Министерство образования и науки Российской федерации
Федеральное государственное автономное образовательное учреждение
высшего образования
«ЮЖНО-УРАЛЬСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»
(национальный исследовательский университет)
Высшая школа электроники и компьютерных наук
Кафедра «Электронные вычислительные машины»

ДОПУСТИТЬ К ЗАЩИТЕ
Заведующий кафедрой,
доцент
К.А. Домбровский
_____2017 г.

Кроссплатформенный сетевой файловый менеджер, позволяющий отправлять, получать и делиться файлами и каталогами через интернет или локальную сеть

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА
К ВЫПУСКНОЙ КВАЛИФИКАЦИОННОЙ РАБОТЕ
ЮУрГУ-09.03.01.2017.382 ПЗ ВКР

Руководитель работы,
доцент
И.Л. Надточий
_____2017 г.

Авторы работы
студенты группы КЭ-445
К.Д. Суханов
В.Ж. Алексеев
_____2017 г.

Нормоконтроллер,
старший преподаватель
В.В. Лурье
_____2017 г.

Челябинск 2017

Аннотация

Алексеев В.Ж., Суханов К.Д. Разработка кроссплатформенного сетевого файлового менеджера. – Челябинск: ФГБОУ ВПО «ЮУрГУ» (НИУ) ВШЭКН; 2017, 54 с., 28 ил. Библиографический список – 24 наименований.

Работа посвящена разработке кроссплатформенного сетевого файлового менеджера, позволяющего осуществить удаленную передачу файлов и папок, настроить общий доступ, а также получить полный доступ к файловой системе на своих устройствах.

Данная работа состоит из введения, шести глав, заключения, библиографического списка.

В первой главе представлен обзор аналогов, общая информация о разрабатываемом программном продукте и описание требований к функциональности. Во второй главе – планирование, описаны сценарии использования, приведены обоснования выбора платформ и сред разработок. В третьей главе описана архитектура сервиса и взаимодействие ее основных компонентов. В четвертой главе рассмотрена реализация программного продукта, инструментарий, использованный при разработке. В пятой главе – функциональное тестирование и результаты тестов. В шестой главе представлено руководство пользователя.

В заключении приведено описание основных результатов работы.

Оглавление

Введение

Стремительное развитие вычислительной и телекоммуникационной техники привело к тому, что большинство людей обладает двумя и более устройствами. По данным исследования Яндекс[1] 84% пользователей интернета в течение месяца используют для выхода в сеть больше одного устройства – например, рабочий и домашний компьютеры или компьютер и мобильное устройство. Операционные системы у этих устройств, как правило, разные. Поэтому возникает проблема совместимости программного обеспечения. Кроссплатформенность решает эту проблему. Ведь в рамках одного программного продукта пользователю не придется устанавливать дополнительное программное обеспечение, чтобы передать данные на другие операционные системы. По данным аналитического сервиса StatCounter за 2017 год [2] самой популярной системой стал Android, с небольшим отрывом второе место занимает Windows, а последние места делят ОС семейства Linux и OS X. И именно на эти платформы делается упор в разработке кроссплатформенного приложения.

В настоящее время существует множество различных программ для передачи файлов. Однако ни одна из них не предоставляет того набора функций, которые бы заметно увеличили возможности пользователя при работе с файлами, а также избавили от установки дополнительных программных продуктов. Поэтому возникает потребность в написании программного обеспечения, которое бы решило существующие проблемы и повысило удобство и эффективность в области удаленной передачи файлов.

Актуальность и необходимость создания такой программы, которая бы значительно облегчила дистанционную работу с файлами и их отправку, как при домашнем использовании, так и на предприятии, повысив качество работы, а также снизив труд человека, обусловлена следующими аспектами:

1. возможностью передачи файлов, как через интернет, так и через локальную сеть для снижения расходов, например, в мобильной сети интернет.
2. наличием двухпанельного файлового менеджера для работы с файлами на удаленном устройстве
3. полным доступом к файловой системе ваших устройств, а также возможностью настроить и предоставить общий доступ к файлам и папкам для устройств друзей.
4. кроссплатформенность – приложение может быть запущено на всех популярных ОС: Windows, Linux, macOS, Android.

Цели и задачи

Цель работы – разработка кроссплатформенного сетевого файлового менеджера, позволяющего осуществить удаленную передачу файлов и папок, настроить общий доступ, а также получить полный доступ к файловой системе на своих устройствах.

Для осуществления поставленной цели необходимо реализовать следующие задачи:

а) исследовать существующие программные продукты для передачи файлов;

б) изучить проблемы удаленной передачи файлов и выделить функциональные возможности приложения;

в) исследовать область разработки десктопных и мобильных приложений;

г) выделить ядро программы и реализовать его в виде кроссплатформенной библиотеки;

д) спроектировать серверную часть: сервер, базу данных и настроить шифрование.

е) написать front-end для персональных компьютеров и мобильных устройств

ж) создать простой и удобный инсталлятор, который включает в себя все необходимые файлы (библиотеки, расширения)

з) протестировать разработанный программный продукт

Основные понятия предметной области

API – Application Programming Interface – интерфейс программирования приложений;

SSL – Secure Sockets Layer — уровень защищённых сокетов — криптографический протокол, который подразумевает более безопасную связь;

JSON – JavaScript Object Notation – простой формат обмена данными, удобный для чтения и написания как человеком, так и компьютером;

БД – база данных;

ООП – объектно-ориентированное программирование;

ОС – операционная система;

ПО – программное обеспечение;

ЛВС – локальная вычислительная сеть, локальная сеть — компьютерная сеть, покрывающая обычно относительно небольшую территорию или небольшую группу зданий [3];

Android NDK – native development kit – это набор инструментов, которые позволяют реализовать часть вашего приложения используя такие языки как C/C++[4];

JNI – Java Native Interface – стандартный механизм для запуска кода, под управлением виртуальной машины Java, который написан на языках C/C++ или Assembler, и скомпонован в виде динамических библиотек, позволяет не использовать статическое связывание.

TCP – Transmission Control Protocol, протокол управления передачей – один из основных протоколов передачи данных интернета, предназначенный для управления передачей данных[5].

UDP – User Datagram Protocol – один из ключевых элементов TCP/IP, набора сетевых протоколов для Интернета. С UDP компьютерные приложения могут посылать сообщения (в данном случае называемые датаграммами) другим хостам по IP-сети без необходимости предварительного сообщения для установки специальных каналов передачи или путей данных [6].

1 Анализ существующих решений

1.1 Обзор приложений для удаленной передачи файлов

На сегодняшний день существует множество программ для передачи файлов. Для анализа мы выбрали самые популярные кроссплатформенные приложения для операционных систем Android, Windows, Linux, OS X (без iOS)

1.1.1 PushBullet

«PushBullet» [7] - программное обеспечение для Android, iOS, Windows, macOS, позволяющее отправлять файлы, обмениваться ссылками, а также отправлять сообщения. Сервис перенаправляет уведомления, которые приходят на смартфон, и отображает их на персональном компьютере. Приложение распространяется бесплатно с возможностью покупки Pro-версии. Загрузить можно как с официального сайта, так и с Play Market и AppStore. Скриншот приложения представлен на рисунке 1.

Достоинства:

— Удобный интерфейс;

— Возможность обмена ссылками;

— Отправка сообщений.

Недостатки:

— Ограничение на размер файла;

— Нет возможности передачи через локальную сеть;

— Отсутствие поддержи ОС Linux.

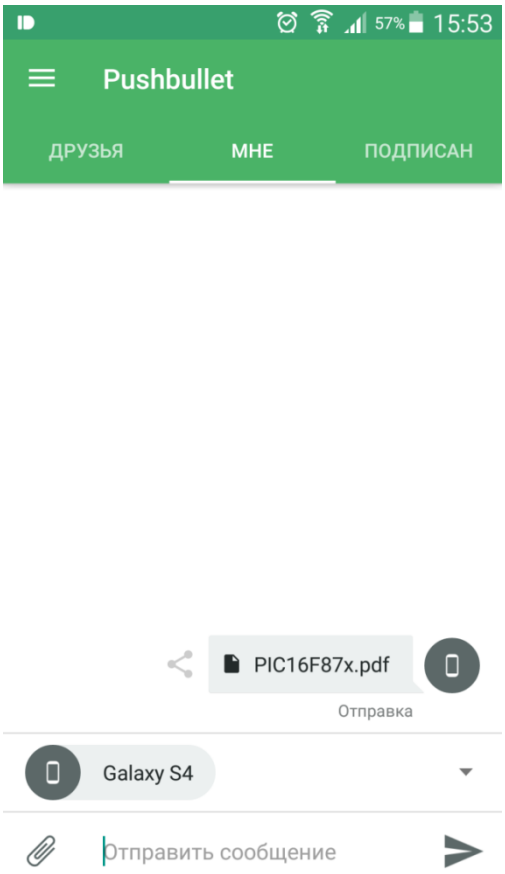

Рисунок 1 – Скриншот приложения «PushBullet», установленного на ОС Android.

1.1.2 Airdroid

«Airdroid» [8] - программное обеспечение позволяющее управлять своим Android устройством: передавать файлы между устройством и компьютером, отправлять сообщения, устанавливать программы, прослушивать музыку. Присутствует возможность просмотра состояния устройства: загруженности процессора и оперативной памяти, заряде батареи, свободной памяти. Приложение доступно для загрузки в магазине приложений Play Market. Скриншот приложения представлен на рисунке 2.

Достоинства:

— Наличие веб-версии, которая позволяет работать на компьютерах с такими операционными системами, как Windows, Linux, Mac;

— Полное управление Android-устройством.

Недостатки:

— Отсутствует возможность передачи файлов между десктопными устройствами, так как продукт заточен только под устройства с ОС Android;

— Ограничение в 100 МБ при передаче через веб.

— Передача папок только через локальное подключение.

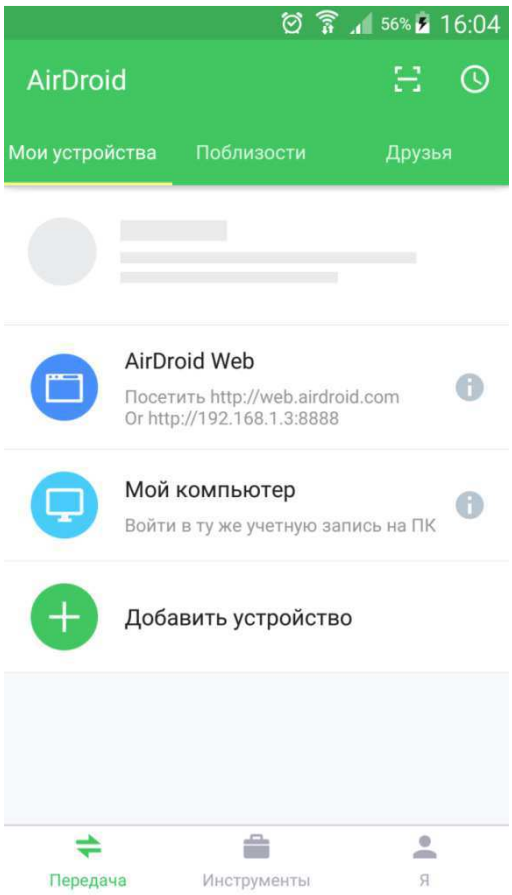

Рисунок 2 – Скриншот приложения «Airdroid», установленного на ОС Android.

1.1.3 SHAREit

«SHAREit» [9] - программный продукт, с помощью которого можно отправлять файлы в локальной сети. Передача осуществляется через каналы Wi-Fi или Bluetooth. Загрузить можно с официального сайта для Windows, с Play Market для Android и с AppStore для OS X. Скриншот приложения представлен на рисунке 3.

Достоинства:

— Нет ограничения на тип и размер передаваемых файлов

— Простой интерфейс

Недостатки:

— Передача осуществляется только в локальной сети

— Для каждой новой передачи перезапускается соединение

— Нет возможности самому выбрать и скачать нужный файл с другого устройства.

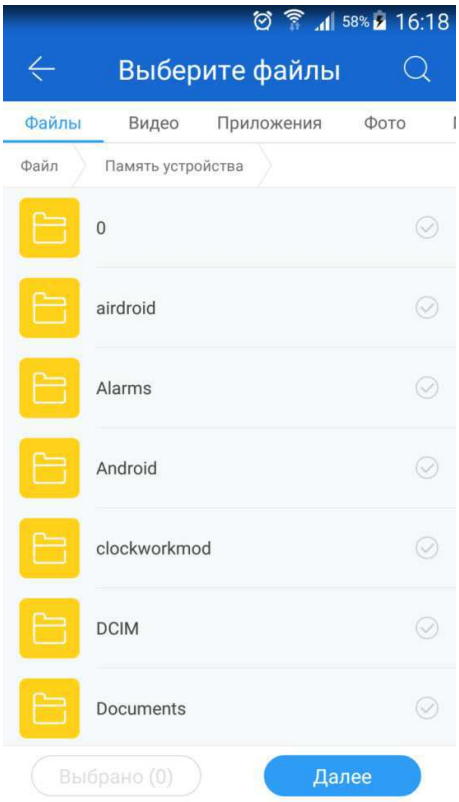

Рисунок 3 – скриншот приложения «SHAREit», установленного на ОС Android

1.1.4 Resilio Sync

«Resilio Sync» [10] - программное обеспечение для синхронизации файлов и резервного копирования по протоколу P2P. Приложение для Windows и OS X можно загрузить с официального сайта, для Linux с репозитория, а для мобильных устройств с Play Market и AppStore. Скриншот приложения представлен на рисунке 4.

Достоинства:

— Peer-to-peer-подход позволяет использовать трафик максимально эффективно

— Кроссплатформенность

Недостатки:

— Отсутствие файлового менеджера, позволяющего выбрать тот файл, который не был изначально указан для синхронизации.

Подходит для синхронизации, но не для передачи отдельных файлов.

