Частное учреждение образования

«Колледж бизнеса и права»

ПРОГРАММА ДЛЯ СИНХРОНИЗАЦИИ ПЕРЕДАЧИ ДАННЫХ "Device Sync"

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

по учебному предмету «Практика по разработки сопровождения»

КП Т.096020.401

Руководитель проекта (В.Ю.Михалевич)

Обучающийся (В.А. Лукашонок)

2023

**СОДЕРЖАНИЕ**

Изм.

Изм.

Изм.

Изм.

Изм.

Изм.

Изм.

Изм.

Изм.

Изм.

Изм.

Изм.

Изм.

Изм.

Изм.

Изм.

Лист

Лист

Лист

Лист

Лист

Лист

Лист

Лист

Лист

Лист

Лист

Лист

Лист

Лист

Лист

Лист

№ докум.

№ докум.

№ докум.

№ докум.

№ докум.

№ докум.

№ докум.

№ докум.

№ докум.

№ докум.

№ докум.

№ докум.

№ докум.

№ докум.

№ докум.

№ докум.

Подпись

Подпись

Подпись

Подпись

Подпись

Подпись

Подпись

Подпись

Подпись

Подпись

Подпись

Подпись

Подпись

Подпись

Подпись

Подпись

Дата

Дата

Дата

Дата

Дата

Дата

Дата

Дата

Дата

Дата

Дата

Дата

Дата

Дата

Дата

Дата

Лист

Лист

Лист

Лист

Лист

Лист

Лист

Лист

Лист

Лист

Лист

Лист

Лист

Лист

Лист

Лист

3

3

3

3

3

3

3

3

3

3

3

3

3

3

3

3

КП Т.096020.401 ПЗ

КП Т.716007.401 ПЗ

КП Т.716009.401 ПЗ

КП Т.716007.401 ПЗ

КП Т.096020.401 ПЗ

КП Т.716007.401 ПЗ

КП Т.716009.401 ПЗ

КП Т.716007.401 ПЗ

КП Т.096020.401 ПЗ

КП Т.716007.401 ПЗ

КП Т.716009.401 ПЗ

КП Т.716007.401 ПЗ

КП Т.096020.401 ПЗ

КП Т.716007.401 ПЗ

КП Т.716009.401 ПЗ

КП Т.716007.401 ПЗ

Разраб.

Разраб.

Разраб.

Разраб.

Разраб.

Разраб.

Разраб.

Разраб.

Разраб.

Разраб.

Разраб.

Разраб.

Разраб.

Разраб.

Разраб.

Разраб.

*Лукашонок В.А.*

*Голубев И.В.*

*Гринь А.М.*

*Голубев И.В.*

*Лукашонок В.А.*

*Голубев И.В.*

*Гринь А.М.*

*Голубев И.В.*

*Лукашонок В.А.*

*Голубев И.В.*

*Гринь А.М.*

*Голубев И.В.*

*Лукашонок В.А.*

*Голубев И.В.*

*Гринь А.М.*

*Голубев И.В.*

Провер.

Провер.

Провер.

Провер.

Провер.

Провер.

Провер.

Провер.

Провер.

Провер.

Провер.

Провер.

Провер.

Провер.

Провер.

Провер.

Михалевич В.Ю.

Банцевич. С.В

Банцевич. С.В

Банцевич. С.В

Михалевич В.Ю.

Банцевич. С.В

Банцевич. С.В

Банцевич. С.В

Михалевич В.Ю.

Банцевич. С.В

Банцевич. С.В

Банцевич. С.В

Михалевич В.Ю.

Банцевич. С.В

Банцевич. С.В

Банцевич. С.В

Т. контр.

Т. контр.

Т. контр.

Т. контр.

Т. контр.

Т. контр.

Т. контр.

Т. контр.

Т. контр.

Т. контр.

Т. контр.

Т. контр.

Т. контр.

Т. контр.

Т. контр.

