

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение
высшего образования

«ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

Факультет

Кафедра

вычислительной техники

«Вычислительная техника»

Специальность: 09.05.01 «Применение и эксплуатация
автоматизированных систем специального
назначения»
Специализация: «Эксплуатация вычислительных машин, комплексов,
систем и сетей специального назначения»

ДИПЛОМНЫЙ ПРОЕКТ

на тему

ПРИЛОЖЕНИЕ ДЛЯ ОБМЕНА СООБЩЕНИЯМИ В
ПИРИНГОВЫХ СЕТЯХ

Студент

Гвалья
(подпись, дата)

13.06.2023

Гвалья Давид Мамукович
(ФИО полностью)

Руководитель

Дубравин
(подпись, дата)

13.06.2023

Дубравин А.В.
(фамилия, инициалы)

Нормоконтролёр

Бычков
(подпись, дата)

13.06.2023

Бычков А.С.
(фамилия, инициалы)

Рецензент

Нач. отдела
Фил. и Юр. «Росстат Волга»
(должность, место работы) «Пенза-эксперт»

Сивастьянов
(подпись, дата)

14.06.2023

Сивастьянов А.
(фамилия, инициалы)

Работа допущена к защите (протокол зас. кафедры от 2.06.2023 г. № 14)

Заведующий кафедрой

Митрохин
(подпись)

Митрохин М.А.
(фамилия, инициалы)

Работа защищена с отметкой

хорошо

(протокол заседания ГЭК от

19.06.2023 № 2)

Секретарь ГЭК

Гурин
(подпись)

Гурин Е.И.
(фамилия, инициалы)

Пенза 2023

Федеральное государственное бюджетное образовательное учреждение
высшего образования
ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ПОЛИТЕХНИЧЕСКИЙ ИНСТИТУТ

Факультет
Вычислительной техники

Кафедра
Вычислительная техника

«Утверждаю»
Заведующий кафедрой
Митрохин М.А.
«26» апреля 2023 г.

**ЗАДАНИЕ
НА ДИПЛОМНЫЙ ПРОЕКТ**

1. Студент Гвалия Давид Мамукович гр. 19ВОЭ1 факультета вычислительной техники.

2. Тема работы: «Приложение для обмена сообщениями в пиринговых сетях».

Тема утверждена приказом ПГУ № 458/0 от 26.04.2023 г.

3. Руководитель работы к.т.н., доцент каф. ВТ «ПГУ» Дубравин А.В.

4. Задание на работу (назначение разработки, исходные данные и т.п.).

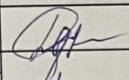
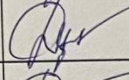
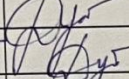
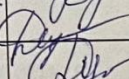
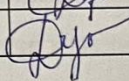
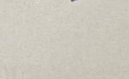
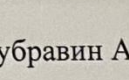
Назначение разработки – Программа предназначена для передачи файлов и сообщений с помощью протоколов Wi-Fi Direct и TCP/IP. В программе необходимо предусмотреть с файловой системой и СУБД SQLite. Программа должна работать в рамках беспроводной одноранговой сети передачи данных.

Среда работы приложения – пиринговая сеть, «точка» - «точка».

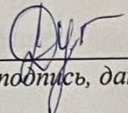
5. Перечень подлежащих разработке вопросов.

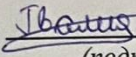
Требуется произвести анализ предметной области, определить функциональный состав программы, выбрать стек технологий для реализации программы, разработать алгоритм работы программы, разработать диаграммы: прецедентов, активностей, последовательностей, объектов, классов, с последующим написанием программы.

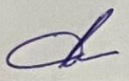
6. Календарный график выполнения работы

№ п/п	Наименование этапов работы	Объем работы	Срок выполнения	Подпись руководителя
1.	Анализ предметной области	10	26.04.2023 - 28.04.2023	
2.	Выбор инструментов проектирования и реализации проекта	10	29.04.2023 - 30.04.2023	
3.	Выбор протокола передачи данных	20	31.04.2023 - 03.05.2023	
4.	Разработка базы данных	10	04.05.2023 - 08.05.2023	
5.	Разработка мобильного приложения	10	09.05.2023 - 13.05.2023	
6.	Тестирование приложения	10	16.05.2023 - 23.05.2023	
7.	Оформление выпускной работы	30	24.05.2023 - 02.06.2023	

Дата выдачи задания 26 «04» 2023 года.

Руководитель дипломного проекта  Дубравин А.В.
(подпись, дата)

Задание к исполнению принял студент  Гвалия Д.М.
(подпись, дата)

Работу к защите допустить
Декан факультета  19.06.2023 Фионова Л.Р.
(подпись, дата)

Реферат

Работа содержит 98 лист., 29 рис., 19 таб., 5 ист., 5 прил.

ОДНОРАНГОВЫЕ СЕТИ, БЕСПРОВОДНЫЕ СЕТИ, МОБИЛЬНОЕ ПРИЛОЖЕНИЕ

Цель ВКР – разработать программное обеспечение, позволяющее передавать данные между двумя устройствами в рамках одноранговой беспроводной сети по принципу «точка - точка».

В ходе выполнения ВКР были разработано мобильное приложение для установления связи между двумя мобильными устройствами на основании протокола *Wi-Fi Direct*.

В результате ВКР разработано приложение, полностью соответствующее поставленной задаче, с возможностью:

- установки соединения между двумя устройствами;
- передачи файлов между двумя устройствами;
- передачи сообщений между двумя устройствами;
- закрытия сеанса.

В пояснительной записке приведен анализ предметной области, процесс разработки программного обеспечения, обзор технологии разработки и отладки программы.

Графическая часть содержит диаграммы, полностью описывающие структуру и реализацию программного обеспечения.

					ПГУ 1.09.05.01.02.001 ПЗ			
Изм.	Лист	№ докум.	Подпись	Дата	Приложение для обмена сообщениями в пиринговых сетях Пояснительная записка	Лит.	Лист	Листов
Разраб.		Гвалья Д.М.					2	98
Провер.		Дубравин А.В.						
Н. Контр.		Бычков А.С.						
Утверд.		Митрохин М.А.				ФВТ, каф. ВТ гр.18ВОЭ1		

Содержание

Перечень сокращений и обозначений.....	4
Введение.....	4
1. Приложение для обмена сообщениями в пиринговых сетях	5
1.1 Предпроектный анализ предметной области.....	5
1.1.1 Анализ предметной области	5
1.1.2 Анализ программ аналогов.....	7
1.1.3Анализ целевой аудитории	8
1.1.4 Выбор стека технологий для реализации приложения.....	9
1.1.4.1 Сравнительный анализ протоколов беспроводной передачи данных.....	10
1.1.4.2 Описание протокола Wi-Fi Direct	12
1.1.4.3. Способ подключения Wi-Fi Direct на уровне операционной системы	13
1.1.4.4 Выбор языка программирования	18
1.4.5 Выбор СУБД	19
1.1.5 Анализ требований	19
1.1.5.1 Функциональные требования к разрабатываемому программному обеспечению.....	20
1.1.6 Нефункциональные требования к разрабатываемому ПО	22
1.1.7Стадии и этапы разработки.....	22
1.1.8. Выводы по главе	23
1.2. Проектирование и реализация программного обеспечения для обмена сообщениями в пиринговых сетях.	24
1.2.1 Проектирование архитектуры мобильного приложения.....	24
1.2.2 Проектирование интерфейса мобильного приложения.....	30
1.2.3 Проектирование базы данных	36
1.2.3.1 Логическая модель разрабатываемой базы данных.....	37
1.2.3.2 Физическая модель разрабатываемой базы данных	38
1.2.4 Реализация мобильного приложения.....	40

1.2.5 Выводы по главе	43
2.Технология отладки программного обеспечения в <i>IDE Android Studio</i>	44
2.1 Запуск режима отладки в <i>IDE Android Studio</i>	44
2.2 Подключение отладчика к запущенному процессу	46
2.3. Просмотр общих сведений в системных журналах	46
2.4 Просмотр системного журнала для выбранного процесса.....	47
2.5 Способы работы с точками останова.....	48
2.6 Фиксация скриншотов и видео в режиме отладки приложения	52
2.7 Выводы по главе	53
3. Экономическое обоснование эффективности проекта	54
4. Безопасность жизнедеятельности.....	58
Заключение	61
Список использованных источников	62
Приложение А – <i>UML-диаграммы</i>	63
Приложение А.1 – Диаграмма вариантов использования	64
Приложение А.2 – Диаграмма развертывания.....	65
Приложение А.3 – Диаграмма деятельности	66
Приложение А.4 – Диаграмма деятельности	67
Приложение А.5 – Диаграмма данных	68
Приложение Б – Листинг программы	69
Приложение В - Презентация	89

Перечень сокращений и обозначений

ПО	-	Программе обеспечение
СУБД	-	Системы управления данными
БД	-	База данных
ЭВМ	-	Электронно-вычислительная машина
ПИ	-	Пользовательский интерфейс
ПЗ	-	Пояснительная записка

					ПГУ 1.09.05.01.02.001 ПЗ	Лист
						4
Изм.	Лист	№ докум.	Подпись	Дата		

Введение

В настоящее время беспроводные технологии прочно вошли в нашу жизнь, как в бытовой, так и в промышленной плоскости.

В настоящий момент существует большой выбор беспроводных протоколов, программного обеспечения (далее - ПО), устройств позволяющих передавать аудио, видео, текстовую информацию и/или управлять «умными» устройствами на расстоянии.

Фактически сейчас наблюдается подъем интереса к «умным» устройствам и ПО, а также интерес к интеграции вышеописанных компонентов в «эко-систему» под названием интернет вещей (IoT).

Данная работа посвящена реализации программного обеспечения эксплуатируемого в рамках мобильных устройств под руководством операционной системы Android с поддержкой протокола *Wi-Fi Direct*.

Реализованное ПО необходимо для прямого объединения гетерогенных мобильных устройств в одноранговую сеть с последующей возможностью передачи файлов между устройствами и/или передачи текстовой информации по схеме «*peer-to-peer*» (далее - P2P).

Выбор данного вида передачи информации между устройствами и необходимость реализации подобного программного обеспечения очевидна:

- большинство информации между людьми в рамках командной работы передается с помощью мессенджеров по схеме (точка-сервер-точка), несмотря на то, что расстояние между людьми не превышает 100 метров, при этом информация проходит через каналы связи и серверы, которые могут быть «скомпрометированы». Для решения данной проблемы решения «точка-точка» подходят идеально;

- практически все открытые решения в рамках беспроводных стандартов в настоящий момент имеют иностранное происхождение, и реализация отечественных аналогов будет способствовать развитию отечественного сегмента *P2P*.

.

					ПГУ 1.09.05.01.02.001 ПЗ	Лист
						4
Изм.	Лист	№ докум.	Подпись	Дата		

1. Приложение для обмена сообщениями в пиринговых сетях

1.1 Предпроектный анализ предметной области

1.1.1 Анализ предметной области

Исследования международных некоммерческих организаций показывают [1], что число пользователей смартфонов с каждым годом стремительно растет.

В 2023 году число пользователей смартфонов достигло 6,8 миллиарда человек по всему миру, что указывает на рост на 4,2% за 1 год [1].

За семь лет в период с 2016 по 2023 год рост составил 86,5%, в абсолютных значениях это прирост на 3,2 миллиарда единиц мобильных устройств [2].

Вышеописанные показатели говорят о том, что смартфоны являются самыми популярными электронными устройствами в мире. Статистика пользователей смартфонов [3] показывает, что из всех используемых сегодня устройств смартфоны являются самыми популярными, ими пользуются 95,8% всех пользователей Интернета.

Согласно отчетам, [4] в рисунке 1.1 мы можем увидеть, что число пользователей смартфонов по всему миру продолжит расти и к 2024 году достигнет 7,1 миллиарда.

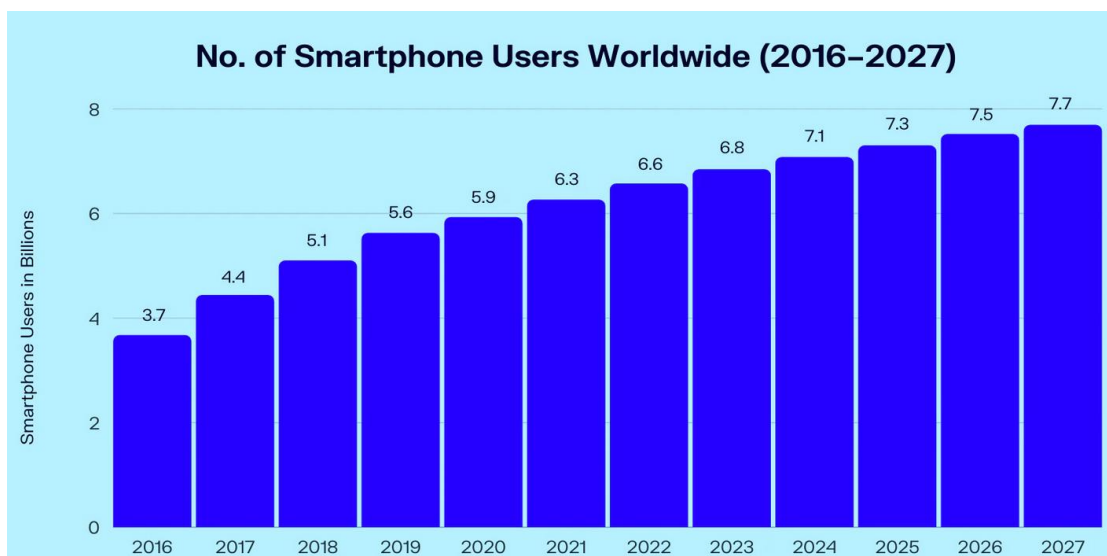


Рисунок 1.1 – Количество смартфонов используемых в мире в период 10 лет

Это означает, что уровень использования смартфонов в 2024 году составит около 90%. Другими словами, почти девять из 10 человек в мире будут оснащены смартфоном.

Выше сказанное показывает насколько важно развивать и использовать «эко систему» смартфонов в части передачи сообщений и иной информации с учетом нагрузки на сеть передачи данных. В частности разрабатывать программное обеспечение для взаимодействия данных устройств, без нагрузки на сеть интернет и коммутационное оборудование, с целью передачи данных и работы в режиме передачи мгновенных сообщений.

В целях передачи данных и коммуникации между смартфонами без использования сторонних коммутационных устройств и каналов используются различные протоколы и способы передачи данных, некоторые из них реализованы в виде программного обеспечения, некоторые представлены аппаратными модулями мобильных устройств.

Способы прямого взаимодействия устройств основаны на способе реализации одноранговой, децентрализованной компьютерной сети, основанной на равноправии участников. Для объединения смартфонов и иных устройств в такие одноранговые сети типа *P2P (peer-to-peer)* могут быть использованы следующие протоколы передачи данных: *Bluetooth, WiFi Direct, BLE, Zigbee, Z-Wave, 6LoWPAN, NFC, WiFi*.

В такой сети отсутствуют выделенные серверы (реализованные в виде компьютера и/или коммутатора), и любой хост (peer) является клиентом, с возможностью функций сервера.

Вывод, мобильное приложение для обмена сообщениями в пиринговых сетях должно предоставлять пользователям следующие основные функции:

- возможность отправки файлов;
- возможность отправки сообщений;
- хранить историю сообщений;
- предоставлять список потенциальных клиентов;
- работать с клиентами по беспроводным сетям;

- работать с клиентами без коммутационного и серверного взаимодействия с локальной и/или глобальной сетью.

1.1.2 Анализ программ аналогов

При разработке приложения важно исследовать аналоги и опыт уже реализованного программного обеспечения. В данном разделе приведены аналоги приложения реализованного в рамках дипломного проекта.

Приложение для обмена файлами *Android Google Files* (*com.google.android.apps.nbu.files*) - «Files by Google».

Первоначально разработанное программное обеспечение в рамках проекта *Google Next Billion Users (NBU)* [7] для смартфонов в развивающихся странах: Бразилия, Китай, Индия, Индонезия и Нигерия. Проект, ориентирован на развивающиеся рынки, активно поддерживается на *Android* и устанавливается в качестве системного приложения, начиная с *Android 8* и *Android 9*.

SHAREit (*com.lenovo.anyshare.gps*) реализован компанией Lenovo в году и стал широко используемым приложением для обмена файлами D2D в мире. По утверждению компании, количество активных пользователей приложения на десктопных и мобильных устройствах превышает 1,5 млрд [10].

Приложения *Xender* (*cn.xender*) и *MiDrop* (*com.xiaomi.midrop*) системные приложения, от производителей, используемые для передачи данных с устройства на устройство в формате D2D.

В таблице 1 перечислены приложения для обмена файлами D2D, которые проанализированы в текущем разделе.

Таблица 1.1 – Приложения передачи данных в формате D2D.

Приложение	адрес приложения в сети Интернет	количество инсталляций, шт.
<i>SHAREit</i>	<i>com.lenovo.anyshare</i>	>10 миллионов
<i>Xender</i>	<i>cn.xender</i>	>10 миллионов
<i>Xiaomi Mi Drop</i>	<i>com.xiaomi.midrop</i>	>10 миллионов
<i>Files by Google</i>	<i>com.google.android.apps.nbu.files</i>	>10 миллионов

В любом случае, несмотря на популярность и пользу данные приложения обладают недостатками:

- все они являются системными приложениями, без возможно добавления функциональных возможностей и в основном идут предустановленными с новыми мобильными устройствами;
- все представленные приложения являются импортными и реализованными производителями устройств.

Для использования приложения с таким же функционалом в условиях и локациях с повышенными требованиями к безопасности данных: закрытые объекты, корпоративные офисы требуется реализация приложения, которое полностью контролируется, имеет открытый код и имеет возможность добавления функционала в соответствии с потребностями заказчика.

1.1.3 Анализ целевой аудитории

В связи с тем, что передача данных в рамках ограниченного пространства, без устройств посредников является актуальной задачей для государственных учреждений, то работать реализуемое программное обеспечение должно под платформы с открытым исходным кодом.

Инструменты разработки должны создавать кроссплатформенный программный продукт.

В таблице 1.2. представлены масштабы использования мобильных ОС в разных регионах Российской Федерации.

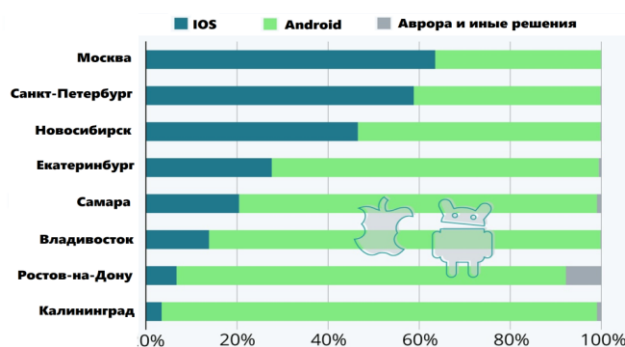


Рисунок 1.2 – процент использования мобильных ОС в различных регионах РФ

Помимо масштабов использования важны и иные характеристики ОС, представленные в таблице 1.2

Таблица 1.2 – Сравнительный анализ мобильных операционных систем

Параметр сравнения	iOS	Android
Доля на российском рынке	30%	70%
Целевая аудитория	обеспеченные пользователи	социальный слой не имеет значения
Количество загрузок приложений в % (2023)	32.3 %	67,7
Сроки разработки	медленно	Быстро
Унификация	унификации устройств и программных средств	широкий технологический стек
Языки программирования	<i>Objective-C, Swift</i>	<i>Java, Kotlin</i>
Инструменты разработки	<i>Apple Code, Xcode</i>	<i>Eclipse, Android Studio</i>
Фрагментация устройства	ограниченная линейка унифицированных дорогих устройств	большое количество дешевых устройств
Каналы распространения	магазин приложений	<i>Google Play</i> , сторонние магазины приложений и ручная загрузка приложений
Плата за распространение	есть	нет

На основании данных представленных на рисунке 1.2 и в таблице 1.2 делается вывод, что выбор в качестве рабочей ОС - *Android* является перспективным с точки зрения выбора потенциальных пользователей.

1.1.4 Выбор стека технологий для реализации приложения

Для реализации программного приложения требуется определить инструментарий:

- протокол позволяющий передавать данные в пиринговый сетях;
- язык программирования для реализации кроссплатформенного приложения;

- СУБД, для хранения данных и служебной информации о приложении.

1.1.4.1 Сравнительный анализ протоколов беспроводной передачи данных

NFC , *Bluetooth* , *Wi-Fi Direct* - это технологии и протоколы беспроводной передачи данных формата *P2P* поддерживаемые современными портативными устройствами.

Проведем сравнительный анализ данных протоколов, цель которого выбор протокола для реализации программного приложения обмена файлами и мгновенными сообщениями в рамках одноранговой сети.

Области применения *NFC* (*Near Field Communication*) привязаны к сфере коммерции, социальных сетей. Прямым родственником *NFC* можно считать технологию *Android beam*, которая, как и *NFC* позволяет передавать данные через связь ближнего радиуса действия. Что позволяет быстро обмениваться веб-закладками, контактной информацией, маршрутами, видео на YouTube и другими данными на небольшом расстоянии.

Преимуществом *NFC* является простые способы программной реализации. Протокол универсален и может быть использован как для обмена файлами, так и для перевода платежей по банковским картам. Его также можно использовать для видеоигр.

Поскольку протокол использует короткий диапазон для передачи данных, он помогает поддерживать безопасность данных, сводя к минимуму кражу.

Недостатки *NFC*:

- для использования технологии *NFC* требуется специально созданный сертификат. С точки зрения аппаратного модуля в смартфоне это дорогостоящая технология, которая сказывается на стоимости устройства;
- протокол может передавать данные только между совместимыми устройствами, необходимо, чтобы оба устройства имели технологию *NFC*.

Области применения *Bluetooth* начинаются от использования в бюджетных беспроводных устройствах: ноутбуки, мобильные телефоны, заканчивая сложными медицинскими устройствами.

Преимущества Bluetooth:

- проверенная временем технология. Программисты, и разработчики аппаратуры знают, как реализовать продукт в рамках данной технологии;
- разработчики, и конечный пользователь могут легко внедрить эту технологию в свои устройства, так как она дешевая;
- версии *Bluetooth* постоянно обновляются.

Недостатки Bluetooth:

- не защищенный канал передачи данных с возможностью передачи информации на расстояния менее 5 метров;
- наличие ограничений в передаче данных. Протокол не позволяет передавать данные более чем с двух устройств.

Wi-Fi Direct позволяет организовывать беспроводные сети между компьютерами или между компьютерами и периферийными устройствами.

Wi-Fi Direct разрабатывается и поддерживается группой *WECA* [4] - альянсом крупнейших производителей *Wi-Fi*-оборудования. Самым близким аналогом *Wi-Fi Direct* являются ранний вариант *Wi-Fi*, который связывал 2 или более устройств без роутера.

В отличие от классического протокола *Wi-Fi* данному протоколу (*Wi-Fi Direct*) не требуется наличия в одноранговой сети специальных управляющих устройств (точки доступа).