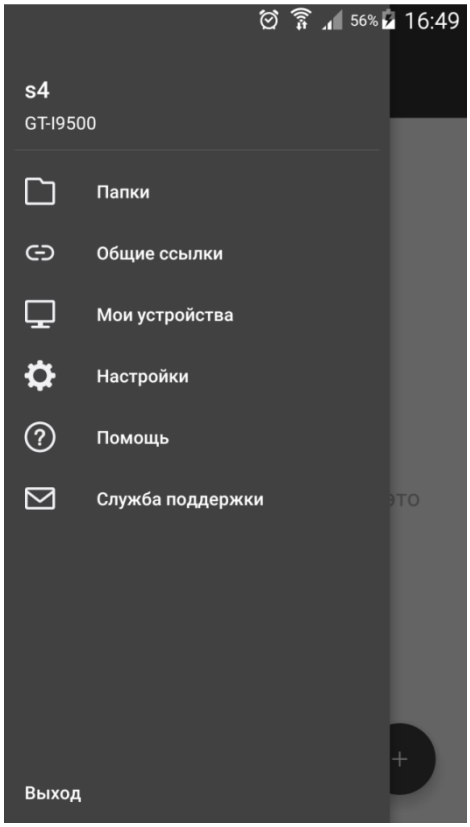

Рисунок 4 – скриншот приложения «Resilio Sync», установленного на ОС Android.

2 Планирование

Проведя исследования существующих программных продуктов в области передачи файлов, можно сделать вывод, что, чтобы программное обеспечение было востребованным и уникальным нужно реализовать следующие функции:

1) Кроссплатформенность. Наличие программы под популярные операционные системы не только уберет необходимость установки дополнительного ПО, но и увеличит базу пользователей.
2) Файловый менеджер. Чтобы пользователь мог выгрузить необходимые документы именно в ту папку, которую он выберет. А при необходимости выбрать нужный файл или папку на удаленном устройстве и загрузить себе.
3) Возможность передачи, как через локальную сеть, так и через интернет. В случае с ЛВС, пользователи, находящиеся в одной сети, могут сэкономить трафик, а также получить максимальную скорость передачи.
4) Предоставление доступа к папкам. Это позволит пользователям делиться файлами, а также принимать файлы от друзей.

2.1 Сценарии использования

События, возникающие в системе с точки зрения пользователя, изображены с помощью диаграммы прецедентов в нотации UML (рисунок 5).

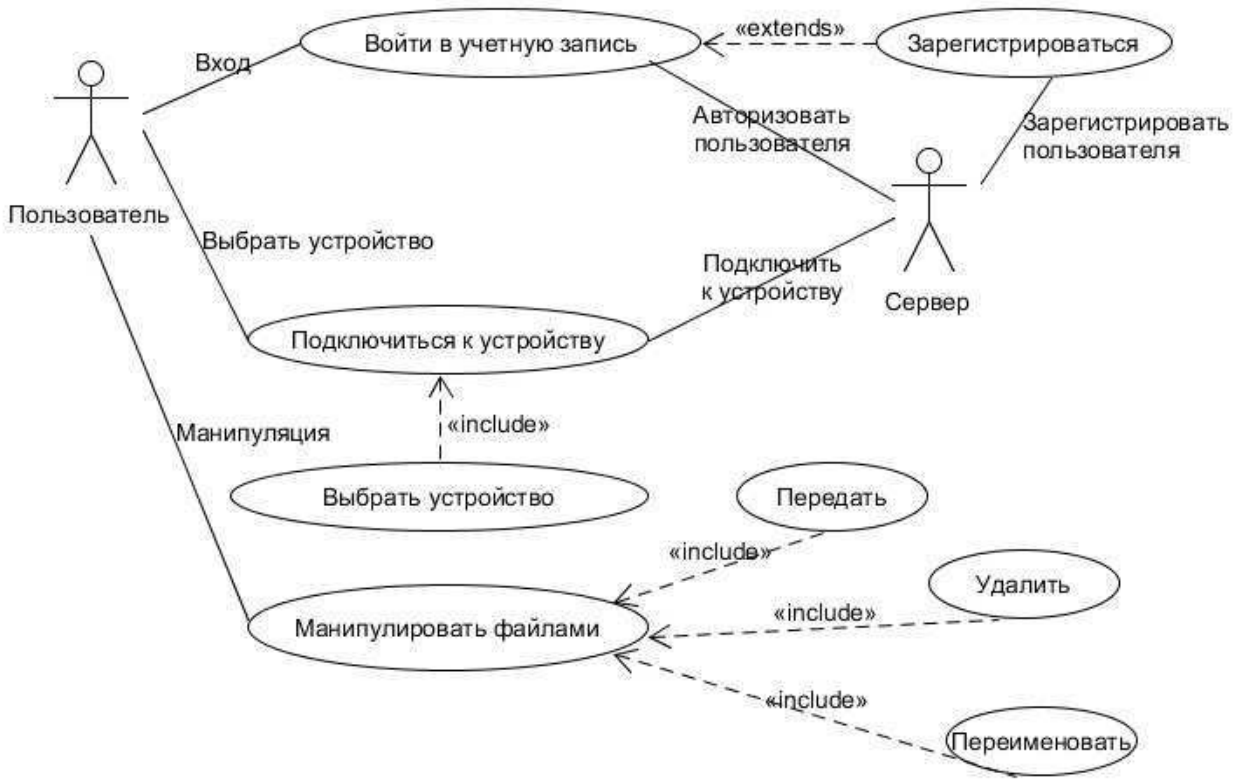

Рисунок 6 – Диаграмма прецедентов

1 Регистрация и авторизация. Описание сценария приведено в таблице 1.

Таблица 1 – Сценарий регистрации и авторизации

Действие пользователя	**Реакция системы**
Запуск приложения	Запускается главное окно приложения. Пользователь нажимает кнопку «Войти/Зарегистрироваться»
Ввод данных, нажатие кнопки «Войти» или кнопки «Зарегистрироваться»	После нажатия кнопки «Войти/Зарегистрироваться», открывается окно «Авторизация/Регистрация». Если пользователь уже зарегистрирован в системе, то ему необходимо заполнить две формы: «имя пользователя» и «пароль». После заполнения необходимо нажать кнопку «Авторизоваться». При совпадении введенной комбинации с комбинацией находящейся в базе данных, окно «Авторизации/Регистрации» закрывается, и на главном окне приложения появляется имя пользователя, что означает успешную авторизацию. Если введенная комбинация в базе не обнаружена, то выводится сообщение об ошибке, а также предложение повторного ввода информации, восстановлении пароля или регистрации. В случае отсутствия учетной записи, для передачи файлов через интернет, пользователю необходимо зарегистрироваться – заполнить форму регистрации. Для регистрации требуется ввести имя пользователя в строку «имя пользователя», пароль в строку «пароль» и повторное ввести в пароль в строку «повторите пароль». После чего происходит закрытие окна «Авторизации/Регистрации», и на главном окне появляется имя пользователя, что соответствует успешному входу в систему.

2 Список друзей, добавление в список друзей. Функция добавления друзей предназначена для передачи файлов и папок и предоставления общего доступа к файлам и папкам. Описание сценария добавления в друзья. Описание сценария приведено в таблице 2.

Таблица 2 – описание сценария добавления в друзья

Действие пользователя	**Реакция системы**
Нажатие на кнопку «Друзья»	Приложение открывает окно «Друзья» и выдает список друзей и список входящих заявок в друзья. Также там находится кнопка «Добавить в друзья», которая предоставляет возможность добавить в друзья пользователя.
Ввод данных, добавление друга	При вводе в поле «Имя пользователя» и нажатии на кнопку «Добавить в друзья», система отправляет заявку пользователю с указанным именем. Если указанный пользователь отсутствует, то система выдаст ошибку. Входящие заявки принимаются кнопкой «Принять», а отклоняются нажатием на кнопку «Отклонить».

3 Подключение к устройству, передача файлов. Пользователь можно без ограничений просматривать файловую систему своих устройств. Для этого ему необходимо авторизоваться на всех своих устройствах, к которым он хочет получить доступ. Просмотр файловой системы друзей ограничен. Друг пользователя сам настраивает, что может видеть пользователь.

Таблица 3 – Сценарий передачи файлов

Действие пользователя	**Реакция системы**
Выбор устройства и подключение к нему	При выборе устройства из списка, происходит подключение к выбранному удаленному устройству. После успешного подключения, доступные файлы и папки отображаются во второй панели файлового менеджера.
Выбор необходимых файлов и папок и их передача	После выбора файлов и папок, и нажатии кнопки «Отправить», файлы передаются на удаленное устройство.

4 Манипуляция с файлами. Пользователю, помимо передачи файлов, доступны функции «Создать папку», «Переименовать», «Удалить». Для своих устройств, пользователи имеют право использовать данные функции без ограничений во всей файловой системе. А для устройств друзей, такое право недоступно. Данные функции будут работать только в папках, для которых предоставлен общий доступ.

Таблица 4 – Манипуляция с файлами.

Нажатие кнопки «Создать папку»	В случае, когда название новой папки уникальное, то папка создается. Если название не уникальное, то выводится сообщение об ошибке, предлагается ввести другое название.
Нажатие кнопки «Переименовать»	Происходит переименование файла или папки. Если введенное новое название не уникально, то выводится сообщение об ошибке, предлагается ввести другое название.
Нажатие кнопки «Удалить»	Выбранный файл удаляется.

2.2 Требования к операционной системе

Клиентская часть программы может быть запущена на операционных средах: Windows, Linux, OS X, Android.

На персональных компьютерах системные программные средства, используемые программой, должны быть представлены лицензионной локализованной версией операционной системы Windows 7 или выше, или OS X или выше, или Linux.

На мобильных устройствах системные программные средства, используемые программой, должны быть представлены версией операционной системы Android 4.1 или выше.

2.3 Требования к среде эксплуатации

Для возможности передачи файлов через интернет требуется подключение к интернету. А для передачи файлов в локальной сети необходимо подключение к локальной сети.

2.4 Обоснование выбора платформ

По данным исследования Яндекс [11] 84% пользователей интернета в течение месяца используют для выхода в сеть больше одного устройства — например, рабочий и домашний компьютеры или компьютер и мобильное устройство. А это значит, что в связи с возрастанием количества устройств у людей, возникает потребность передачи файлов с одного устройства на другое. На мобильных устройствах и персональных компьютерах обычно разные операционные системы, и пользователь сталкивается с несовместимостью приложений. Для решения этой проблемы, было принято решение сделать продукт кроссплатформенным.

2.5 Обоснование выбора платформ Windows, Linux, OS X

Данные платформы были выбраны, так как они покрывают абсолютное большинство персональных компьютеров, а именно 96% процентов всех пользователей по данным аналитического сервиса StatCounter Global Stats - Browser, OS, Search Engine including [12] (см. диаграмму на рисунке 7) [13].

Рисунок 7 – Анализ распространенности операционных систем на персональных компьютерах

2.6 Обоснование выбора платформы Android

По данным аналитического сервиса StatCounter с апреля 2016 года по апрель 2017 года самой популярной операционной системой стал Android (см. рисунок 8) [14].

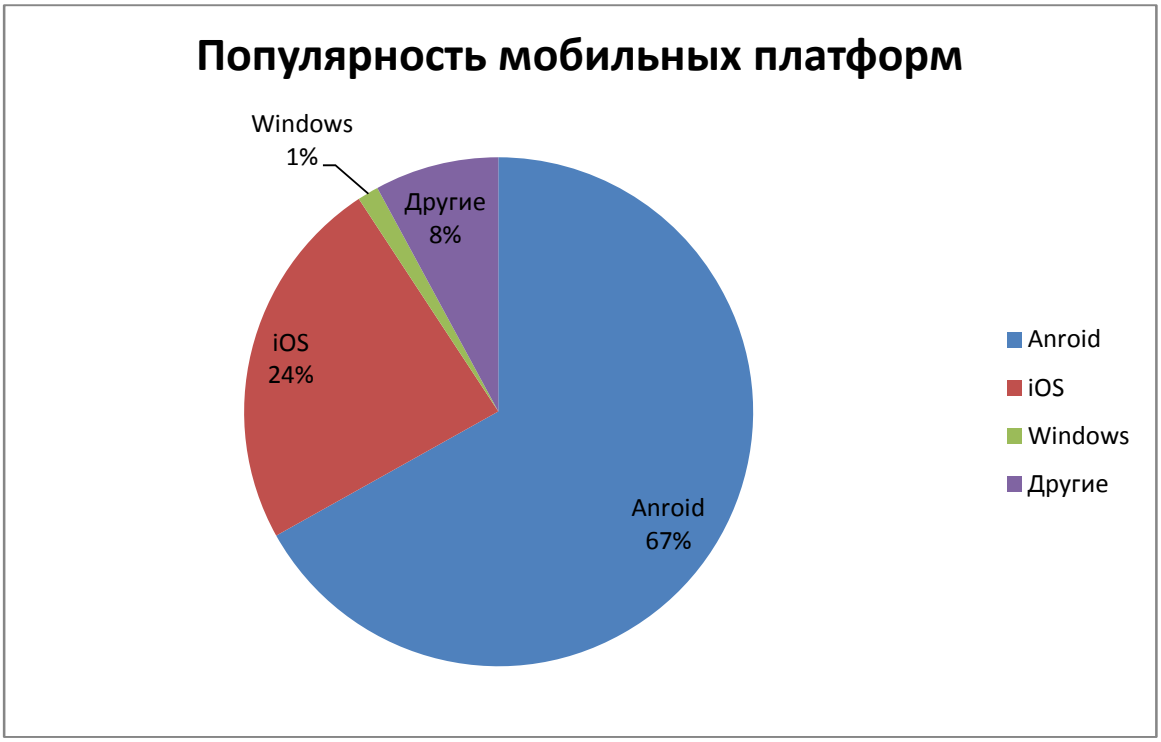

Рисунок 8 – Анализ Распространенности операционных систем

2.7 Обоснование выбора сред разработки

Для написания библиотеки, сервера и интерфейса под персональные компьютеры, использовалась среда разработки Clion.