Т. контр.

Н. контр.

Н. контр.

Н. контр.

Н. контр.

Н. контр.

Н. контр.

Н. контр.

Н. контр.

Н. контр.

Н. контр.

Н. контр.

Н. контр.

Н. контр.

Н. контр.

Н. контр.

Н. контр.

Утверд.

Утверд.

Утверд.

Утверд.

Утверд.

Утверд.

Утверд.

Утверд.

Утверд.

Утверд.

Утверд.

Утверд.

Утверд.

Утверд.

Утверд.

Утверд.

Программа для синхронизации передачи данных "Device Sync"

Программа для синхронизации передачи данных "Device Sync"

Программа для синхронизации передачи данных "Device Sync"

Программа для синхронизации передачи данных "Device Sync"

Лит.

У

Лит.

У

Лит.

У

Лит.

У

Лит.

У

Лит.

У

Лит.

У

Лит.

У

Лит.

У

Лит.

У

Лит.

У

Лит.

У

Лит.

У

Лит.

У

Лит.

У

Лит.

У

Листов

Листов

Листов

Листов

Листов

Листов

Листов

Листов

Листов

Листов

Листов

Листов

Листов

Листов

Листов

Листов

37

50

55

50

60

50

55

50

60

50

55

50

60

50

55

50

КБП

КБП

КБП

КБП

КБП

КБП

КБП

КБП

КБП

КБП

КБП

КБП

КБП

КБП

КБП

КБП

[**Введение** 3](#_Toc137784828)

[1 Описание задачи 5](#_Toc137784829)

[1.1 Анализ предметной области 5](#_Toc137784830)

[1.2 Постановка задачи 6](#_Toc137784831)

[2 Проектирование системы 7](#_Toc137784832)

[2.1 Требования к приложению 7](#_Toc137784833)

[2.2 Проектирование модели 7](#_Toc137784834)

[2.3 Концептуальный прототип 10](#_Toc137784835)

[3 Описание реализации программного средства 12](#_Toc137784836)

[3.1 Инструменты разработки и применяемые технологии 12](#_Toc137784837)

[3.2 Функции: логическая и физическая организация 12](#_Toc137784838)

[3.3 Функциональное тестирование 14](#_Toc137784839)

[4 Применение 17](#_Toc137784840)

[4.1 Назначение программного средства 17](#_Toc137784841)

[4.2 Условия применения 17](#_Toc137784842)

[Заключение 18](#_Toc137784843)

[Список использованных источников 19](#_Toc137784844)

[Приложение А 20](#_Toc137784845)

[Приложение Б 32](#_Toc137784847)

**Введение**

Фото, видео, сообщения – в современном мире не проходит и дня без взаимодействия с теми или иными данными, и вопрос об их передачи обстоит особенно остро. Возможность быстро передать необходимы файл или сообщения на устройство является незаменимой, именно для таких целей создаются экосистемы позволяющие “бесшовно” перемещается между устройствами.

Существует множество программных средств, решающих данную проблему в той или иной мере начиная от мессенджеров заканчивая как было оговорено экосистемами. Спрос на быстрый и удобный способ обмена данными постоянный и большой, что доказывает существования бесчисленного количество мессенджеров и альтернативных им способов обмена данными.

Целью курсового проекта на тему «Программа для синхронизации передачи данных "Device Sync"» является разработка программы, с помощью которой пользователи могли бы передавать текстовыми сообщениями и файлы. Для того, чтобы пользователь мог передавать данные, ему необходимо настроить соединение между устройствами. Для этого необходимо указать внешний Ip адрес получателя порт по которому он сможет принимать данные, а также собственный порт для получения данных.

Решение поставленных задач отражено в пояснительной записке, которая состоит из четырех разделов и содержит необходимую и достаточную информацию по использованию данного программного средства.

В первом разделе «Описание задачи» рассматривается сущность и актуальность поставленной задачи, описание существующих аналогов, описание основных функций для разработки программного средства.

