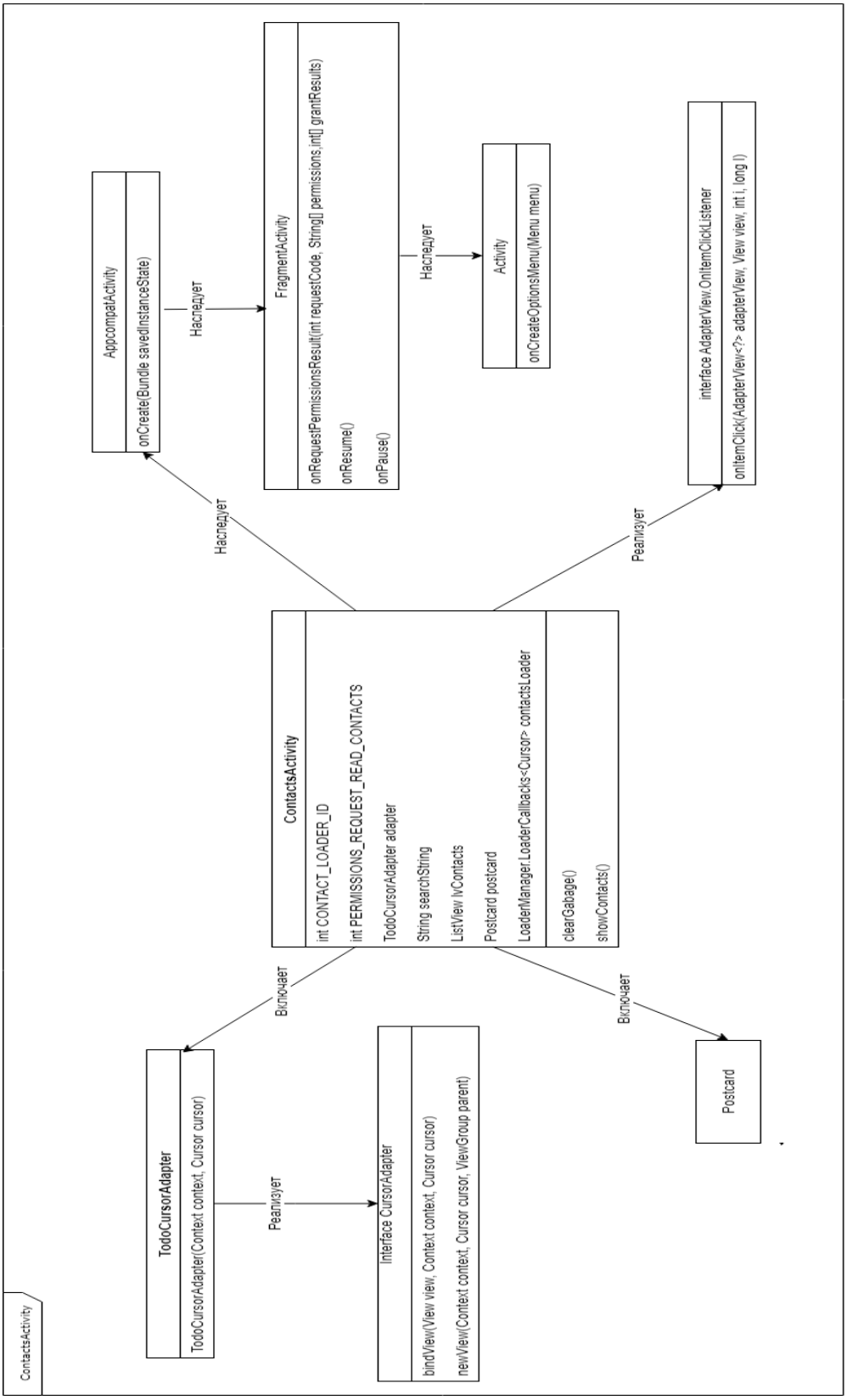


Рисунок А.2 – Диаграмма классов ContactsActivity

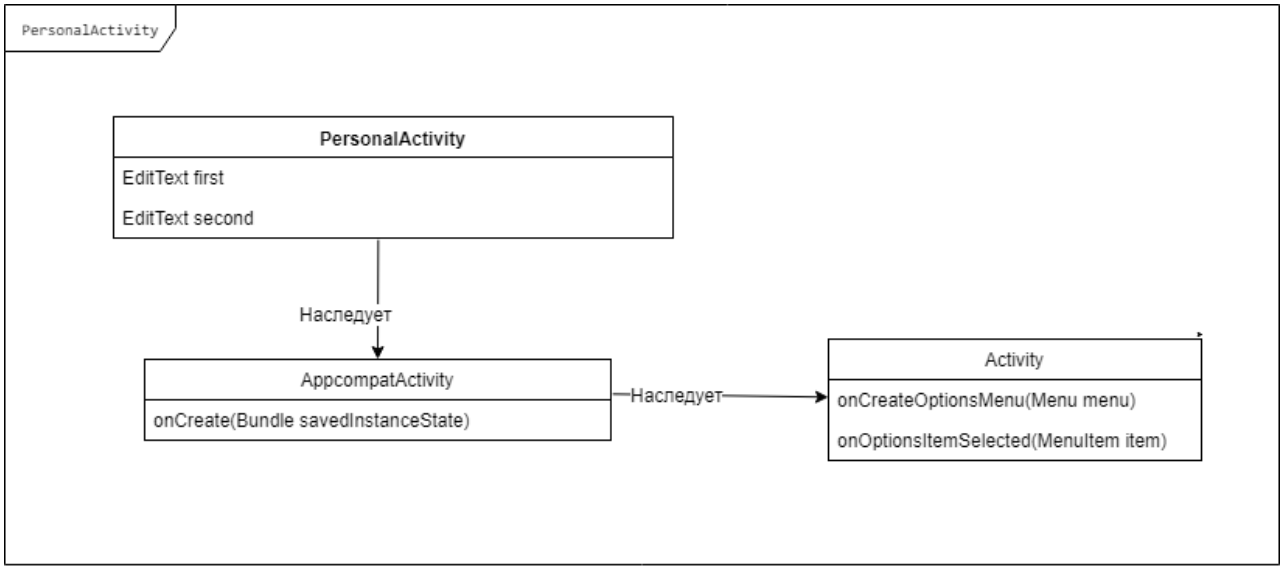


Рисунок А.3 – Диаграмма классов PersonalActivity

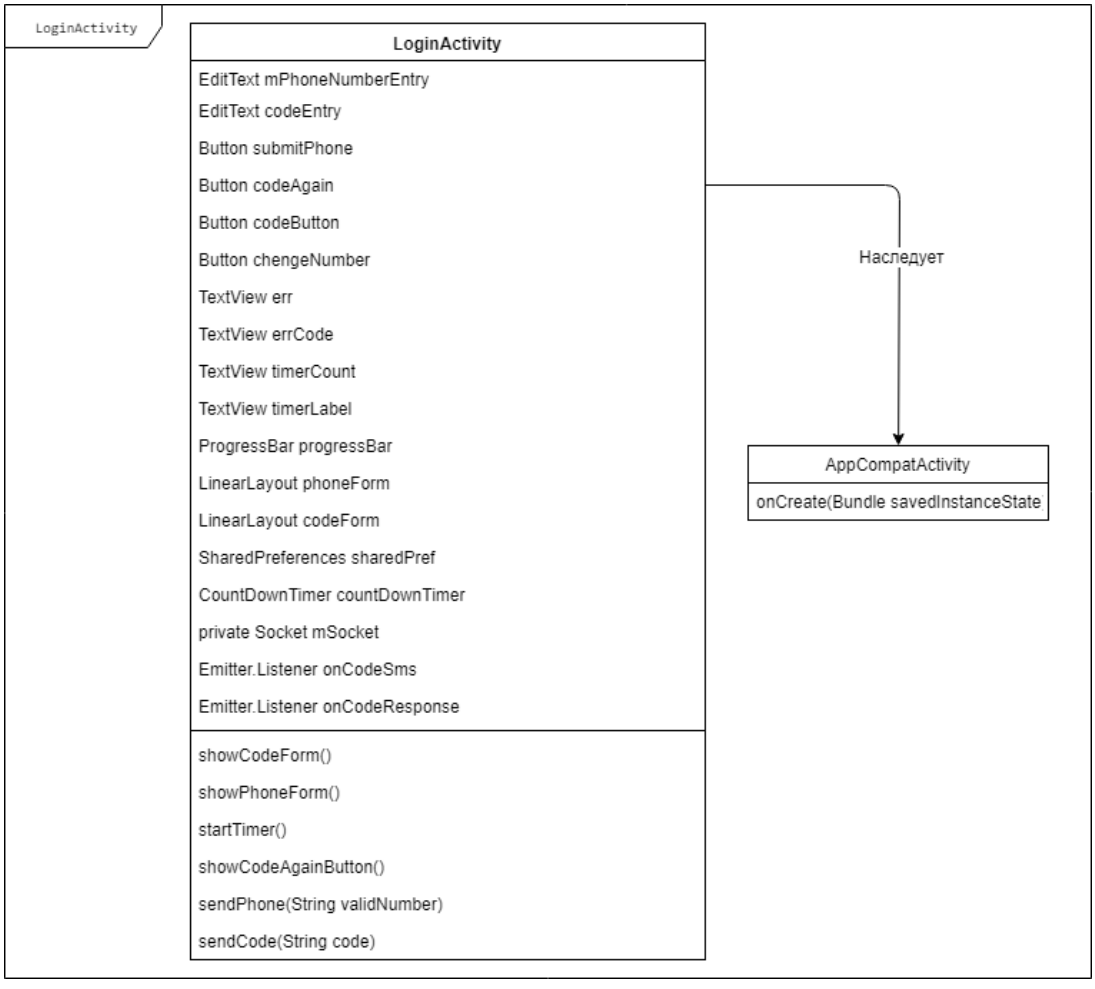


Рисунок А.4 – Диаграмма классов LoginActivity

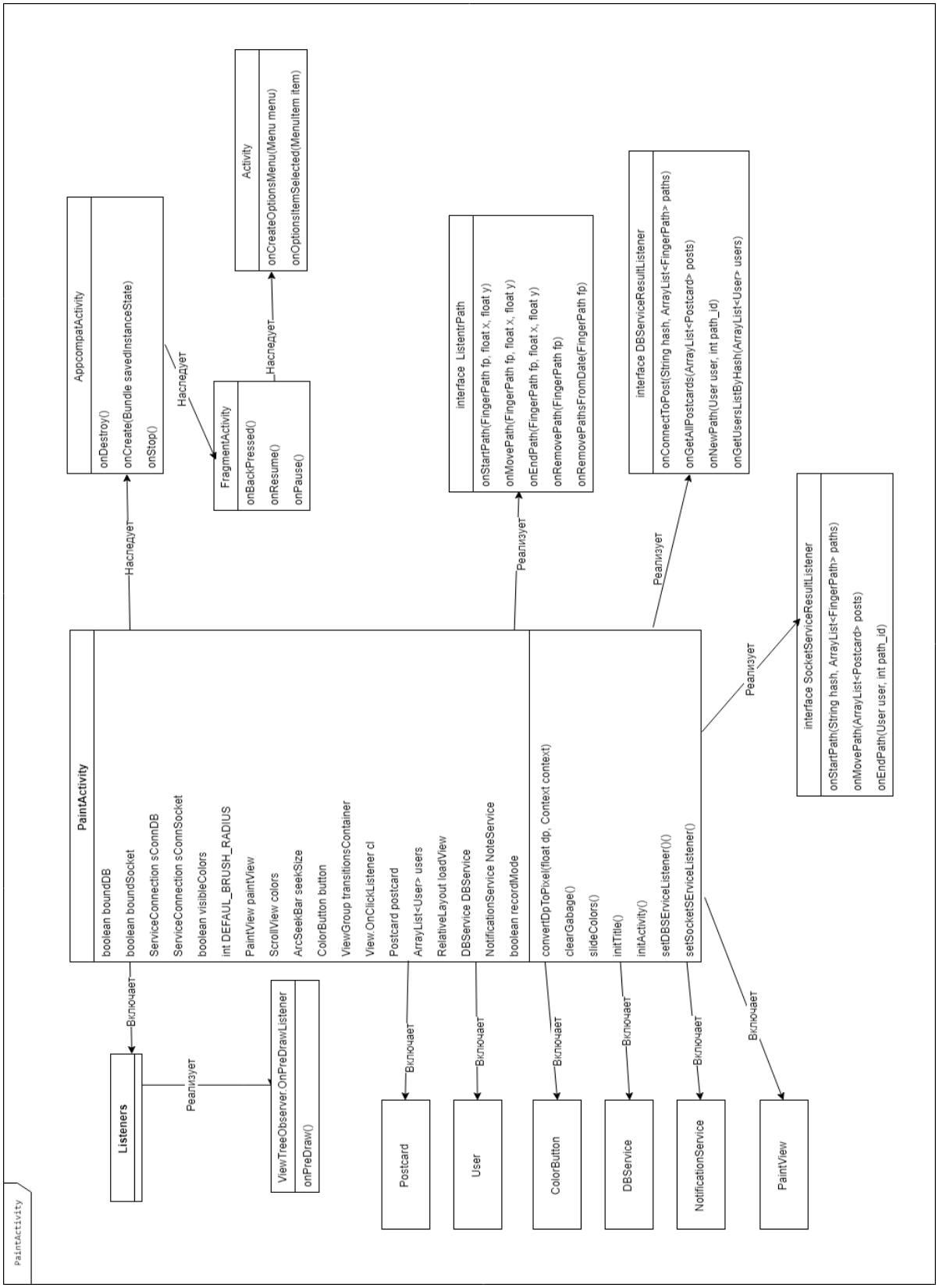


Рисунок А.5 – Диаграмма классов PaintActivity

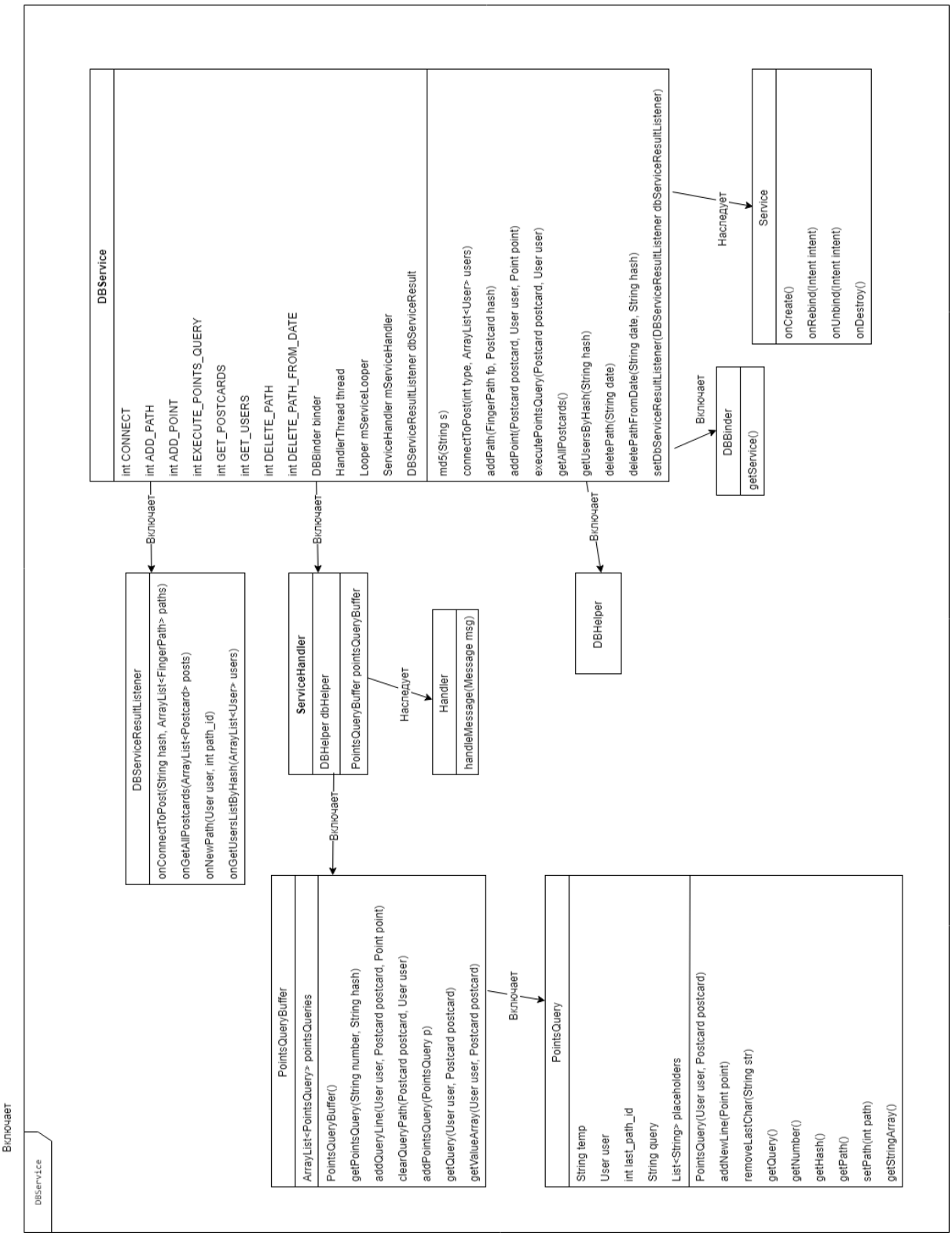


Рисунок А.6 – Диаграмма классов DBService

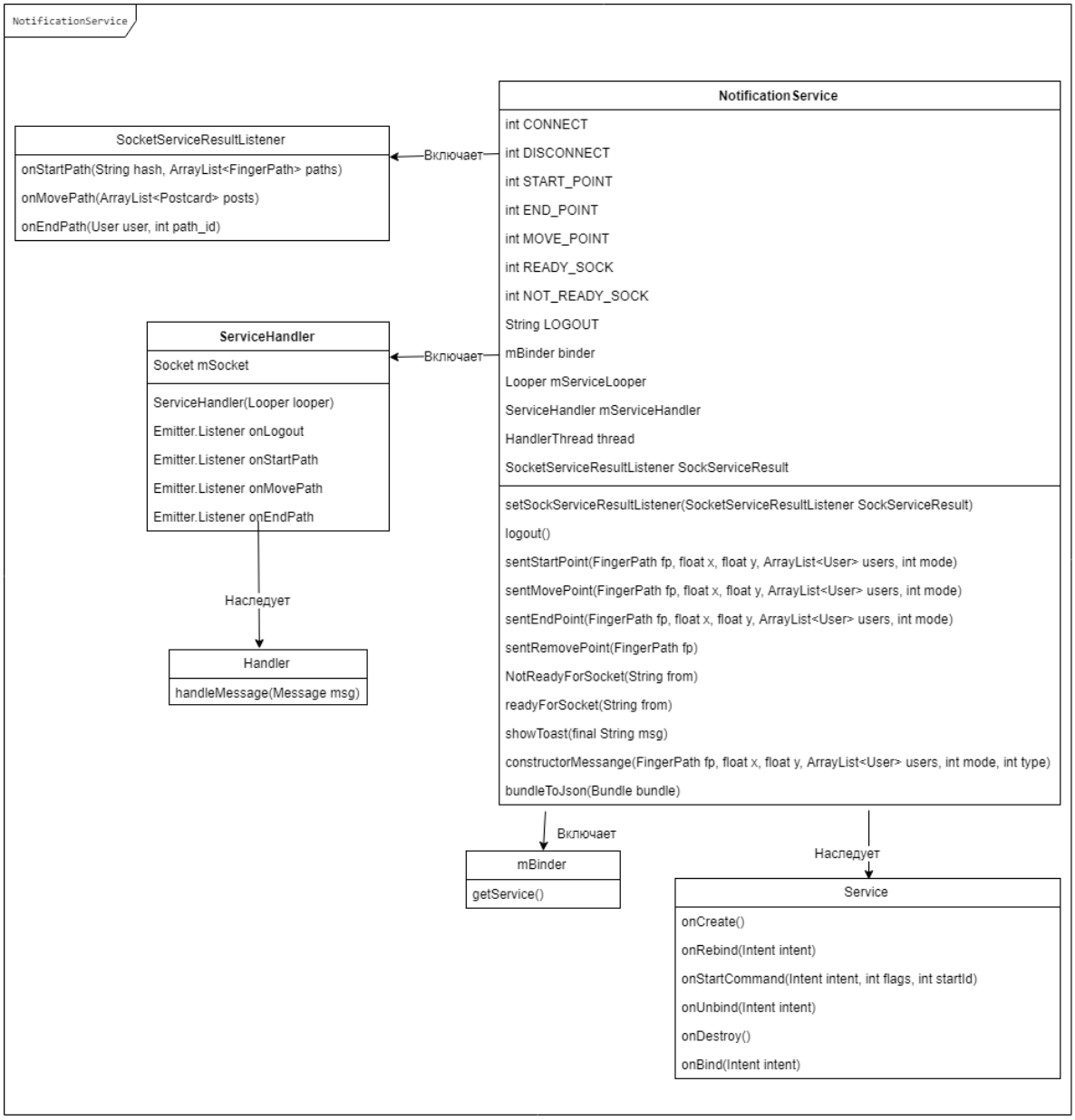


Рисунок А.7 – Диаграмма классов NotificationService

ПРИЛОЖЕНИЕ Б

Диаграммы вспомогательных классов

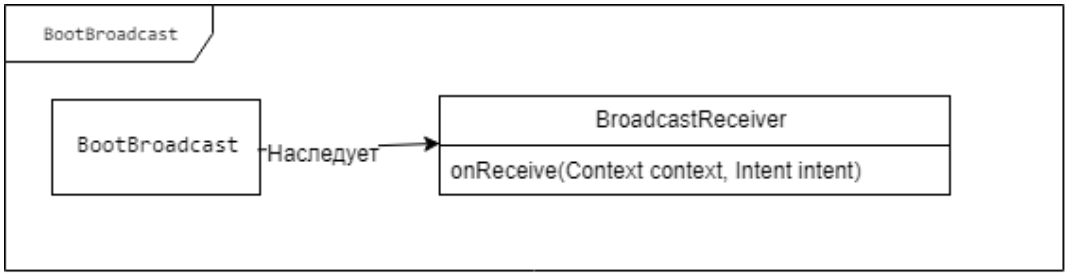


Рисунок Б.1 – Диаграмма класса BootBroadcast