Clion – это среда разработки для языка C++, ориентированная на кроссплатформенную разработку, с использованием GCC или Clang. Для Windows требуется MinGW. Присутствует поддержка CMake – система сборки кроссплатформенных проектов. В качестве отладчика используется GDB, который позволяет выставлять точки остановки, отслеживать значения выделенных переменных, отображать структуру STL контейнеров, а также поддерживает удаленную отладку.

Для разработки приложения под Android, использовалась среда разработки Android Studio.

Android Studio — это официальная среда разработки для работы с платформой Android, что означает присутствие всех необходимых функций для написания под Android. Поддерживает разработку на Java [15], а также на C++

(NDK). Поддержка отладки как на эмуляторе, так и на реальном устройстве, что позволяет провести тестирование более тщательно.

2.8 Обоснование выбора СУБД

Критериями выбора СУБД были: производительность, стабильность, наличие полной документации. В результате анализа рынка были выбраны следующие СУБД:

— MySQL;

— MariaDB;

— MSSQL.

Microsoft SQL Server 2016 Express SP1 – имеет следующие ограничения: 1 процессор, 1ГБ оперативной памяти, размер файлов баз данных 10ГБ, а также не поддерживается платформа Linux.

MySQL – поддерживает Linux и очень популярна, но данная СУБД имеет закрытые части, а также платна для коммерческого использования.

MariaDB – свободно распространяемая полноценная серверная СУБД. Полностью совместима с MySQL, полностью открыт код. Лучше поддерживается разработчиками. За счет упрощения некоторых стандартов увеличена скорость работы.

Исходя из результатов сравнения, для использования в проектируемом программном продукте была выбрана СУБД MariaDB 10.2.6.

3 Проектирование системы

Ключевой этап разработки приложения – проектирование. Для нашего программного продукта мы выделили следующие пункты:

— проектирование архитектуры приложения

— проектирование платформонезависимой библиотеки

— проектирование сервера

— проектирование мобильного приложения

— проектирование приложения для ПК

3.1 Проектирование архитектуры приложения

Программный продукт состоит из:

— Платформонезависимая библиотека, инкапсулирующая функционал системы;

— Приложение под мобильные устройства с операционной системой ОС Android;

— Приложения для персональных компьютеров, на которых установлена одна из операционных систем: Windows, Linux, OS X;

— Сервер, который обеспечивает взаимодействие устройств через интернет;

— База данных MySQL, которая отвечает за хранение учетных записей пользователей и список друзей пользователя.

Архитектура разрабатываемого приложения предоставлена на рисунке 9.

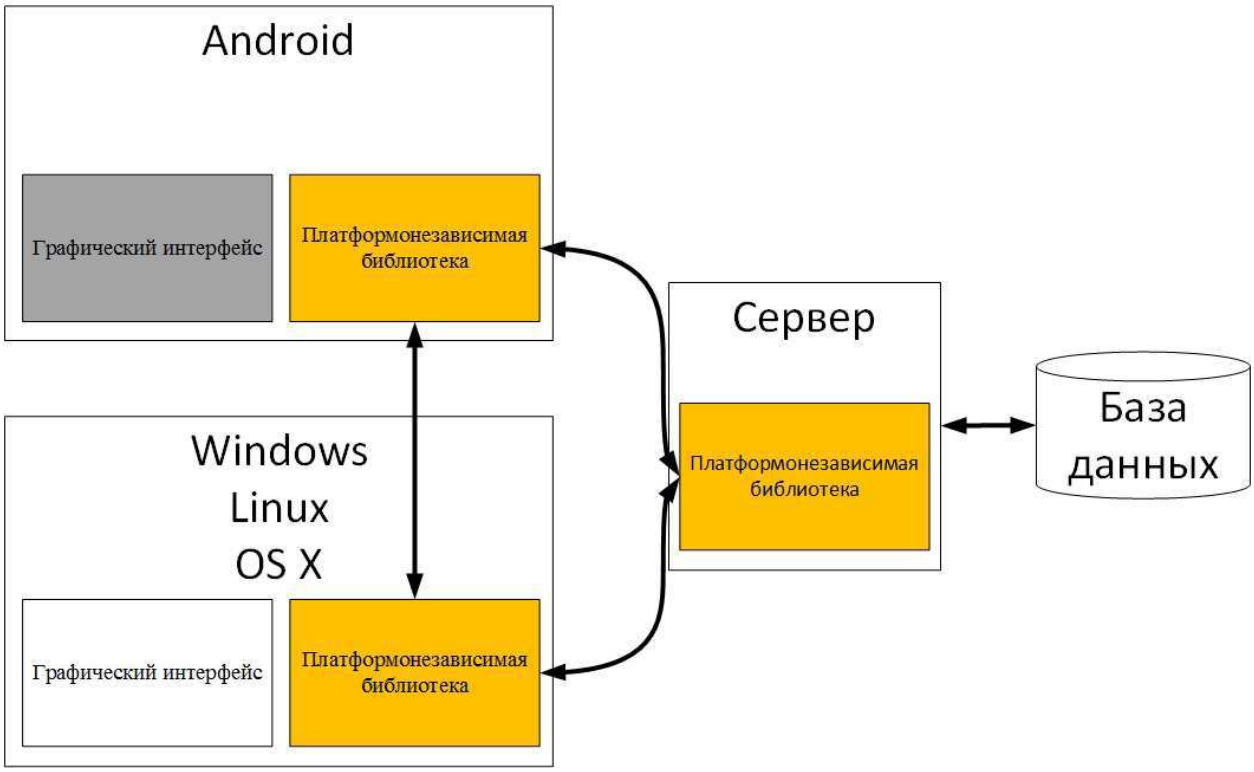

Рисунок 9 – Архитектура разрабатываемого приложения.

3.2 Проектирование платформонезависимой библиотеки

Для достижения кроссплатформенности нам необходимо вынести общую часть функционала программного продукта в отдельную библиотеку независимую от операционной системы.

В качестве языка описания библиотеки был выбран C++. Потому что он компилируется в наивный код, а нативный код обладает высокой производительностью, также позволяет скрыть внутреннюю реализацию системы и открыть доступ только к не обходимым интерфейсам.

Язык C++ - это компилируемый, статистически типизированный язык программирования общего назначения. C++ используется для разработки ПО. Является одним из самых популярных языков программирования.

Один из основных плюсов – возможность компилирования в нативный код. Нативный код – код, компилируемый в машинные инструкции и выполняемый

непосредственно процессором устройства скрыть внутреннюю реализацию системы.

3.3 Архитектура под Android, Windows, Linux, OS X

Приложением является Front-end, который представляет собой взаимодействие с библиотекой и пользователем.

Клиент под ПК написано на C++ с использование библиотеки Qt. Qt используется для графического интерфейса, а язык C++ обеспечивает кроссплатформенность.

Интерфейс клиента под Android написан на Java, промежуточный слой между интерфейсом и библиотекой на C++ (NDK).

3.4 Проектирование сервера

Для функционирования приложения вне локальной сети необходимо создание сервера, который обеспечивает взаимодействие устройств через интернет. Также неотъемлемая часть сервера – база данных. Она хранит в себе учетные записи пользователей и список друзей пользователя.

Сервер ведет список авторизованных устройств и перенаправляет траффик между девайсами.

В таблице users хранится информация о пользователях необходимая для их авторизации: логин, пароль, электронная почта. Таблица friends хранит и логин самого пользователя, и логин его друга. Таблица pending_friends содержит в себе запросы на добавления в друзья.

Скриншот схемы базы данных представлен на рисунке 10.

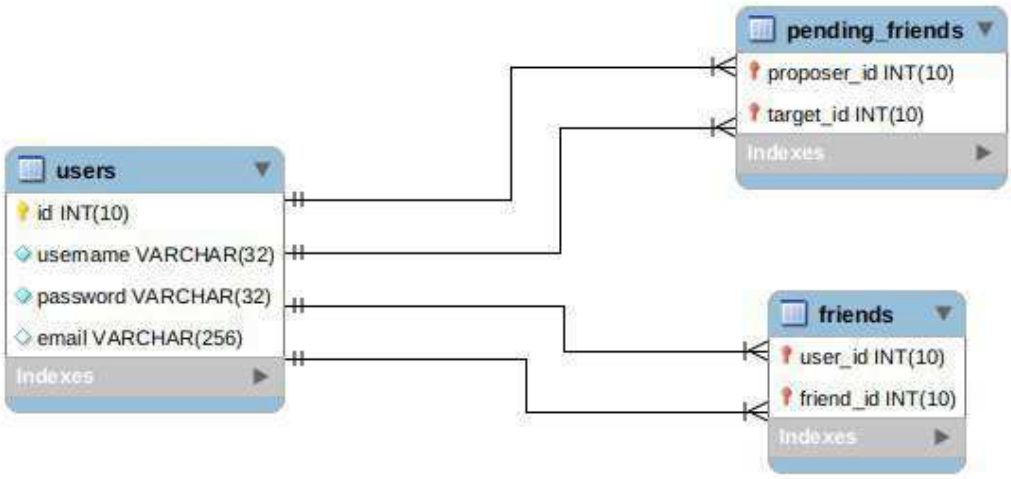

Рисунок 10 – Схема базы данных

4 Реализация

4.1 Реализация платформонезависимой библиотеки

Библиотека написана на языке C++.[16] Для сетевых взаимодействий используется библиотека Boost.Asio. Boost.Asio – это кросс-платформенная библиотека C++ для сетевого и низкоуровневого I/O программирования, которая предоставляет разработчикам асинхронную модель, используя современный подход C++.

Выбор данной библиотеки обоснован нижеперечисленными причинами:

1) Переносимость. Библиотека поддерживает все основные операционные системы, и обеспечивает устойчивое поведение вне зависимости от операционной систем.
2) Расширяемость. Библиотека поддерживает разработку сетевых приложений, в которых может происходить тысячи одновременных соединений. Реализация библиотеки должна использовать механизм, которым расширяемость достигается лучшим способом.
3) Эффективность. Поддерживает такие технологии как scatter-gather I/O и позволяет программам минимизировать копирование данных.

4) Используется хорошо зарекомендовавший API такой, как BSD sockets. API BSD сокетов – это широко используемый и описан во множестве литературных источников. Другие языки программирования часто используют подобный интерфейс для сетевых операций.
5) Простота использования. Библиотека предоставляет инструментарий, а не целый фреймворк.

Помимо сетевых возможностей Boost.Asio предоставляет таймеры, отложенный вызов кода в потоке.

Программе требуется, как минимум, один объект io_service. Io_service представляет связь программы с I/O сервисами операционной системы.

Пример использования библиотеки Boost.Asio изображен на рисунке 11.

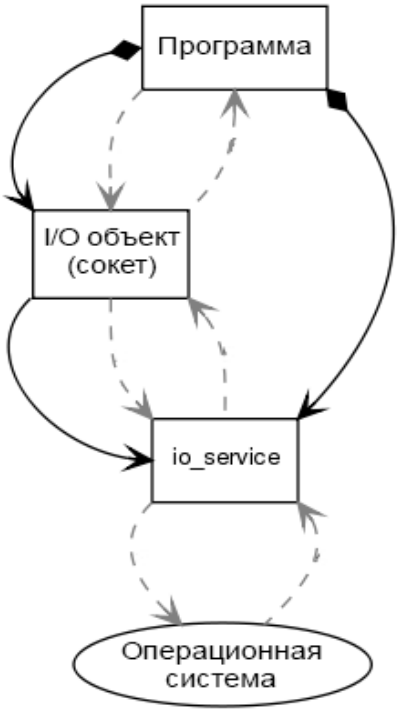

Рисунок 11 – Пример использования библиотеки Boost.Asio

В данном случае мы использовали сокеты UDP, TCP и SSL.

```
   SSLChannel * SSLChannel::mkChannel(boost::asio::io_service& ioService,
endpoint_t endpoint, boost::asio::ssl::context& ctx)
   {
       SSLChannel* ch;
       try
       {
           ch = new SSLChannel(ioService, endpoint, ctx); // создание объекта
SSL канала
           ch->_stream.lowest_layer().connect(ch->_remote); // установление
соединения
           ch->client_handshake(); // процедура верификации
       }
       catch(std::exception& e) // обработка ошибок
       {
           LOG_D(e.what());
           delete ch;
           ch = nullptr;
       }
       return ch;
   }
```

Листинг 1 – создание SSL канала.

```
   void Acceptor::do_accept()
   {
       static boost::asio::ip::tcp::endpoint remote;
       _acceptor.async_accept(_socket, remote, [this](const
boost::system::error_code& e)
   // ожидание входящего подключения
       {
           if(!e) // Если ошибок нет, то создаем объект SSL канала
           {
              auto ch = new SSLChannel(_ioService,
                                       std::move(_socket),
                                       remote,
                                       _ssl_ctx);
                        ch->server_handshake(); // процедура верификации
                        _channelHandler(ch); // отдаем канал внешнему
обработчику

               do_accept(); // заново начинаем процедуру ожидания
           }
           else
           {
               LOG_D(e.message().c_str());
           }
       });
   }
```

Листинг 2 – Прием входящего подключения

Библиотека Boost.Asio предоставляет таймеры и функции для работы с ними (см. листинг 3).

```
    void Dispatcher::do_repeat(const error_t& error_code)
    {
        request_from_server(); // проверка входящих подключений от сервера
        _timer.expires_from_now(boost::posix_time::seconds(1));    //    установка
таймера на 1 секунду от текущего времени
        _timer.async_wait(boost::bind(&Dispatcher::do_repeat, this,
                                      boost::asio::placeholders::error));      //
установка обработчика срабатывания таймера
    }
```

Листинг 3 – пример использования таймера.

Для взаимодействия с файловой системой была использована библиотека Boost.Filesystem. Данная библиотека предоставляет возможности для манипуляции файлами, директориями и работой с путями.