Второй раздел «Проектирование системы» содержит проектирование модели на основе выбранных диаграмм, требования к аппаратным и операционным ресурсам, средства защиты, организация данных и описание концептуального прототипа.

Третий раздел «Описание реализации программного средства» включает описание избранной технологии для реализации программы и избранные инструменты разработки, физическая организация данных, способы реализации функции пользователя, формат входных и выходных данных, функциональное тестирование, описание справочной системы.

Четвертый раздел «Применение» содержит информацию, необходимую в процессе эксплуатации программного средства: его назначение и условия применения.

В заключении описывается выполнение поставленной задачи, степень соответствия проектных решений задания, причины несоответствия, если таковые имеются.

В приложении А представлен текст программных модулей, в приложении Б – формы выходных документов.

Графическая часть представлена диаграммами вариантов использования, классов, деятельности.

**1 Описание задачи**

**1.1 Анализ предметной области**

В современном мире передача данных стала неотъемлемой частью нашей жизни. Мы все больше и больше зависим от средств коммуникации, которые позволяют нам обмениваться информацией, начиная от мгновенных сообщений в мессенджерах и заканчивая отправкой электронной почты и использованием файлообменников. На данный момент на рынке существует огромное разнообразие инструментов и платформ, предназначенных для передачи данных в различных форматах.

Программное средство, которое мы планируем разработать, будет ориентировано на людей, которые ежедневно сталкиваются с необходимостью передачи данных. Практически каждый человек, использующий смартфон, планшет, компьютер или другие устройства связи, активно использует интернет для передачи информации. Будь то личные или деловые цели, передача данных является неотъемлемой частью нашей современной жизни.

Данные — это формы представления информации, с которыми работают информационные системы и их пользователи. Они могут быть представлены в различных форматах, включая текстовую информацию, изображения, аудио- и видеоматериалы, документы и многое другое. Передача данных может осуществляться как синхронно, в режиме реального времени, так и асинхронно, с возможностью сохранения и обработки информации в удобное время.

Основные функции программного средства для передачи данных включают:

* Обмен текстовыми сообщениями, который является одним из наиболее распространенных и удобных способов коммуникации. Это может быть как простой текстовый чат, так и более сложные системы мгновенных сообщений с поддержкой эмодзи, смайликов и других элементов форматирования.;
* Возможность прикрепления к сообщению различных файлов, таких как фотографии, изображения, документы, видео и аудио. Это позволяет пользователям обмениваться не только текстовой информацией, но и мультимедийными материалами, делая коммуникацию более насыщенной и интересной;
* Иногда может быть необходимостью в потоковой передаче данных, такой как видео- и аудиовещание, проведение онлайн-конференций или стриминг музыки и видео. Это требует специальных алгоритмов и протоколов передачи данных, чтобы обеспечить стабильную и качественную передачу информации в режиме реального времени.

Наше программное средство будет предназначено для широкого круга пользователей, которым необходимо передавать данные с помощью интернета. Оно предоставит возможность пользователям отправлять и получать фотографии, видеоматериалы и текстовые сообщения. Для пользователей десктопной версии программы также будет доступна функция получения команд, которые позволят им выполнять различные операции и действия с переданными данными.

Мы стремимся создать программное средство, которое будет простым и удобным в использовании, но в то же время обеспечит высокую степень безопасности и надежности передачи данных. Мы уверены, что наше решение сможет удовлетворить потребности пользователей и стать надежным партнером в их повседневной коммуникации и обмене информацией.

**1.2 Постановка задачи**

Исходя из анализа предметной области можно выделить следующие задачи:

* Соединение пользователей;
* переписка пользователей текстовыми сообщениями, с использованием текстовых сообщений;
* передача фалов;
* передача команд и последующее выполнение в десктопной версии.