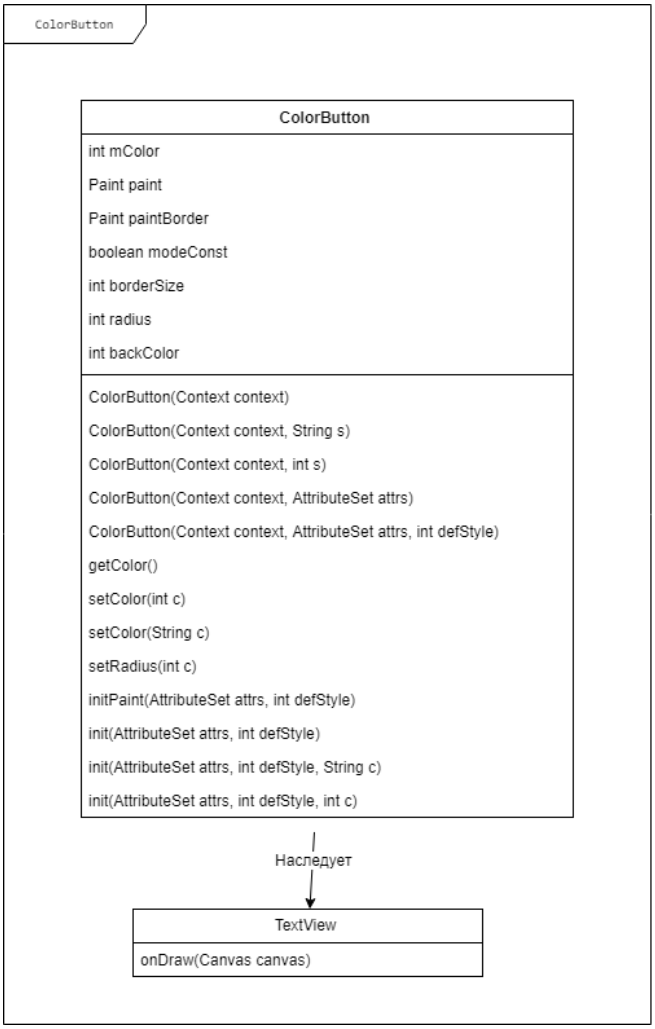


Рисунок Б.2 – Диаграмма класса ColorButton

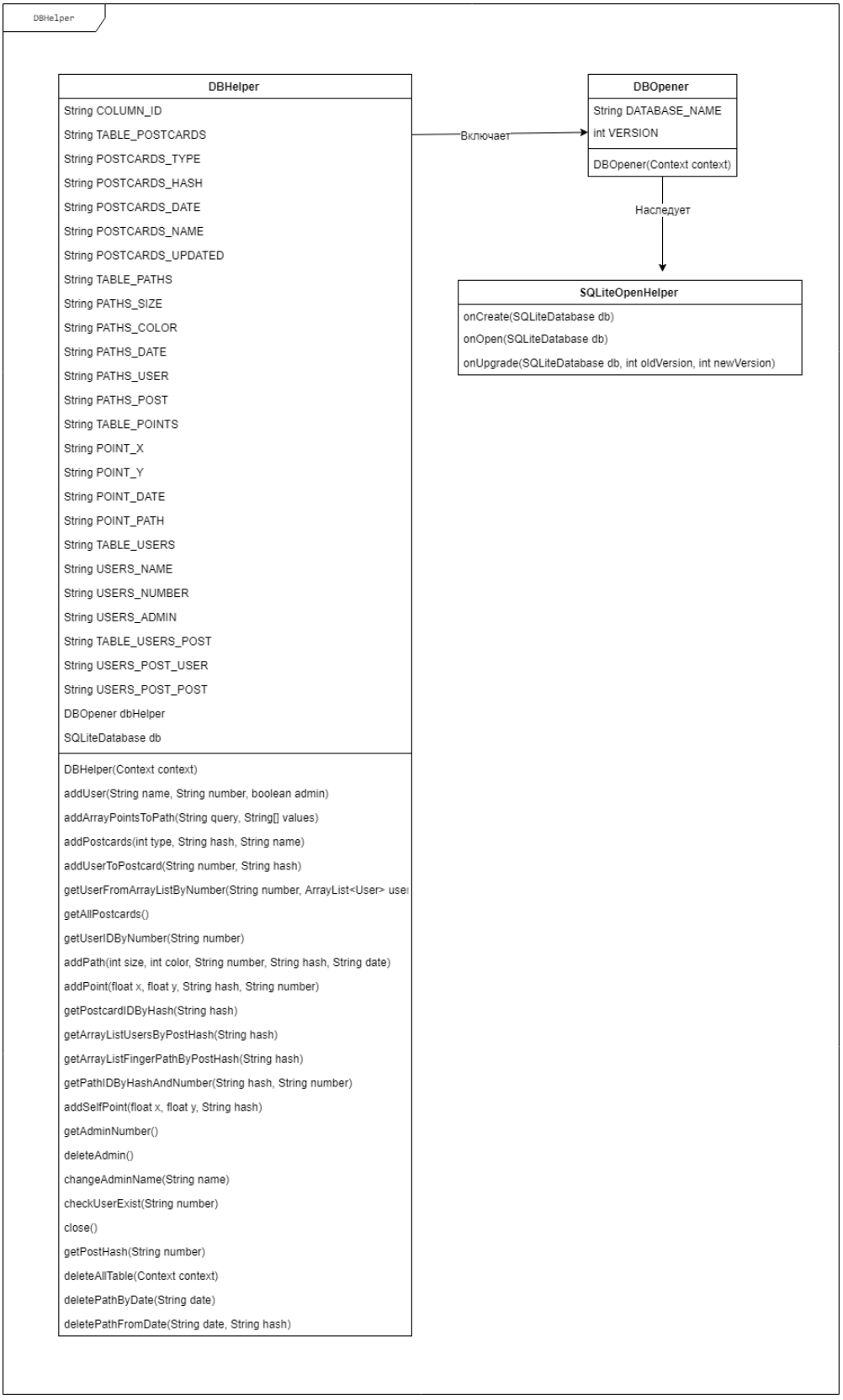


Рисунок Б.3 – Диаграмма класса DBHelper

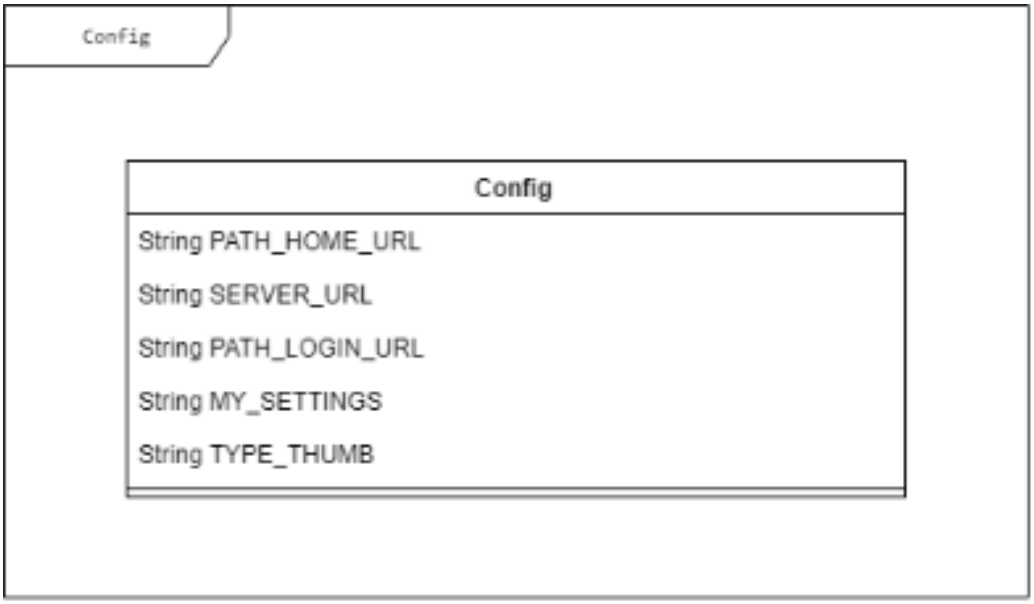


Рисунок Б.4 – Диаграмма класса Config

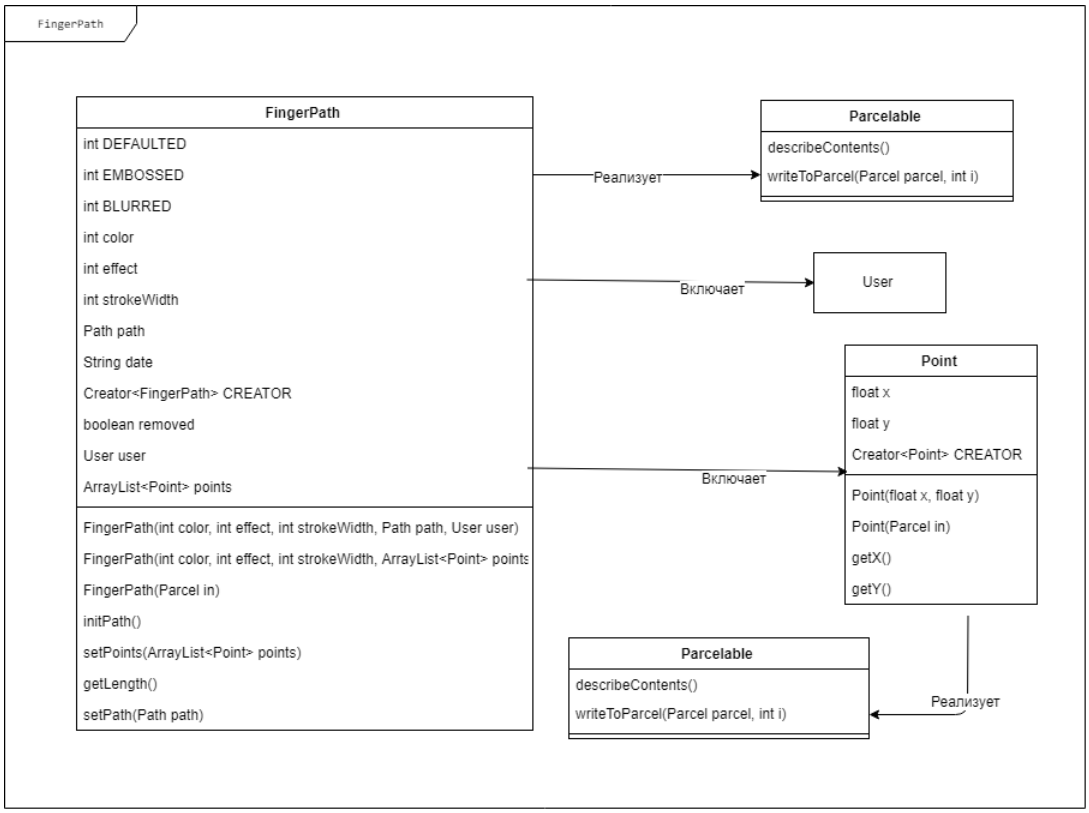


Рисунок Б.5 – Диаграмма класса FingerPath

PaintView	PaintView
PaintView(Context context)	int DEFAULT_COLOR
PaintView(Context context, AttributeSet attrs)	int DEFAULT_CARD_COLOR
setPaths(ArrayList<FingerPath> paths)	int DEFAULT_BG_COLOR
getPaths()	int DEFAULT_SHADOW_COLOR
setUsers(ArrayList<User> users)	float DEFAULT_PADDING