Выбор данной библиотеки обоснован данными причинами:

1) Современный C++ интерфейс, полностью совместим со стандартной библиотекой C++;
2) Переносимость между операционными системами;
3) Обработка ошибок на основе C++ исключений или на основе кодов ошибок;
4) Подходит для широкого спектра приложений: от простейших до очень сложных;

```
    void FileBrowserBackend::get_content(const pack_t &pack)
    {
        json rsp;
        try
        {
            for(fs::directory_entry& e : fs::directory_iterator(_path)) // цикл
прохождения по всему содержимому директории
            {
                rsp.clear();
                uint32_t flags = 0;
                rsp["name"]  =  e.path().filename().string();  //  получаем  имя
текущего файла
                rsp["size"] = 0;
                try
                {
                    flags |= fs::is_directory(e.path()) ? entry_flags_t::F_DIR :
0;
                    rsp["size"]  =  (flags  &  entry_flags_t::F_DIR)  ?  0  :
fs::file_size(e.path()); // получаем размер файла, если это папка, то размер
будет 0.
                }
                catch (boost::filesystem::filesystem_error& e)
                {
                    if(e.code()  ==  boost::system::errc::permission_denied)  //
если произошла ошибка доступа, то устанавливаем соответствующий флаг
                        flags |= entry_flags_t::F_ACCESSDENIED;
                }
                catch (std::exception& e)
                {
                    LOG_D(e.what());
                }
                rsp["flags"] = flags;
                rsp["is_directory"] = (bool)(flags & entry_flags_t::F_DIR);
                rsp["status"] = "in_process";
                write(rsp);
            }
            rsp["path"] = _path.string();
            rsp["status"] = "ok";
            write(rsp);
        }
        catch (std::exception& e)
        {
            rsp["status"] = "failed";
            rsp["reason"] = e.what();
            write(rsp);
        }
    }
```

Листинг 4 – код, отвечающий за получение списка файлов в директории

4.1.1 Внутренняя архитектура библиотеки

Так как C++ объектно-ориентированный язык, то архитектуру можно отразить в иерархии классов.

Сетевая модель состоит из следующих классов (см. рисунок 12):

1) Channel. Абстрактный класс, представляющий канал передачи данных. Поддерживает следующие операции: отправить n-ное количество байт, принять n-ное количество байт, отравить JSON-сообщение, принять JSON-сообщение, асинхронно отправить JSON-сообщение, асинхронно отправить JSON-сообщение. Данные операции являются виртуальными и их необходимо переопределить в классах-наследниках.
    - UdpChannel – отправка и прием сообщений по UDP протоколу;
    - TcpChannel – отправка и прием сообщений по TCP протоколу;
    - SSLChannel – отправка и прием сообщений по защищенному каналу;
    - ProxyChannel – отправка и прием сообщений через сервер, работает поверх любого другого канала.
2) Discover. Класс, занимающийся поиском устройств в локальной сети по средством broadcast-рассылки. Если есть подключение к серверу, то также запрашивает список удаленных устройств с сервера.
3) Acceptor. Принимает входящие TCP и SSL подключения.
4) Dispatcher. Обрабатывает входящие запросы от других устройств и запускает соответствующие сервисы.

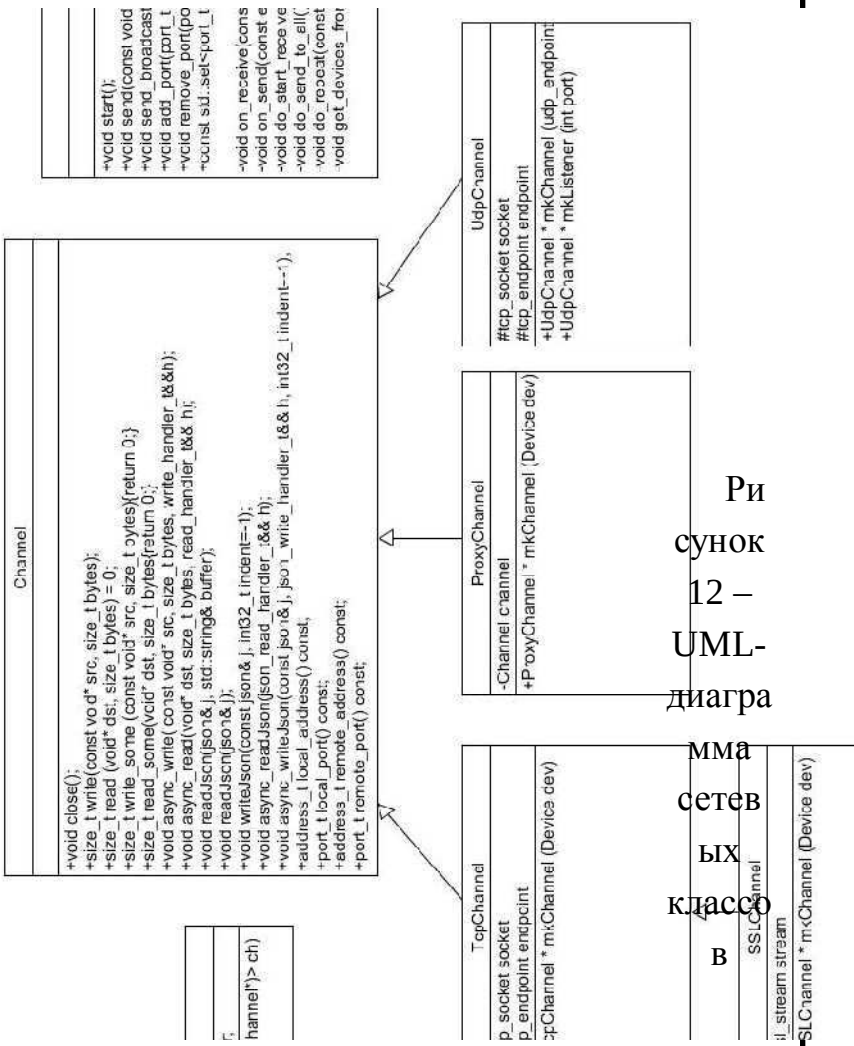

Рисунок 12 – UML-диаграмма сетевых классов

```
    void Dispatcher::as_service(Channel *channel, const json &data)
    {
        std::string  name  =  data["name"].asString();  //  извлекаем  имя  от
требуемого сервиса
        auto it = _reg.find(name); // ищем сервис в списке зарегистрированных
        if(it == _reg.end())
        {
            LOG_D("invalid service\n");
            return;
        }
        auto& h = it->second; // извлекаем функцию создания сервиса
        std::thread t([channel, h]() // создаем новый поток
        {
            ServiceBackend* sb = h(channel); // создаем сервис
            sb->run(); // запускаем сервис
            delete sb; // когда сервис отработал - удаляем
        });
        t.detach();
    }
```

Листинг 5 – Функция запуска backend’а сервиса

Операции системы разделены на различные сервисы, которые не зависят

друг от друга, и это позволяет расширять функционал, не затрагивая существующий код. Каждый сервис имеет Front-end и Back-end составляющие. Front-end выполняется на вызывающем устройстве, а Back-end на принимающем.

Сервисная модель состоит из следующих классов (см. рисунок 13):

1) ServiceFrontend. Абстрактный класс, который является базой для всех остальных Front-end сервисов. Позволяет запускать и останавливать сервис, а также вызывать методы этого сервиса. Вызов метода происходит динамически, то есть существует список имен методов и соответствующих им функций в классе-наследнике. Если реализуемая операция выполняется только в Back-end, то вместо реализации данной функции в Front-end можно использовать перенаправление (forward method (см. листинг 6));
2) ServiceBackend. Абстрактный класс, который является базой для всех остальных Back-end сервисов. Вызывается диспетчером, когда стартуем соответствующий ему Front-end и выполняется на удаленном устройстве. Вызов методов происходит динамически – аналогично Front-end'у.
3) ServiceManager. Позволяет создавать объекты зарегистрированных сервисов.
4) ServiceInterface. Инкапсулирует объект сервиса, позволяет вызвать его методы, отключаться и подключаться, проверять наличие подключения.
5) TransferManager. Ведет учет текущих пересылок, а также информирует об их прогрессе.

На данный момент существуют такие сервисы как:

— FileBrowser – навигация по файловой системе и выдача содержимого директории;
— FileManipulator – различные манипуляции с файлами (удаление, переименование, создание директорий);
— FileSend – выгрузка и загрузка файла;

— Informator – получение информации об устройстве;

— Server – взаимодействие с сервером (авторизоваться, зарегистрироваться, добавить пользователя в друзья, принять или отклонить входящую заявку в друзья);

— LocalService – вариант сервиса без Back-end;

— TransferManager – локальный сервис для добавления новых пересылок и получения информации о прогрессе пересылки.

```
static method_map_t methods
            {
                    __GEN_METHOD_ENTRY("get_dir", forward_method),
                    __GEN_METHOD_ENTRY("change_dir", forward_method),
                    __GEN_METHOD_ENTRY("get_content", get_content),
                    __GEN_METHOD_ENTRY("get_entry_points", forward_method)
            };
```

Листинг 6 – Пример создания списка методов и соответсвующих функций

Каждый Backend исполняется в отдельном потоке, вызовом методов управляет следующий код (см. листинг 7)

```
   void ServiceBackend::run()
   {
       auto& methods = get_methods(); // получить список методов данного
сервиса
       json J;
       std::string buffer;
       do
       {
           try
           {
               _channel->readJson(J, buffer); // получить сообщение от
Frontend'a
               std::string action = J["action"].asString();
               if(action == "stop") // если запрошено действие СТОП, то
остановить backend
                   break;
               if(J["action"] != "invoke" || J["method"].isNull()) // проверки
               {
                   LOG_D("undefined action: \"%s\" or method \"%s\"", action,
J["method"]);
                   LOG_J(J);
                   //break;
               }
               std::string method = J["method"].asString(); // извлекаем имя
метода
               if(methods.find(method) != methods.end()) // ищем метод в списке
               {
                   const method_t& m = methods.at(method);
                   (this->*m)(J); // вызываем метод, передаем аргументы
               }
               else
                   LOG_D("undefined method: %s", method.c_str());
           }
           catch (std::exception& e)
           {
               LOG_D("error: %s\n", e.what());
               LOG_J(J);
               break;
           }
       }while(true);
   }
```

Листинг 6 – Код управления Back-end

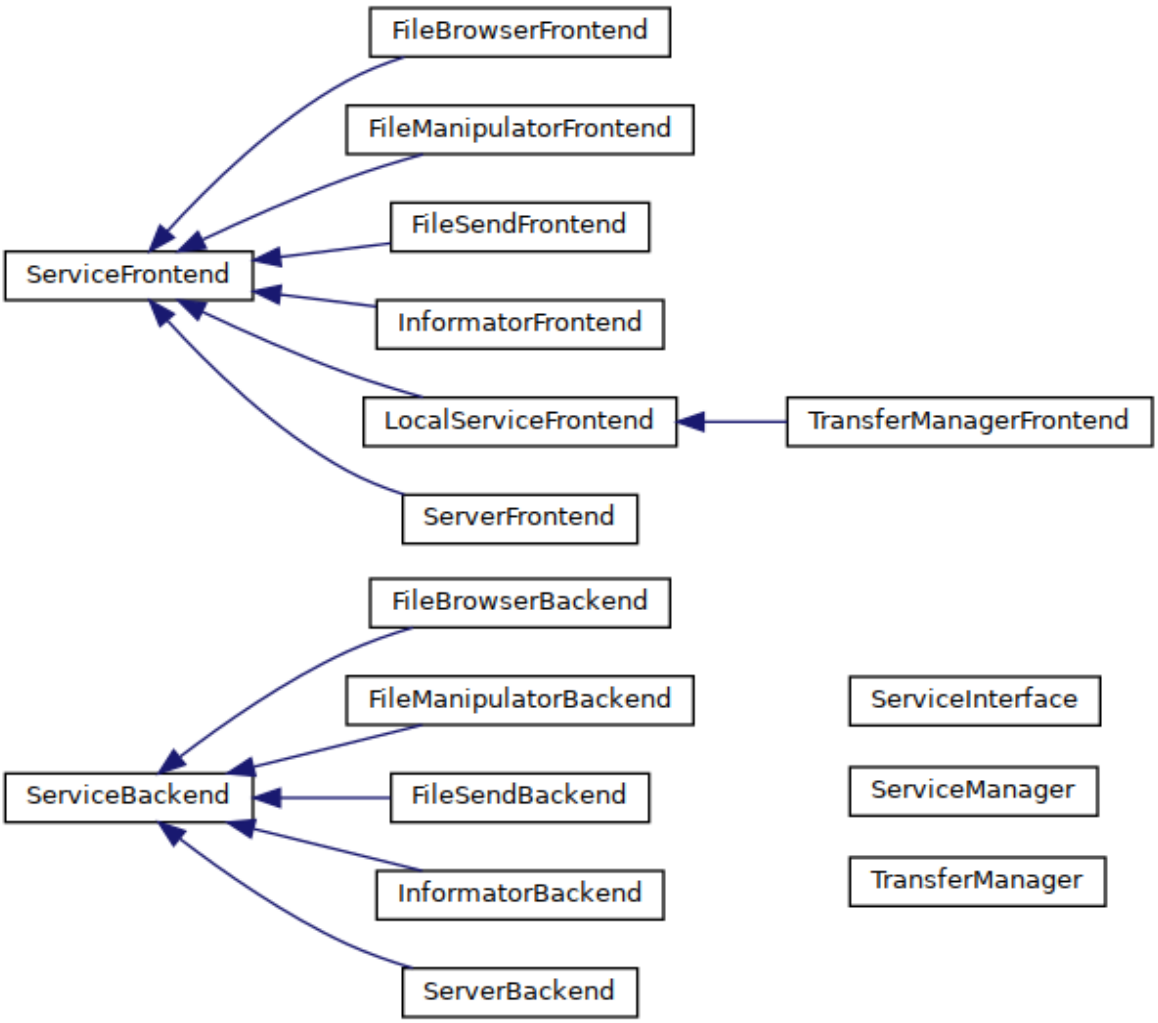

Рисунок 13 – Список классов сервисной модели

Остальные классы включают такие классы как:

— Global – содержит важные объекты для доступа к ним из различных частей библиотеки;

— Device – представляет собой устройство, к которому может быть получен доступ;

— dev_id – идентификатор устройства, состоит из имени пользователя и названия устройства;

— DeviceManager – содержит список устройств;

— Reporter – служит для оповещения пользователя библиотеки о событиях внутри библиотеки;

— AbstractReportConsumer – абстрактный класс (интерфейс), который представляет собой базу для обработчиков сообщений.