Существует довольно много аналогов для данного приложения, такие как KdeConnect, MS Связь с телефоном и другие. Разрабатываемый мессенджер направлен на пользователей с Windows 10 и выше, Android 8.1 и выше. Программное средство сильно упрощено и не имеет больших массивов данных.

**2 Проектирование системы**

**2.1 Требования к приложению**

Разрабатываемое программное средство должно иметь понятный и удобный в использовании интерфейс, чтобы взаимодействие между программой и пользователем было максимально упрощено.

Согласно общим требованиям, графический интерфейс разрабатываемого программного средства должен:

* ориентироваться на пользователя, который общается с программой на внешнем уровне взаимодействия;
* сохранять стандартизированное назначение и местоположение на экране графических объектов, работающих в среде Windows и Android.

Интерфейс программного средства будет разрабатываться с учетом общих требований к пользовательскому интерфейсу. Пользователь может изменять настройки по своему усмотрению при работе с программным средством.

Существуют общие требования, которые предъявляются к программным приложениям:

* соответствие стандартам организации интерфейса: использование многооконного подхода, реализация управления работой программного приложения с помощью элементов управления;
* требования к выбранной цветовой схеме рабочего экрана;
* выполнение одной функции с помощью разных элементов управления.

Для обучения пользователя необходимо разработать справочную систему, в которой должны быть раскрыты все аспекты работы с программой, возможные трудности, возникшие во время работы и пути их решения.

Для удобства работы пользователя с программным средством необходимо при разработке форм придерживаться единого стиля оформления. Формы не должны быть перегружены излишней информацией или содержать информацию, не относящуюся к данной форме. Также необходимо предусмотреть защиту данных от удаления и изменения, а также от ввода некорректных данных. В случае ввода некорректных данных или попытке совершить запрещенные действия, пользователь должен быть проинформирован о своих действиях с помощью диалоговых окон.

**2.2 Проектирование модели**

Цель моделирования данных состоит в обеспечении разработчика информационной системы концептуальной схемой базы данных в форме одной или нескольких локальных моделей, которые относительно легко могут быть отображены в любую систему без данных.

Наиболее распространенным средством моделирования данных является диаграмма «Сущность-связь» (ERD). С их помощью определяется важные для предметной области объекты (сущности), их свойства (атрибуты) и отношения друг с другом (связи). ERD непосредственно используются для проектирования реляционных баз данных.

Диаграмма «Сущность-связь» в нотации Чена представлена на рисунке 2.1.