Точки доступа (коммутаторы, роутеры) выполняют три функции: физическую поддержку беспроводных и проводных сетей, коммутацию и маршрутизацию между устройствами в сети, а также службу, позволяющую добавлять и удалять устройства из сети. Этот функционал устройство с поддержкой *Wi-Fi Direct* реализует самостоятельно.

В отличие от *NFC* и *Bluetooth*, *Wi-Fi Direct* способен передавать большие объемы данных со скоростями свыше 200 Мбит/сек на расстояния до 100

метров. Данный факт делает технологию идеальной для реализации одноранговой сети, в рамках которой можно реализовать:

- обмен файлами между подключенными устройствами;
- беспроводную печать;
- режим потокового видео;
- автономную синхронизацию устройств.

В связи с тем, что реализуемое в рамках дипломного проекта приложение должно осуществлять передачу данных и сообщений между устройствами в формате P2P преимущества протокола Wi-Fi Direct по нагрузке, скорости и площади покрытия будут использованы при разработке программы.

1.1.4.2 Описание протокола *Wi-Fi Direct*

В связи с тем, что основой *D2D* приложения будет установление соединения по протоколу *Wi-Fi Direct*, ниже приведено описание работы данного протокола.

Устройства в рамках работы по протоколу *Wi-Fi Direct* могут работать в двух режимах: *P2P*, *P2PGO*. Этапы взаимодействия устройств в рамках протокола показаны на рисунке 1.3.

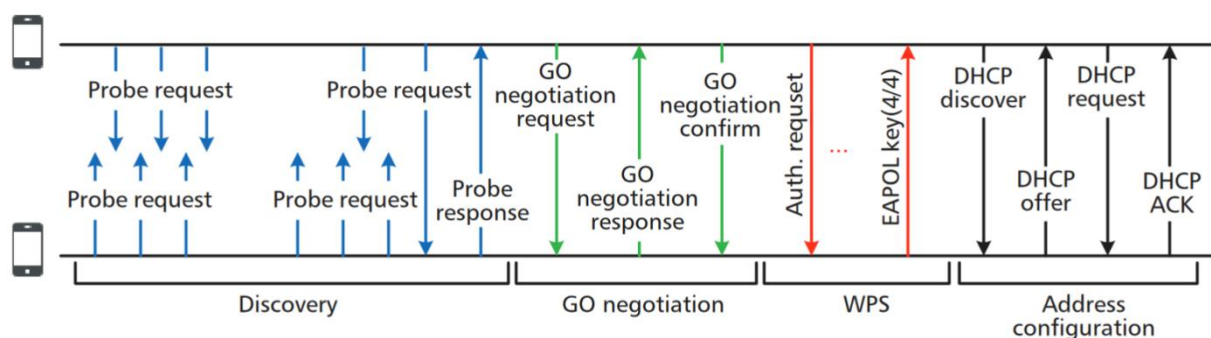


Рисунок 1.3 – Этапы работы устройств в рамках протокола *Wi-Fi Direct*

Протокол *WiFi Direct* позволяет двум устройствам установить соединение *D2D* без помощи точек доступа (*Wi-Fi* роутера). На рисунке 1.3 показан порядок соединения *D2D* с помощью *WiFi Direct*. Во-первых, два устройства выполняют зондирование канала (стандартные

каналы доступа №№ 1, 6, 11). Каждое устройство делает три запроса, по одному на каждом канале и если два устройства на одном канале определили друг друга они обмениваются «рукопожатием».

На втором этапе, два устройства ведут переговоры, чтобы определить, кто будет владельцем группы (*GO*) (владелец действует как точка доступа (*AP*)), для этого производится сравнение *MAC* адресов устройств и устройство с наименьшим *MAC* адресом становится владельцем группы.

После того, как устройства договорились об их ролях, при наличии библиотек авторизации и аутентификации (имеется в виду библиотеки в программном приложении) устройства могут выполнить настройку безопасности *Wi-Fi* соединения. Если таковые библиотеки отсутствуют, то данный шаг пропускается и соединение остается не зашифрованным.

Далее для передачи данных между устройствами необходимо установить классическое соединение *TCP/IP* для чего устройства используют протокол динамической конфигурации хостов (*DHCP*). Данный протокол поддерживает любое современное мобильное устройство.

Процесс работы двух мобильных устройств в рамках протокола показан на рисунке 1.3.

1.1.4.3. Способ подключения *Wi-Fi Direct* на уровне операционной системы

При этом работа данных приложений зачастую возможна, если в настройках операционной системы включен режим *Wi-Fi Direct* и имеется специализированное устройство.

Для того чтобы использовать *Wi-Fi Direct* потребуется портативное устройство с установленной операционной системой, в которой предусмотрена поддержка стандарта *Wi-Fi Direct*.

Для проверки поддержки устройством протокола *Wi-Fi Direct* можно использовать командную строку *Windows PowerShell* (рисунок 1.4).

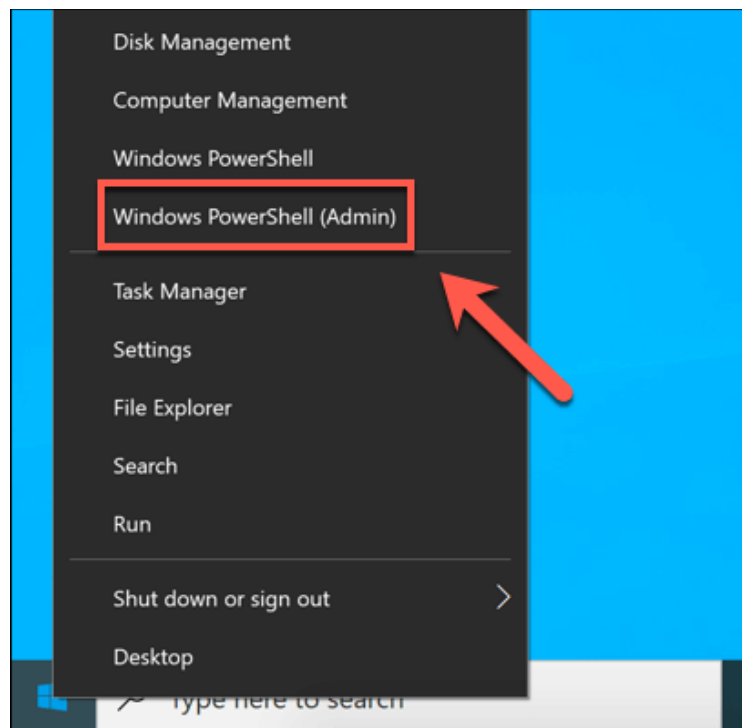


Рисунок 1.4 – Активизированное меню «Пуск»

В активированном окне *Windows PowerShell* требуется ввести команду *ipconfig/all* и нажмите «Enter». Появится список сетевых адаптеров и подключений в операционной системе. В появившемся списке адаптеров требуется найти виртуальный адаптер *Microsoft Wi-Fi Direct Virtual Adapter*.

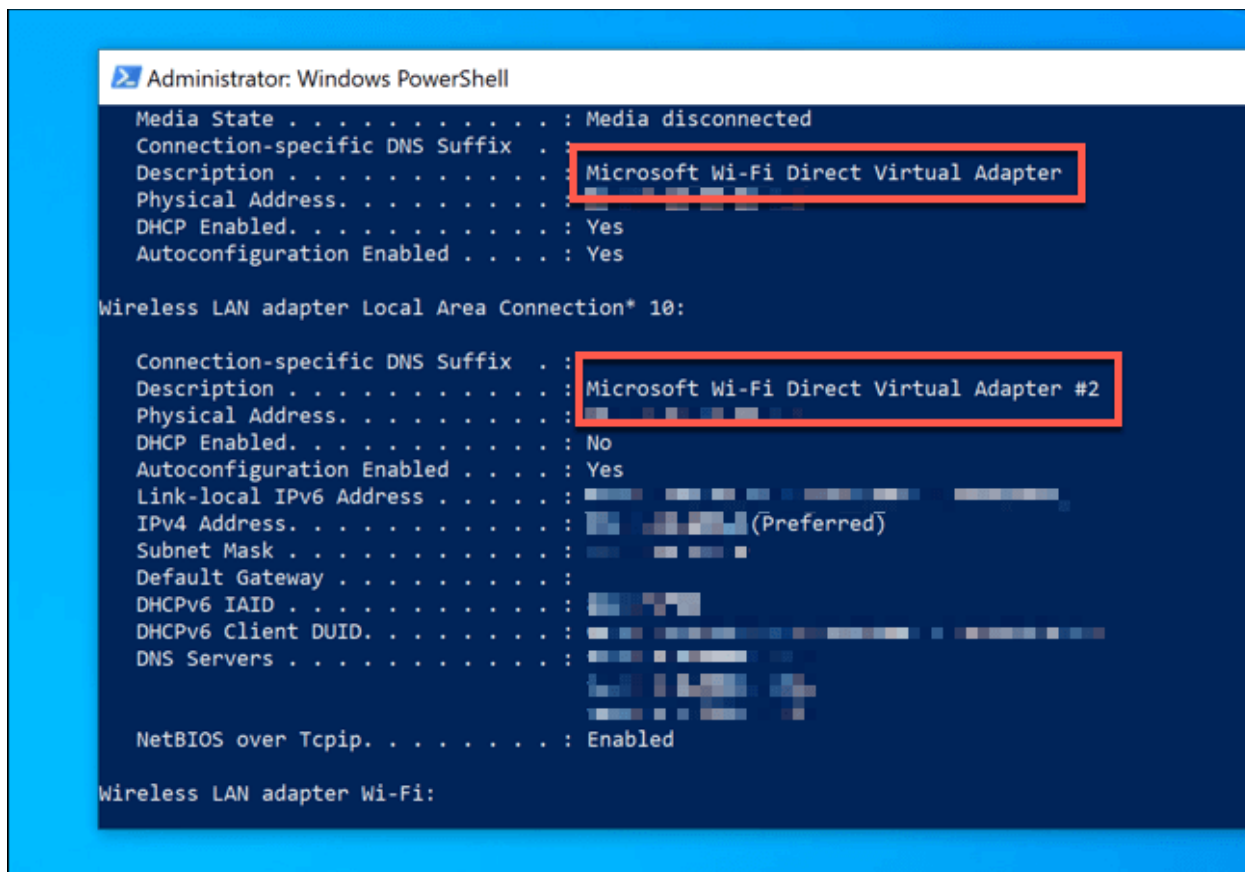


Рисунок 1.5 – Список доступных *Wi-Fi Direct* в ОС

На устройстве используемом как стенд для данного дипломного проекта виртуальный адаптер *Microsoft Wi-Fi Direct* установлен (рисунок 1.5.). Если на другом стендовом компьютере устройство *Wi-Fi Direct* нет, то необходимо подключить стороннее USB-устройство с поддержкой протокола *Wi-Fi Direct*, чтобы включить данную функцию.

Для передачи данных стандартными средствами операционной системы требуется перейти в диалог «Настройки» (рисунок 1.6) и выбрать опцию «Устройства» (рисунок 1.7.).

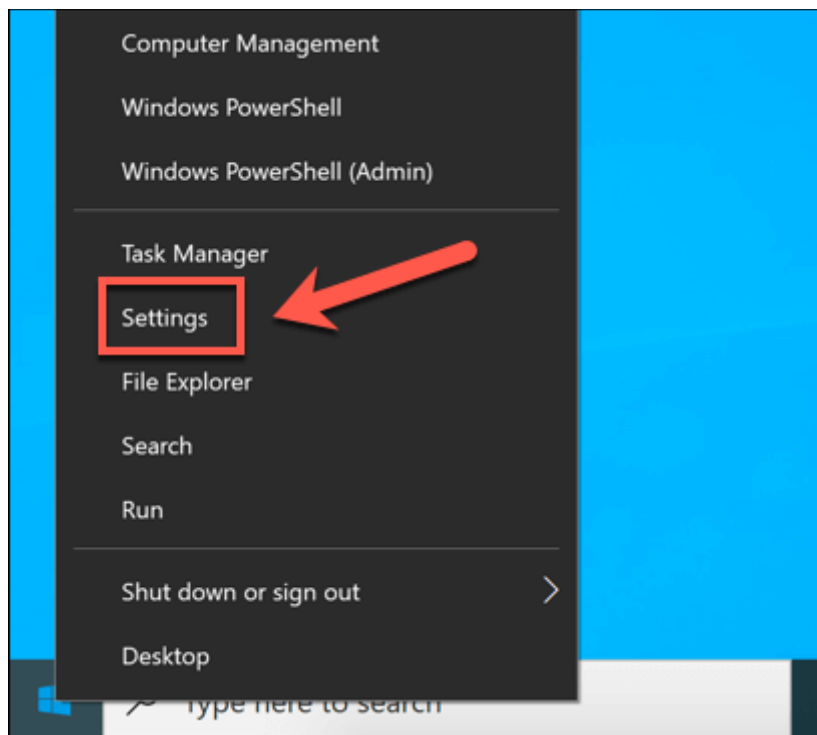


Рисунок 1.6 – Настройки *Wi-Fi Direct*

Далее требуется выбрать опцию «Добавить *Bluetooth* или другое устройство» (рисунок 1.7).

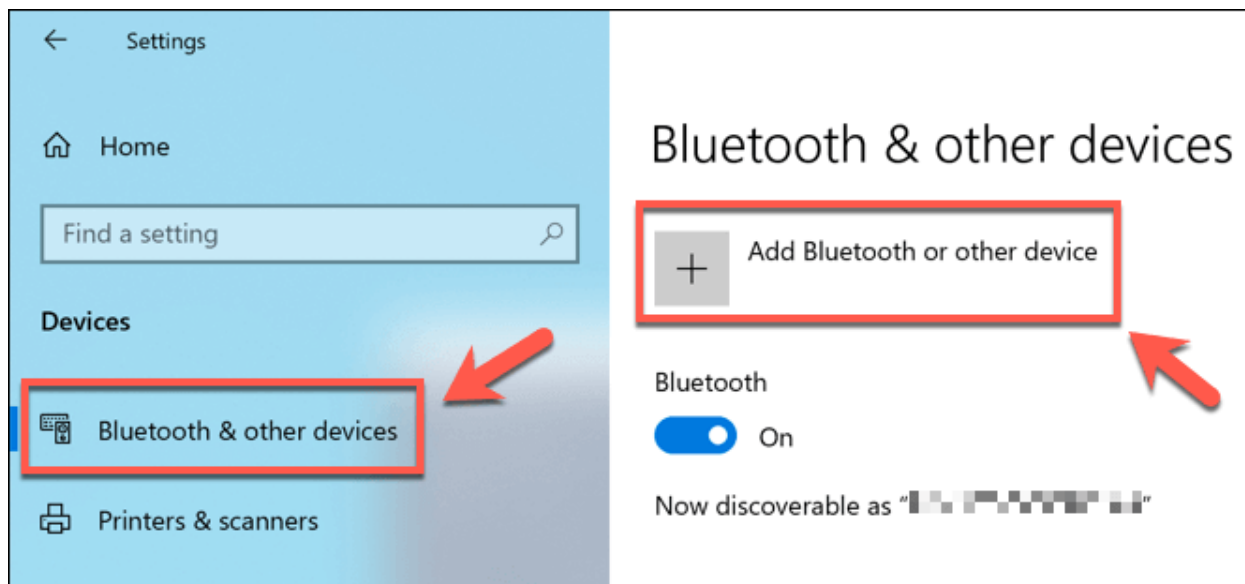


Рисунок 1.7 – Выбор настроек *Wi-Fi Direct*

Во всплывающем меню «Добавить устройство» требуется выбрать тип устройства, к которому хотите подключиться - *Wi-Fi Direct*.

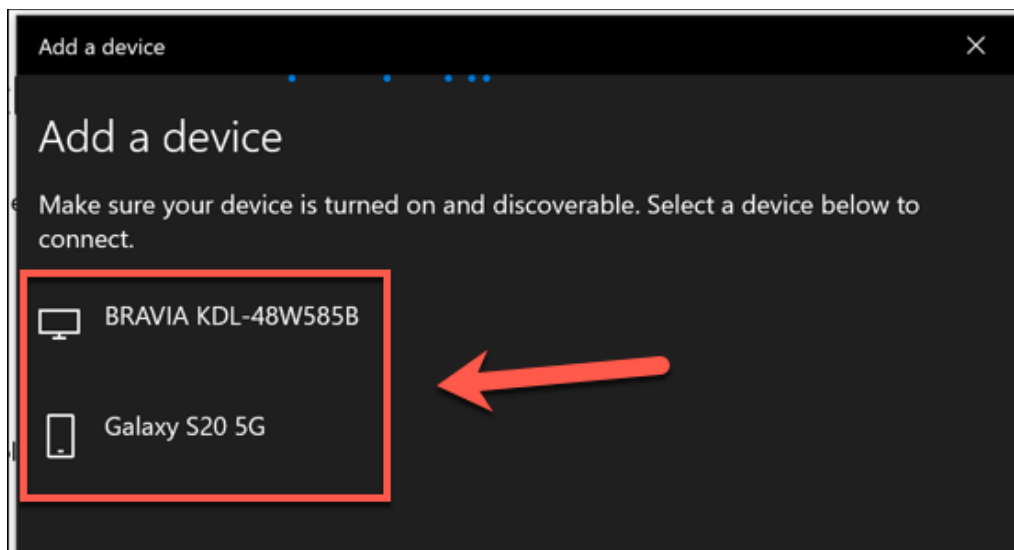


Рисунок 1.8 – Список устройств готовых к подключению

При успешном подключении будет активировано всплывающее меню (рисунок 1.9)

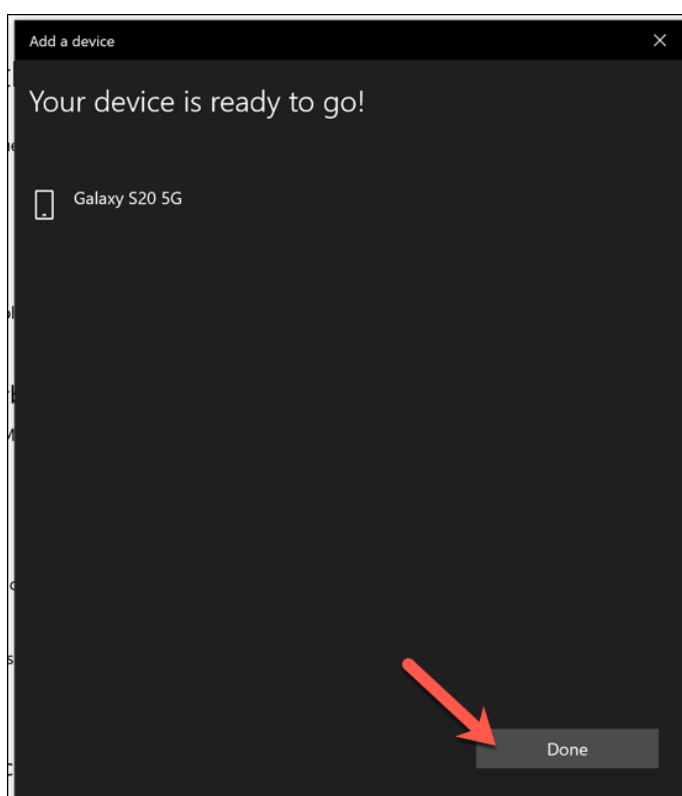


Рисунок 1.9 – Уведомление о том, что устройства успешно
подсоединены

На базе мобильной платформы под управлением ОС Android настройка и подключение *Wi-Fi Direct* производится аналогичным способом.

1.1.4.4 Выбор языка программирования

Следующим шагом является выбор языка программирования между *Java* и *Kotlin* (поддерживаются ОС *Android*).

Язык *Java* является мощным инструментом и поддерживает разработку программных приложений, операционных систем, и даже новых языков программирования, плюс *Java* - его высокая производительность.

Достоинства языка *Java*:

- кроссплатформенный язык, работает практически на любом устройстве, сервере или операционной системе;
- имеет строгую типизацию и механизмы работы с памятью, которые делают вероятность ошибки исполнения низкой;
- поддерживает механизмы создания модульных приложений отсюда появляется возможность повторного использования кода;
- обладает способностью использовать сторонний код;
- стандарты безопасности *Java* выше чем *Kotlin*.

Недостатки языка *Java*:

- синтаксис сложнее для понимания начинающего разработчика по сравнению с такими языками, как *Python*;
- много деприкатного функционала из-за чего часть контента приложения может быть недоступен конечному пользователю;
- разработка требует написания большего количества кода и имеет более высокую вероятность ошибок по сравнению с *Kotlin* или *Python*;
- требователен к количеству памяти устройства.

Kotlin - это типизированный язык программирования с открытым исходным кодом, который работает на *JVM* и является официальным языком для разработки приложений для *Android*.

Достоинства языка *Kotlin*:

- разработка приложений на *Kotlin* занимает меньше времени по сравнению с *Java*;
- код, написанный на *Kotlin*, имеют меньший размер;

- байт-код, который компилируется *Kotlin*, может быть выполнен с помощью *JVM*. Таким образом, библиотеки и фреймворки из *Java* можно настроить для запуска в проекте, основанном на *Kotlin*;
- *Kotlin* совместим с *Javascript* для веб-разработки.

Недостатки языка *Kotlin*:

- сообщество разработчиков меньше по численности по сравнению с известными языками программирования;
- маленькая база пользователей и небольшая служба поддержки.

В связи с тем, что поддержка на этапе разработки и эксплуатации программного обеспечения является важным этапом, было принято решение использовать проверенный язык программирования, для которого реализованы библиотеки и *API* для сети *Wi-Fi Direct – Java*.

1.4.5 Выбор СУБД

Для работы мобильного приложения, в котором планируется хранить данные и служебную информацию требуется легковесная, встраиваемая реляционная система управления базами данных. СУБД должна работать без настроек и управления сервером базы данных и физически находится в одном файле на устройстве пользователя.

Такой СУБД является *SQLite*. Эта СУБД позволяет выполнять стандартные операции с базами данных: создание, добавление, изменение и удаление данных. *SQLite* поддерживает основные типы данных языка *SQL* и приспособлена для использования в различных сред программирования: *Python, Java, C ++*.

При разработке приложений для ОС *Android*, *SQLite* используется для сохранения любых данных: текстовые сообщения, изображения, видео, контакты и прочее. *SQLite* для *Android* обеспечивает быстрый доступ к данным и высокую производительность в приложениях.

1.1.5 Анализ требований

Применим в дипломном проекте два типа требований:

					ПГУ 1.09.05.01.02.001 ПЗ	Лист
						19
Изм.	Лист	№ докум.	Подпись	Дата		

- функциональные требования – описывают поведение, которое должно предлагать ПО;
- нефункциональные требования – определяют свойства и/или ограничения, накладываемые на ПО стеком технологий, условиями использования и/или пользователями.