Взаимодействия кода графического интерфейса с библиотекой должно производиться через 2 класса: ServiceManager и ServiceInterface. Они позволяют создать подключение к сервису и вызывать необходимые методы (см. листинг 8).

```
    ServiceManager* sm = Global::get()->get_service_manager(); // получаем
объект ServiceManager
    auto s = sm->mkService("filemanipulator", get_device_id()); // запрашиваем
ServiceInterface
    s->connect(); // подключаемся к сервису
    json data; // указываем аргументы для вызова
    data["path"] = get_current_path();
    data["name"] = dir_name.toStdString();
    s->invoke("new_directory", data); // производим вызов
    s->disconnect(); // оключаемся от сервиса
    delete s;
```

Листинг 8 – Пример вызова сервиса

Вызов методов сервисов, а также получение результатов выполнения этих методов происходит с помощью объектов JSON. Так как вызов методов происходит динамически, то было необходимо передавать аргументы и результаты единообразными объектами. Для этих целей существует два основных формата данных: JSON и XML. В данном случае был выбран JSON, потому что он менее избыточный, и более удобен для заполнения и чтения, как человеком, так и компьютером.

Принципы работы JSON таковы:

— Объект – неупорядоченный набор пар ключ-значение (см. рисунок 14);

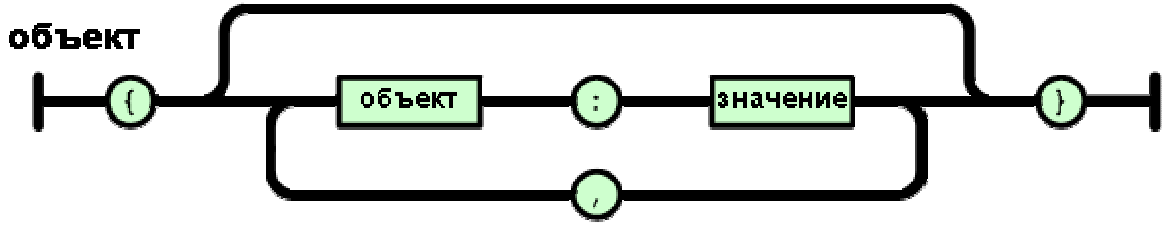

Рисунок 14 – Объект в нотации JSON

— Массив – упорядоченная коллекция значений (см. рисунок 15);

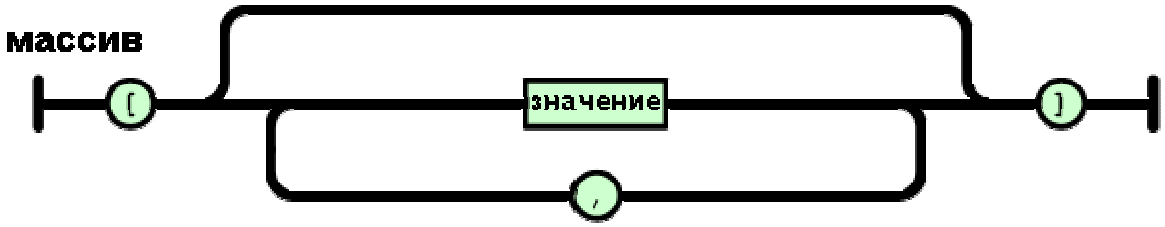

Рисунок 15 – Массив в нотации JSON

— Значение, которое может быть представлено строкой, числом, логическим значением, объектом или массивом (см. рисунок 16).

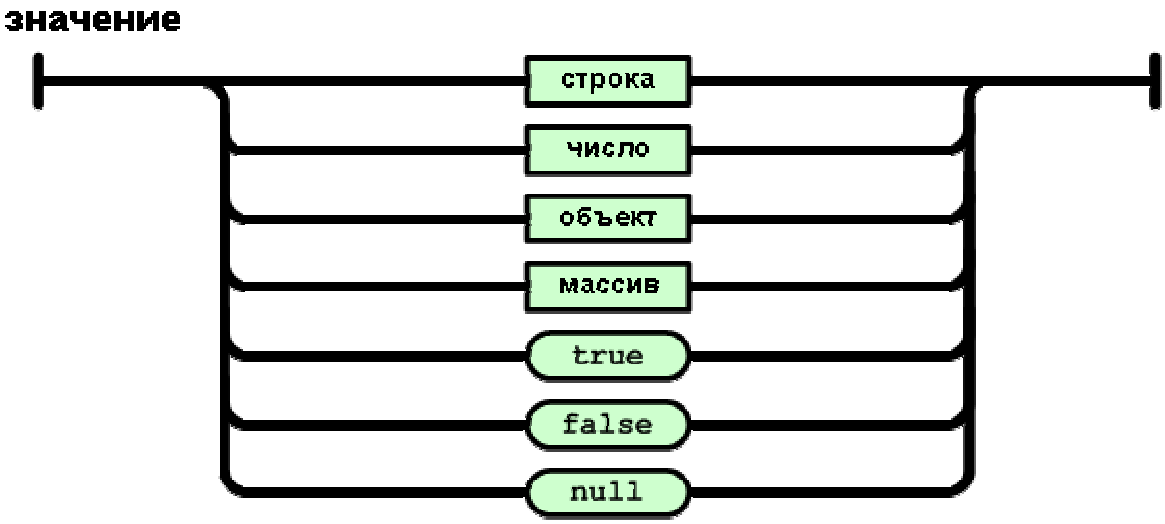

Рисунок 16 – Значение в нотации JSON

Для безопасной передачи информации на сервер мы используем TLS, основанный на протоколе SSL. SSL – это криптографический протокол, обеспечивающий защищенную передачу информации, который позволяет передавать зашифрованную информацию по незасекреченным каналам.

В нашем программном продукте TLS версии 1.2 реализованный библиотекой OpenSSL.

4.2 Реализация серверной части

Сервер нужен для связи устройств и передачи данных через интернет.

Сервер является отдельным приложением, написанный на языке C++. Представляет собой консольное приложение, использующее вышеописанную платформонезависмую библиотеку.

Для подключения сервера к базе данных используется библиотека MySQL++.

MySQL++ - это C++ обертка для API MySQL. Построен по тем же принципам, что и стандартная C++ библиотека, чтобы сделать работу с базой данных такой же простой, как и работу с STL контейнерами. Также MySQL++ обеспечивает средства, которые позволяют наиболее лаконично выразить SQL код (см. листинг 9).

```
    using namespace mysqlpp;
    Connection con; //объект подключения к базе данных
    con.connect(DB_NAME, DB_IP, DB_USER, DB_PASSWORD, DB_PORT); // подключение к базе данных
    Query query = con.query("create table if not exists users(id INT(10) not null auto_increment," // формировани запроса
       " username varchar(32) unique not null,"
        " password varchar(32) not null,"
        " primary key(id));");
    SimpleResult res = query.execute(); // испольнение запроса
```

Листинг 9 – Создание таблицы с использование библиотеки MySQL++

Взаимодействия клиентов с сервером реализовано в классе ServerBackend. Он поддерживает следующие операции:

- Зарегистрировать нового пользователя – добавления нового пользователя в базу данных;
- Авторизоваться – проверка корректности пользовательских и разрешение использования остальных операций сервера;
- Запрос наличия входящих подключений – выдает количество ожидающих подключения для данного устройства;

— Получить список удаленных устройств – выдача списка устройств, доступных через сервер, как своих, так и устройств друзей;

— Отправить заявку на добавление в друзья;

— Получить список друзей;

— Получить список входящих заявок в друзья;

— Принять запрос дружбы;

— Отклонить запрос дружбы.

Проксирование траффика.

Для установления соединения не находящихся в одной сети устройств, необходимо направить трафик через сервер (см. рисунок 17).

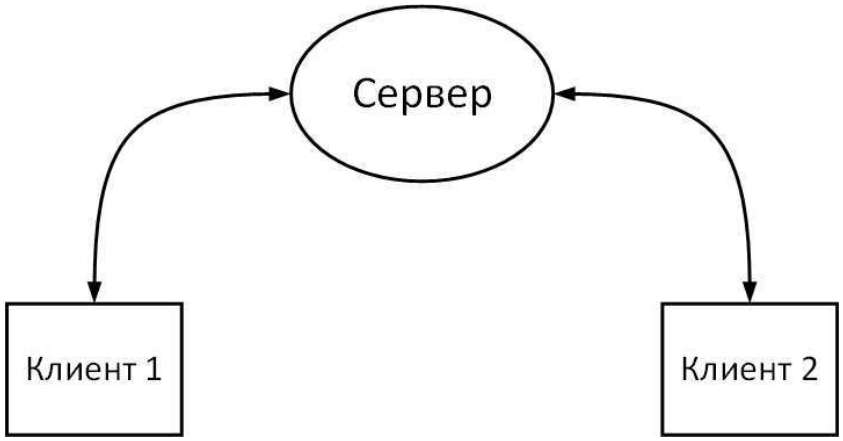

Рисунок 17 – Схема передачи данных через север

При подключении первого клиента к серверу с целью проксирования трафика, возникает проблема – в начальный момент подключения нет канала связи от сервера до удаленного устройства. Эта проблема была решена с помощью очереди ожидания (см. листинг 10)

```
    void Proxy::dispatch(Channel *source, const json& data)
    {
        dev_id device_id = data["device"]; // извлекаем id запрашиваемого
удаленного устройства
        ServerDeviceManager* dm = ServerGlobal::get()->get_device_manager();
        const Device* device = dm->get_device_by_id(device_id); // ищем
устройство в списке устройств
        if(device != nullptr)
        {
            proxy_waiter pw; // создаем объект ожидания
            dm->add_to_wait_queue(device->getId(), &pw); // добавляем в очередь
ожидания
            if(!pw.try_lock()) // захватываем мьютекс
                throw std::runtime_error("couldn't lock pw");
            pw.lock(); // захватываем мьютекс еще раз. эта операция заблокирует
дальнейшее исполнение кода в данном потоке, до тех пор пока мьютекс не будет
разблокирован извне
            Channel* destination = pw.channel(); // извлекаем канал до
запрашиваемого удаленного устройства
            Proxy* proxy = new Proxy(source, destination);
            proxy->do_proxy_job(); // начинаем проксирование трафика
            delete proxy;
        }
        else
        {
            source->close();
        }
    }
```

Листинг 10 – очередь ожидания

После установления соединения, сервер начинает проксирование трафика от одного устройства к другому. Так как канал двунаправленный, то необходимо организовать передачу в двух потоках (см. листинг 11)

```
    void __read_and_write(Channel* src, Channel* dst, size_t buffer_size) //
функция цикла копированя данных из одного канала в другой
    {
        char buffer[buffer_size];
        try
        {
            size_t bytes_read;
            for(;;) // бесконечный цикл, выход и которого производет при закрытии
одного из каналов
            {
                bytes_read = src->read_some(buffer, buffer_size); // Считать
данные с приемника
                dst->write_some(buffer, bytes_read); // записать в источник
            }
        }
        catch (std::exception& e)
        {
        }
    }

    void Proxy::do_proxy_job() // выполнить проксирование
    {
        std::thread destination_to_source_thread([this]() // отдельный поток для
передачи данных из Б в А
```

```
            {
                __read_and_write(_destination,
                 _source,
                 _buffer_size);
            });

        __read_and_write(_source, _destination, _buffer_size); // передача
данных из А в Б
        destination_to_source_thread.join(); // ожидание завершения потока
передачи данных из Б в А
        try
        {
            _source->close();
            _destination->close();
        }
        catch (std::exception& e)
        {
        }
    }
```

Листинг 11 – Проксирование трафика

4.3 Реализация клиентской части приложения для ПК

Программный продукт работает под следующими операционными системами: Windows, Linux, OS X.

Для реализации графического интерфейса используется библиотека Qt 5.8. Qt 5.8 был выбран по следующим причинам:

— Кроссплатформенность. Позволяет запускать написанное с его помощью ПО в большинстве современных операционных систем путём простой компиляции программы для каждой ОС без изменения исходного кода;

— Поддержка концепции слотов и сигналов использующаяся для коммуникации между объектами;

— Является полностью объектно-ориентированным, легко расширяемым и поддерживающим технику компонентного программирования.

Макет интерфейса приложения под персональные компьютеры изображен на рисунке 18.