setPathListener(ListenerPath listenerPath)	int DEFAULT_SHIFT_SHADOW
init(int brushColor, int size, int cardCol, int backgroundColor, float allov	float TOUCH_TOLERANCE
getXMLAttributeSet(AttributeSet attrs)	int WIDTH_CARD
init()	int HEIGHT_CARD
setSeekForHistory(ArcSeekBar a)	float RATIO_CARD
setTextForHistory(TextView a)	int DEFAULT_BRUSH_SIZE
setInitionPosition()	float maxScale
alignCenter()	int POINTER_INDEX
saveThumb(String fileName, Context context)	int TRANSLATE_SUCCSESS
getLastPathOfNumber(String number)	int TRANSLATE_FAIL_X
postTranslate(float x, float y, boolean autoAling)	int TRANSLATE_FAIL_XY
prev()	int TRANSLATE_FAIL_X_ALIGN
next()	int TRANSLATE_FAIL_Y
setBrushColor(int c)	int TRANSLATE_FAIL_YX
setCardColor(int c)	int TRANSLATE_FAIL_Y_ALIGN
setStrokeWidth(int c)	int TRANSLATE_SUCCSESS_OVER_SCALE
normal()	int TRANSLATE_SUCCSESS_OVER_FAIL_X
setMaxScale()	int TRANSLATE_SUCCSESS_OVER_FAIL_XY
setMinScale()	int TRANSLATE_SUCCSESS_OVER_FAIL_Y
setNormalScale()	int TRANSLATE_SUCCSESS_OVER_FAIL_YX
touchStart(float x, float y)	int TRANSLATE_SUCCSESS_OVER_FAIL_X_ALING
drawLastPath()	ListenerPath listenerPath
drawPath(FingerPath fp)	float WIDTH_VIEW
drawPath(FingerPath fp)	float mX, mY
updateCardMatrix()	boolean isScrolling
removeRemovedPath()	History history
touchMove(float x, float y)	GestureDetector mGestureDetector
touchUp()	...