1.1.5.1 Функциональные требования к разрабатываемому программному обеспечению

Разработанное в рамках данного проекта программное обеспечение должно содержать следующий функционал (рисунок 1.10):

- инициализация программы;
- выбор роли, в которой будет работать устройство;
- выбор режима работы (установка соединения, выбор и передача файла, проверка статуса файлов, проверка статуса соединений);
- соединение двух и более устройств по протоколу *WiFi Direct*;
- поддерживать операции с файлами операционной системы *Android*;
- иметь возможность работать в режиме клиент серверного приложения (поставщик – потребитель);
- иметь возможность закончить сеанс передачи данных;
- имеет возможность логирования действий, с сохранением фиксируемой информации в БД.

Диаграмма вариантов использования мобильного приложения представлена на рисунке 1.10.

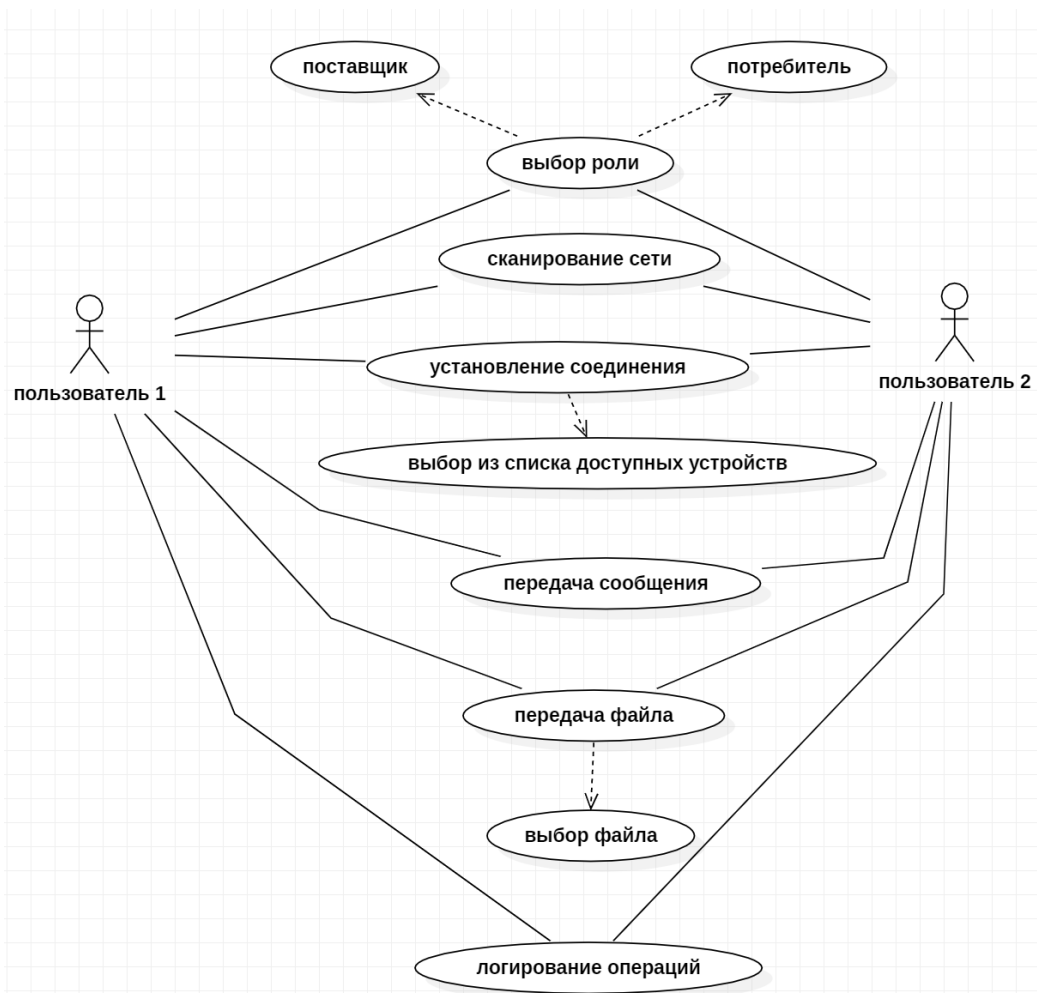


Рисунок 1.10 – Диаграмма вариантов использования мобильного приложения

Исходя из данных рисунка 1.10 и функциональных требований к программе, работа системы представляется следующим образом.

В системе работают два пользователя. Пользователь 1 является инициатором связи, для чего он с помощью функции «Выбор роли» выбирает роль поставщика услуг.

После выбора роли у него появляется возможность сканировать сеть в поисках Пользователя 2. Как только в процессе сканирования Пользователю 1 поступит сигнал о доступности Пользователя 2 - Пользователь 2 появится в списке доступных устройств, Пользователь 1 установит соединение с Пользователем 2.

Отметим, что в системе протокол *Wi-Fi Direct* используется только как поисковая система устройств в беспроводной сети, без участия устройств посредников. Всю основную работу по передачи данных будет выполнять

классическое *TCP/IP* соединение двух устройств, выполненное в рамках одной одноранговой сети. Для чего устройство Пользователя 1 после установления соединения с помощью протокола *WiFi Direct* присваивает себе *IP* адрес 192.168.90.2, а устройству Пользователя 2 предлагает присвоить *IP* адрес 192.168.90.3.

После установления *TCP/IP* соединения двух устройств, Пользователь 1 может выбрать файл в своей файловой системе и передать его Пользователю 2. В системе разработчиком установлено ограничение на объем передаваемых файлов (не более 2 Мб на время тестового периода).

Передача сообщений может быть инициирована как поставщиком, так и потребителем (рисунок 1.10).

Все операции, производимые в рамках сеанса связи, логируются в базу данных устройств Пользователей.

1.1.6 Нефункциональные требования к разрабатываемому ПО

Реализуемое программное обеспечение работает только с устройствами, содержащими модули *Wi-Fi Direct*.

Мобильное приложение развертывается только с устройствами на базе ОС Android.

Расстояние между устройствами при передаче данных не должно быть более 50 метров (ограничение протокола *Wi-Fi Direct*).

1.1.7 Стадии и этапы разработки

Стадии и этапы разработки программного обеспечения описаны в таблице 1.2

Таблица 1.3- Стадии и этапы разработки ПО

наименование этапов работы	объем работы	срок выполнения
Анализ предметной области	6%	
Составление технического задания	6%	
Выбор инструментов разработки	8%	
Проектирование системы	12%	

Таблица 1.3- Стадии и этапы разработки ПО

Разработка программы	23%	
Разработка СУБД	23%	
Выполнение задания по БЖД	6%	
Разработка экономического обоснования проекта	8%	
Оформление пояснительной записки	8%	

1.1.8. Выводы по главе

В результате исследования проведенного в первой главе был сделан вывод, что реализация программного продукта для мобильных устройств является перспективным направлением в связи с популярностью данных устройств.

Передача информации в формате *D2D* в рамках одноранговых сетей является недооцененным с точки зрения насыщенности программными продуктами поле деятельности.

Ниша занята в основном только системным программным обеспечением или пользовательским программным обеспечением производителей.

Данное направление является перспективным в связи с тем, что имеется множество локаций даже в рамках массовых мероприятий, где отсутствие интернета приводит к необходимости соединения устройств коммуникации напрямую для передачи видео, аудио и текстовой информации.

1.2. Проектирование и реализация программного обеспечения для обмена сообщениями в пиринговых сетях.

Проектирование программного приложения состоит из этапов:

- проектирование и реализация приложения;
- проектирование и реализация базы данных.

1.2.1 Проектирование архитектуры мобильного приложения

Для реализации программного приложения, удовлетворяющего требованиям, представленным в первой главе необходимо использовать фреймворк *WifiP2P* [4].

На диаграмме развертывания показаны блоки реализуемого приложения (рисунок 2.1)

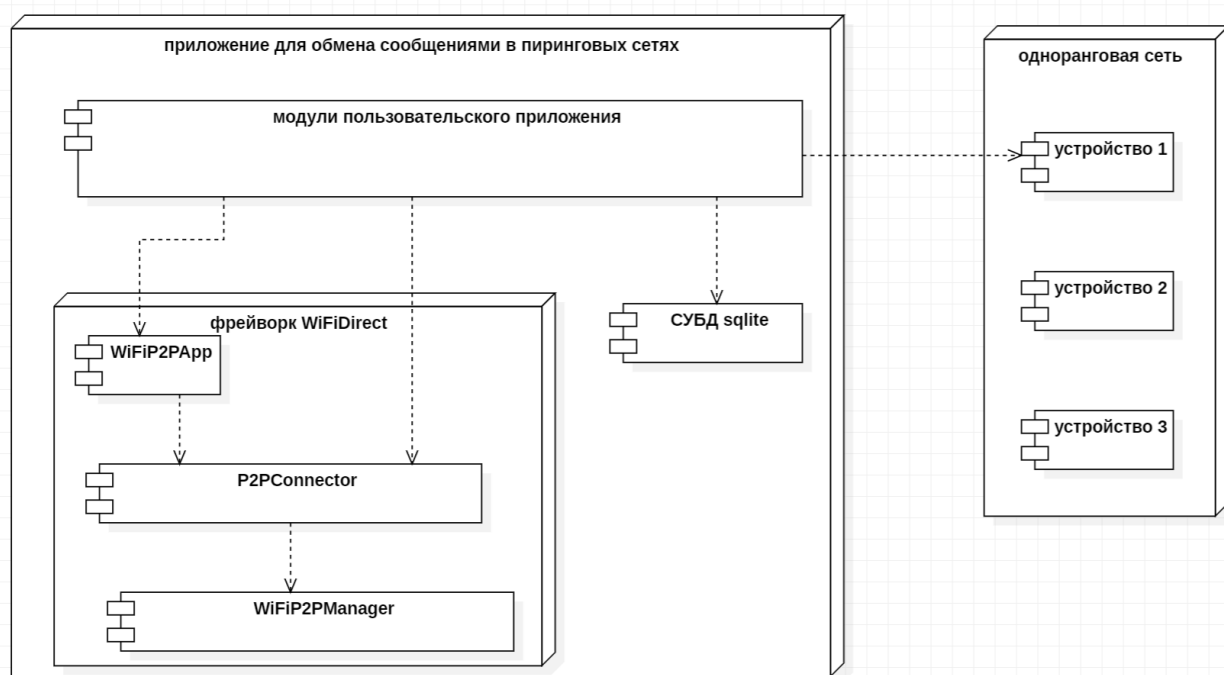


Рисунок 2.1 – Диаграмма развертывания приложения

На рисунке 2.1 показаны компоненты приложения, система окружения и их взаимодействие для реализации функционала приложения.

Приложение состоит из трех компонентов: пользовательские классы приложения, фреймворк для работы с функциями WiFi Direct, СУБД Sqlite.

Фреймворк состоит из трех наборов классов, каждый из которых выполняет определенную функцию: *WifiP2PManager*, *P2PConnector*, *WifiP2PApp*.

Набор классов *WifiP2PManager* является интерфейсом для сканирования устройств *Wi-Fi P2P*, содержит функции подключения, отключения и регистрации устройств *P2P*.

Набор классов *P2PConnector* является надстройкой над низкоуровневым пакетом *WifiP2PManager*. Пакет *P2PConnector* запрашивает у *WifiP2PManager* состояние соединения *Wi-Fi P2P* и получает от него сведения об обнаруженных службах *P2P*.

Набор классов *P2PConnector* работает на уровне абстракции поставщик, потребитель услуги. *P2PConnector* реализует канал связи с использованием TCP/IP.

Пакет *WifiP2PApp* - это набор классов, с помощью которого пользовательское приложение управляет *P2PConnector*. *WifiP2PApp* отображает состояние подключения *P2P Wi-Fi* и роль пользователя и взаимодействует с *P2PConnector*, когда пользователь выбирает стать потребителем или поставщиком.

Далее с помощью диаграмм последовательности опишем следующие этапы работы приложения: запуск приложения, подключение к сети в роли поставщика, подключение к сети в роли потребителя, организация канала TCP/IP для обмена файлами и сообщениями.

При запуске приложения и установке роли «Потребитель» (рисунок 2.2) происходит самотестирование приложения на предмет доступности модулей фреймворка и СУБД, от которых зависит работоспособность основного функционала приложения.

Тут же происходит чтение параметров настройки приложения. Затем при необходимости происходит запуск функций *Wi-Fi Direct*, по команде происходит передача управлению экземпляру пакета *WiFiP2PApp*, который запускает процесс определения роли и поиска устройств. Как только будет

определена роль – поставщик или потребитель услуг, будет запущен механизм сканирования доступных устройств.

Как только устройства будут обнаружены, список устройств, со служебной информацией (MAC адреса) будет передан по цепочке уровней сканирования для отображения в пользовательском приложении.

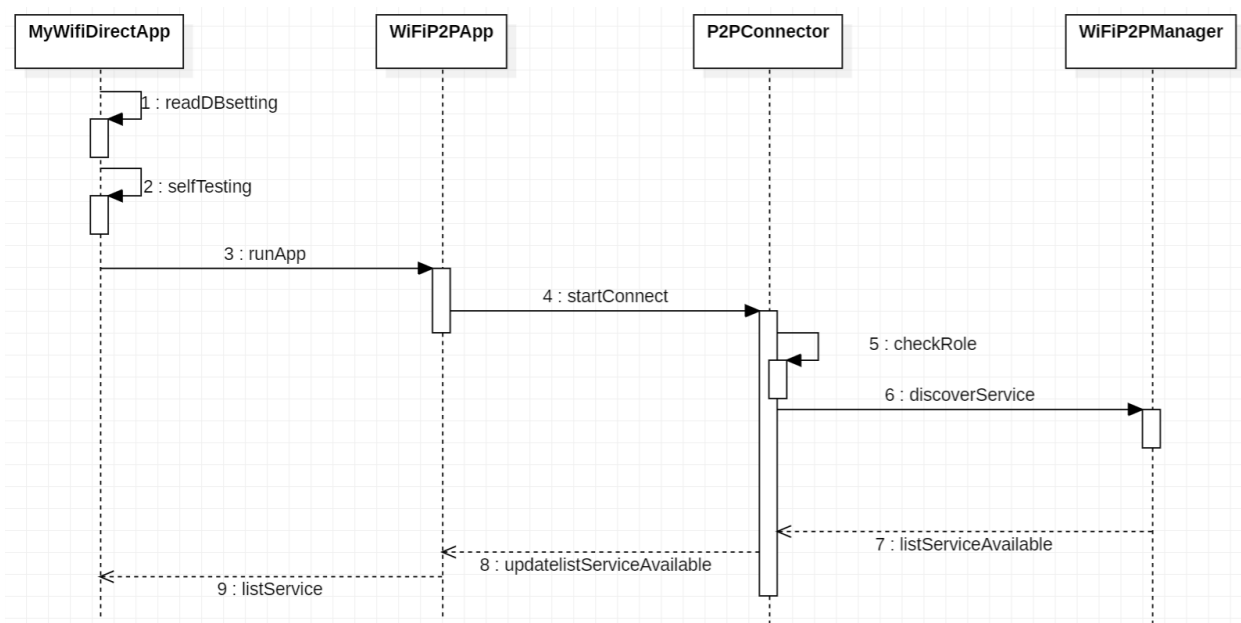


Рисунок 2.2 – Диаграмма последовательности, запуск приложения.

Роль «Потребитель»

При запуске приложения и установке роли «Поставщик» (рисунок 2.3) происходит самотестирование приложения на предмет доступности модулей фреймворка и СУБД, от которых зависит работоспособность основного функционала приложения. Затем приложение считывает параметры настройки приложения из СУБД. При необходимости происходит запуск функций *Wi-Fi Direct*, по команде происходит передача управлению экземпляру пакета *WiFiP2PApp*, который запускает процесс определения роли и поиска устройств. В роли «Поставщик» приложение производит поиск устройств с признаком «Потребитель». Как только такие устройства будут обнаружены экземпляр объекта *P2PConnector* производит соединение с устройством «Потребитель» по протоколу *WiFi Direct*, как только такое соединение

установлено все остальные действия по передачи данных можно осуществлять по протоколу *TCP/IP* так как вместе с признаком связи устройство «Потребитель» передает свой *IP* адрес и порт соединения.

На основании полученной информации объект *MyWiFiDirectApp* создает отдельную нить для работы с подключенным устройством «Потребитель».

При необходимости объект *MyWiFiDirectApp* может подать сигнал стоп для прекращения связи с устройством «Потребитель», освобождения памяти и прекращения связи по протоколам *TCP/IP* и *WiFi Direct*.

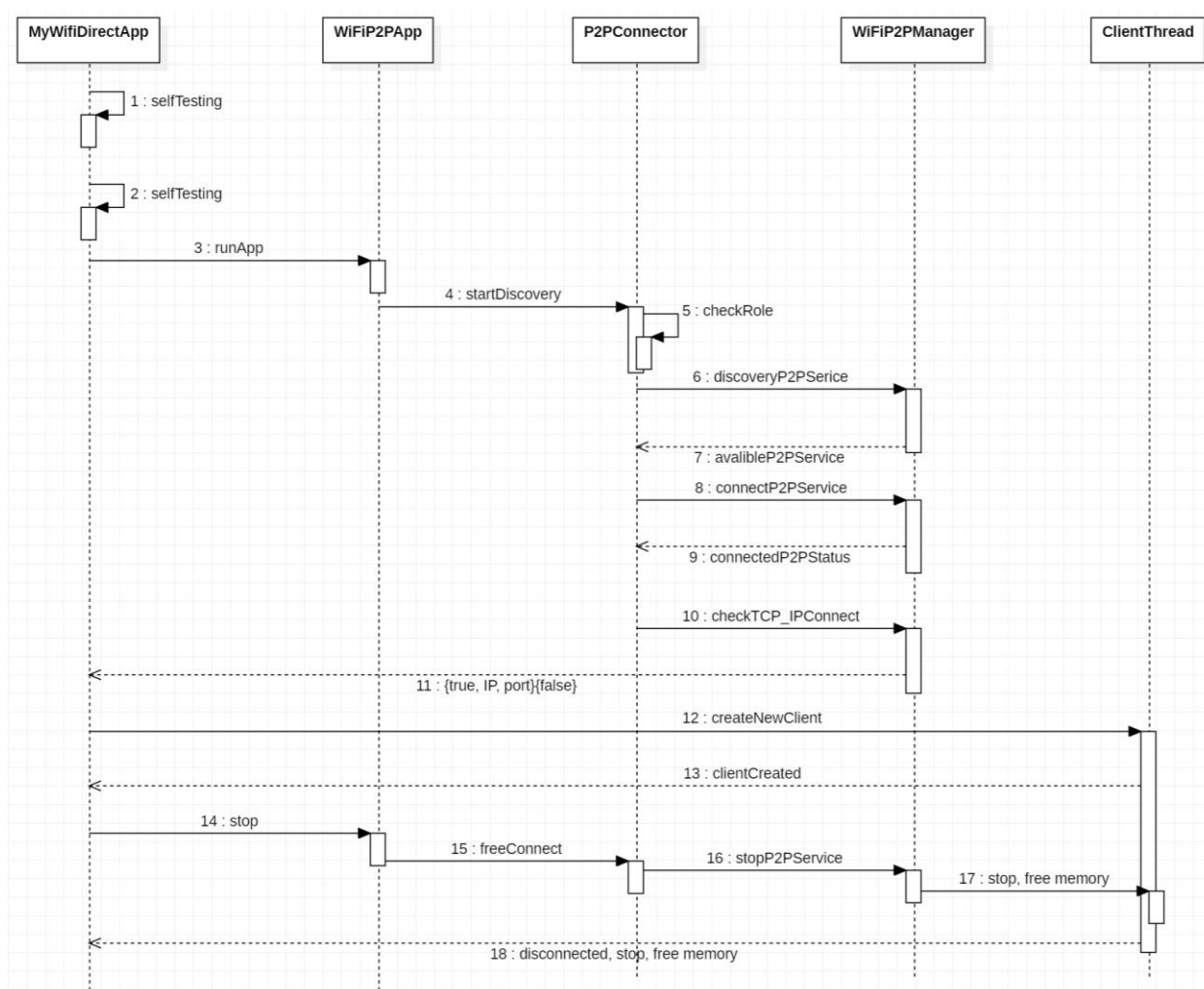


Рисунок 2.3 – Диаграмма последовательности, запуск приложения. Роль «Поставщик»

Как только устройство «Потребитель» установило связь с устройством «Поставщик» на уровне протокола *Wi-Fi Direct*, ему необходимо перейти в

режим работы с протоколом *TCP/IP*, для этого объект приложения *MyWiFiDirectApp* создает новую нить приложения и выполняет функции сокета, для работы в режиме сервер (рисунок 2.4).

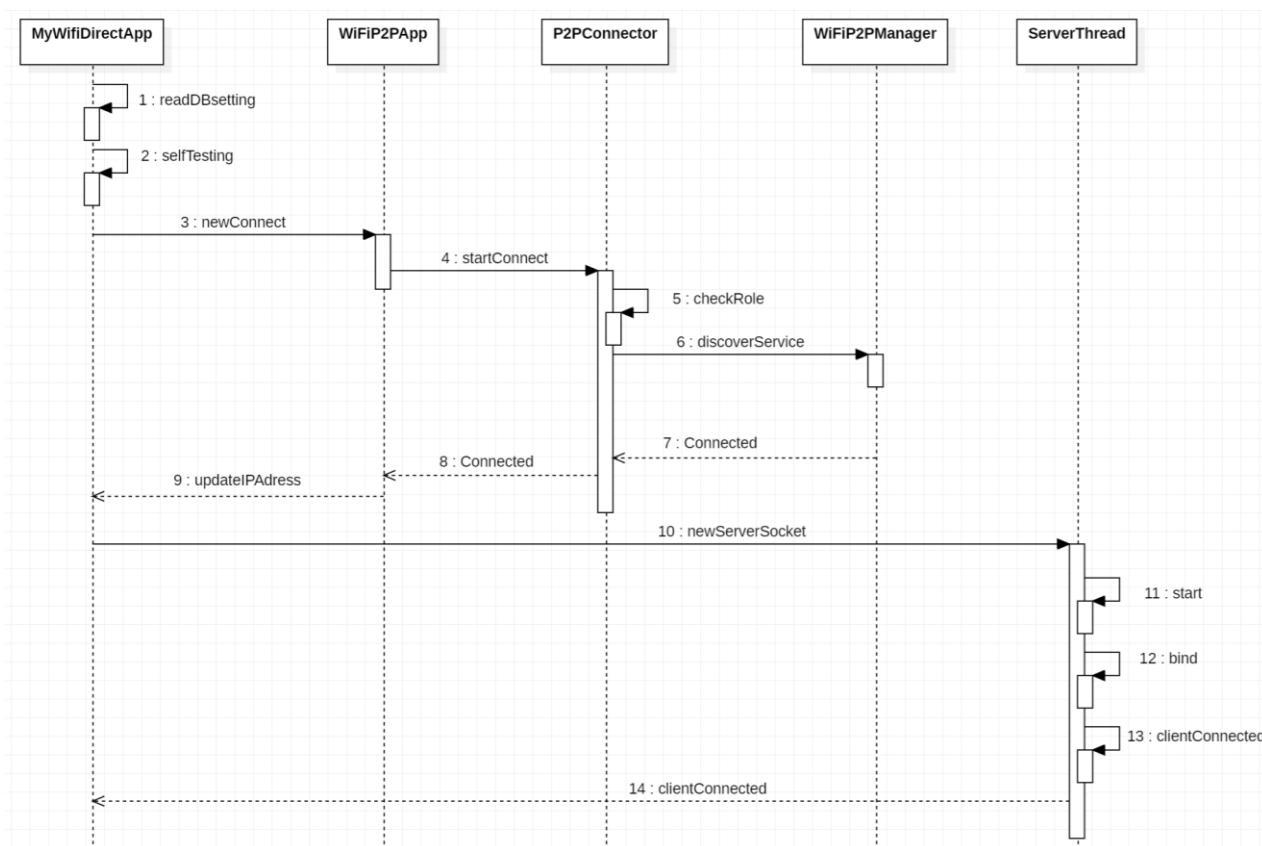


Рисунок 2.4 – Диаграмма последовательности переход устройства в режим *TCP/IP*. Роль «Потребитель»

Исходя из вышеописанного алгоритма работы приложения (рисунки 2.2-2.4) опишем работу приложения в режиме *FTP* сервера (рисунок 2.5) и в режиме приема передачи сообщений (рисунок 2.6).