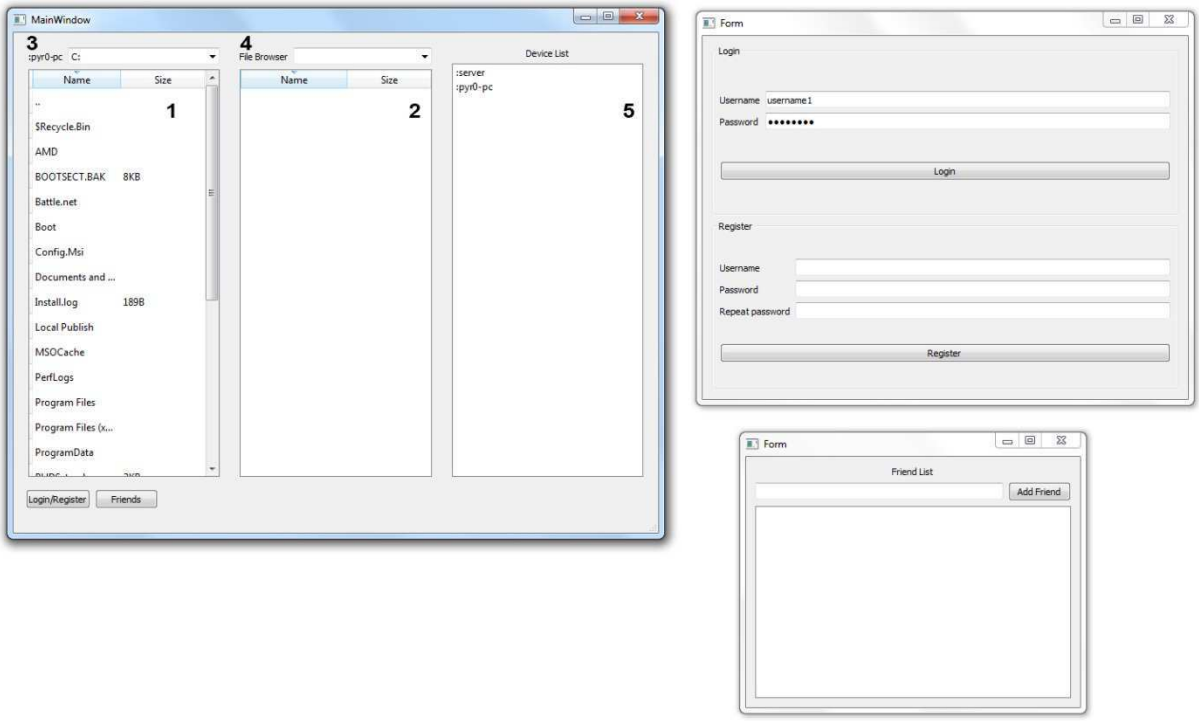

Рисунок 18 – Макет интерфейса приложения под персональные компьютеры

На макете изображено главное окно приложения, окно списка друзей (6), окно входа и регистрации (7). На главном окне присутствуют следующие элементы:

1. список файлов на своем устройстве;
2. список файлов на удаленном устройстве;
3. текущий путь и список точек входа на своем устройстве;
4. текущий путь и список точек входа на удаленном устройстве;
5. список устройств.

4.3 Реализация клиентской части приложения для мобильных устройств

При разработке под Android системы необходимо выбрать уровень минимально поддерживаемого API. Уровень API – это число, которое идентифицирует ревизию программных библиотек, которые предоставляются версией платформы Android [17]. С ростом уровня API появляются новые удобные средства разработки и функции поддерживаемые операционной

системы, но при этом количество устройств поддерживающих данный уровень уменьшается.

В разрабатываемом приложении минимальным поддерживаемым уровнем выбран API 16. Это обусловлено тем, что по официальным данным 95% устройств поддерживаются данным API. [18]

Приложение использует разрешения:

1) INTERNET для сетевых возможностей;
2) READ_EXTERNAL_STORAGE для чтения файловой системы;
3) WRITE_EXTERNAL_STORAGE для записи в файловую систему.

Стандартным языком для написания программного продукта под Android является Java. Java – объектно-ориентированный язык программирования, обладающий такими особенностями как:

— Независимость от платформы;

— Отсутствие прямой работы с памятью;

— Сборка мусора;

Так используется язык Java, а библиотека написана на языке C++, необходимо реализовать некую прослойку для взаимодействия клиента на Java с библиотекой на C++. Для этого необходимо использовать JNI – стандартный механизм для запуска кода под управлением виртуальной машины Java, который написан на языках C/C++[19].

Для создания и управления сервисами из Java кода реализован класс ServiceInterface, который инкапсулирует в себе объект класса ServiceInterface из нативной библиотеки, а также содержит JNI-обертки методов этого класса.

Для передачи сообщений о события из библиотеки в интерфейс на Java присутствует реализация интерфейса AbstractReportConsumer.

```
   extern "C" JNIEXPORT jstring JNICALL // функция должна соответсвовать стандарту вызовов C
   Java_com_example_package_ServiceInterface_invoke(JNIEnv* env,
       jobject thiz,
       jstring j_method,
       jstring j_data)
   {
       ServiceInterface* si = get_service_from_java_object(env, thiz); // получить объект ServiceInterface, инкапсулированный в Java-объекте
       if(!si)
           return 0;
       std::string method, data;
       GetJStringContent(env, j_method, method); // Получить строку из Java-строки
       GetJStringContent(env, j_data, data); // Получить строку из Java-строки
       json j;
       if(!data.empty())
       {
           Json::Reader reader;
           reader.parse(data, j); // аргументы для вызова парсятся в JSON-объект
       }
       auto ret = si->invoke(method, j); // вызов метода сервиса
       if(!ret)
           return env->NewStringUTF(""); // если вызов не удался, то возвращаем пустую строку
       Json::StreamWriterBuilder builder;
       return env->NewStringUTF(Json::writeString(builder, ret.get()).c_str()); // возвращаем результат вызова метода сервиса обратно в Java-код
   }
```

Листинг 12 – Код JNI-обертки метода invoke

Для сохранения указателя на C++ -объект ServiceInterface в Java-классе, используется объект DirectByteBuffer, который представляет собой регион памяти и, соответственно, имеет адрес.

```
   jobject create_service_interface(JNIEnv* env, ServiceInterface* si)
   {
       jobject bb = env->NewDirectByteBuffer(si, sizeof(*si)); // создаем объект DirectByteBuffer по адресу si
       return env->NewObject(__glob_ServiceInterface_class, __glob__ServiceInterface_ctor, bb); // создаем Java-объект ServiceInterface
   }

   ServiceInterface* get_service_from_java_object(JNIEnv* env, jobject obj)
   {
       jobject bb = env->GetObjectField(obj, __glob_ServiceInterface_ptr_field); // извлекаем DirectByteBuffer из Java-объекта
       ServiceInterface* si = (ServiceInterface*)env->GetDirectBufferAddress(bb); // производим преобразование типов
       return si;
   }
```

```
    public class ServiceInterface
    {
        private ByteBuffer _native_service = null; // объект, который необходим
для сохранения указателя на C++ объект ServiceInterface

        private ServiceInterface(ByteBuffer ns)
        {
            _native_service = ns;
        }

        protected void finalize() throws Throwable // деструктор
        {
            try
            {
                destroy();
            }
            finally
            {
                super.finalize();
            }
        }

        static public native ServiceInterface mkService(String service, String
Device); // нативные методы, реализация которых, написана на C++
        static public native ServiceInterface mkServiceToSelf(String service);
        static public native ServiceInterface mkServiceToServer(String service);
        static public native String thisDeviceID();
        public native String device();
        public native boolean connect();
        public native boolean connectTo(String device);
        public native boolean isConnected();
        public native boolean disconnect();
        public native String invoke(String method, String data);
        private native void destroy();
    }
```

Листинг 13 – взаимодействие Java-кода с C++-кодом

На макете изображено главное окно приложения, окно списка устройств, окно входа и регистрации (см. рисунок 19). Список устройств открывается свайпом. На главном окне присутствуют следующие элементы:

1 – список файлов на своем устройстве;

2 – список файлов на удаленном устройстве.

Рисунок 19 – Макет интерфейса приложения под мобильные устройства

5 Функциональное тестирование

Тестирование разработанного программного продукта проводилось с помощью реальных устройств на операционных системах Android, Linux, OS X, Windows.

5.1 Тест 1 – Регистрация и авторизация пользователя

Тестирование программы для операционной системы семейства Linux проводилось на дистрибутиве Arch Linux.

В таблице 5 описано тестирование регистрации и авторизации пользователя.

Таблица 5 – Тестирование регистрации и авторизации пользователя

Свойство	Значение
Выполняемые действия	Запуск приложения, нажатие на кнопку «Войти/Регистрация». Ввод данных для регистрации пользователя в соответствующие поля. После регистрации, выполняем авторизацию пользователя. Вводим данные пользователя, указанные при регистрации, нажимаем кнопку «Войти».
Ожидаемые результаты	При регистрации информация о новом пользователе заносится в базу данных. При авторизации, введенная комбинация имени пользователя и пароля, сравнивается с комбинацией в базе данных, после чего происходит авторизация пользователя.
Полученные результаты	Регистрация проходит успешно – новый пользователь добавляется в базу данных. Авторизация проходит успешно – пользователь успешно заходит под своей учетной записью.

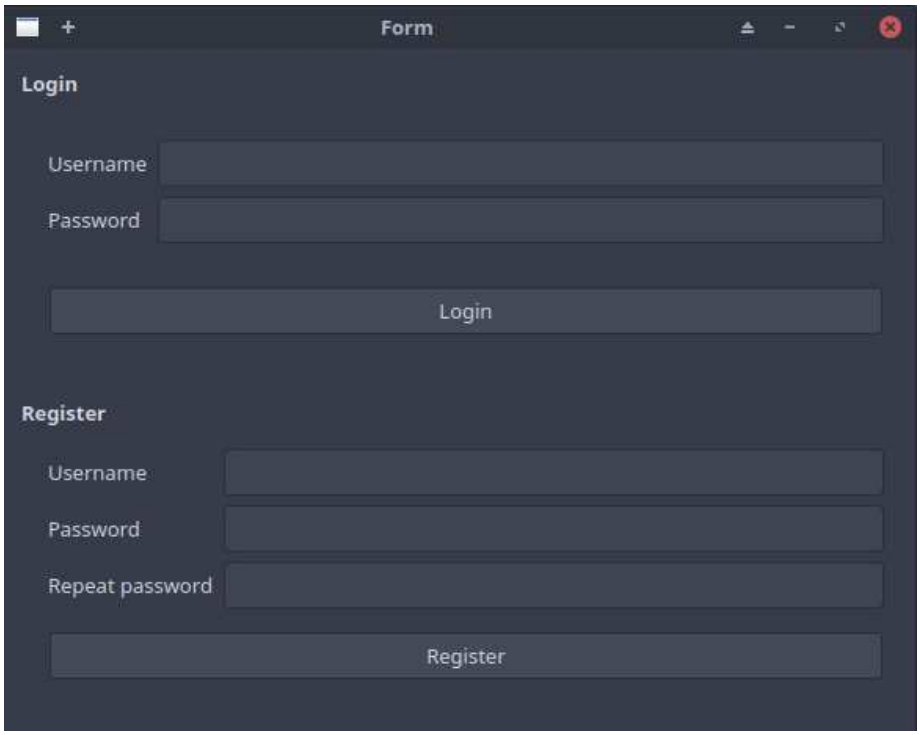

Рисунок 20 – скриншот окна авторизации и регистрации

5.2 Тест 2 – Подключение к устройству

В таблице 6 описано тестирование подключения к устройству.

Таблица 6 – тест подключения к устройству

Свойство	Значение
Выполняемые действия	Приложение запущено, пользователь авторизован. Происходит выбор удаленного устройства в области списка устройств, нажатие правой кнопки мыши, далее «Подключиться».
Ожидаемые результаты	Успешное подключение к устройству. Вывод файлов и папок во второй панели файлового менеджера.
Полученные результаты	Происходит подключение к удаленному устройству, а во второй панели файлового менеджера появляются файлы и папки подключенного устройства.

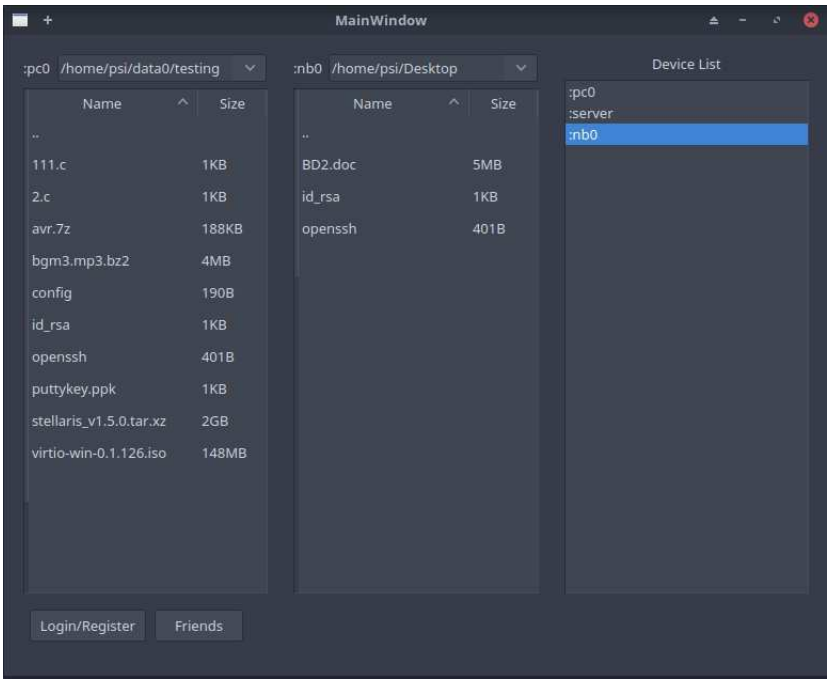

Рисунок 21 – Скриншот подключенного устройства

5.3 Тест 3 - Навигация по файловой системе устройства

В таблице 7 описано тестирование навигации по файловой системе устройства

Таблица 7 – тест навигации по файловой системе

Свойство	Значение
Выполняемые действия	Приложение запущено, пользователь авторизован, удаленное устройство подключено. Производим нажатие на две точки «..» вверху панели файлового менеджера. Около имени устройства, в панели навигации пишем необходимый путь. После также проверяем выпадающий список с доступными путями.
Ожидаемые результаты	При нажатии на «..», пользователь должен оказаться на уровень выше текущей папки. При вводе пути в панели навигации, пользователь должен перейти по заданному пути. А при нажатии

	на выпадающий список – получить доступные пути.
Полученные результаты	Нажатие на «..» переносит на уровень выше. Самостоятельный ввод пути работает корректно, введенный путь переносит пользователя по нему. Выпадающий список показывает доступные пути.

5.4 Тест 4 - Просмотр списка друзей и добавление новых друзей

В таблице 8 описано тестирование просмотра списка друзей и добавление новых друзей.

Таблица 8 – тест навигации по файловой системе

Свойство	Значение
Выполняемые действия	Приложение запущено, пользователь авторизован, Производим нажатие на кнопку «Друзья». Вводим имя пользователя, затем нажимаем кнопку «Добавить». Также принимаем входящую заявку в друзья.
Ожидаемые результаты	При нажатии на кнопку «Друзья», пользователь должен увидеть список друзей. Друзья должны добавиться в список друзей.
Полученные результаты	Окно со списком друзей отображается корректно. В списке друзей появляются два новых друга: с исходящей заявки и с входящей заявки.