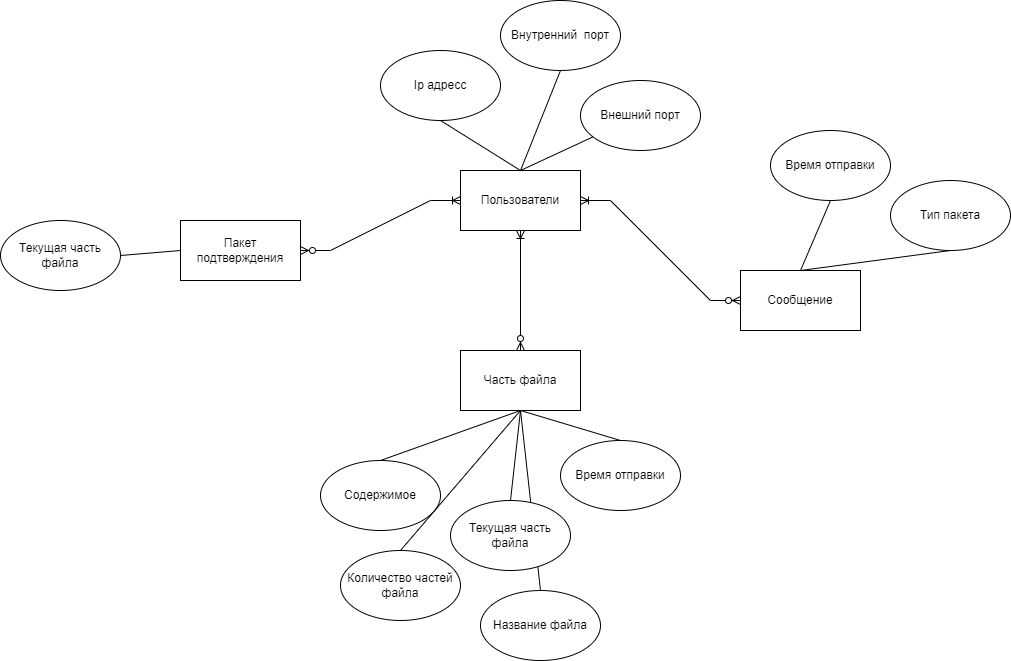


Рисунок 2.1

Исходя из предметной области можно выделить следующие сущности разработки: «Пользователи», «Часть файл», «Сообщение», «Пакет подтверждения».

Для сущности «Пользователь» атрибутами будут являться:

* Ip адрес;
* Внутренний порт;
* Внешний порт.

Для сущности «Сообщение» атрибутами будут являться:

* Тип пакета;
* Время отправки.

Для сущности «Часть файла» атрибутами будут являться:

* Содержимое;
* Название файла;
* Время отправки;
* Количество частей файла;
* Текущая часть файла.

Для сущности «Пакет подтверждения» атрибутами будут являться:

* Текущая часть файла.

Суть диаграммы вариантов использования заключается в том, что проектируемая система представляется в виде множества сущностей или актеров, взаимодействующих с системой с помощью, так называемых вариантов использования.

К «extend» относится информация о неудаче:

* отправки файла;
* подключения;
* отправки сообщения;
* выполнения команды.

К «include» относится:

* статус отправляемого файла.

Диаграмма вариантов использования представлена на листе 1 графической части.

Диаграмма классов служит для представления статической структуры модели системы в терминологии классов объектно-ориентированного программирования. Диаграмма может отражать различные взаимосвязи между отдельными сущностями предметной области, такими как объекты и подсистемы, а также описывать их внутреннюю структуру и типы отношений.

При реализации программного средства были реализованы следующие основные классы:

* Message;
* BytesMessage;
* FileChunk;
* AcknowledgePackage;
* StringMessage.

Класс «Message» является базовым для всех остальных классов, хранит тип передаваемого пакета и время передачи. Класс «ByteMessage» необходим для передачи не текстовых данных и содержит массив байтов, отвечающих за содержимое сообщения. Класс «StringMessage» необходим для передачи текстового сообщения. Класс «FileChunk» наследуется от класса «ByteMessage» и используется для передачи части от файла, содержит название файла. Класс «AcknowledgePackage» необходим для подтверждения получения необходимой части файла и разрешения продолжения отправки.

Диаграмма классов представлена на листе 2 графической части.

При моделировании поведения проектируемой или анализируемой системы возникает необходимость детализировать особенности алгоритмической и логической реализации выполняемых системой операций. Для моделирования процесса выполнения операций в языке UML используются так называемые диаграммы деятельности.

Диаграммы деятельности – частный случай диаграмм состояний. Основная цель использования таких диаграмм – визуализация особенностей реализации операций классов, когда необходимо представить алгоритмы их выполнения. Диаграмма деятельности представлена на листе 3 графической части.

**2.3 Концептуальный прототип**

Концептуальный прототип состоит из описания внешнего пользовательского интерфейса, а именно, элементов управления.

При создании данного приложения важную роль играют формы, так как они являются основным средством работы с пользователем. Разрабатываемое приложение будет содержать несколько форм.

Все формы будут содержать стандартные пользовательские элементы управления.

В рабочем режиме программы, пользователю, для удобной навигации, будет предоставлено меню.