Для работы в режиме передачи файлов необходимо произвести подключение двух устройств, одно в режиме «Поставщик», другое в режиме «Потребитель». После процедуры образования одноранговой сети на уровне протокола *Wi-Fi Direct*, оба устройства переходят в режим работы на уровне протокола *TCP/IP*, для чего обмениваются *IP* адресами и устанавливают единый порт для работы сокетов. При успешном соединении на уровне протокола *TCP/IP* устройство «Потребитель» получает файл от устройства

«Поставщик». Прием и передача файла осуществляется в отдельном потоке процесса программы.

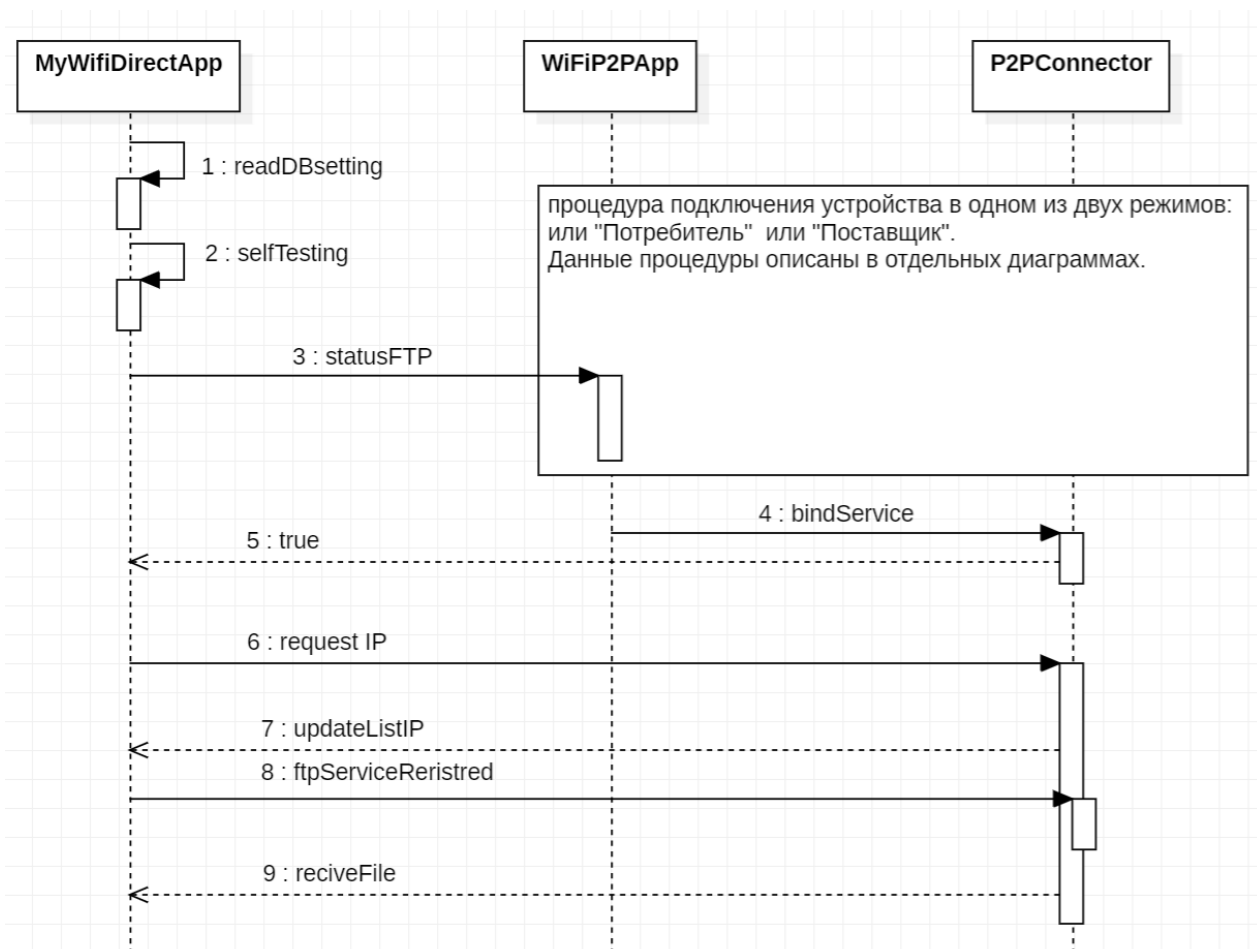


Рисунок 2.5 – Диаграмма последовательности режим работы «FTP сервер»

В режиме передачи текстовых сообщений (рисунок 2.6) шаги подключения и установления соединения такие же как и описанных режимах работы (рисунок 2.5).

После установления соединения, программа создает две параллельно работающие нити, одна на прием сообщений, другая на отправку сообщений.

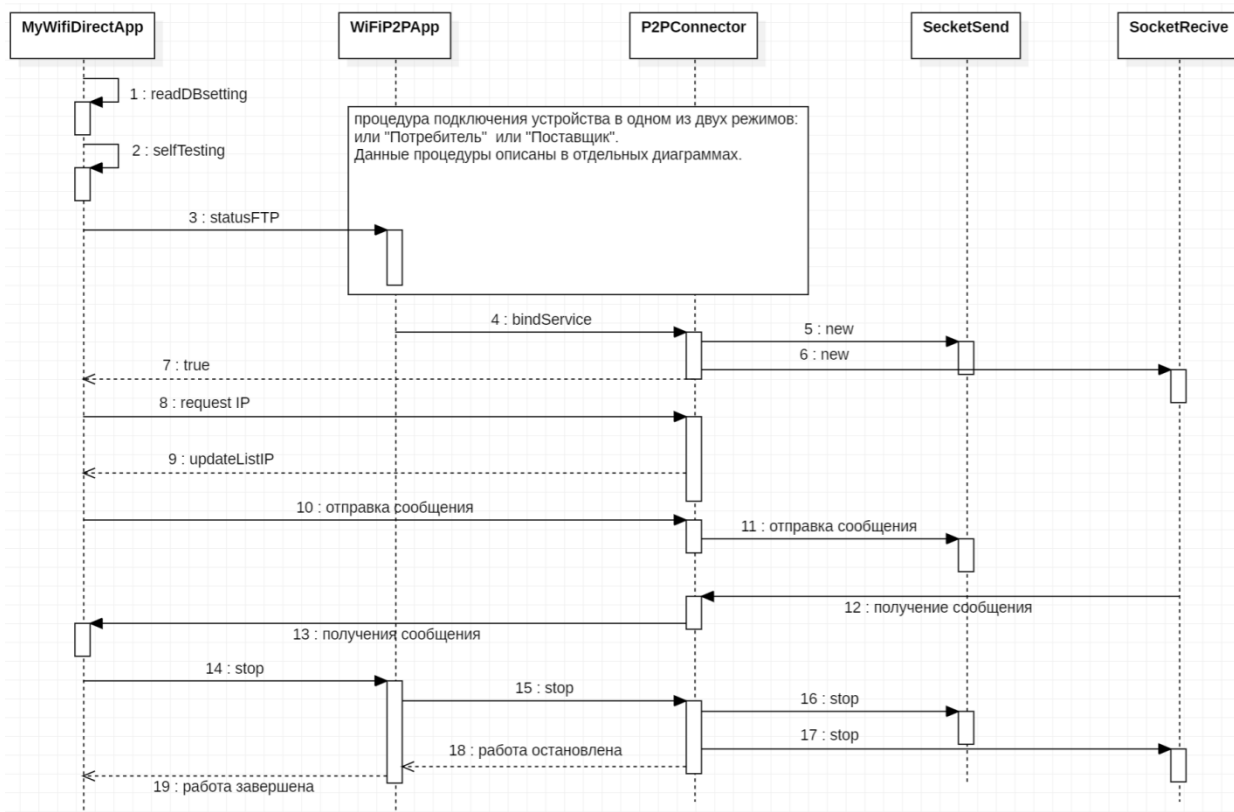


Рисунок 2.6 – Диаграмма последовательности режим работы
«Передача сообщений»

1.2.2 Проектирование интерфейса мобильного приложения

Пользовательский интерфейс приложения (рисунок 2.7- 2.11) - это все, что пользователь приложения может видеть и с чем взаимодействовать. Интерфейс устанавливает связь между приложением и пользователями, что, следовательно, делает важным этап разработки интерфейса при проектировании приложения. Пользовательский интерфейс гарантирует, что пользователи приложения будут взаимодействовать с функционалом приложения.

Пользовательский интерфейс в основном направлен на обеспечение легкого и приятного взаимодействия между приложением и его пользователями. Процесс разработки пользовательского интерфейса включает в себя выбор схемы перехода между диалоговыми окнами, элементов управления.

На рисунках 2.8 – 2.11 представлены макеты диалоговых окон реализуемого приложения.

При запуске приложения должно открываться главное окно приложения (рисунок 2.7), на котором должна отражаться информация об устройстве: MAC адрес, IP-адрес, имя устройства.



Рисунок 2.7 – Главное окна приложения

Ниже панели с информацией об устройстве находится панель управления, на которой располагаются кнопки перехода в окна, каждая из которых отвечает за определенную функцию приложения:

- кнопка «Сканирование устройств» - отвечает за запуск и отражение результата сканирования окружающего пространства по протоколу Wi-Fi Direct с целью нахождения устройств поддерживающих связь по данному протоколу;

- кнопка «Соединение с устройством» - отвечает за соединение с выбранным устройством по протоку *TCP/IP* и отражение результата соединения;
- кнопка «Передать файл» - отвечает за работу устройства в режиме FTP;
- кнопка «Передать сообщение» - отвечает за работу устройства в режиме чата;
- кнопка «Выйти из подключений» - отвечает за отключение всех подключенных устройств и закрытие возможности подключения по протоколам Wi-Fi Direct и TCP/IP.

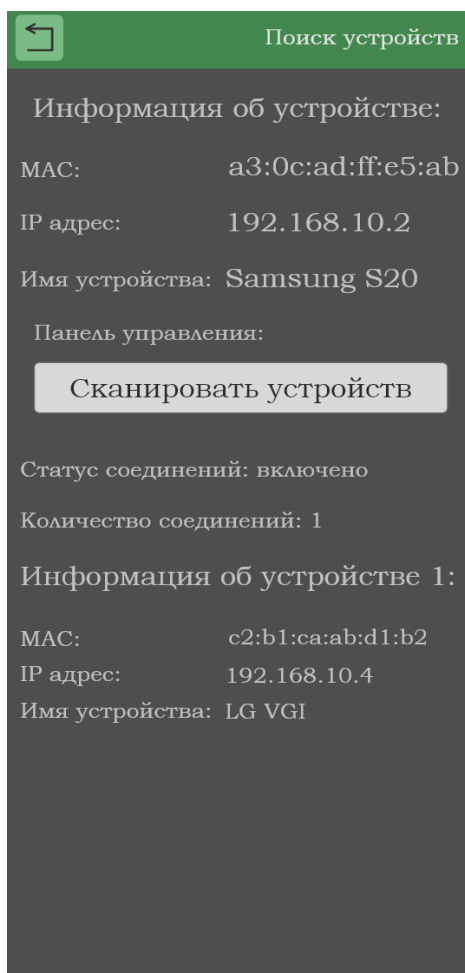


Рисунок 2.8 – Окно приложения «Поиск устройств»

Окно приложения «Поиск устройств» (рисунок 2.8) должно содержать информацию названия окна, далее должно отображать информацию о текущем устройстве: *MAC* адрес, *IP*-адрес, имя устройства.

Ниже информации о текущем устройстве находится панель управления с кнопкой «Сканировать устройства», при нажатии на которую по протоколу Wi-Fi Direct производится сканирование пространства.

При обнаружении устройств, на информационной панели появляется информация о потенциальных клиентах, которые пронумерованы в порядке их регистрации. Помимо регистрационного номера устройства, отображается информация потенциального клиента: *MAC* адрес, *IP*-адрес, имя устройства.

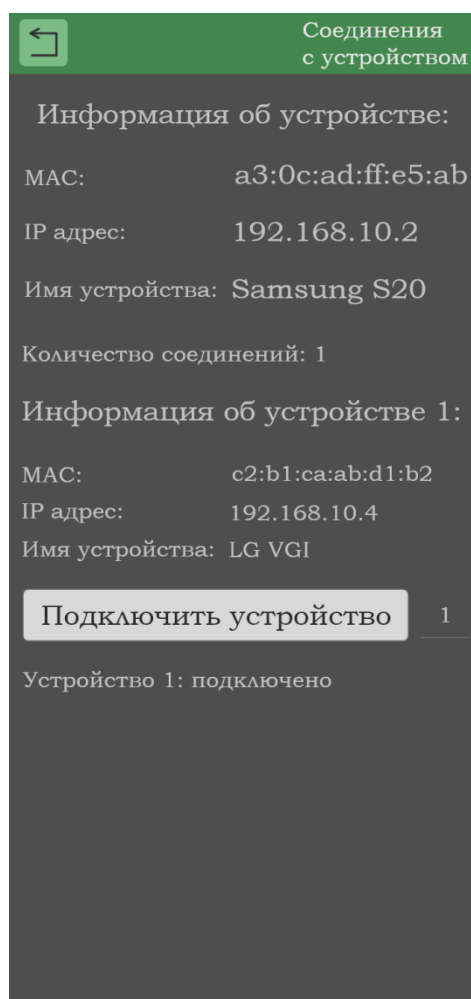


Рисунок 2.9 – Окно приложения «Установить соединение с устройством»

Окно приложения «Установить соединение с устройством» (рисунок 2.9) должно содержать информацию названия окна, далее должно отображать информацию о текущем устройстве: *MAC* адрес, *IP*-адрес, имя устройства.

Ниже информационной панели находится панель управления, на которой с помощью поля ввода можно выбрать регистрационный номер

одного из зарегистрированных потенциальных клиентов (рисунок 2.8). После ввода регистрационного номера и нажатия на кнопку «Подключить устройство», отображается статус: устройство «регистрационный номер» подключено и появляется в режиме «только для чтения» информация о подключенном клиенте: *MAC* адрес, *IP*-адрес, имя устройства.

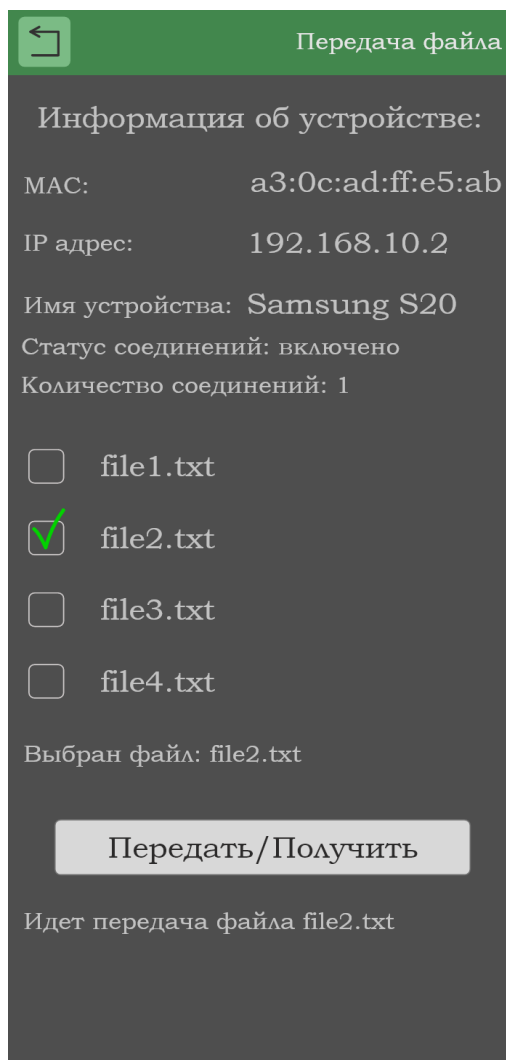


Рисунок 2.10 – Окно приложения «Установить соединение с устройством»

Окно приложения «Передать файл» (рисунок 2.10) должно содержать информацию названия окна, далее должно отображать информацию о текущем устройстве: *MAC* адрес, *IP*-адрес, имя устройства.

Ниже информационной панели находится панель управления, на которой находится элемент работы с файловой системой *OC Android* – *OpenFileDialog*. Данный элемент позволяет выбрать требуемый для передачи

файл, если устройство в режиме «Поставщик» или посмотреть полученный по сети файл, если устройство в режиме «Потребитель».

После выбора файла информация о наименовании файла отражается в отдельной строке в режиме «только для чтения». С помощью кнопки «Передать файл» устройство, находящееся в режиме «Поставщик», может передать файл другому устройству.

Если устройство находится в режиме «Потребитель» кнопка «Передать файл» находится в состоянии недоступно.

Окно приложения «Передать сообщение» (рисунок 2.11) должно содержать информацию названия окна, далее должно отображать информацию о текущем устройстве: *MAC* адрес, *IP*-адрес, имя устройства.

На панели «Передать сообщение» находится текстовое поле для ввода сообщения для отправки. Размер передаваемого сообщения ограничен 64 символами. Введенное сообщение будет отправлено при нажатии кнопки «Передать сообщение».

На панели «Получить сообщение » находится текстовое поле для полученного от другого устройства сообщения. Данное текстовое поле находится в режиме «только для чтения» и имеет ограничение – полученное сообщение должно содержать не более 64 символа.

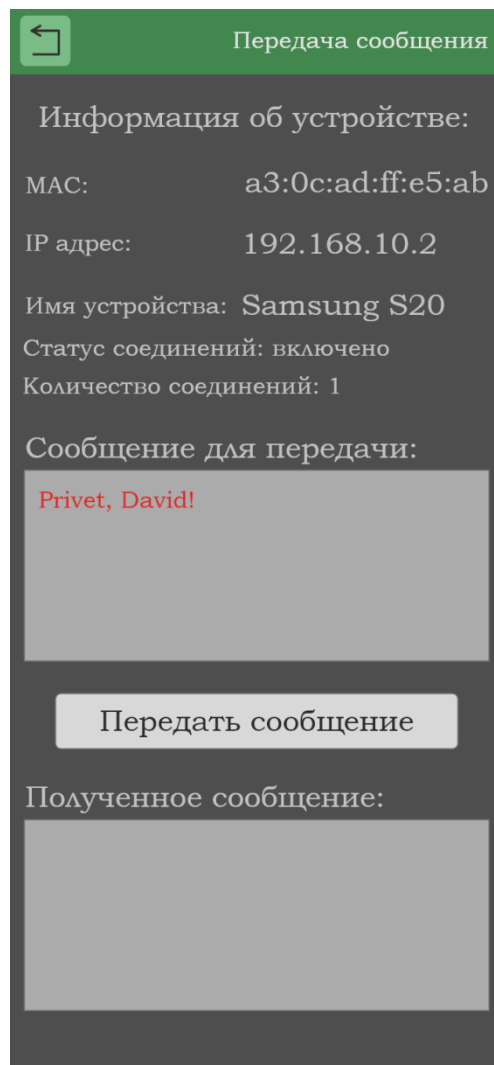


Рисунок 2.11 – Окно приложения «Установить соединение с устройством»

1.2.3 Проектирование базы данных

Проектирование базы данных - это совокупность процессов, которые включают проектирование, разработку, внедрение и обслуживание систем управления данными. Правильно спроектированные базы данных просты в обслуживании, повышают согласованность данных.

Основными задачами проектирования базы данных: создание логических и физических проектных моделей предлагаемой системы БД.

Логическая модель концентрируется на требованиях к данным и данным, подлежащих хранению, независимо от физических соображений. Это не касается того, как будут храниться данные или где они будут храниться физически.

Модель физического проектирования данных включает в себя перевод логического проектирования базы данных на физический уровень с использованием аппаратных ресурсов и свойств программных систем, включая типы данных.

1.2.3.1 Логическая модель разрабатываемой базы данных

Таблицы БД для хранения информации о файлах необходимы для хранения информации о том, какое устройство передало файл.

Таблицы БД для хранения информации о действиях позволяют сохранить информацию о том, какое именно действие совершало то или иное устройство: отправка файла, отправка или получение сообщения.

Таблица 2.1 – Атрибуты сущности «Сервер»

Наименование атрибута	Тип	Обязательный
Идентификатор устройства	Целый	Да
Наименование устройства	Строка	Да
IP адрес устройства	Целый	Нет
MAC адрес устройства	Целый	Да
Тип устройства	Строка	Да

Таблица 2.2 – Атрибуты сущности «Файл»

Наименование атрибута	Тип	Обязательный
Идентификатор файла	Целый	Да
Наименование файла	Строка	Да
Размер файла	Целый	Нет
Тип файла	Строка	Нет
Дата создания	Временная метка	Нет
Идентификатор устройства	Целый	Да
Путь к файлу	Строка	Нет
Состояние (доступен / недоступен)	Логический	Да

Таблица 2.2 – Атрибуты сущности «Файл»

Время изменения состояния (доступен / недоступен)	Временная метка	Нет
Комментарий	Строка	Нет

Таблица 2.3 – Атрибуты сущности «Логирование»

Наименование атрибута	Тип	Обязательный
Идентификатор записи лога	Целый	Да
Идентификатор сервера	Целый	Да
Идентификатор события для логирования	Целый	Да
Время создания записи лога	Временная метка	Да

Таблица 2.4 – Атрибуты сущности «Событие для логирования»

Наименование атрибута	Тип	Обязательный
Идентификатор события	Целый	Да
Идентификатор записи лога с которым связано событие	Целый	Да
Время возникновения события	Временная метка	Нет
Описание события	Строка	Да

Реализованная на основании логической модели физическая модель базы данных может содержать дополнительные служебные, промежуточные таблицы, для связи логических сущностей.