Рисунок Б.6 – Диаграмма класса PaintView

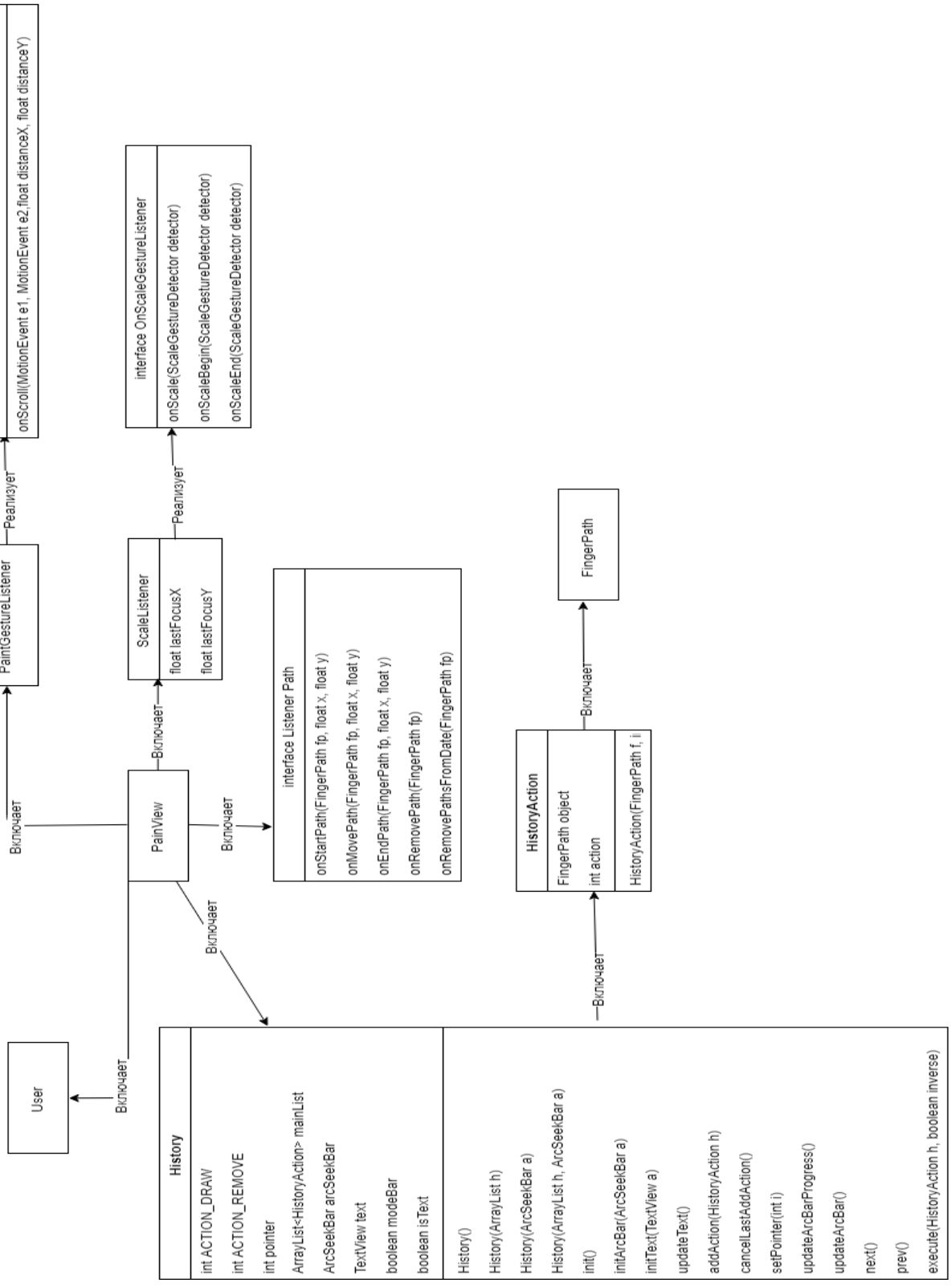


Рисунок Б.7 – Диаграмма классов PaintView

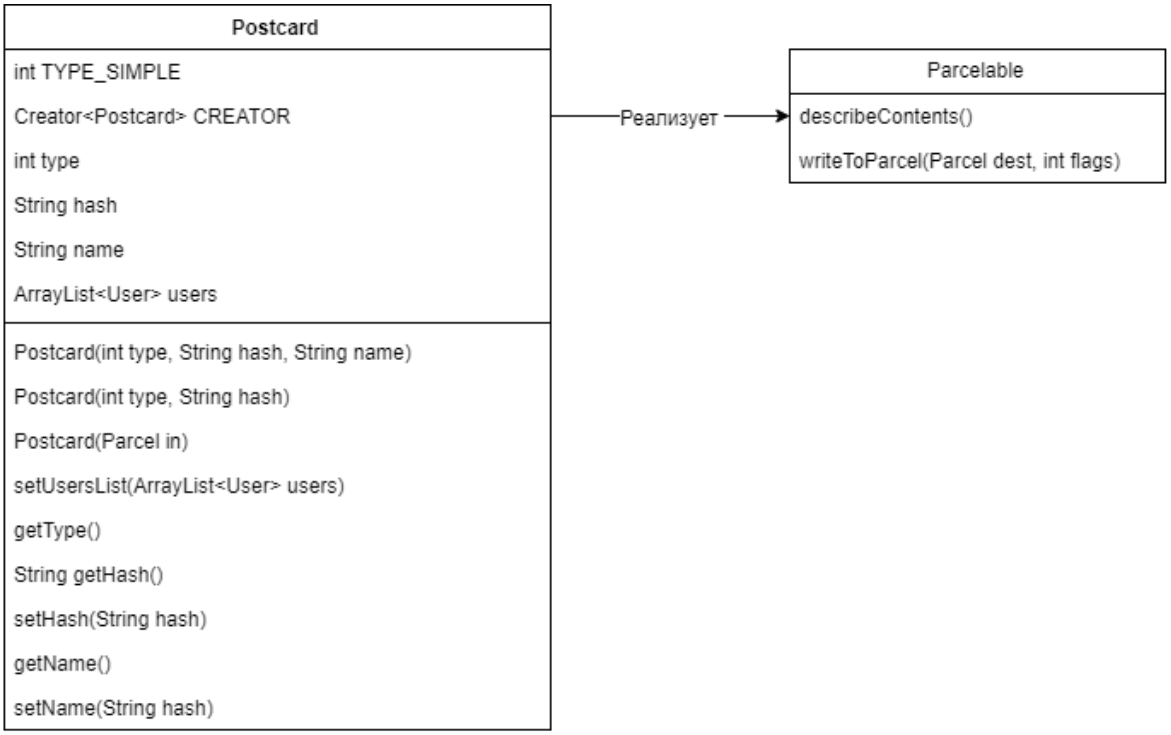


Рисунок Б.8 – Диаграмма классов Postcard

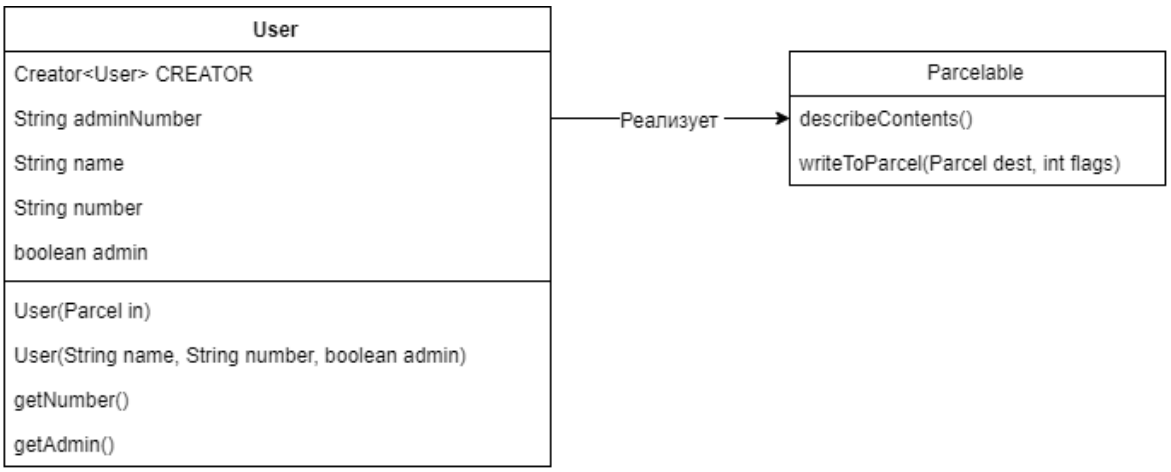


Рисунок Б.9 – Диаграмма классов User

ПРИЛОЖЕНИЕ В

Диаграмма «сущность-связь»

Рисунок В.1 – Диаграмма «сущность-связь»

ПРИЛОЖЕНИЕ Г

Диаграммы функциональных зависимостей

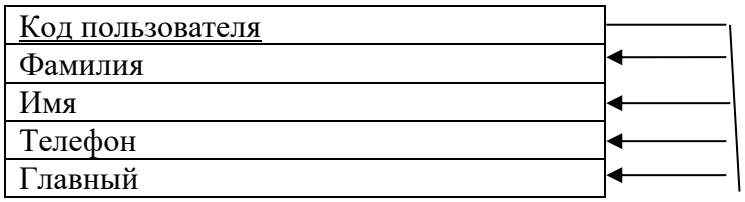


Рисунок Г.1 – Диаграмма функциональной зависимости атрибутов отношения Пользователь