Первая форма, которую увидит пользователь будет форма Подключения, с помощью которой пользователь сможет выполнить соединение в приложении. Данная форма будет иметь три текстовых поля для ввода удаленного Ip адреса, удалённого и внешнего портов, а также кнопка «Соединить». Кнопка «Соединить» будет открывать главное окно и соединение с другим пользователем. Макет формы представлен на рисунке 2.2.

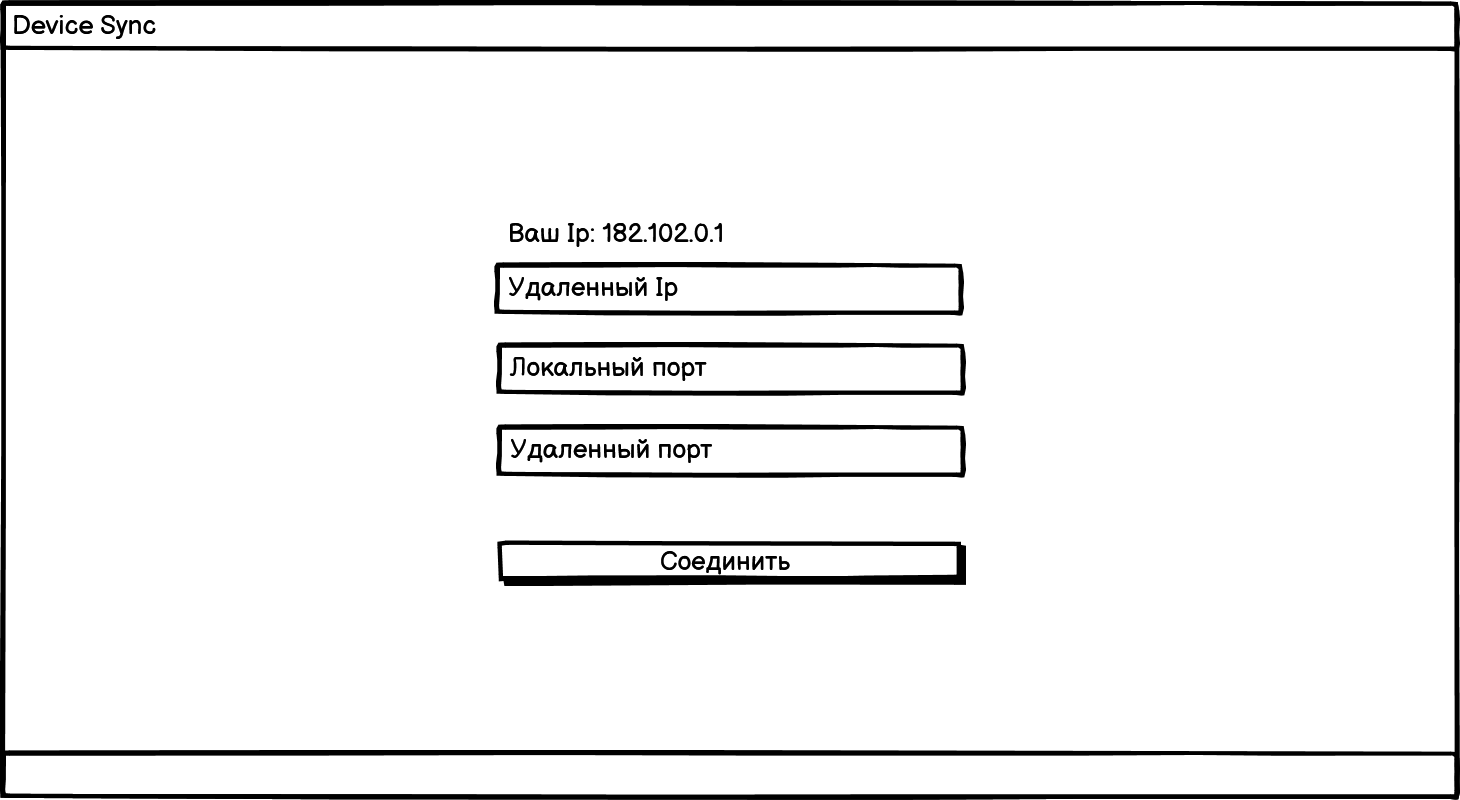


Рисунок 2.2

После успешного входа, будет открыта главная форма. Данная форма будет иметь главное меню с двумя кнопками для отправки сообщения и файла, а также поле ввода текста сообщения и список результата получения/отправки данных. Макет формы главного окна предоставлен на рисунке 2.3.