1.2.3.2 Физическая модель разрабатываемой базы данных

Физическая модель данных (рисунок 2.12) - это специфичная для базы данных модель, которая представляет реляционные объекты данных (например, таблицы, столбцы, первичный и внешние ключи) и их взаимосвязи (таблицы №№).

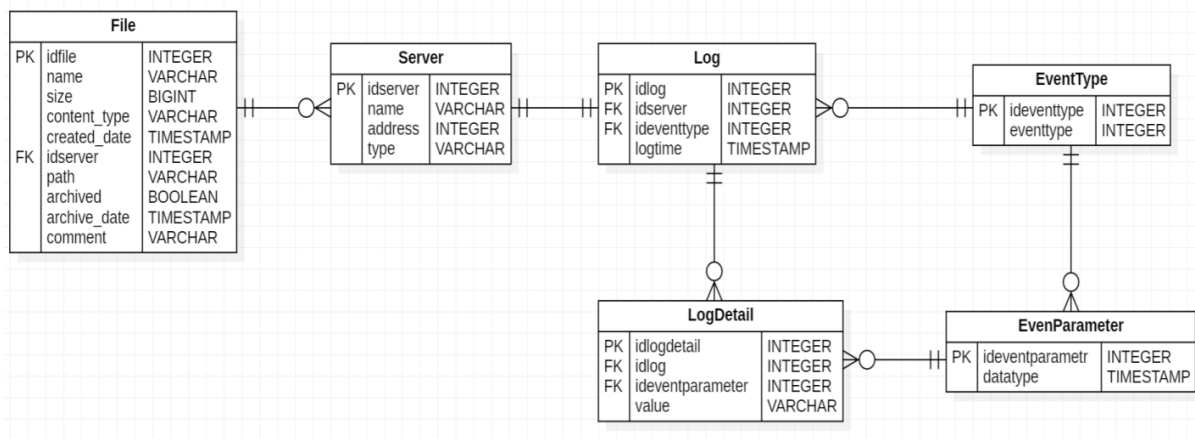


Рисунок 2.12 – Таблицы БД для хранения информации о файлах

Таблица 2.5 – Описание полей таблицы «Server»

Имя поля	Тип	Примечание
<i>idserver</i>	<i>integer</i>	РК, обязательный
<i>name</i>	<i>varchar(255)</i>	Обязательный
<i>IPadress</i>	<i>integer</i>	Не обязательный
<i>MAC</i>	<i>integer</i>	Обязательный
<i>type</i>	<i>varchar(255)</i>	Обязательный

Таблица 2.6 – Описание полей таблицы «File»

Имя поля	Тип	Примечание
<i>idfile</i>	<i>integer</i>	РК, обязательный
<i>name</i>	<i>varchar(255)</i>	Обязательный
<i>size</i>	<i>bigint</i>	Не обязательный
<i>content_type</i>	<i>varchar(255)</i>	Не обязательный
<i>created_date</i>	<i>timestamp</i>	Не обязательный

Таблица 2.6 – Описание полей таблицы «File»

<i>idserver</i>	<i>integer</i>	FK, обязательный
<i>path</i>	<i>varchar(255)</i>	Не обязательный
<i>archived</i>	<i>boolean</i>	Обязательный
<i>artchived_date</i>	<i>timestamp</i>	Не обязательный
<i>comment</i>	<i>varchar(255)</i>	Не обязательный

Таблица 2.7 – Описание полей таблицы «Log»

Имя поля	Тип	Примечание
<i>idlog</i>	<i>integer</i>	РК, обязательный
<i>idserver</i>	<i>integer</i>	FK, обязательный
<i>ideventtype</i>	<i>integer</i>	FK, обязательный
<i>logtime</i>	<i>timestamp</i>	Обязательный

Таблица 2.8 – Описание полей таблицы «LogDetail»

Имя поля	Тип	Примечание
<i>idlogdetail</i>	<i>integer</i>	РК, обязательный
<i>idlog</i>	<i>integer</i>	FK, обязательный
<i>ideventparameter</i>	<i>integer</i>	FK, обязательный
<i>value</i>	<i>varchar(255)</i>	Обязательный

Таблица 2.9 – Описание полей таблицы «EventType»

Имя поля	Тип	Примечание
<i>ideventtype</i>	<i>integer</i>	РК, обязательный
<i>eventtype</i>	<i>integer</i>	Обязательный

Таблица 2.10 – Описание полей таблицы «EventParameter»

Имя поля	Тип	Примечание
<i>ideventparameter</i>	<i>integer</i>	РК, обязательный
<i>datatype</i>	<i>timestamp</i>	Обязательный

Экземпляры событий и экземпляры записей логов образом будут храниться в двух таблицах, для повышения возможностей масштабирования БД при добавлении дополнительных словарей и функционала.

1.2.4 Реализация мобильного приложения

Согласно модели предметной области, была спроектирована диаграмма программных классов, представленная на рисунке 2.13.

Условно классы в данном мобильном приложении можно разделить на две группы: классы, наследуемые от класса *AppCompatActivity* (так называемы

активности) и все остальные. Краткое описание представленных на диаграмме классов приведено в таблице 2.11.

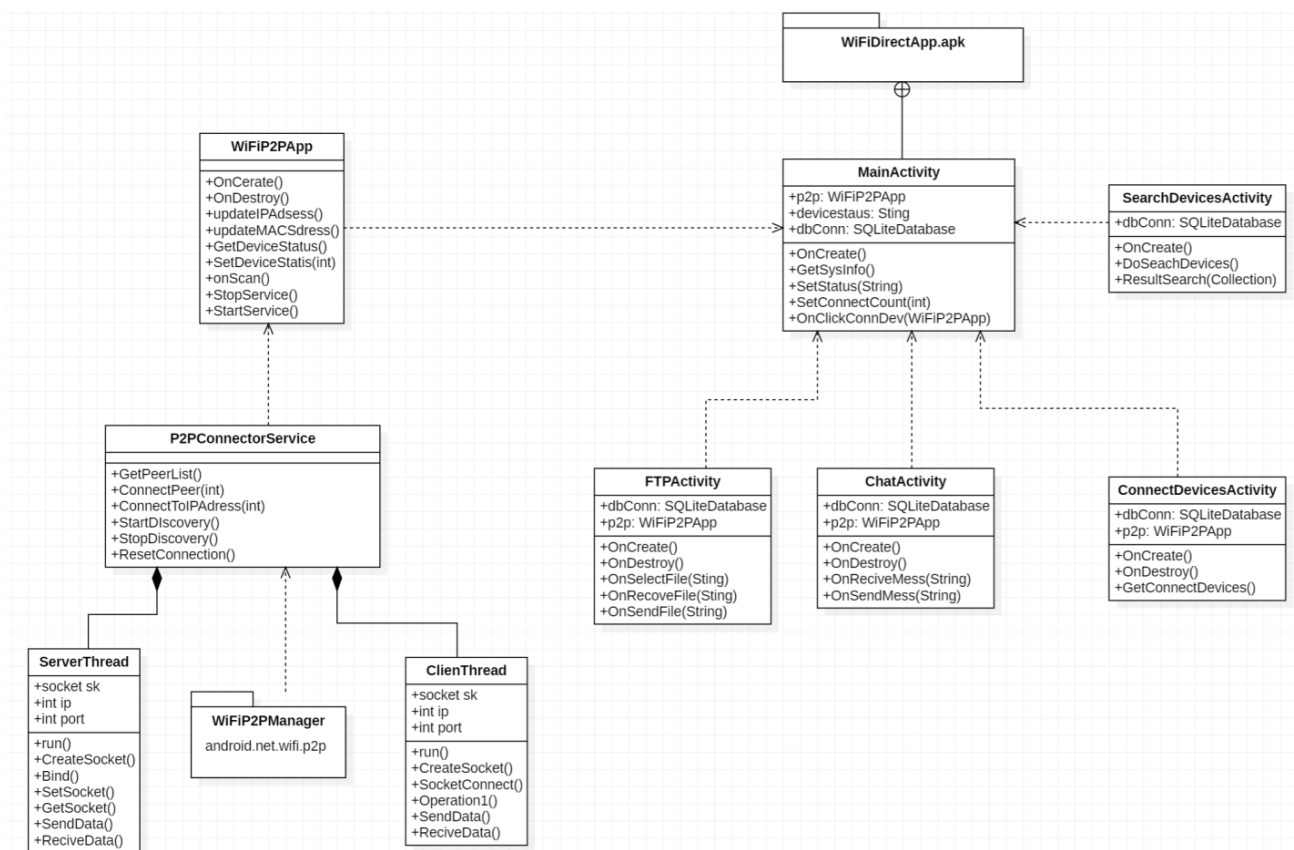


Рисунок 2.13 – Диаграмма классов программы

Таблица 2.11 – Описание классов программы

Класс	Описание
<i>MainActivity</i>	Точка входа в программу. Класс главного окна приложения. На окне находятся информационные поля об устройстве и кнопки перехода на рабочие окна приложения
<i>FTPActivity</i>	Класс окна приложения, приема и передачи файлов. На окне находятся информационные поля для работы с файловой системой устройства и кнопки приема или передачи файлов в зависимости от режима работы программы.

Таблица 2.11 – Описание классов программы

<i>ChatActivity</i>	Класс окна приложения, приема и передачи сообщений. На окне находятся информационные поля для работы с полученными и переданными сообщениями и кнопка передачи сообщений.
<i>ConnectDevicesActivity</i>	Класс окна приложения, для поиска потенциальных клиентов приложения.
<i>WiFIP2PApp</i>	<p>Класс <i>WifiP2PApp</i> получает состояние подключения <i>P2P Wi-Fi</i> через пакет <i>Android WifiP2PReceiver</i> (реализующий <i>Android BroadcastReceiver</i>).</p> <p>В дополнение к состоянию подключения <i>WifiP2PApp</i> также читает <i>IP</i>-адрес текущего и подключенных устройств.</p> <p>Отвечает за состояние потребителя или поставщика.</p> <p>Отвечает за обнаружение <i>Wi-Fi P2P</i> (активное обнаружение или остановлено).</p>
<i>P2PConnectorService</i>	<p><i>P2PConnectorService</i> нижний слой абстракции инфраструктуры <i>P2P Wi-Fi</i>.</p> <p>Класс взаимодействует с пакетами <i>Android</i> через <i>IP2PConnectorService</i> и инкапсулирует в себе все низкоуровневые способы работы по протоколу <i>Wi-Fi Direct</i>.</p> <p>Сведения о состоянии <i>P2P Wi-Fi</i> получаются с помощью <i>P2PConnectorReceiver</i>, прослушивающего асинхронные обновления состояния <i>P2P Wi-Fi</i> от <i>WifiP2PManager</i>.</p> <p><i>P2PConnectorService</i> также создает</p>

	экземпляры
--	------------

Таблица 2.11 – Описание классов программы

	<i>ServerThread</i> и <i>ClientThread</i> в зависимости от роли устройства.
<i>ServerThread</i>	<p>Класс отвечает за поддержание TCP-соединения с подключенным устройством P2P.</p> <p>Устройство-потребитель создает экземпляр <i>ServerThread</i>.</p> <p>Класс инкапсулирует методы работы устройства в режиме <i>Wi-Fi P2P GO</i>.</p>
<i>ClientThread</i>	<p>Класс отвечает за поддержание TCP-соединения с подключенным устройством P2P.</p> <p>Устройство-поставщик создает экземпляр <i>ClientThread</i>.</p>

В ходе разработки мобильного приложения будет реализовано 8 классов, часть из которых отвечает за визуальную функциональность приложения, другая часть за работу с данными предметной области.

1.2.5 Выводы по главе

В результате работ проведенных в рамках второй главы реализованы:

— мобильное приложение в соответствии с техническим заданием; базы данных, необходимая для функционирования и логирования приложения.

2.Технология отладки программного обеспечения в *IDE Android Studio*

Реализованное приложение для обмена сообщениями в пиринговых сетях в процессе разработки требовало инструментов для отладки, чтобы отследить возникающие в процессе реализации приложения ошибки. В данном разделе приведена работа с инструментами отладчика IDE

IDE Android Studio позволяет производить отладку приложения запущенного в эмуляторе или устройстве под управлением *OC Android*.

Для этого *IDE Android Studio* позволяет:

- выбрать устройство для отладки приложения;
- увидеть данные системных журналов;
- установить точки останова в разрабатываемом коде;

Чтобы отладить приложение, *IDE Android Studio* создает отладочную версию приложения, подключает его к устройству или эмулятору, устанавливает приложение и запускает его. *IDE Android Studio* показывает системный журнал пока приложение запущено и предоставляет средства отладки для фильтрации сообщений журнала.

2.1 Запуск режима отладки в *IDE Android Studio*

Чтобы запустить приложение в режиме отладки, требуется создать файл с расширением *APK*, подписанный ключом отладки, и установить его на физическое устройство *Android* или на эмулятор *Android*. Далее для старта отладки разрабатываемого приложения в *Android Studio* требуется :

- Открыть проект в *Android Studio*;
- Нажать кнопку «Отладка»  на панели инструментов;
- В окне «Выбор устройства» выбрать аппаратное устройство из списка или выберите виртуальное устройство (рисунок 3.1);
- Нажать кнопку «ОК». Выбранное приложение запустится на выбранном устройстве;

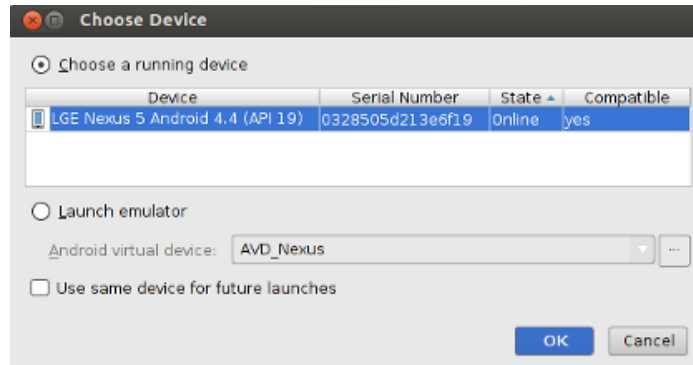


Рисунок 3.1 - Окно выбора устройства

В списке (рисунок 3.1) показаны все доступные устройства *Android* подключенные к компьютеру. Так как отладка планируется на уровне эмуляции, необходимо выбрать «Запустить эмулятор», чтобы использовать виртуальное устройство *Android*. Щелкните по многоточию  чтобы открыть диалог «Диспетчер виртуальных устройств» *Android*.

IDE Android Studio открывает окно инструмента «Отладка». Чтобы открыть окно «Отладка», нажмите кнопку «Отладка»  **5: Debug**. В окне отладки отображаются (рисунок 3.2):

- потоки и переменные;
- состояние устройства;
- консоль и системный терминал для работы с командами;
- системный журнал, журналы логирования.

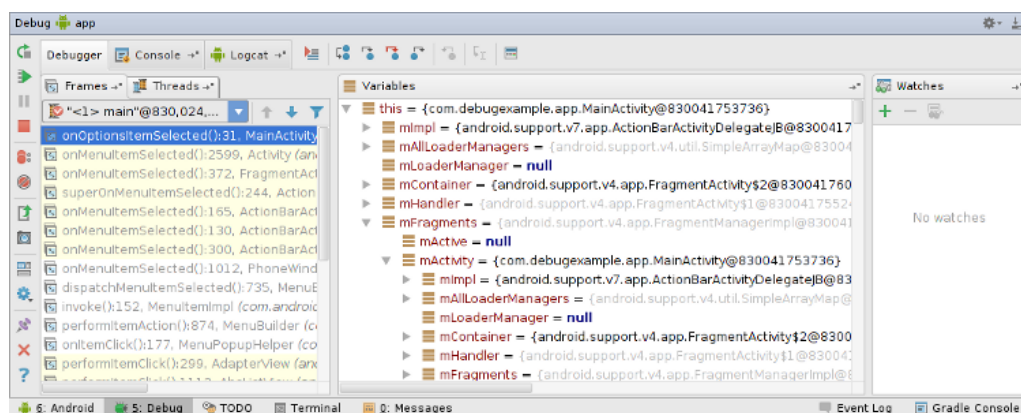


Рисунок 3.2 - Окно отладки кода приложения

2.2 Подключение отладчика к запущенному процессу

Для отладки уже запущенного процесса приложения подключенного к IDE перезапуск приложения не требуется. Для отладки приложения, которое уже в режиме исполнения требуется:

- нажать кнопку «Прикрепить отладчик к процессу» *Android*  ;
- в окне «Выбор процесса» выбрать устройство и приложение, к которым требуется подключить отладчик;
- открыть окно инструмента *Debug* для трассировки кода требуется нажать кнопку «Отладка» .

2.3. Просмотр общих сведений в системных журналах

В системном журнале отображаются системные сообщения во время отладки приложения. Эти сообщения включают в себя информацию из приложения, запущенного на устройстве. Для использования системного журнала для отладки приложения требуется сделать так, чтобы код записывал сообщения журнала и формировал исключения во время отладки приложения.

Для записи сообщения в журнал в своем коде, используйте класс *Log*. Регистрация системных сообщений помогает анализу процесса выполнения, собирая выходные данные отладки системы во время взаимодействия с библиотеками *API*. Сообщения журнала могут указать, какой именно модуль приложения работает с ошибкой.

Далее показаны участки кода приложения, которые добавляют функционал логирования в системный журнал.

```

import android.util.Log;
...
public class MyActivity extends Activity {
    private static final String TAG = MyActivity.class.getSimpleName();
    ...
    @Override
    public void onCreate(Bundle savedInstanceState) {
        if (savedInstanceState != null) {
            Log.d(TAG, "onCreate() Restoring previous state");
            /* restore state */
        } else {
            Log.d(TAG, "onCreate() No saved state available");
            /* initialize app */
        }
    }
}

```

Во время разработки код приложения может перехватывать исключения и записывать трассировку стека в систему журнал, как показано в следующем примере кода:

```

void someOtherMethod() {
    try {
        ...
    } catch (SomeException e) {
        Log.d(TAG, "someOtherMethod()", e);
    }
}

```

2.4 Просмотр системного журнала для выбранного процесса

Инструмент отладки *Android DDMS (Debug Monitor Server)* и инструмент отладки *Debug* показывают общие сведения системного журнала; однако окно инструмента *Android DDMS* позволяет просматривать только сообщения журнала для конкретного процесса.

Для просмотра системного журнала в окне инструмента *Android DDMS* требуется:

- запустить приложение, в режиме отладки;
- нажать кнопку *Android*  *Android* для запуска отдельного окна

Android DDMS и панели инструментов;

- если системный журнал пуст, требуется нажать кнопку  Перезапустить, для его формирования и наполнения информацией.

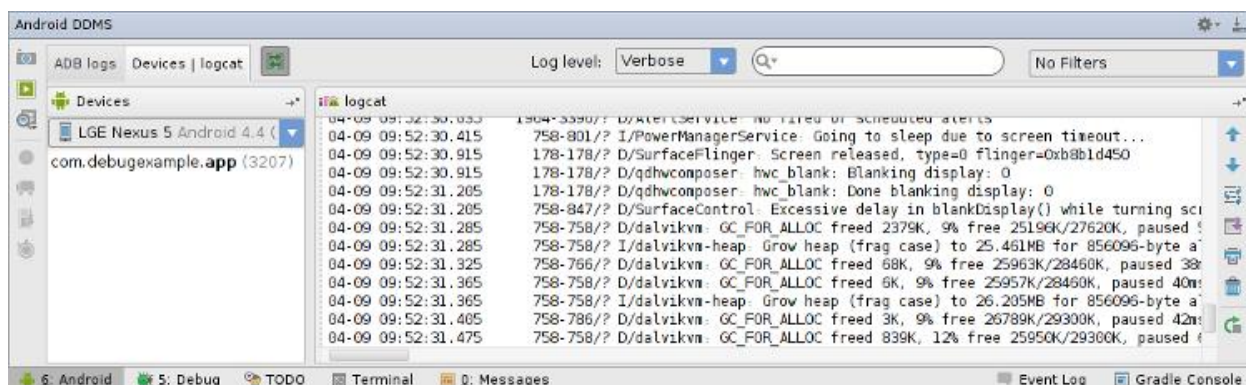


Рисунок 3.3 - Окно инструмента *Android DDMS*

Окно инструментов *Android DDMS* предоставляет доступ к специальным функциям отладки в *Android Studio*.

В системном журнале отображаются сообщения от служб ОС *Android* и других приложений ОС *Android*. Для фильтрации журнала сообщений, для просмотра только необходимых данных, необходимо использовать инструменты в *Android DDMS*:

- требуется выбрать процесс в окне «Просмотр устройств»;
- затем требуется нажать кнопку «Показывать Logcat» ;
- если статус устройства «Недоступен», необходимо нажать кнопку «Восстановить просмотр устройств»;
- далее требуется фильтровать сообщения в журнале, для чего выбрать нужный журнал из окна *Android DDMS*;
- Нажмите кнопку «Ввод» и просмотрите полученный результат.

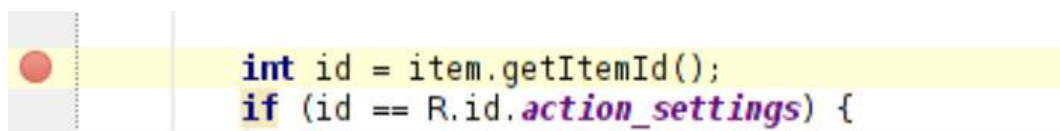
2.5 Способы работы с точками останова

Точки останова позволяют приостановить выполнение приложения на определенной строке кода, изучить переменные и затем продолжить выполнение построчно. Использование точек останова, необходимо для

определения причины ошибок во время выполнения, которые трудно исправить, просмотрев только код приложения.

Для использования точки останова необходимо выполнить следующие действия:

- открыть исходный файл, в котором необходимо установить точку останова,
- найти строку, в которой хотите установить точку останова, и щелкнуть по ней левой кнопкой мыши,
- нажать на желтую часть боковой панели слева от этой строки,
- запустить приложение в режиме отладки.



IDE Android Studio приостанавливает выполнение приложения, когда оно достигает точки останова. Тогда можно будет использовать инструменты в окне инструмента «Отладка», чтобы определить причину ошибки.

Чтобы просмотреть все точки останова и настроить параметры точки останова, нажмите кнопку «Просмотреть точки останова»  в левой части инструмента «Отладка». Появится окно точек останова, как показано на рисунке 3.3.