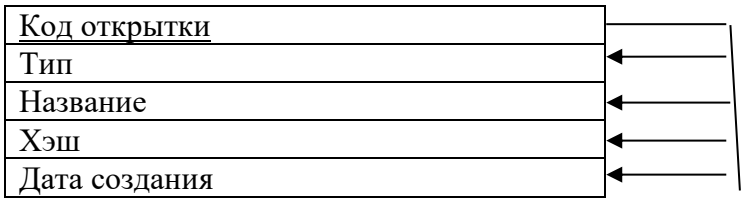


Рисунок Г.2 – Диаграмма функциональной зависимости атрибутов отношения Открытка

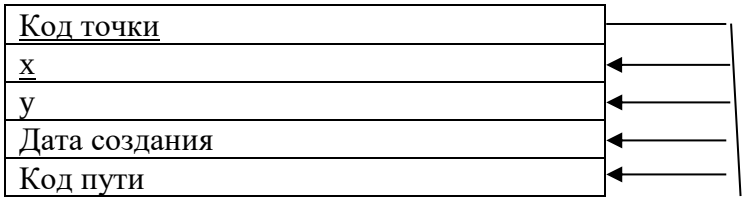


Рисунок Г.3 – Диаграмма функциональной зависимости атрибутов отношения Точка

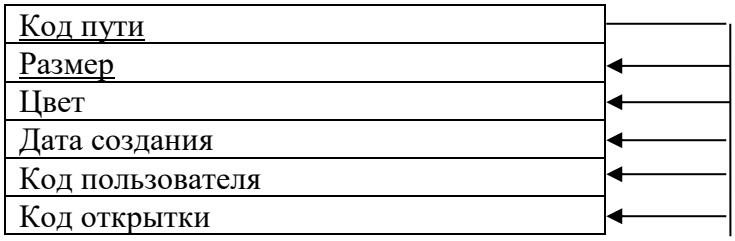


Рисунок Г.4 – Диаграмма функциональной зависимости атрибутов отношения Путь

ПРИЛОЖЕНИЕ Д

Реляционная модель отношений

Отношение 1 – Открытка

Код открытки	Название	Хэш	Дата создания
Обновление			

Отношение 2 – Пользователь

Код пользователя	Фамилия	Имя
Телефон	Главный	

Отношение 3 – Путь

Код пути	Размер	Цвет	Дата создания	**Код открытки**
Код пользователя				

Отношение 4 – Точка

Код точки	x	y	Дата создания	**Код пути**

Отношение 5 – Пользователь - Открытка

Код пользователя	**Код открытки**

ПРИЛОЖЕНИЕ Е

Логическая схема клиентской БД

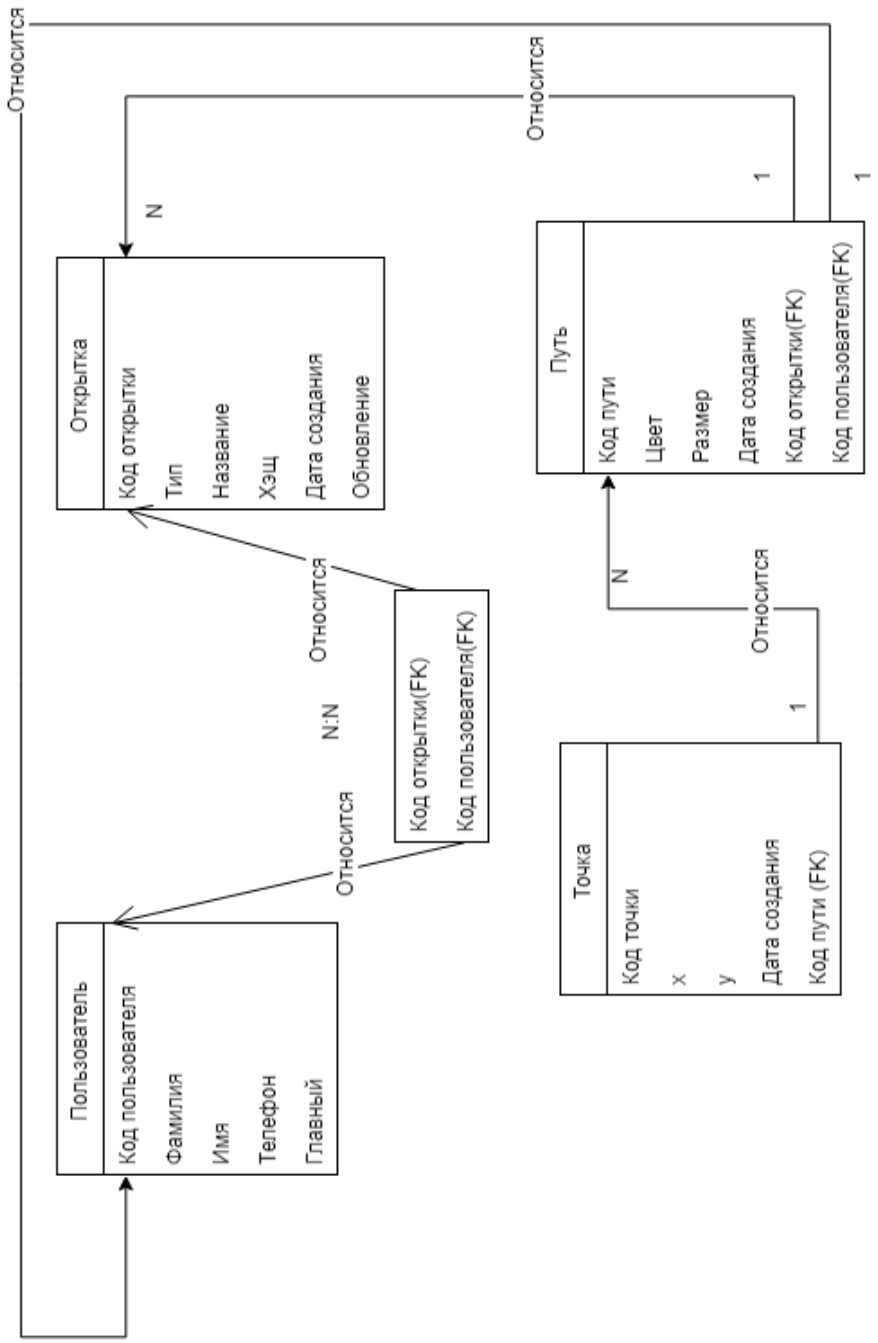


Рисунок Е.1 – Логическая схема клиентской БД

ПРИЛОЖЕНИЕ Ж

Физическая схема клиентской БД

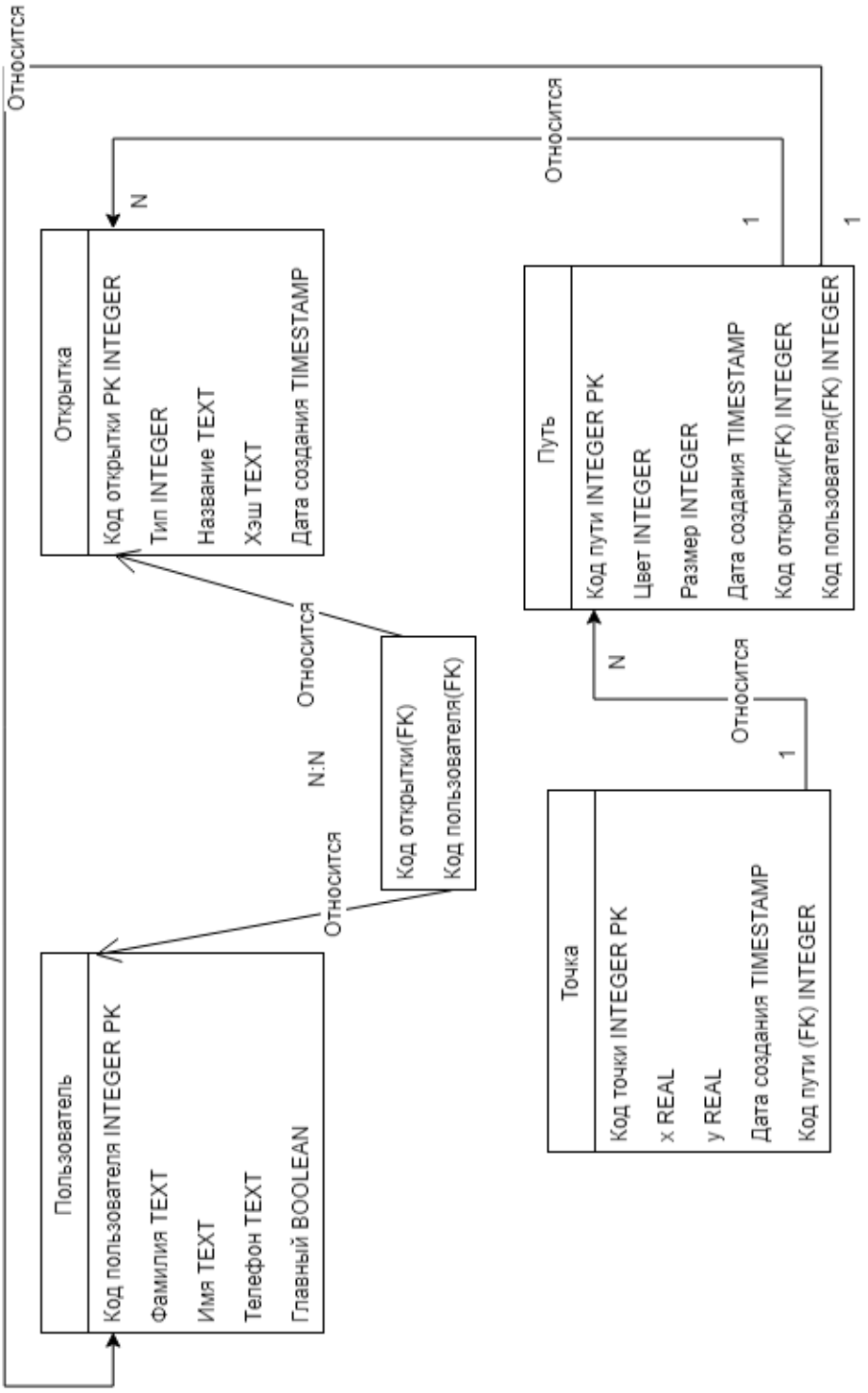