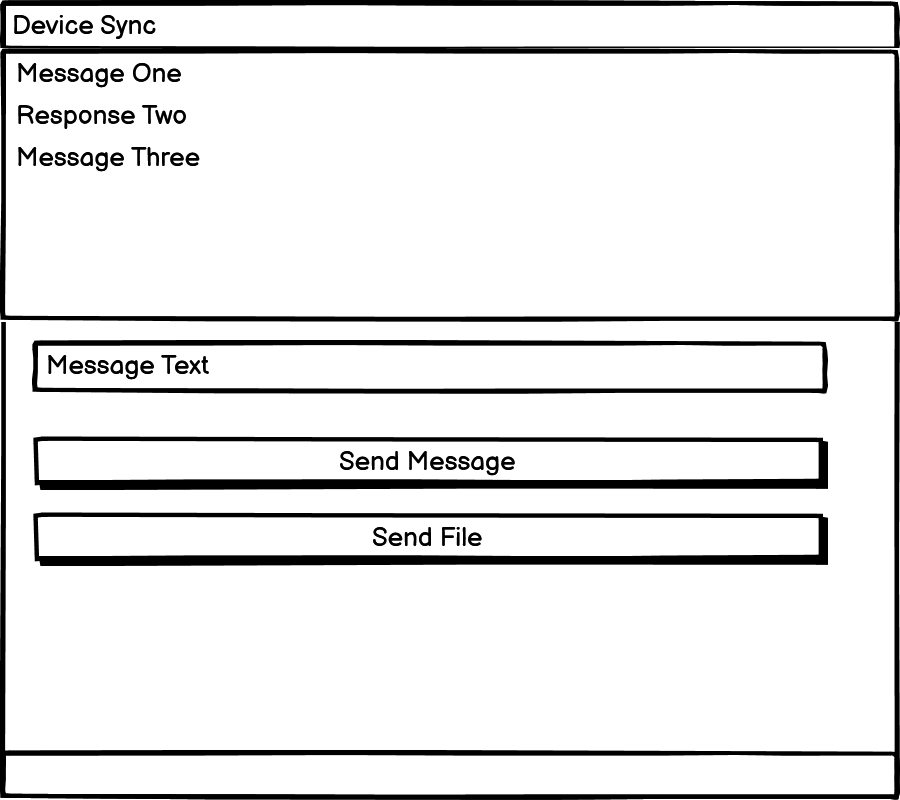


Рисунок 2.3

**3 Описание реализации программного средства**

**3.1 Инструменты разработки и применяемые технологии**

Инструментами для разработки данного программного средства будут являться:

* операционная система Windows 10;
* среда программирования Visual Studio;
* язык программирования C#.
* система контроля версий GitHub;
* средство для разработки диаграмм Diagrams net.

Для разработки программного средства выбрана операционная система Windows 10.

Интегрированная среда разработки Visual Studio – это стартовая площадка для написания, отладки и сборки кода, а также последующей публикации приложений. Интегрированная среда разработки (IDE) представляет собой многофункциональную программу, которую можно использовать для различных аспектов разработки программного обеспечения. Помимо стандартного редактора и отладчика, которые существуют в большинстве сред IDE, Visual Studio включает в себя компиляторы, средства автозавершения кода, графические конструкторы и многие другие функции для упрощения процесса разработки [7].