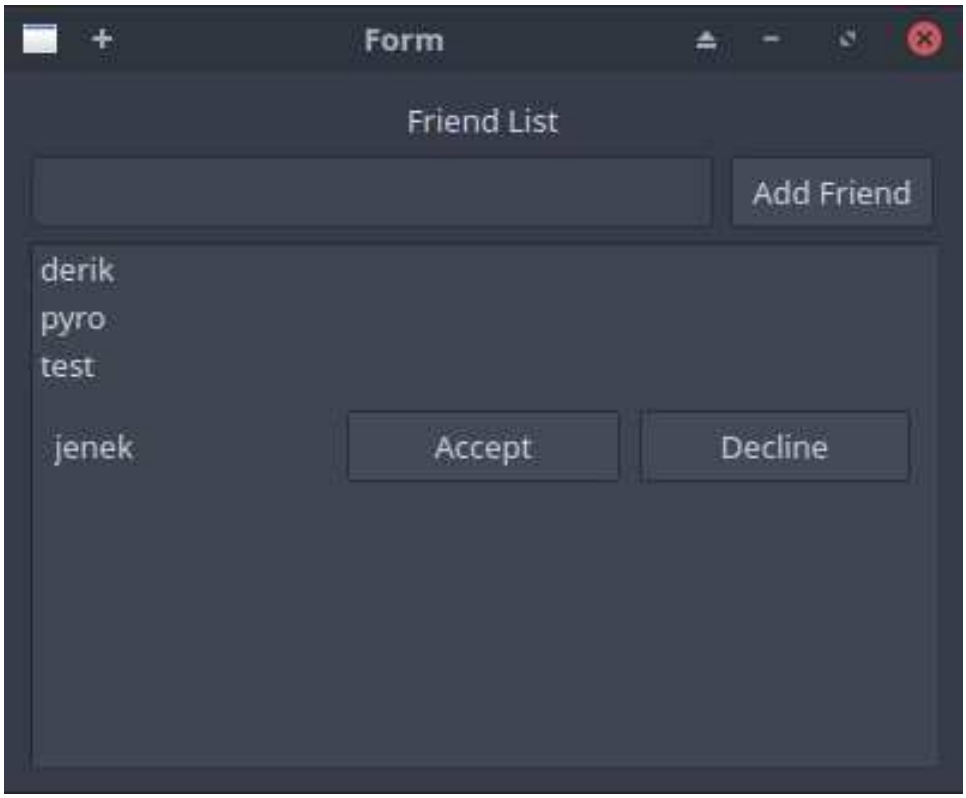

Рисунок 22 – Скриншот списка друзей с входящей заявкой

5.5 Тест 5 - Отправка и прием файлов

В таблице 9 описано тестирование отправки и приема файлов.

Таблица 9 – тест отправки и приема файлов.

Свойство	Значение
Выполняемые действия	Приложение запущено, пользователь авторизован, удаленное устройство подключено. Выбираем файл и папку, нажимаем «Отправить».
Ожидаемые результаты	Выбранные данные отправляются на удаленное устройство.
Полученные результаты	Файлы появились на удаленном устройстве.

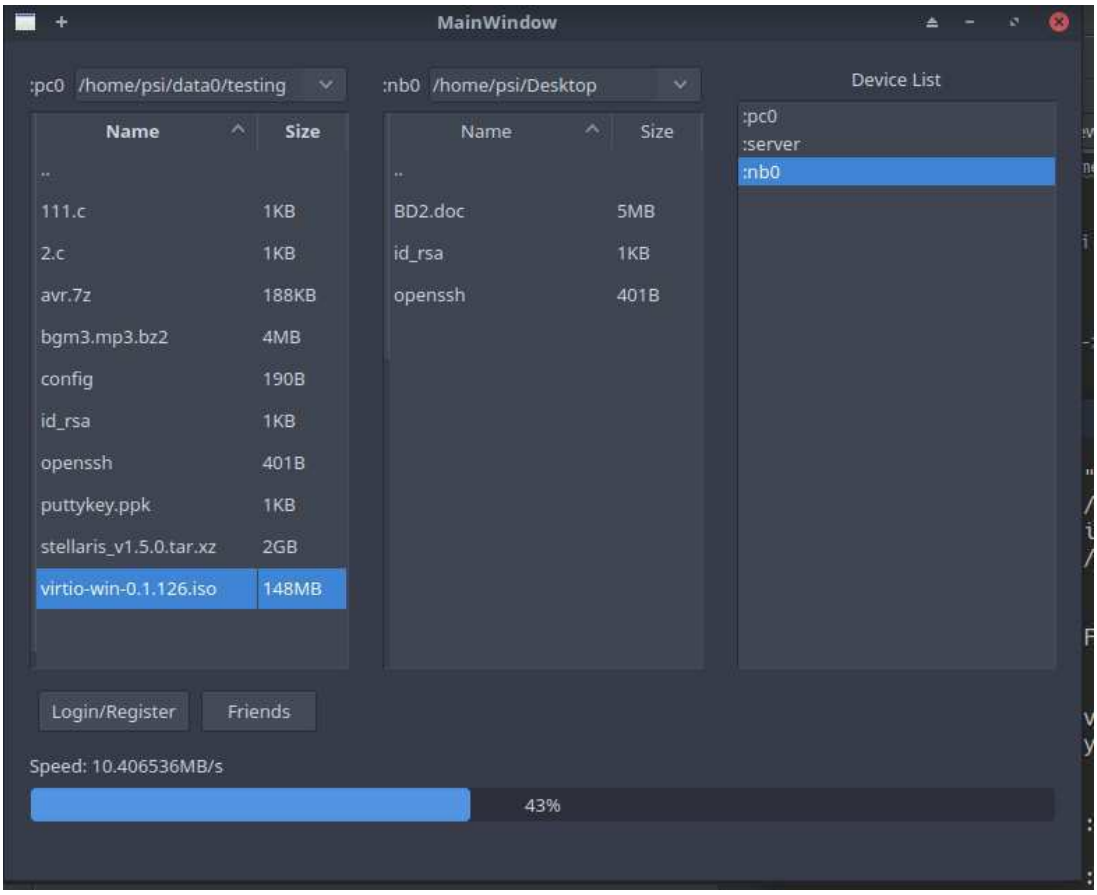

Рисунок 23 – Отправка файла

5.6 Тест 6 – Манипуляция с файлами и папками

В таблице 10 описано тестирование манипуляций с файлами и папками.

Таблица 10 – тест манипуляций с файлами и папками.

Свойство	Значение
Выполняемые действия	Приложение запущено, нажатие правой кнопкой «Удалить». Приложение запущено, нажатие правой кнопкой «Создать папку». Приложение запущено, нажатие правой кнопкой «Переименовать».
Ожидаемые результаты	Файл\папка удаляется. Папка создается. Происходит переименование файла\папки.
Полученные результаты	Файл\папка пропадает из списка. Появилась новая папка. Имя файла\папки поменялось.

Результаты тестирования показали, что программа под персональные компьютеры работает корректно: файлы передаются, авторизация проходит успешно, также работает список друзей.

5.7 Тестирование работоспособности программного продукта на разных операционных системах

Проверим работоспособность на других операционных системах для персональных компьютеров: Windows, OCC Linux, OS X.

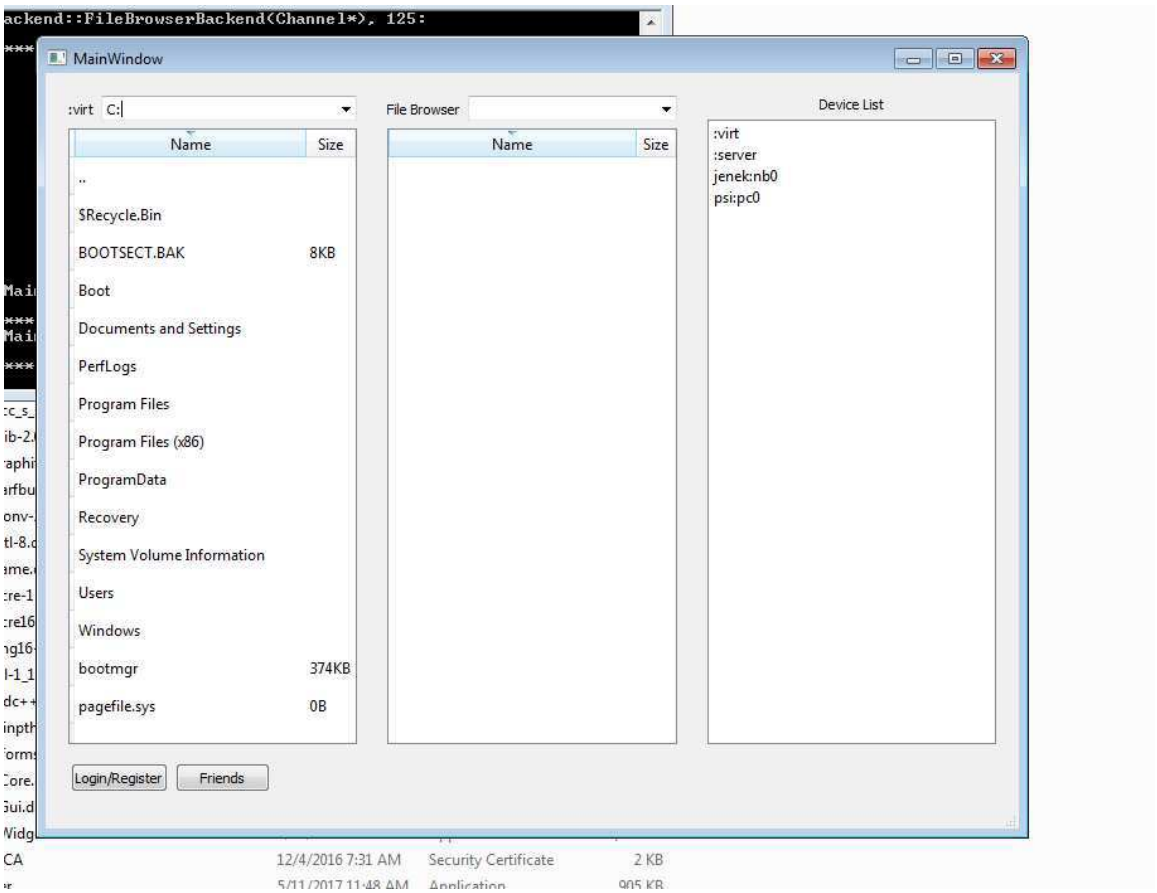

Рисунок 24 – Результат тестирования работоспособности приложения на ОС Windows 7

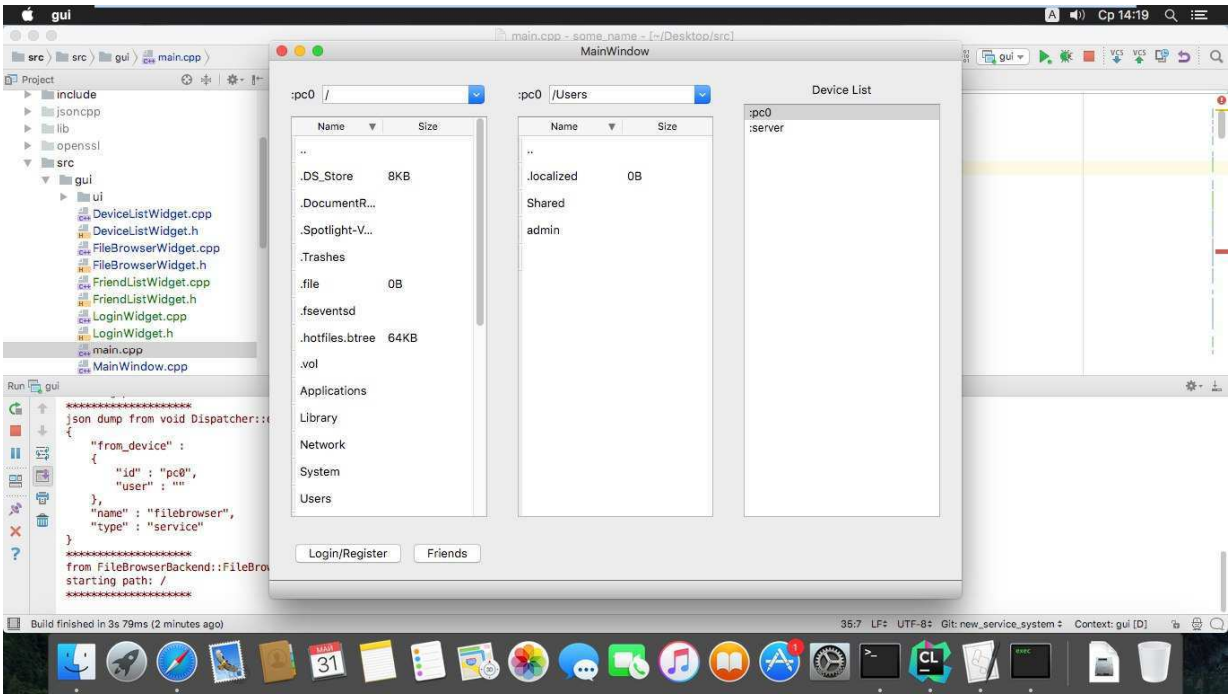

Рисунок 25 – Результат тестирования работоспособности приложения на OS X El Capitan 10.11.6

Тестирование проводилось на Windows 7 и OS X El Capitan 10.11.6 и Arch Linux. Результаты тестирования на рисунках 24 и 25, показывают, что приложение работает корректно на этих операционных системах.

Протестируем разработанный программный продукт под Android. Для этого будем использовать устройство Xiaomi Redmi 2 note с Android 5.1.

Скриншот главной страницы приложения под Android показан на рисунке 26.