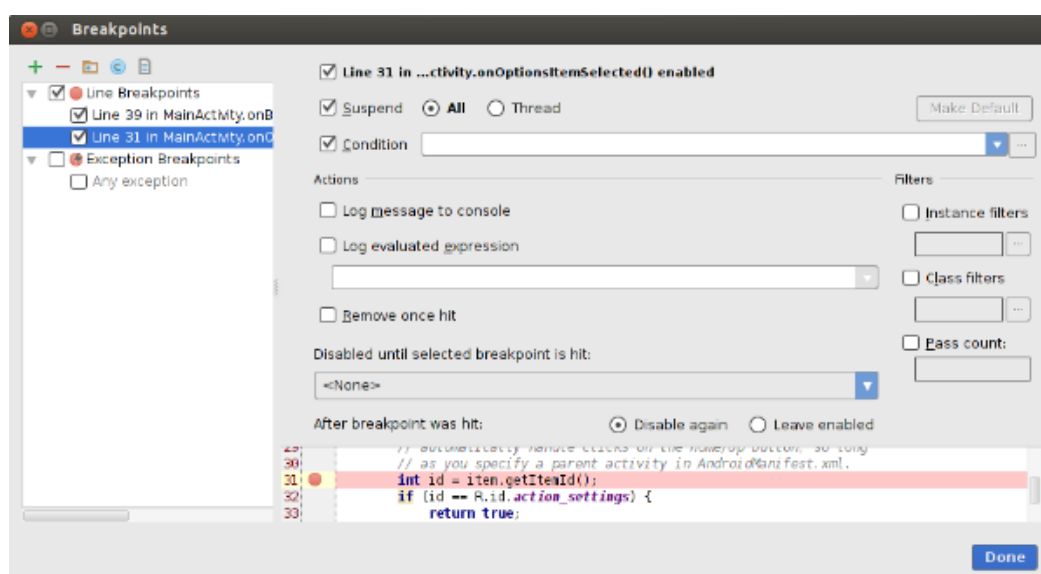


Рисунок 3.4 - Окно точек останова

В окне «Точки останова» перечислены все текущие точки останова, включая настройки поведения для каждой из них. Окно «Точки останова» позволяет включать или отключать каждую точку останова из списка. Если точка останова отключена, *IDE Android Studio* не приостанавливает приложение при ее достижении.

После того как точки останова установлены в коде приложения, нажмите кнопку «Повторить»  чтобы снова запустить приложение.

Когда точка останова достигнута *IDE Android Studio* приостанавливает работу приложения и выделяет точку останова в исходном коде. Инструменты в окне «Отладка» позволяют проверять переменные и управлять шагом выполнения с помощью выполнения следующих шагов:

- чтобы просмотреть дерево объектов на предмет наличия переменной, разверните его в представлении «Переменные». Если представление «Переменные» не отображается, нажмите кнопку «Восстановить представление переменных» ;

- чтобы вычислить выражение в текущей точке выполнения, нажмите кнопку «Вычислить выражение» ;

- чтобы перейти к следующей строке кода, нажмите кнопку «Переход» ;

- чтобы перейти к первой строке внутри метода, нажмите кнопку «Переход В» ;

- чтобы перейти к следующей строке за пределами текущего метода, нажмите «Перейти в...» ;

- чтобы продолжить обычную работу приложения, нажмите кнопку «Возобновить исполнение» .

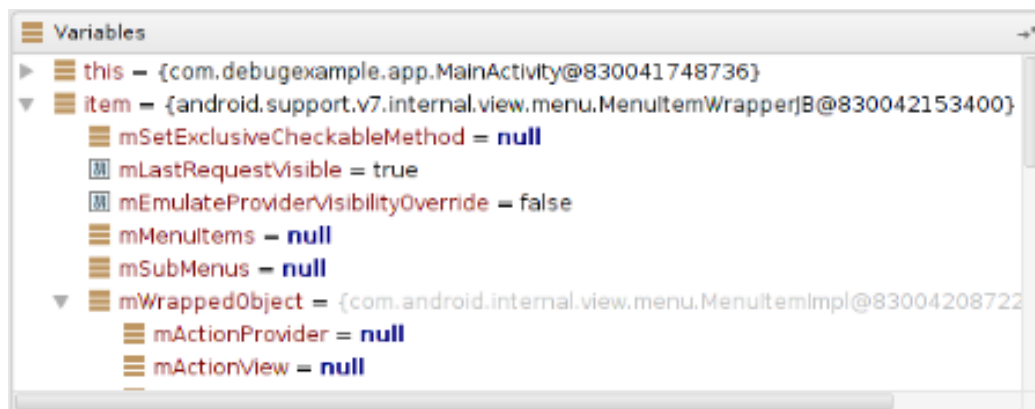


Рисунок 3.5 - Представление переменных в режиме отладки

IDE Android Studio позволяет отслеживать объекты, переменные и методы, которые выделяются в памяти виртуальной машины Java, и наблюдать, какие классы и потоки выделяют эти объекты (рисунок 3.5).

Это позволяет увидеть список объектов в течение интересующего периода исполнения кода. Эта информация нужна для оценки использования памяти, что может повлиять на производительность приложения. Для отслеживания распределения памяти объектов требуется:

- запустить приложение в режиме отладки;
- нажать кнопку «Android»  «Android», чтобы открыть окно инструментов «Android DDMS»;
- В окне инструмента «Android DDMS» выберите вкладку «Устройства Logcat»;
- выбрать нужное устройство из выпадающего списка;
- выбрать приложение по названию пакета из списка запущенных приложений;
- нажать кнопку «Начать отслеживание распределения памяти» ;
- собрать нужную информацию по отладке;
- нажать кнопку «Остановить отслеживание распределения» .

IDE Android Studio показывает объекты, которым система выделила следующую информацию: размер распределения памяти, идентификатор

потока, метод выделения, класс и номер строки, трассировка стека в точке выделения.

Allocation Order	Allocated Class	Allocation Size	Thread Id	Allocation Site
2162	java.util.Arrays\$ArrayList	16	1	java.util.Arrays.asList(Arrays.java:155)
2161	android.animation.Keyframe\$Float	32	1	java.lang.Object.internalClone(Native Method)
2160	java.lang.Object[]	64	1	java.util.ArrayList.addAll(ArrayList.java:198)
2159	java.util.HashMap	48	1	android.animation.ValueAnimator.setValues(ValueAnimator.java:49)
2158	java.util.HashMap\$HashMapEntry	32	1	java.util.HashMap.makeTable(HashMap.java:550)

```

at android.animation.KeyframeSet.ofFloat(KeyframeSet.java:97)
at android.animation.PropertyValuesHolder.setFloatValues(PropertyValuesHolder.java:600)
at android.animation.PropertyValuesHolder$FloatPropertyValuesHolder.setFloatValues(PropertyValuesHolder.java:1234)
at android.animation.PropertyValuesHolder$FloatPropertyValuesHolder.<init>(PropertyValuesHolder.java:1221)
at android.animation.PropertyValuesHolder.ofFloat(PropertyValuesHolder.java:281)
at android.animation.ObjectAnimator.setFloatValues(ObjectAnimator.java:716)
at android.animation.ObjectAnimator.ofFloat(ObjectAnimator.java:409)

```

Рисунок 3.6 – Окно отслеживания распределения памяти объектов

Даже если приложение не генерирует ошибки во время выполнения, это не означает, что оно не имеет семантических ошибок и ошибок уровня исполнения.

Во время тестирования и отладки приложения требуется создать тест лист и фиксировать в нем ответы на следующие вопросы о приложении:

- эффективность и оптимальность расходования памяти приложения;
- какой трафик генерирует приложение;
- методы приложения с минимальной скоростью исполнения.

Для анализа использования памяти, методов профилирования, слежения за сетевым трафиком и имитацией входящих звонков и сообщений имеется специальный инструмент «Монитор».

2.6 Фиксация скриншотов и видео в режиме отладки приложения

IDE Android Studio позволяет фиксировать снимок экрана или сделать видео с экрана устройства и запущенного приложения.

Скриншоты и видеоролики полезны в качестве технических материалов для команды разработки.

Чтобы сделать снимок экрана разрабатываемого приложения необходимо:

- запустить приложение в режиме отладки;
- нажать кнопку «Android»  6: Android чтобы открыть окно инструментов «Android DDMS»;
- нажать кнопку «Захват экрана»  на левой стороне окна инструментов «Android DDMS»;
- чтобы добавить рамку устройства вокруг снимка экрана, включите режим «Рамка снимка экрана»;
- Нажмите кнопку «Сохранить».

Чтобы сделать видеозапись приложения требуется:

- запустить приложение в режиме отладки;
- нажать кнопку «Android»  6: Android на левой стороне окна инструментов «Android DDMS»;
- нажать кнопку «Запись экрана»  на левой стороне окна;
- инструментов «Android DDMS»;
- нажать кнопку «Начать запись»;
- провести взаимодействие с приложением;
- нажать кнопку «Остановить запись»;
- ввести имя файла для записи и нажмите кнопку «ОК».

2.7 Выводы по главе

В результате данной главы, описана технология отладки, примененная при реализации приложения реализованного в рамках дипломной работы.

3. Экономическое обоснование эффективности проекта

Разработка программного обеспечения осуществлена на базе собственной площади 40 кв.м. Содержание помещения в месяц составляет 6 000 рублей.

Для разработки программного обеспечения приобретена вычислительная техника, лицензионное программное обеспечение и расходные материалы (обучающие материалы).

Данные по потребности в оборудовании и лицензионном программном обеспечении представлены соответственно в таблице 4.1 и 4.2.

Таблица 4.1 – Потребность в оборудовании

Оборудование	Организация	Кол-во	Ориентировочная цена за единицу, руб
Компьютер самостоятельная сборка	ООО «Технолинк»	5	60000
МФУ лазерный Conica 1100	ООО «Ситилинк»	1	11500
Клавиатура и мышь	ООО «ДНС»	5	1000
Монитор LG 27 дюймов	ООО «ДНС»	5	27500

Таблица 4.2 – Потребность в лицензионном программном обеспечении

Лицензионное программное обеспечение	Источник	Цена, рублей	Срок действия лицензии, дней
Microsoft Windows 10 Pro	интернет-магазин ООО «Технолинк»	31000	бессрочная

Таблица 4.2 – Потребность в лицензионном программном обеспечении

Антивирус <i>Kaspersky 2023/2024</i>	интернет-магазин ООО «Технолинк»	2100	365
---	-------------------------------------	------	-----

Разработка мобильного программного обеспечения осуществлялась командой разработчиков, состав которой представлен в таблице 10.4. Состав коллектива разработчиков и информация по заработной плате разработчиков программного обеспечения и их количественному составу приведены в таблице 4.4.

Таблица 4.4 – Занятость и заработная плата разработчиков

Специалист	Количество, человек	Сумма заработной платы, рублей
Руководитель проекта	1	80000
Тимлидер	1	60000
Программист	1	60000
Тестировщик	1	50000

Себестоимость разработки программного продукта включает следующие статьи затрат:

- затраты на приобретение оборудования;
- затраты на приобретение лицензионного программного обеспечения;
- затраты на дополнительные материалы;
- затраты на заработную плату;
- затраты на обслуживание.

Затраты на оборудование в период разработки составят 454000 рублей на основании данных в таблице 4.1.

Затраты на программное обеспечение в период разработки составят 165500 рублей и представлены в таблице 4.2.

Затраты на дополнительные материалы в период разработки составят 0 рублей в соответствии с условием что потребности в закупке нет.

Затраты на заработную плату включают затраты на основную и дополнительную заработную плату разработчиков программного обеспечения и отчисления на страховые взносы.

Затраты на основную заработную плату разработчиков определяются исходя из установленной заработной платы, количества специалистов и их занятости в процессе разработки программного продукта (продолжительность работы в днях).

Дополнительная заработная плата составляет 15% от затрат на основную заработную плату разработчиков.

С учетом выходных дней при 5-дневной рабочей неделе количество рабочих дней в месяце в среднем составляет 21 день.

Затраты на основную и дополнительную заработную плату разработчиков программного обеспечения представлены в таблице 4.5.

Таблица 4.5 – Затраты на основную и дополнительную заработную плату разработчиков

Исполнитель	Количество, человек	Сумма заработной платы, рублей	Период занятости, дней	Основная заработная плата, рублей	Дополнительная заработная плата, рублей	Всего рублей
Руководитель проекта	1	80000	42	240000	36000	276000
Тимлидер	1	60000	42	180000	27000	207000

Таблица 4.5 – Затраты на основную и дополнительную заработную плату разработчиков

Программист	1	60000	42	180000	27000	207000
Тестировщик	1	50000	42	150000	22500	172500
Итого	5	250000	168	750000	112500	862500

Таким образом, затраты на заработную плату разработчиков составят 862500 рублей.

Отчисления на страховые взносы по нормам, установленным законодательством в ПРФ, ФОМС, ФСС РФ вычисляются исходя из данных представленных ниже.

Отчисления на страховые взносы производится согласно «Налоговому кодексу Российской Федерации (часть вторая)» от 05.08.2000 № 117-ФЗ (ред. От 26.03.2022).

Действующие тарифы на страховые взносы работодателей установлены статьей 34 НК РФ и составляют: ОПС (обязательное пенсионное страхование) – 22%, ВниМ (страхование временной нетрудоспособности и в связи с материнством) – 2,9%, ОМС (обязательное медицинское страхование) – 5,1%.

Общая сумма отчислений на страховые взносы составит 258750 рублей.

Общая сумма затрат является суммой: затрат на оплату труда включая налоги и сборы, затраты на закупку оборудования и лицензионного ПО и составляет 1740750 рублей.

В процессе эксплуатации пользователи получают свободно распространяемое ПО и техническую поддержку от команды разработки.

4. Безопасность жизнедеятельности

Современный человек привык к вычислительной технике и просто не может представить свою жизнь без персональных компьютеров и мобильных устройств. Эти устройства имеют множество преимуществ, однако, если не соблюдать ряда предостережений, можно заметить влияние на организм множества вредных факторов.

Вредные факторы бывают физическими, химическими и психофизиологическими.

К физическим вредным факторам чаще всего относят:

- повышенный уровень электромагнитного излучения;
- повышенный уровень рентгеновского излучения;
- повышенный уровень ультрафиолетового излучения;
- повышенный уровень инфракрасного излучения;
- повышенный уровень статического электричества и запыленности воздуха рабочей зоны;
- повышенный уровень светового загрязнения;
- неравномерность распределения яркости в поле зрения;
- повышенная яркость экрана монитора.

К химическим факторам относятся:

- повышенное содержание в воздухе рабочей зоны двуокиси углерода;
- повышенное содержание в воздухе рабочей зоны озона;
- повышенное содержание в воздухе рабочей зоны аммиака;
- повышенное содержание в воздухе рабочей зоны фенола;
- повышенное содержание в воздухе рабочей зоны формальдегида.

К психофизиологическим относятся:

- напряжение зрения и внимания;
- длительные эмоциональные и интеллектуальные нагрузки.
- монотонность труда;

Соблюдение определенных правил способно сберечь здоровье человека и предотвратить воздействие вредных факторов на него.

Помещения, предназначенные для размещения рабочих мест пользователей персональных компьютеров, должны иметь как естественное, так и искусственное освещение. Естественное освещение помещений должно осуществляться через световые проемы, ориентированные преимущественно на север и северо-восток.

В случае иной ориентации световых проемов, необходимо предусматривать эффективные средства регулирования интенсивности естественного освещения. Оконные проемы необходимо оборудовать жалюзи или занавесями. Искусственное освещение в помещениях должно осуществляться системой общего равномерного освещения.

Для использования в качестве источников местного освещения пригодны лампы, позволяющие избежать возникновения ослепления и бликов.

Местное освещение на рабочих местах пользователей персональных компьютеров не должно создавать бликов на поверхности экрана. Применять светильники без сертифицированных рассеивателей и экранирующих решеток не допускается.

Общее освещение следует выполнять в виде сплошных или прерывистых линий светильников, расположенных сбоку от рабочих мест, параллельно линии зрения пользователя при рядном расположении рабочих мест. При периметральном расположении рабочих мест линии светильников должны располагаться локализовано над рабочим столом ближе к его переднему краю, обращенному к оператору.

Для обеспечения нормируемых значений освещенности в помещениях следует проводить чистку стекол оконных рам и светильников не реже двух раз в год и проводить своевременную замену перегоревших ламп.

Так как при работе на компьютере основная нагрузка ложится на глаза, поэтому большие требования предъявляются к экранам устройств. Предпочтительным является плоский экран, позволяющий избежать наличие на нем ярких пятен за счет отражения световых потоков. Особенно важен цвет

экрана. Он должен быть нейтральным. Допустимы ненасыщенные светло-зеленые, желто-зеленые, желто-оранжевые, желто-коричневые тона.

Для устранения бликов и снижения влияния электромагнитного излучения экраны должны быть снабжены защитными фильтрами.

Режим труда и отдыха при работе с компьютером должен организовываться в зависимости от вида и категории деятельности. Виды деятельности подразделяются на следующие группы:

- группа А - работа по считыванию информации с монитора с предварительным запросом;
- группа Б - работа по вводу информации;
- группа В - творческая работа в режиме диалога.

Соблюдение вышеописанных правил сохранит здоровье оператора ЭВМ и увеличит производительность труда.

					ПГУ 1.09.05.01.02.001 ПЗ	Лист
						60
Изм.	Лист	№ докум.	Подпись	Дата		

Заключение

В ходе выполнения дипломного проекта была изучена предметная область, в результате чего было сформулировано техническое задание на разработку приложения для обмена сообщениями в пиринговых сетях.

Был произведен выбор инструментария для реализации мобильных приложений, изучен протокол *Wi-Fi Direct*, в результате было разработано программное приложение.

Приложение реализовано в соответствии с техническим заданием.

В разделе безопасности жизнедеятельности были описаны факторы воздействия вычислительной техники на человека.

В экономическом разделе был произведен расчет трудоемкости и затрат на разработку мобильного приложения.

Задание на дипломное проектирование выполнено в полном объеме.

					ПГУ 1.09.05.01.02.001 ПЗ	Лист
						61
Изм.	Лист	№ докум.	Подпись	Дата		

Список использованных источников

1. Белов В. А. Основы объектного программирования на Java / - М.: ИТУИТ, Бином., 2019. - 584 с.
2. Гарнаев А. М. Программирование на Java в среде Android / - М.: БХВ-Петербург, 2018. - 198 с.
3. Голощапов А. С. Программирование для мобильных устройств/ - М.: БХВ-Петербург, 2015. - 138 с.
4. Пертиков И. С. Создание приложения на базе беспроводных сетей- М.: Издательство «Эксмо», 2018. - 514 с.
5. Мартынов В.Ф. Основы беспроводных сетей - М.: "Издательство «Известия», 2015. - 327 с.

					ПГУ 1.09.05.01.02.001 ПЗ	Лист
						62
Изм.	Лист	№ докум.	Подпись	Дата		

Приложение А

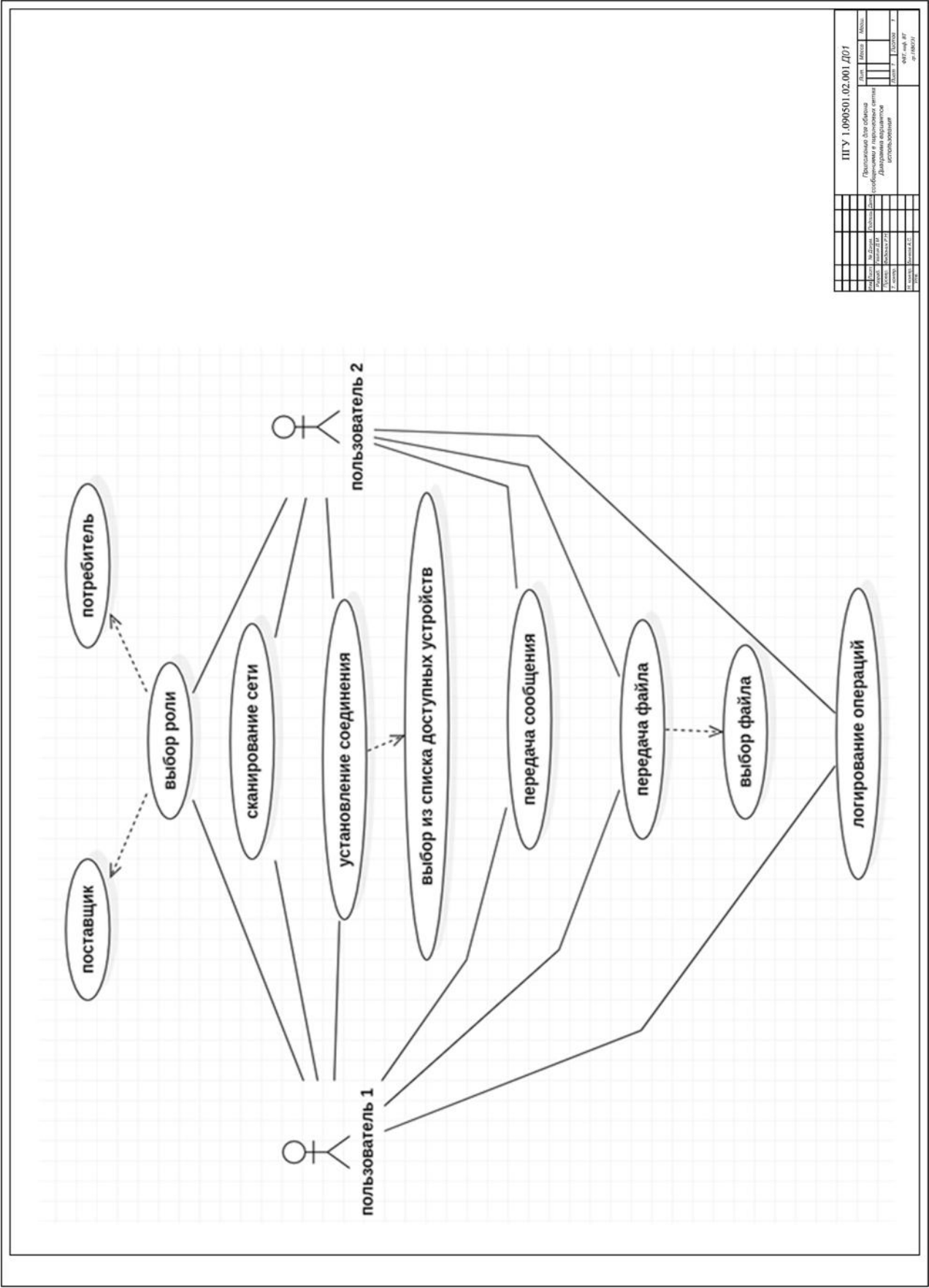
(Обязательное)