Рисунок Ж.1 – Физическая схема клиентской БД

ПРИЛОЖЕНИЕ И

Диаграмма объектов серверной БД

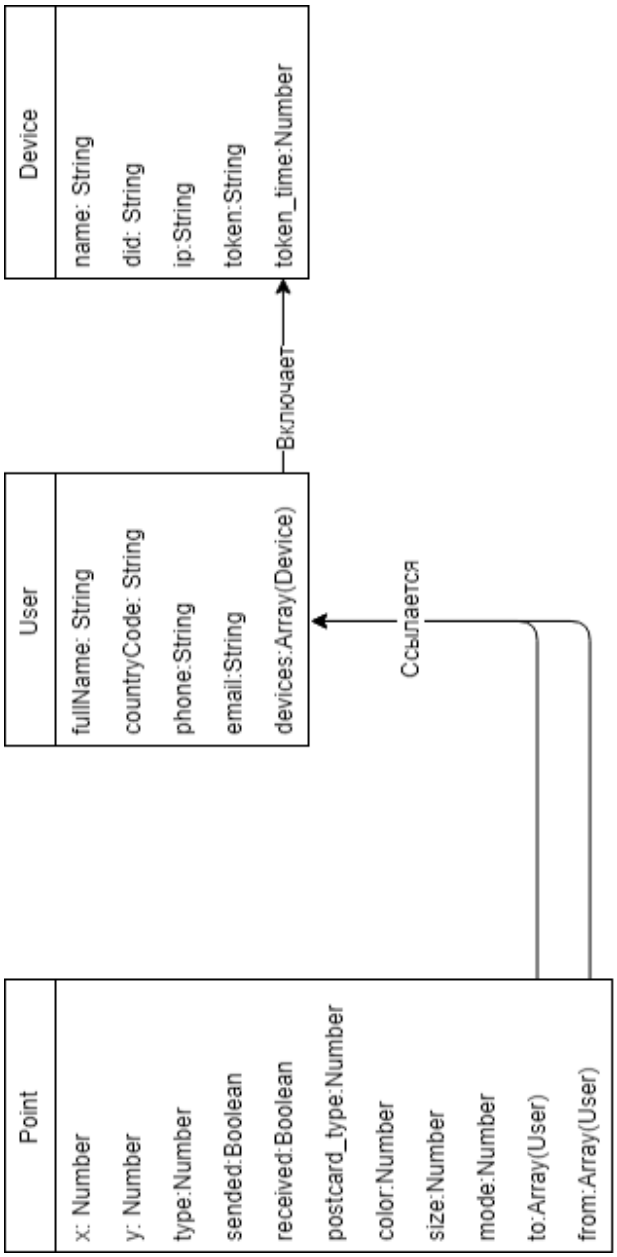


Рисунок И.1 – Структура серверной БД

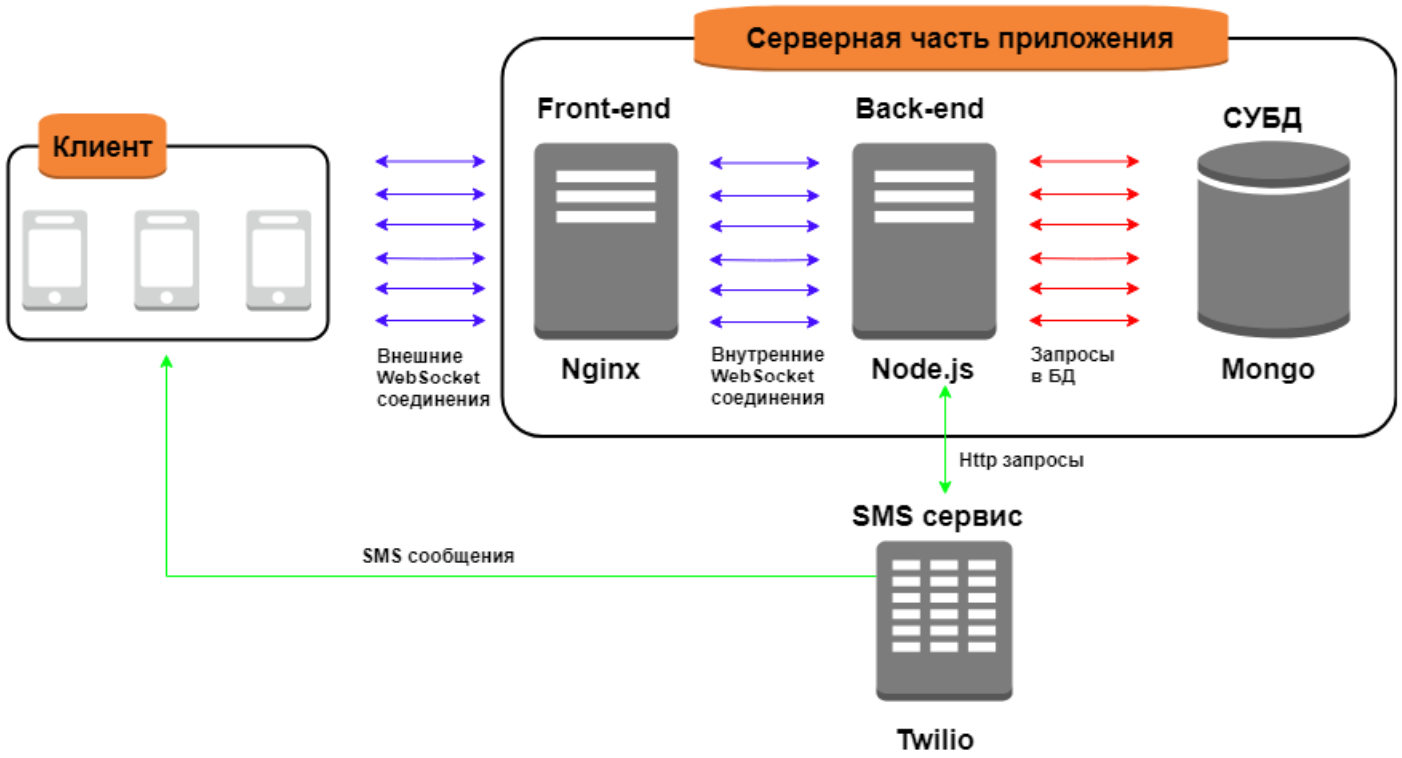


Рисунок 1 – Общая архитектура серверной части приложения

					ВКР.145328.09.03.02.СХ		
Изм.	Лист	№ докум.	Подп.	Дата			
Разраб.		Саяпин В.А.			РАЗРАБОТКА ANDROID ПРИЛОЖЕНИЯ МЕССЕНДЖЕР	Лит.	Масса / Масштаб
Пров.		Бушманов А.В.				У	
Консульт.		Булавков А.Б.					
Н.контр.		Романико В.В.				Лист 1	Листов 8
Зав.каф.		Бушманов А.В.				АмГУ Кафедра ИУС	

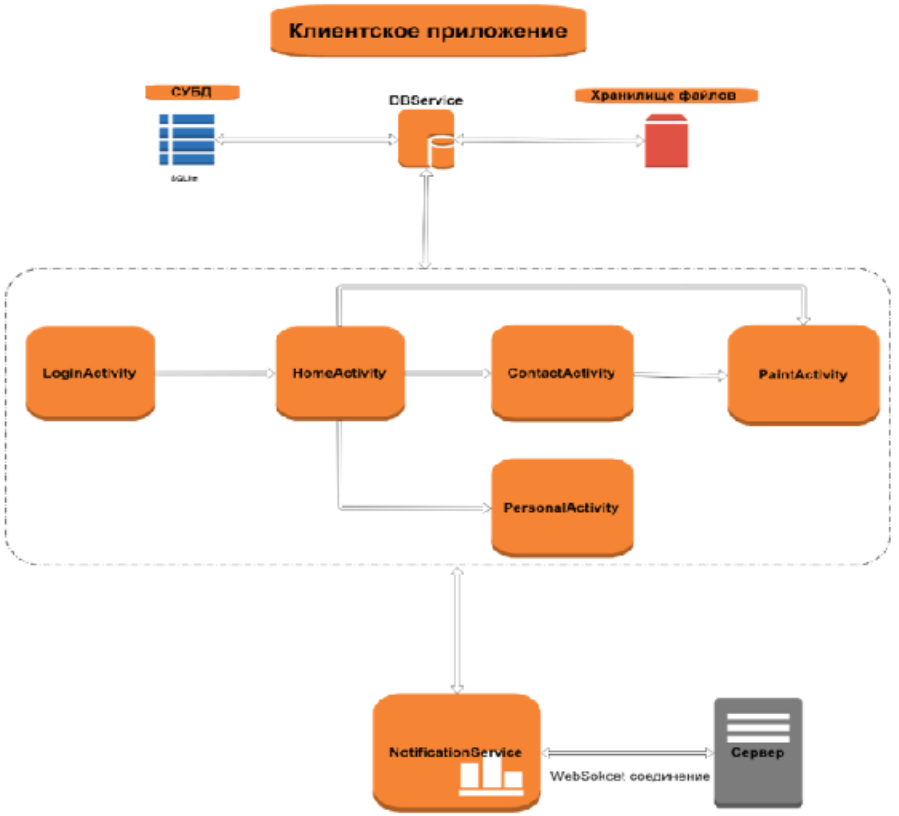


Рисунок 2 – Архитектура клиентского приложения
с указание всех действующих компонентов

					ВКР.145328.09.03.02.СХ			
Изм.	Лист	№ докум.	Подп.	Дата		Лит.	Масса	Масштаб
Разраб.		Саяпин В.А.			РАЗРАБОТКА ANDROID ПРИЛОЖЕНИЯ МЕССЕНДЖЕР	У		
Пров.		Бушманов А.В.						
Консульт.		Булавков А.Б.				Лист 2	Листов 8	
Н.контр.		Романико В.В.				АмГУ Кафедра ИУС		
Зав.каф.		Бушманов А.В.						