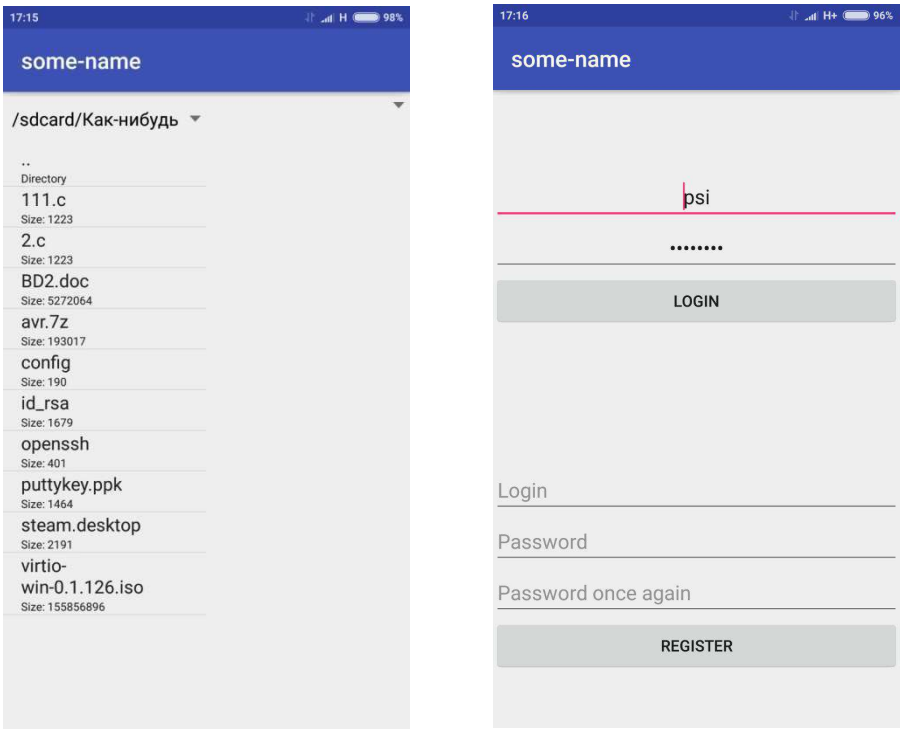

Рисунок 26 – главная страница приложения под Android

Чтобы убедиться в работоспособности разработанного приложения, проведем полное тестирование функционала.

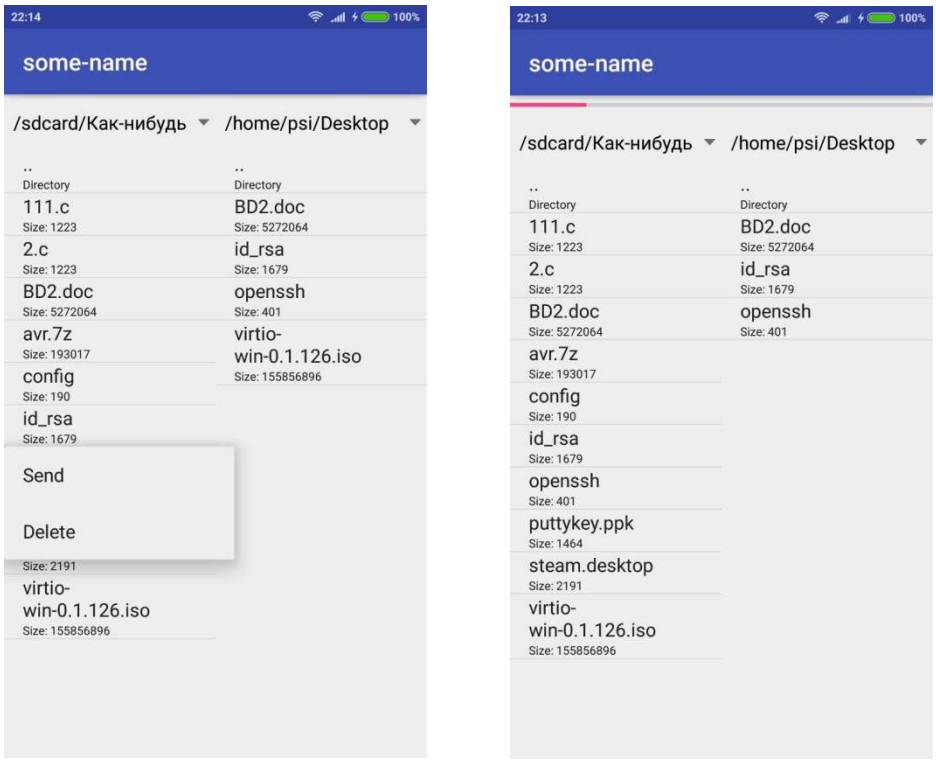

Рисунок 27 – результат тестирования отправки

Функция авторизации работает корректно. Протестируем отправку файла и функции файлового менеджера.

Результаты тестирования на рисунках показывают корректную работу программного продукта. Прогресс бар на рисунке 27 показывает текущее состояние отправки файла.

6 Руководство пользователя

6.1 Назначение разработки

Разрабатываемое программное обеспечение предназначено как для загрузки файлов и папок с удаленного устройства, так и для выгрузки файлов и папок в память удаленного устройства. Программа предоставляет возможность поделиться файлами с друзьями, предварительно настроив общий доступ.

6.2 Подготовка к работе

Для установки программы на персональный компьютер с операционной системой Windows необходимо запустить инсталлятор, далее выбрать каталог установки. По завершению инсталляции программу можно запустить через ярлык на рабочем столе. Скриншот инсталлятора представлен на рисунке 28.

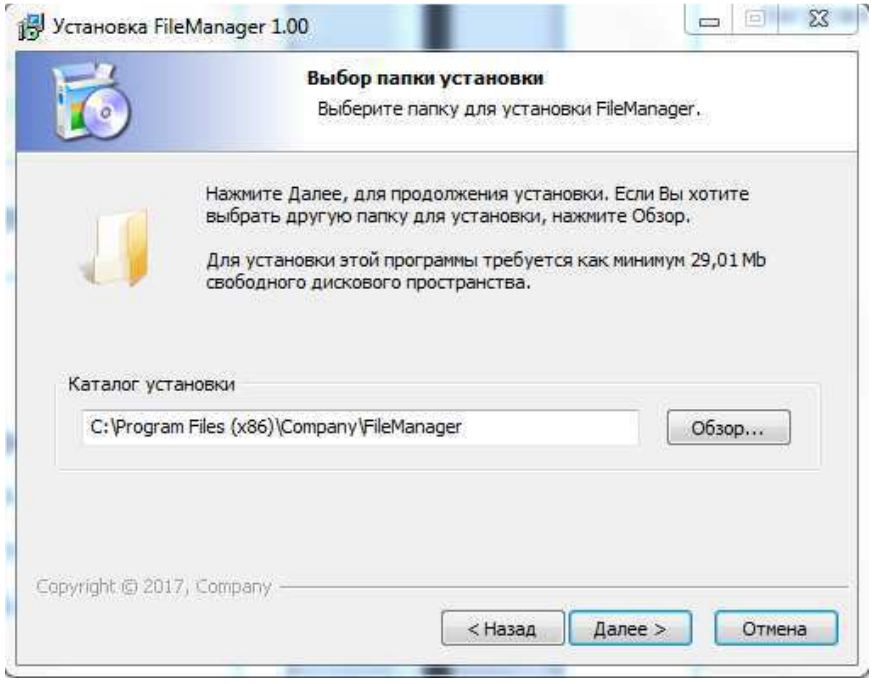

Рисунок 28 – скриншот инсталлятора

Для установки приложения на мобильно устройство с операционной системой Android необходимо загрузить apk-файл и установить его.

ЗАКЛЮЧЕНИЕ

В данном проекте был разработан продукт, предназначенный для удаленной работы с файлами и папками, и их передаче на другое устройство. Простой интерфейс программы не вызовет трудности для начинающего пользователя. Набор функций позволяет использовать приложение на предприятии для обмена документами между рабочими в офисе и на выезде, есть возможность получить доступ к необходимым файлам, которые находятся в общей папке.

В ходе выполнения работы решены следующие задачи:

1) проведен анализ существующего программного обеспечения для удаленной работы с файлами и папками;
2) спроектирована структура приложения;
3) реализован кроссплатформенный программный продукт для удаленной передачи файлов и папок;
4) проведено тестирование программного продукта;

Результат работы – программный продукт, который позволяет загружать, выгружать, делиться и работать с файлами и папками на удаленном устройстве.

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. Исследования Яндекса — Развитие интернета в регионах России [Электронный ресурс]. – Режим доступа: https://yandex.ru/company/researches/2016/ya_internet_regions_2016#dostupvinternetsraznyxustrojjstv. – Заглавие с экрана. – (Дата обращения: 16.05.2017).
2. Operating system market share Worldwide [Электронный ресурс]. – Режим доступа: http://gs.statcounter.com/os-market-share. – Заглавие с экрана. – Англ. – (Дата обращения 16.05.2017).
3. Студенческая наука – взгляд в будущее, 2015 год. Часть 2. [Электронный ресурс]. – Режим доступа: http://kgau.ru/new/all/science/04/content/konf_02_04_2015(2). – Заглавие с экрана. – (Дата обращения: 23.05.2017).
4. Введение в Android NDK / Хабрахабр. [Электронный ресурс]. – Режим доступа: http://habrahabr.ru/post/203014/. – Заглавие с экрана. – (Дата обращения: 23.05.2017).
5. Примеры транспортных протоколов. [Электронный ресурс]. – Режим доступа: http://ww.lektsii.com/3-48411.html/. – Заглавие с экрана. – (Дата обращения: 17.05.2017).
6. Программный комплекс, сети, сетевые атаки, защита. [Электронный ресурс]. – Режим доступа: http://diss.seluk.ru/pr-bezopasnost/1092107-1-v-pervom-razdele-rassmatrivayutsya-postanovka-zadachi-vidi-setey-klassifikaciya-setey-osnovnie-tipi-setevih-at.php/. – Заглавие с экрана. – (Дата обращения: 17.05.2017).
7. Pushbullet [Электронный ресурс]. – Режим доступа: https://play.google.com/store/apps/details?id=com.pushbullet.android&hl=ru. – Заглавие с экрана. – (Дата обращения 16.05.2017).
8. AirDroid [Электронный ресурс]. – Режим доступа: https://play.google.com/store/apps/details?id=com.sand.airdroid&hl=ru. – Заглавие с экрана. – (Дата обращения 16.05.2017).

9. SHAREit - Поделиться Файлами [Электронный ресурс]. – Режим доступа: https://play.google.com/store/apps/details?id=com.lenovo.anyshare.gps&hl=ru. – Заглавие с экрана. – (Дата обращения 16.05.2017).
10. Resilio Sync [Электронный ресурс]. – Режим доступа: https://play.google.com/store/apps/details?id=com.resilio.sync&hl=ru. – Заглавие с экрана. – (Дата обращения 16.05.2017).
11. Развитие интернета в регионах России [Электронный ресурс]. – Режим доступа: https://yandex.ru/company/researches/2016/ya_internet_regions_2016#dostupvinternetsraznyxustrojjstv. – Заглавие с экрана. – (Дата обращения: 19.05.2017).
12. Desktop Operating System Market Share Worldwide Apr 2016 to Apr 2017 [Электронный ресурс]. – Режим доступа: http://gs.statcounter.com/os-market-share/desktop/worldwide. – Заглавие с экрана. – Англ. – (Дата обращения 16.05.2017).
13. Mobile Operating System Market Share Worldwide Apr 2016 to Apr 2017 [Электронный ресурс]. – Режим доступа: http://gs.statcounter.com/os-market-share/mobile/worldwide/. – Заглавие с экрана. – Англ. – (Дата обращения 16.05.2017).
14. Васильев, А.Н. Самоучитель Java с примерами и программами. [Электронный ресурс]. – Электрон. дан. – СПб. : Наука и Техника, 2016. – 368 с. – Режим доступа: http://e.lanbook.com/book/90231 – Загл. с экрана.
15. Страуструп Б. Язык программирования C++ / Б. Страуструп – М.: Бином, 2011. – 1136 с.
16. Уровень API. [Электронный ресурс] – Режим доступа: http://microsin.net/programming/android/what-is-api-level.html. – Заглавие с экрана. – (Дата обращения 15.05.2017).
17. Dashboard Developers Android [Электронный ресурс]. – Режим доступа: https://developer.android.com/about/dashboards/index.html. – Заглавие с экрана. – Англ. – (Дата обращения: 15.05.2017).
18. Сьерра К. Изучаем Java / К. Сьерра, Б. Бейтс – М.: Эксмо, 2012. – 708 с.

19. Таненбаум Э. Компьютерные сети / Э. Таненбаум, Д. Уэзэрол – 5-е изд. – СПб.: Питер, 2012. – 960 с.
20. Как подружить Java и C++ / Хабрахабр [Электронный ресурс]. – Режим доступа: https://habrahabr.ru/post/49660/. – Заглавие с экрана. – (Дата обращения: 19.05.2017).
21. Boost.Asio C++ Network Programming / Хабрахабр [Электронный ресурс]. – Режим доступа: https://habrahabr.ru/post/192284/. – Заглавие с экрана. – (Дата обращения: 15.05.2017).
22. Secure programming with the OpenSSL API [Электронный ресурс]. – Режим доступа: https://www.ibm.com/developerworks/linux/library/l-openssl/index.html/. – Заглавие с экрана. – Англ. – (Дата обращения: 15.05.2017).
23. MySQL++ v3.2.2 User Manual [Электронный ресурс]. – Режим доступа: https://tangentsoft.net/mysql++/doc/html/userman/. – Заглавие с экрана. – Англ. – (Дата обращения: 18.05.2017).
24. 10 reasons to migrate to MariaDB (if still using MySQL) [Электронный ресурс]. – Режим доступа: https://seravo.fi/2015/10-reasons-to-migrate-to-mariadb-if-still-using-mysql/. – Заглавие с экрана. – Англ. – (Дата обращения: 20.05.2017).