Приложение для обмена сообщениями в пиринговых сетях

UML-диаграммы

					ПГУ 1.09.05.01.02.001 ПЗ	Лист
						63
Изм.	Лист	№ докум.	Подпись	Дата		

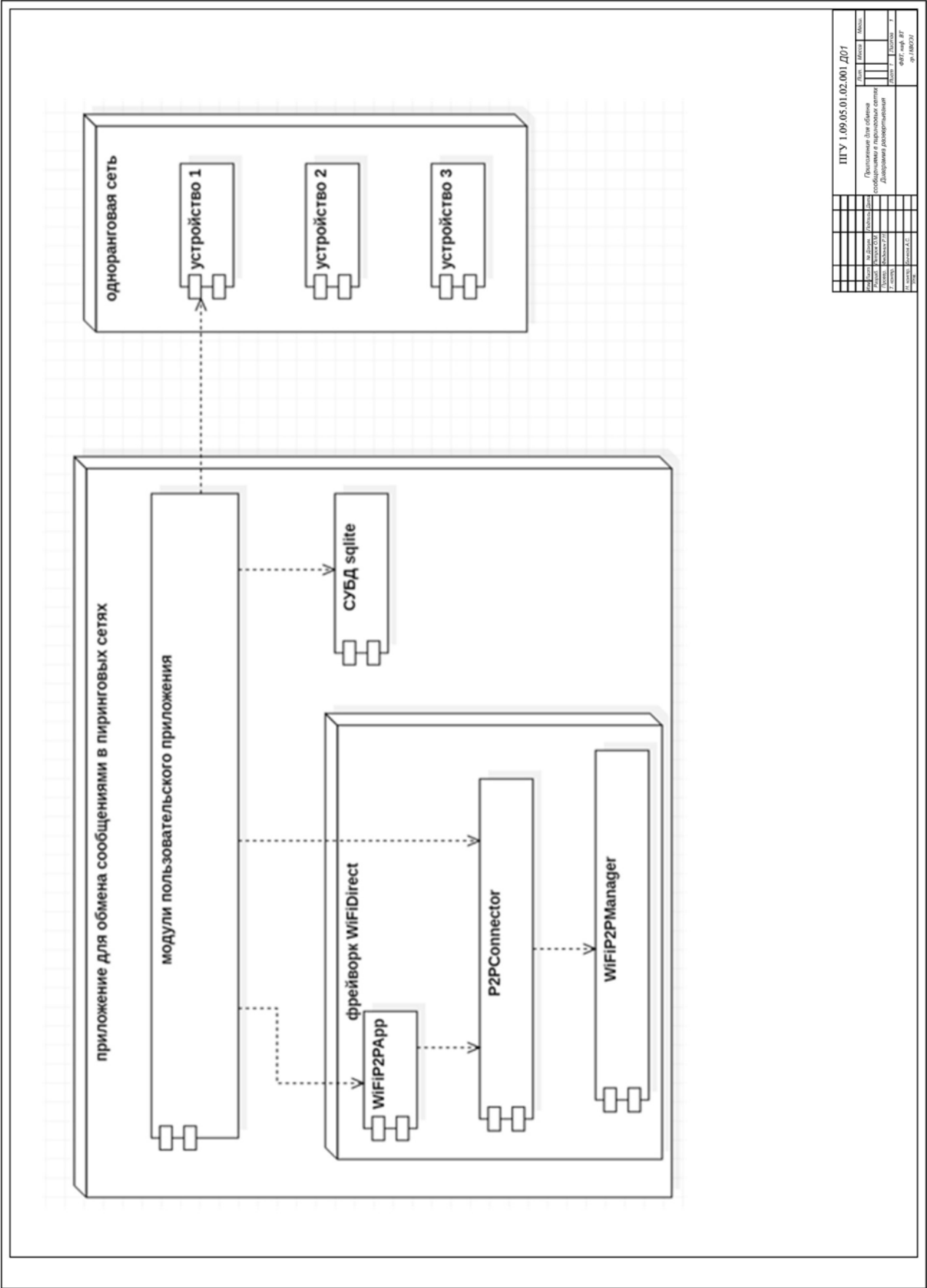
Приложение А.1 – Диаграмма вариантов использования



ПГУ 1.09.05.01.02.001 Д01									
Детализация для описания									
варианта использования									
Диаграмма вариантов									
использования									
Файл: 1.09.05.01.02.001 Д01									
Стр. 1 из 1									
Файл: 1.09.05.01.02.001 Д01									

Изм.	Лист	№ докум.	Подпись	Дата

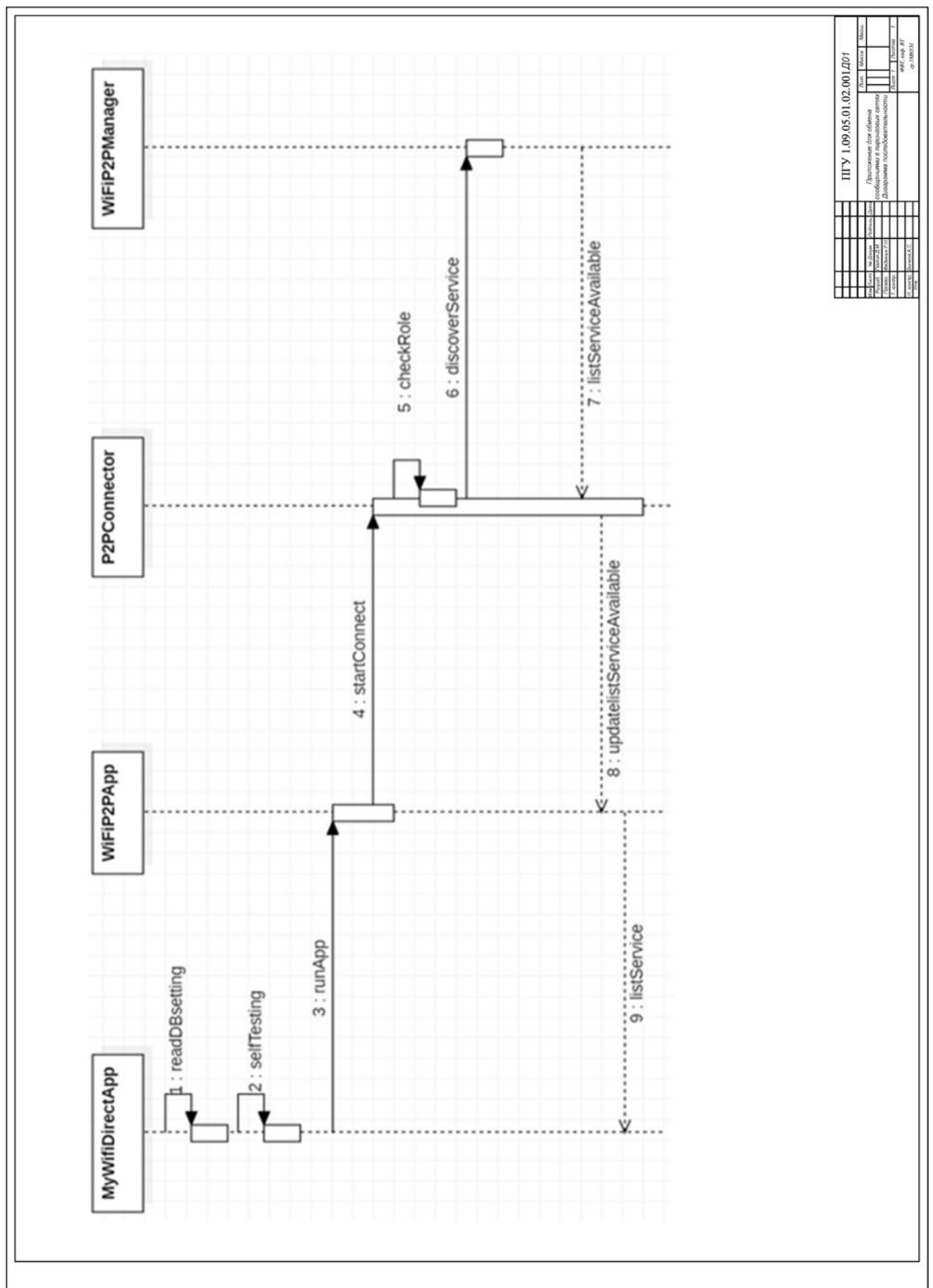
Приложение А.2 – Диаграмма развертывания



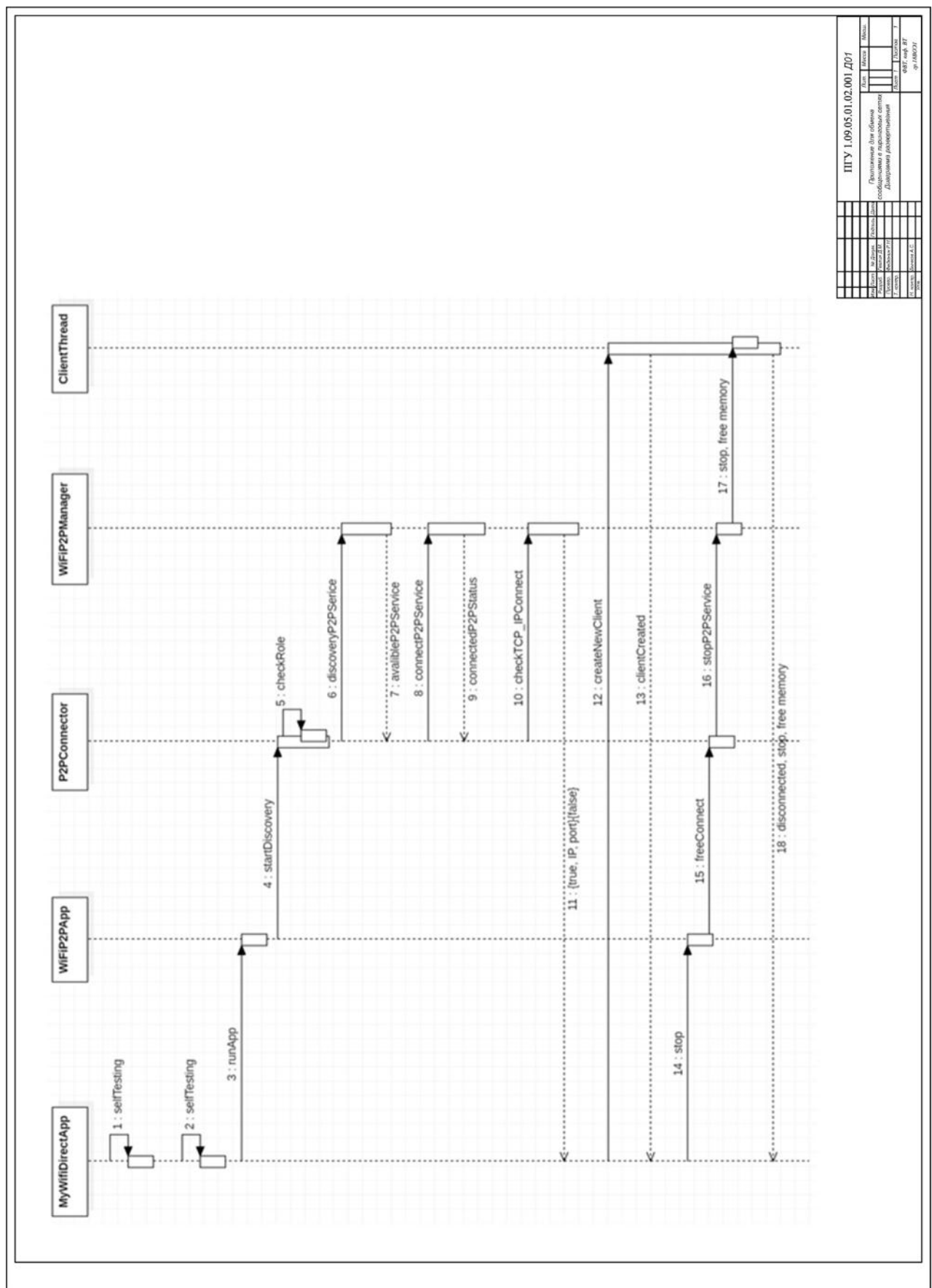
ПГУ 1.09.05.01.02.001 Д01									
Исполн.	И.И.Иванов	Исполн. зам.	И.И.Иванов	Исполн. зам.	И.И.Иванов	Исполн. зам.	И.И.Иванов	Исполн. зам.	И.И.Иванов
Исполн.	И.И.Иванов	Исполн. зам.	И.И.Иванов	Исполн. зам.	И.И.Иванов	Исполн. зам.	И.И.Иванов	Исполн. зам.	И.И.Иванов
Исполн.	И.И.Иванов	Исполн. зам.	И.И.Иванов	Исполн. зам.	И.И.Иванов	Исполн. зам.	И.И.Иванов	Исполн. зам.	И.И.Иванов
Исполн.	И.И.Иванов	Исполн. зам.	И.И.Иванов	Исполн. зам.	И.И.Иванов	Исполн. зам.	И.И.Иванов	Исполн. зам.	И.И.Иванов
Исполн.	И.И.Иванов	Исполн. зам.	И.И.Иванов	Исполн. зам.	И.И.Иванов	Исполн. зам.	И.И.Иванов	Исполн. зам.	И.И.Иванов
Исполн.	И.И.Иванов	Исполн. зам.	И.И.Иванов	Исполн. зам.	И.И.Иванов	Исполн. зам.	И.И.Иванов	Исполн. зам.	И.И.Иванов
Исполн.	И.И.Иванов	Исполн. зам.	И.И.Иванов	Исполн. зам.	И.И.Иванов	Исполн. зам.	И.И.Иванов	Исполн. зам.	И.И.Иванов
Исполн.	И.И.Иванов	Исполн. зам.	И.И.Иванов	Исполн. зам.	И.И.Иванов	Исполн. зам.	И.И.Иванов	Исполн. зам.	И.И.Иванов
Исполн.	И.И.Иванов	Исполн. зам.	И.И.Иванов	Исполн. зам.	И.И.Иванов	Исполн. зам.	И.И.Иванов	Исполн. зам.	И.И.Иванов

Изм.	Лист	№ докум.	Подпись	Дата

Приложение А.3 – Диаграмма деятельности

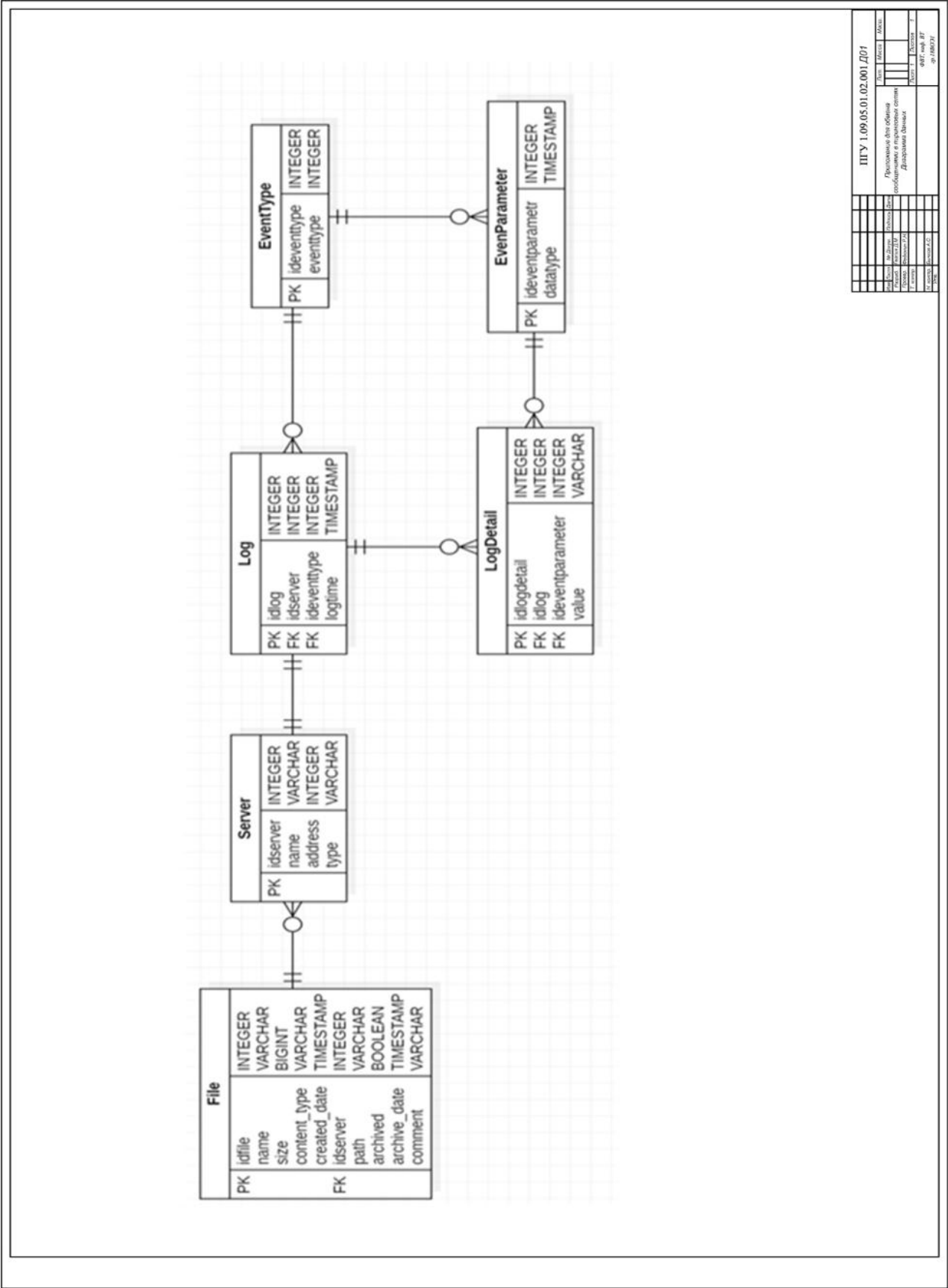


Приложение А.4 – Диаграмма деятельности

[illegible]

Изм.	Лист	№ докум.	Подпись	Дата

Приложение А.5 – Диаграмма данных



Приложение Б

(Обязательное)

Приложение для обмена сообщениями в пиринговых сетях

Листинг программы

Модуль MainActivity.java

```
import java.io.BufferedReader;
import java.io.BufferedWriter;
import java.io.File;
import java.io.FileReader;
import java.io.FileWriter;
import java.io.IOException;
import java.util.HashMap;
import java.util.Iterator;
import java.util.Set;

import android.app.Activity;
import android.app.AlertDialog;
import android.content.BroadcastReceiver;
import android.content.Context;
import android.content.DialogInterface;
import android.content.Intent;
import android.content.IntentFilter;
import android.net.wifi.p2p.WifiP2pConfig;
import android.net.wifi.p2p.WifiP2pDevice;
import android.net.wifi.p2p.WifiP2pManager;
import
android.net.wifi.p2p.WifiP2pManager.ActionListener;
import android.net.wifi.p2p.WifiP2pManager.Channel;
```

```

import
android.net.wifi.p2p.WifiP2pManager.ChannelListener;
import android.os.Bundle;
import android.provider.Settings;
import android.util.Log;
import android.view.Menu;
import android.view.MenuInflater;
import android.view.MenuItem;
import android.view.View;
import android.widget.TextView;
import android.widget.Toast;

    private WifiP2pManager manager;
    private Channel channel;
    private BroadcastReceiver receiver;
    private final IntentFilter intentFilter = new
IntentFilter();
    private boolean isWifiP2pEnabled = false;
    Context CONTEXT=this;
    private boolean retryChannel=false;
    public final static int BYTESPERCHUNK=100000;
    static HashMap<String, String> allFileList= new
HashMap<String, String>();
    static HashMap<String, Integer> numOfChunks= new
HashMap<String, Integer>();
    static String fileNeeded=new String();

    private WifiP2pDevice device;
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);

```

```

        File file= new File("/sdcard/ee579");
        if(!file.exists()) file.mkdirs();
        initialization();
        fileNeeded=getFileNeeded();
intentFilter.addAction(WifiP2pManager.WIFI_P2P_STATE_CHANGED_ACTION);

intentFilter.addAction(WifiP2pManager.WIFI_P2P_PEERS_CHANGED_ACTION);

intentFilter.addAction(WifiP2pManager.WIFI_P2P_CONNECTION_CHANGED_ACTION);

intentFilter.addAction(WifiP2pManager.WIFI_P2P_THIS_DEVICE_CHANGED_ACTION);

        manager = (WifiP2pManager)
getSystemService(Context.WIFI_P2P_SERVICE);
        channel = manager.initialize(this,
getMainLooper(), null);
        receiver = new
WifiDirectBroadcastReceiver(manager, channel, this);
        registerReceiver(receiver, intentFilter);
        //searchPeer();
    }
    public void onResume() {
        super.onResume();
        registerReceiver(receiver, intentFilter);
    }

    public void onPause() {

```

					ПГУ 1.09.05.01.02.001 ПЗ	Лист
						71
Изм.	Лист	№ докум.	Подпись	Дата		

```

        super.onPause();
        unregisterReceiver(receiver);
    }

    //дополнительные опции
    public boolean onCreateOptionsMenu(Menu menu) {
        MenuInflater inflater = getMenuInflater();
        inflater.inflate(R.menu.menuitem, menu);
        return true;
    }
    @Override
    public boolean onOptionsItemSelected(MenuItem item)
    {
        switch (item.getItemId()) {
            case R.id.exit:                //Кнопка закрыть
                disconnect();
                cancelDisconnect();
                unregisterReceiver(receiver);
                this.finish();
                System.exit(0);
                return true;
            case R.id.folder:              // файлы достать
для пересылки
                Intent browseFolder= new
Intent(this,BrowserFolder.class);
                startActivity(browseFolder); // открыть папку
                default:
                    return
super.onOptionsItemSelected(item);
        }
    }
}

```

```

        //вайфай директ доступен для работы
        public void setIsWifiP2pEnabled(boolean
isWifiP2pEnabled) {
            this.isWifiP2pEnabled = isWifiP2pEnabled;
            return;
        }

        public void showMessage(String str){
            Toast.makeText(CONTEXT,
str,Toast.LENGTH_SHORT).show();
        }

        //поиск устройств
        public void searchButton(View view){
            searchPeer();
            return;
        }

        public void searchPeer(){
            //check if wifi direct is enabled
            if(!isWifiP2pEnabled){
                new AlertDialog.Builder(this)
                    .setIcon(R.drawable.ic_launcher)
                    .setTitle("Wi-Fi Direct не доступен")
                    .setPositiveButton("Setting", new
DialogInterface.OnClickListener() {
                        public void onClick(DialogInterface
dialog, int whichButton) {
                            startActivity(new
Intent(Settings.ACTION_WIRELESS_SETTINGS));

```

					ПГУ 1.09.05.01.02.001 ПЗ	Лист
						73
Изм.	Лист	№ докум.	Подпись	Дата		

```

        }

        }).show();

        return;

    }

    if(fileNeeded==null){

        return;

    }

    if(devicefragment.device!=null&&devicefragment.isConnected){

        showMessage("Отбой соединения.");

        return;

    }

    final DeviceListFragment fragment =
(DeviceListFragment) getFragmentManager()

        .findFragmentById(R.id.devicelist);

    fragment.onInitiateDiscovery();

    fragment.getView().setVisibility(View.VISIBLE);

    manager.discoverPeers(channel, new
WifiP2pManager.ActionListener() {

        @Override

        public void onSuccess() {

            //Toast.makeText(CONTEXT.show());

            return;

        }

        public void onFailure(int reasonCode) {

            Toast.makeText(CONTEXT, "Поиск не
удался).show();

            return;

        }

    });

}