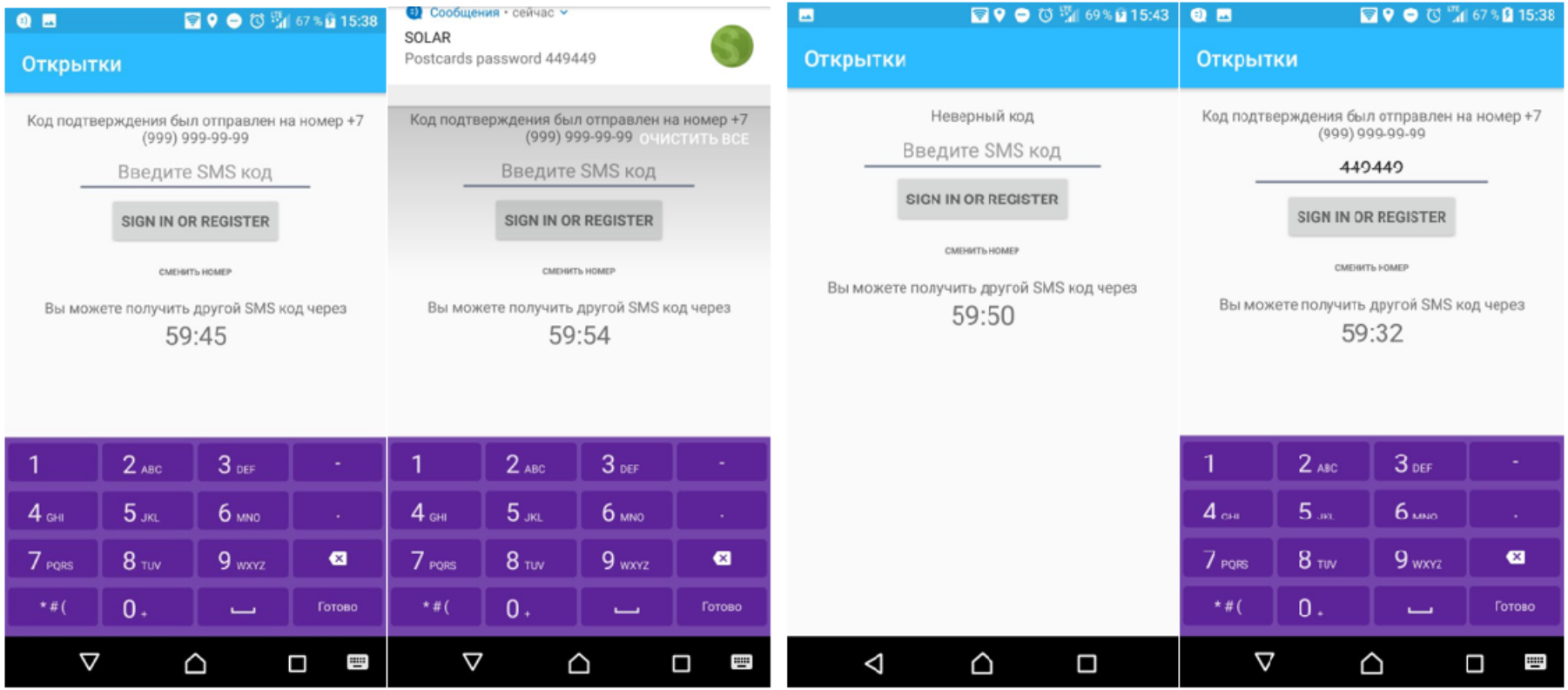


Рисунок 3 – Пример ввода, получения SMS кода и ошибки

					ВКР.145328.09.03.02.СХ			
Изм.	Лист	№ докум.	Подп.	Дата				
Разраб.		Саяпин В.А.			РАЗРАБОТКА ANDROID ПРИЛОЖЕНИЯ МЕССЕНДЖЕР	Лит.	Масса	Масштаб
Пров.		Бушманов А.В.				У		
Консульт.		Булавков А.Б.						
Н.контр.		Романико В.В.				Лист 3	Листов 8	
Зав.каф.		Бушманов А.В.				АмГУ Кафедра ИУС		

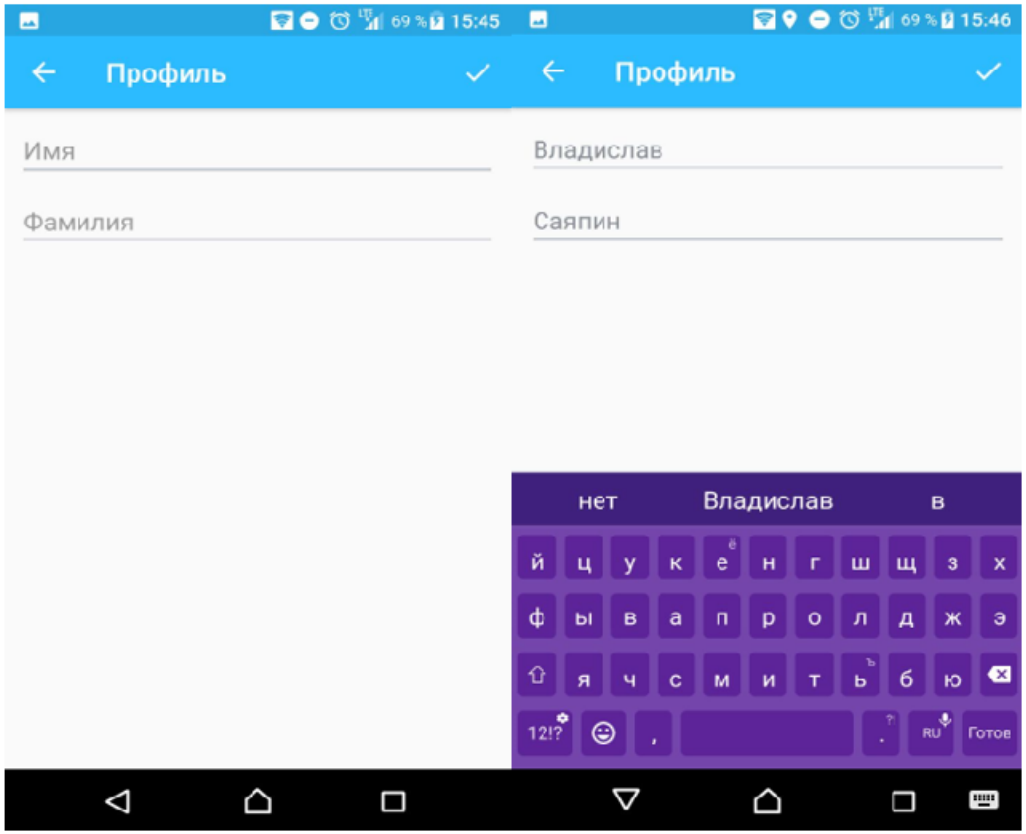


Рисунок 4 – Пример изменения личных данных

Рисунок 5 – Пример работы с HomeActivity

Рисунок 6 – Пример работы с ContactsActivity

					ВКР.145328.09.03.02.СХ			
Изм.	Лист	№ докум.	Подп.	Дата		Лит.	Масса	Масштаб
Разраб.		Саяпин В.А.			РАЗРАБОТКА ANDROID ПРИЛОЖЕНИЯ МЕССЕНДЖЕР	У		
Пров.		Бушманов А.В.						
Консульт.		Булавков А.Б.				Лист 6	Листов 8	
Н.контр.		Романико В.В.				АмГУ Кафедра ИУС		
Зав.каф.		Бушманов А.В.						

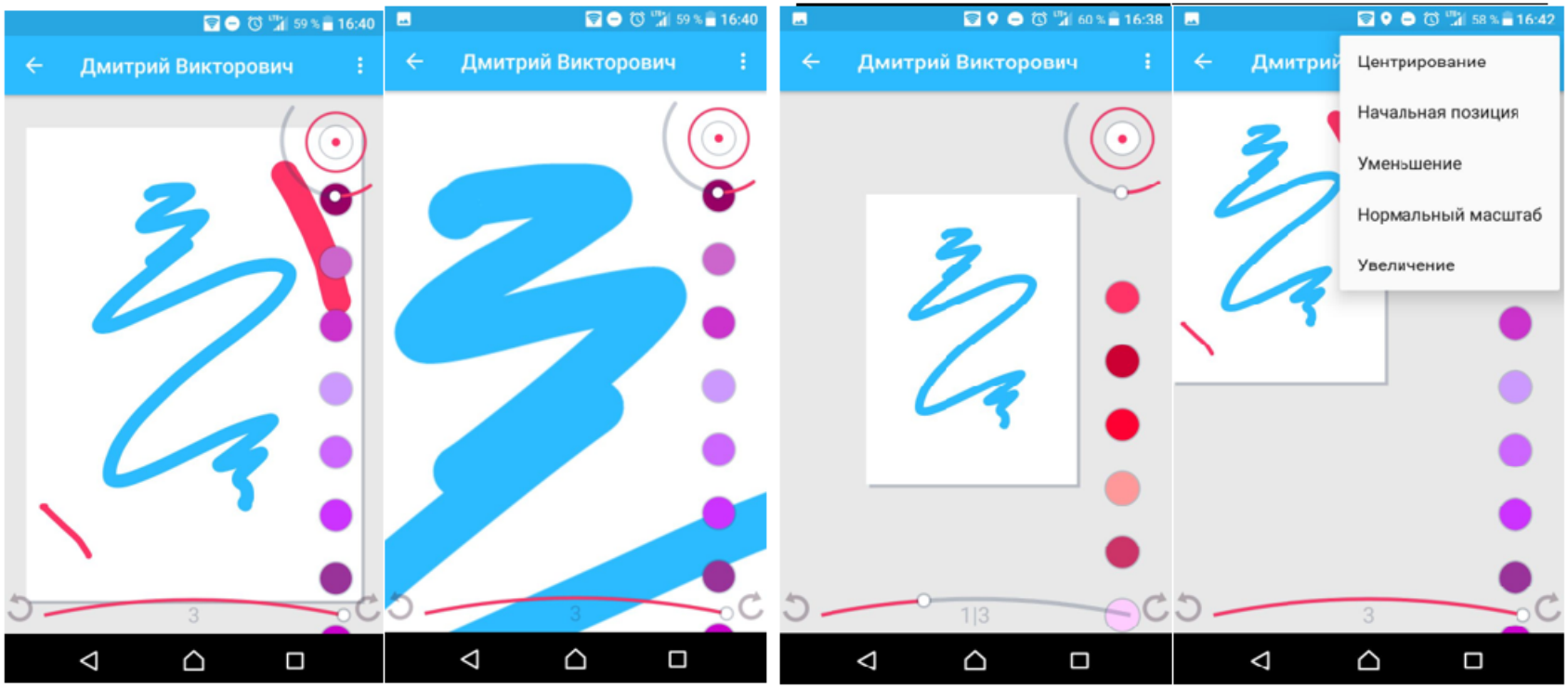


Рисунок 7 – Пример функционала PaintActivity

					ВКР.145328.09.03.02.СХ		
Изм.	Лист	№ докум.	Подп.	Дата			
Разраб.		Саяпин В.А.			РАЗРАБОТКА ANDROID ПРИЛОЖЕНИЯ МЕССЕНДЖЕР	Лит.	Масса / Масштаб
Пров.		Бушманов А.В.				У	
Консульт.		Булавков А.Б.				Лист 7	Листов 8
Н.контр.		Романико В.В.				АмГУ Кафедра ИУС	
Зав.каф.		Бушманов А.В.					

Рисунок 8 – Уведомления о новых сообщениях