```

```

    public static String getDeviceStatus(int
deviceStatus) {
        switch (deviceStatus) {
            case WifiP2pDevice.AVAILABLE:
            case WifiP2pDevice.CONNECTED:
                return "Соединение есть";
            case WifiP2pDevice.FAILED:
                return "Ошибка";
            case WifiP2pDevice.UNAVAILABLE:
                return "Нет доступа";
            default:
                return "Unknown";

        }
    }

    public static int divRoundUp(int n,int s){
        return (((n) / (s)) + (((n) % (s)) > 0) ? 1 : 0));
    }

    public void updateThisDevice(WifiP2pDevice device) {
        TextView view =
        (TextView)findViewById(R.id.mystatus);
        view.setText("My Name: "+device.deviceName+"\nMy
Address: "+device.deviceAddress+"\nMy Status:
"+getDeviceStatus(device.status));
        return;
    }

    public void connect(WifiP2pConfig config){

```

					ПГУ 1.09.05.01.02.001 ПЗ	Лист
						75
Изм.	Лист	№ докум.	Подпись	Дата		

```

        manager.connect(channel, config, new
ActionListener() {
            @Override
            public void onSuccess() {
                }

            @Override
            public void onFailure(int reason) {
                showMessage("Доступ отменен: "+reason);
            }
        });
        return;
    }

//wifi direct disconnect function
public void disconnect(){
    final DeviceDetailFragment fragment =
(DeviceDetailFragment) getFragmentManager()
        .findFragmentById(R.id.devicedetail);
    fragment.blockDetail();
    updateRecord();
    manager.removeGroup(channel, new
ActionListener() {
        @Override
        public void onFailure(int reasonCode) {
            }

        public void onSuccess() {

//fragment.getView().setVisibility(View.GONE);
            showMessage("Дисконект.");

```

```

        }

    });

    return;
}

public void onChannelDisconnected() {
    if (manager != null && !retryChannel) {
        Toast.makeText(this, "Канала нет",
Toast.LENGTH_LONG).show();
        retryChannel = true;
        manager.initialize(this, getMainLooper(),
this);
    } else {
        Toast.makeText(this,
            Toast.LENGTH_LONG).show();
    }
    return;
}

public void cancelDisconnect() {
    if (manager != null) {
        final DeviceDetailFragment fragment =
(DeviceDetailFragment) getFragmentManager()

.findFragmentById(R.id.devicedetail);
        if (fragment.device == null
            || fragment.device.status ==
WifiP2pDevice.CONNECTED) {
            disconnect();
        } else if (fragment.device.status ==
WifiP2pDevice.AVAILABLE

```

```

        || fragment.device.status ==
WifiP2pDevice.INVITED) {

        manager.cancelConnect(channel, new
ActionListener() {

            public void onSuccess() {
                showMessage("обрыв передачи");
                return;
            }

            public void onFailure(int
reasonCode) {

            }

        });

    }

    return;
}

```

```

    static HashMap<String, HashMap<String, Boolean>>
availableFileChunks= new HashMap<String, HashMap<String,
Boolean>>();

    static HashMap<String, HashMap<String, Boolean>>
neededFileChunks= new HashMap<String, HashMap<String,
Boolean>>();

    void initialization(){
        File listFile=new File("/sdcard/ee579filelist.txt");
        File recordFile=new File("/sdcard/ee579record.txt");
        try {

```

```

        BufferedReader inputReader = new
BufferedReader(new FileReader(listFile));
        String buffer = new String();
        while((buffer=inputReader.readLine())!=null){
            String [] fileInfo= buffer.split(",");
            allFileList.put(fileInfo[0], fileInfo[1]);
            int
num=divRoundUp(Integer.parseInt(fileInfo[2]),BYTESPERCHU
NK);

            numOfChunks.put(fileInfo[0],num);
        }
        inputReader.close();
        inputReader=new BufferedReader(new
FileReader(recordFile));
        while((buffer=inputReader.readLine())!=null){
            String [] fileInfo= buffer.split(",");
            if((buffer=inputReader.readLine())!=null){
                HashMap<String, Boolean> chunkMap= new
HashMap<String, Boolean>();
                String [] chunkNum= buffer.split(",");
                for(int i=0;i<chunkNum.length;i++){
                    chunkMap.put(chunkNum[i], true);
                }
                availableFileChunks.put(fileInfo[0],
chunkMap);
            }
        }
        inputReader.close();
        Set<String>files= allFileList.keySet();
        Iterator<String> it=files.iterator();
        while(it.hasNext()){

```

```

        buffer=it.next();
        if(availableFileChunks.get(buffer)==null){
            HashMap<String, Boolean> chunkMap= new
HashMap<String, Boolean>();
            for(int i=0;i<numOfChunks.get(buffer);i++){
                chunkMap.put(new Integer(i).toString(),
true);
            }
            neededFileChunks.put(buffer, chunkMap);
        }else{
            HashMap<String, Boolean>
chunkMap=availableFileChunks.get(buffer);
            HashMap<String, Boolean> neededChunkMap= new
HashMap<String, Boolean>();
            for(int i=0;i<numOfChunks.get(buffer);i++){
                if(chunkMap.get(new
Integer(i).toString())==null){
                    neededChunkMap.put(new
Integer(i).toString(), true);
                }
            }
            neededFileChunks.put(buffer,
neededChunkMap);
        }
    }
} catch (IOException e) {
    showMessage("IO Error.");
}

}

public static String getFileNeeded(){

```

```

String result=new String();
Set<String> files=neededFileChunks.keySet();
if(files.isEmpty()) return null;
Iterator<String> it=files.iterator();
int i=0;
while(it.hasNext()){
    if(i++!=0) result+=",";
    String buffer=it.next();
    result+=buffer+", ";
    HashMap<String, Boolean>
neededChunkMap=neededFileChunks.get(buffer);
    Set<String> chunks=neededChunkMap.keySet();
    Iterator<String> chunkit=chunks.iterator();
    int j=0;
    while(chunkit.hasNext()){
        if(j++!=0) result+=" ";
        result+=chunkit.next();
    }
}
return result;
}

public void updateRecord(){
    File recordFile=new File("/sdcard/ee579record.txt");
    try {
        BufferedWriter outputWriter = new
BufferedWriter(new FileWriter(recordFile,false));
        Set<String>files= availableFileChunks.keySet();
        Iterator<String> it=files.iterator();
        while(it.hasNext()){
            outputWriter.write(it.next()+"\n");

```

```

        HashMap<String, Boolean>
chunkMap=availableFileChunks.get(it.next());
        Set<String> chunks= chunkMap.keySet();
        Iterator<String> chunkit=chunks.iterator();
        String chunkList=new String();
        int i=0;
        while(chunkit.hasNext()){
            if(i++!=0) chunkList+=",";
            chunkList+=chunkit.next();
        }
        outputWriter.write(chunkList+"\n");
    }
    outputWriter.flush();
    outputWriter.close();
} catch (IOException e) {
    showMessage("IO Error.");
}

public static HashMap<String, BitMap>
availableChunkMap= new HashMap<String, BitMap>();
    public static HashMap<String, BitMap>
neededChunkMap= new HashMap<String, BitMap>();
    void initialization(){
        File listFile=new File("/sdcard/ee579filelist.txt");
        File recordFile=new
File("/sdcard/ee579bitmaprecord.txt");
        try {
            if(!listFile.exists()){ // need to have a
config file list all files available
                showMessage("Fatal Error: Config file not
found.");
            }
            return;
        }
    }

```

```

    }

    BufferedReader inputReader = new
BufferedReader(new FileReader(listFile));

    String buffer = new String();
    while((buffer=inputReader.readLine())!=null){
        String [] fileInfo= buffer.split(",");
        allFileList.put(fileInfo[0], fileInfo[1]);
        int
num=divRoundUp(Integer.parseInt(fileInfo[2]),BYTESPERCHU
NK);

        numOfChunks.put(fileInfo[0],num);
        // if the files on the list exist on the phone.
        record them on a hashtable-bitmap.
        // can check this to see if it has the file or
        not when there are requests
        File oneFile= new
File("/sdcard/ee579/"+fileInfo[1]);
        if(oneFile.exists()){
            BitMap chunkMap= new BitMap(num);
            for(int i=0;i<num;i++) chunkMap.Mark(i);
            availableChunkMap.put(fileInfo[0],
chunkMap);
        }
    }

    inputReader=new BufferedReader(new
FileReader(recordFile));

    while((buffer=inputReader.readLine())!=null){
        String [] fileInfo= buffer.split(",");
        if(availableChunkMap.get(fileInfo[0])!=null){
            // if we have already had the record about
            this file skip it.

```

```

        buffer=inputReader.readLine();
        continue;
    }
    if((buffer=inputReader.readLine())!=null){
        BitMap chunkMap= new BitMap(buffer);

        if(chunkMap.length()!=numOfChunks.get(fileInfo[0])){
            showMessage("Error: BitMap length not
correct");
            return;
        }
        availableChunkMap.put(fileInfo[0],
chunkMap);
    }
}

inputReader.close();
Set<String>files= allFileList.keySet();
Iterator<String> it=files.iterator();
while(it.hasNext()){
    buffer=it.next();
    if(availableChunkMap.get(buffer)==null){
        BitMap chunkMap=new
BitMap(numOfChunks.get(buffer));
        for(int i=0;i<numOfChunks.get(buffer);i++){
            chunkMap.Mark(i);
        }
        neededChunkMap.put(buffer, chunkMap);
    }else{

        BitMap
chunkMap=availableChunkMap.get(buffer);

```

```

if(chunkMap.numMarked()==numOfChunks.get(buffer))
continue;

        BitMap neededChunk= new
BitMap(numOfChunks.get(buffer));

        for(int i=0;i<numOfChunks.get(buffer);i++){
            if(!chunkMap.Test(i)){
                neededChunk.Mark(i);
            }
        }
        neededChunkMap.put(buffer, neededChunk);
    }
}

} catch (IOException e) {
    showMessage("IO Error: "+e.toString());
}
}

public static String getFileNeeded(){
    String result=new String();
    Set<String> files=neededChunkMap.keySet();
    if(files.isEmpty()) return null;
    Iterator<String> it=files.iterator();
    int i=0;
    while(it.hasNext()){
        if(i++!=0) result+=",";
        String buffer=it.next();
        result+=buffer+",";
        result+=neededChunkMap.get(buffer).toString();
    }
    return result;
}

```

```

    }

    public static void updateRecord() {
        File recordFile=new
File("/sdcard/ee579bitmaprecord.txt");
        try {
            BufferedWriter outputWriter = new
BufferedWriter(new FileWriter(recordFile,false));
            Set<String>files= availableChunkMap.keySet();
            Iterator<String> it=files.iterator();
            while(it.hasNext()){
                String buffer=it.next();
                // write out the file number+ file name, and the
bitmap of that file as string.

                outputWriter.write(buffer+", "+allFileList.get(buffer
)+"\n");

                BitMap chunkMap=availableChunkMap.get(buffer);
                outputWriter.write(chunkMap.toString()+"\n");
            }
            outputWriter.flush();
            outputWriter.close();
        } catch (IOException e) {
            Log.d("EE579", "IO Error.");
        }
    }

    public static BloomFilter availableChunks=new
BloomFilter();

    void initialization(){
        File listFile=new File("/sdcard/ee579filelist.txt");
        try {

```

```

        if(!listFile.exists()){
            showMessage("Fatal Error: Config file not
found.");
            return;
        }

        BufferedReader inputReader = new
BufferedReader(new FileReader(listFile));
        String buffer = new String();
        while((buffer=inputReader.readLine())!=null){
            String [] fileInfo= buffer.split(",");
            allFileList.put(fileInfo[0], fileInfo[1]);

num=divRoundUp(Integer.parseInt(fileInfo[2]),BYTESPERCHU
NK);

            numOfChunks.put(fileInfo[0],num);
            File oneFile= new
File("/sdcard/ee579/"+fileInfo[1]);
            if(oneFile.exists()){
                for(int i=0;i<num;i++){
                    availableChunks.mark(fileInfo[0]+"-"+i);
                }
            }
            inputReader.close();
            File oneFile= new File("/sdcard/ee579/tmp");
            if(oneFile.exists()){
                File[] files=oneFile.listFiles();
                for(int i=0;i<files.length;i++){
                    String fileName=
files[i].getName().split("\\.")[0];
                    availableChunks.mark(fileName);

```

```

    }
    }
} catch (IOException e) {
    showMessage("IO Error: "+e.toString());
}
}

public static String getFileNeeded() {
    String result=new String();
    Set<String>files= allFileList.keySet();
    Iterator<String> it=files.iterator();
    while(it.hasNext()){
        String buffer=it.next();
        for(int i=0;i<numOfChunks.get(buffer);i++){
            if(!availableChunks.test(buffer+"-"+i)){
                result+=buffer+"-"+i+", ";
            }
        }
    }
    return result; }

```

Приложение В

(Обязательное)

Приложение для обмена сообщениями в пиринговых сетях

Презентация

					ПГУ 1.09.05.01.02.001 ПЗ	Лист
						89
Изм.	Лист	№ докум.	Подпись	Дата		

Приложение для обмена сообщениями в пиринговых сетях

Автор работы:

студент гр. 19ВОЭ1 Гвалия Давид Мамукович

Научный руководитель:

к.т.н., доцент Федюнин Роман Николаевич

Пенза, 2023

					ПГУ 1.09.05.01.02.001 ПЗ	Лист
						90
Изм.	Лист	№ докум.	Подпись	Дата		

Цель работы и объекты

Цель работы – Разработать мобильное приложение позволяющее передавать файлы и короткие сообщения по протоколу Wi-Fi Direct.

Объектом проекта является клиент-серверное приложение, выполняющие следующие функции:

- поиск потенциального клиента;
- Соединение с клиентом по протоколу Wi-Fi Direct;
- Установление одноранговой сети с IP адресами 192.168.XX.XX;
- Передача данных по протоколу TCP/IP;
- Конец сеанса связи.

Задачи приложения

Приложение реализовано в виде клиент серверной архитектуры.

При которой один экземпляр приложения выступает в роли ведущего, а второй экземпляр приложения в роли ведомого.

Передача сообщений осуществляется на короткие расстояния до 100 метров без участия устройств посредников.

Сравнение аналогов

Приложение	адрес приложения в сети Интернет	количество инсталляций, шт.
SHAREit	com.lenovo.anyshare	>10 миллионов
Xender	cn.xender	>10 миллионов
Xiaomi Mi Drop	com.xiaomi.midrop	>10 миллионов
Files by Google	com.google.android.apps.nbu.files	>10 миллионов

Аналоги обладают недостатками:

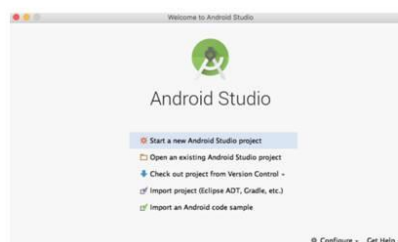
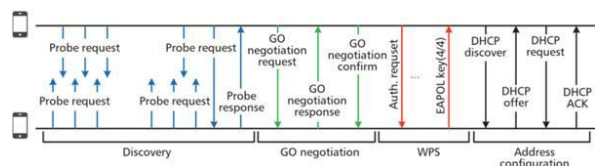
являются системными приложениями;

без возможно добавления функциональных возможностей и в основном идут предустановленными с новыми мобильными устройствами ;

являются импортными и реализованными производителями устройств.

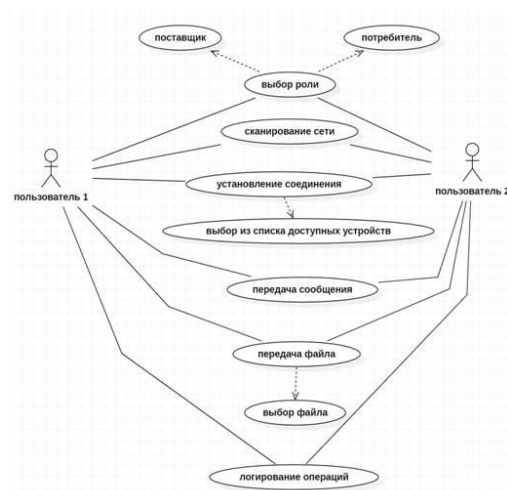
Стек технологий

Gradle



Android Monitor

Основные функции приложения



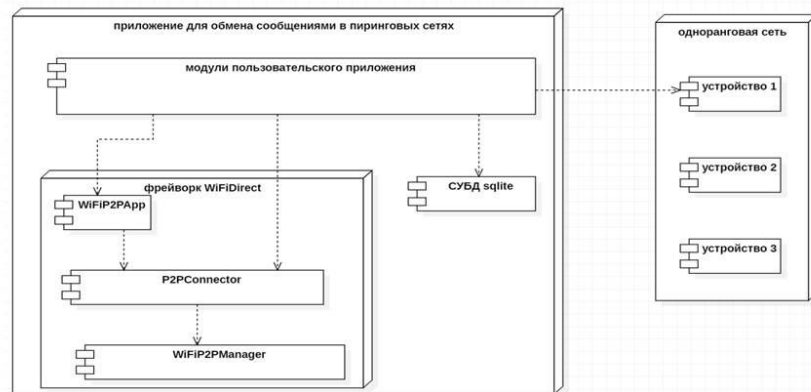
Изм.	Лист	№ докум.	Подпись	Дата

ПГУ 1.09.05.01.02.001 ПЗ

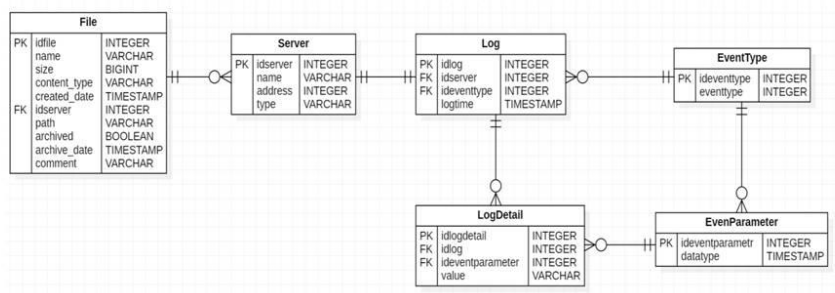
Лист

92

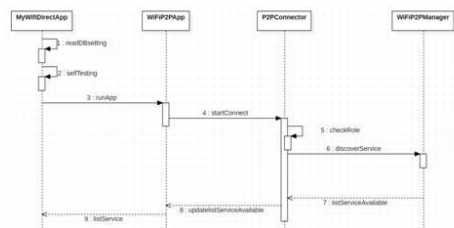
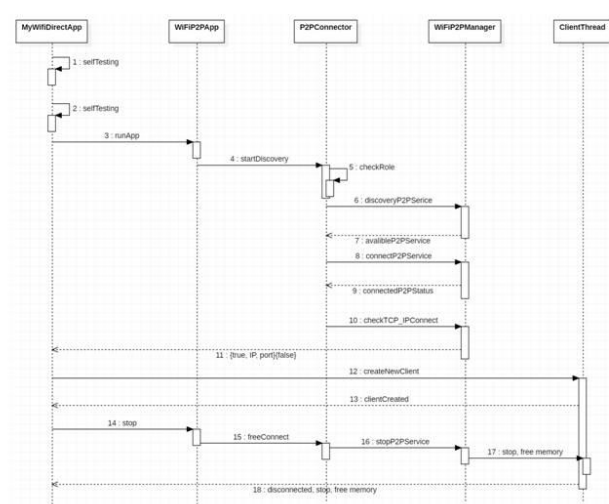
Архитектура приложения



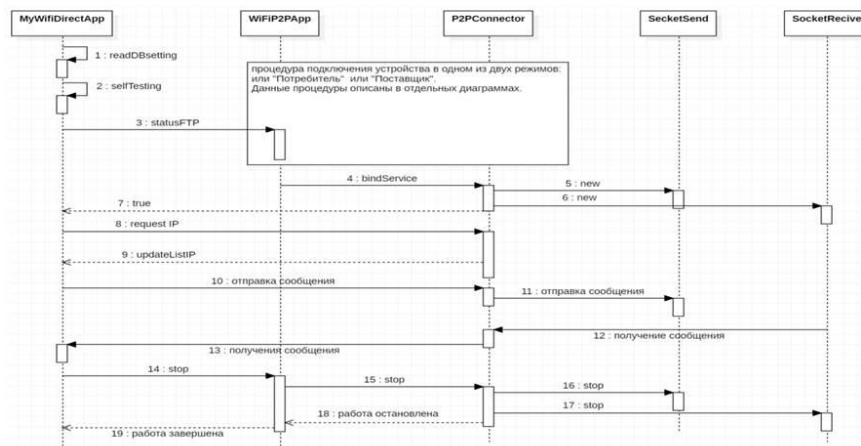
База данных приложения



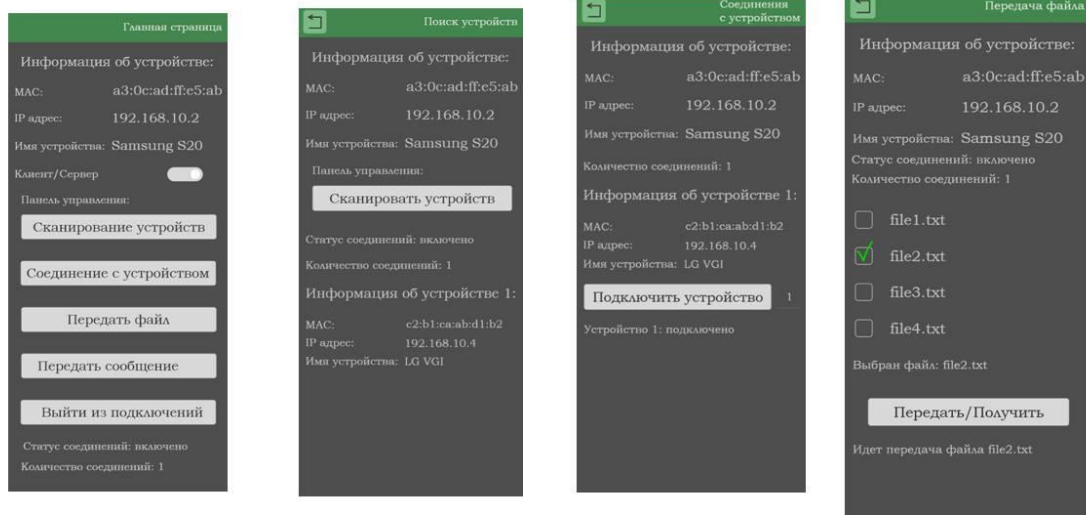
Работа приложения в различных режимах



Режим передачи сообщений



Скриншоты запущенного приложения



Результаты работы

- Разработано мобильное приложение полностью соответствующее техническому заданию;
- Проведено функциональное тестирование (успешно);
- Намечены задачи по развитию функциональных возможностей программы.

Спасибо за внимание!

					ПГУ 1.09.05.01.02.001 ПЗ	Лист
						96
Изм.	Лист	№ докум.	Подпись	Дата		