Язык программирования C# – это объектно- и **компонентно-ориентированный** язык программирования. Он предоставляет языковые конструкции для непосредственной поддержки такой концепции работы. Благодаря этому C# подходит для создания и применения программных компонентов. С момента создания язык C# обогатился функциями для поддержки новых рабочих нагрузок и современными рекомендациями по разработке ПО [2].

GitHub  – крупнейший [веб-сервис](https://ru.wikipedia.org/wiki/%D0%92%D0%B5%D0%B1-%D1%81%D0%B5%D1%80%D0%B2%D0%B8%D1%81) для [хостинга](https://ru.wikipedia.org/wiki/%D0%A5%D0%BE%D1%81%D1%82%D0%B8%D0%BD%D0%B3) [IT-проектов](https://ru.wikipedia.org/wiki/IT) и их совместной разработки. Веб-сервис основан на системе контроля версий [Git](https://ru.wikipedia.org/wiki/Git) и разработан на [Ruby on Rails](https://ru.wikipedia.org/wiki/Ruby_on_Rails) и [Erlang](https://ru.wikipedia.org/wiki/Erlang) компанией GitHub, Inc (ранее Logical Awesome). Сервис бесплатен для проектов с [открытым исходным кодом](https://ru.wikipedia.org/wiki/%D0%9E%D1%82%D0%BA%D1%80%D1%8B%D1%82%D0%BE%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%BD%D0%BE%D0%B5_%D0%BE%D0%B1%D0%B5%D1%81%D0%BF%D0%B5%D1%87%D0%B5%D0%BD%D0%B8%D0%B5) и (с 2019 года) небольших частных проектов, предоставляя им все возможности (включая [SSL](https://ru.wikipedia.org/wiki/SSL)), а для крупных корпоративных проектов предлагаются различные платные тарифные планы [11].

Diagrams.net – это условно бесплатная утилита, разработанная компанией Metaproducts для работы под управлением операционной системы Microsoft Windows, которая позволяет легко контролировать текущую деятельность Интернета на компьютере [6].

**3.2 Функции: логическая и физическая организация**

В программном средстве реализованы функции открытия двунаправленного соединения между двумя приложениями.

Рассмотрим данные функции программы.

Для того, чтобы открыть соединение в приложении, пользователю необходимо ввести имя Ip адрес внутренний и внешний порты. Также пользователю будет выведена информация о его текущем Ip адресе для открытия соединения на другом устройстве. После нажатия кнопки «Соединить» клиент отправляет запрос на подключенное устройство, на котором уже производится дальнейшая обработка запроса.

Код функции открытия соединения предоставлен ниже.

public partial class MainPage : ContentPage

{

private const string IpAddressKey = "IpAddress";

private const string LocalPortKey = "LocalPort";

private const string RemotePortKey = "RemotePort";

public MainPage()

{

InitializeComponent();

RemoteAddress.Text = Preferences.Default.Get(IpAddressKey, "192.168.0.1");

LocalPort.Text = Preferences.Default.Get(LocalPortKey, "5001");

RemotePort.Text = Preferences.Default.Get(RemotePortKey, "5002");

Address.Text = "Your IP Address: ...";

Task.Run(() =>

{

using var socket = new Socket(SocketType.Stream, ProtocolType.Tcp);

socket.Connect("example.com", 80);

string IPAdress = ((IPEndPoint)socket.LocalEndPoint).Address.ToString();

Dispatcher.Dispatch(() =>

{

Address.Text = "Your IP Address: " + IPAdress;

});

});

}

private void Message\_Clicked(object sender, EventArgs e)

{

Singleton.Instance.IpAddress = IPAddress.Parse(RemoteAddress.Text);

Singleton.Instance.LocalPort = Convert.ToInt32(LocalPort.Text);

Singleton.Instance.RemotePort = Convert.ToInt32(RemotePort.Text);

Preferences.Default.Set(IpAddressKey, RemoteAddress.Text);

Preferences.Default.Set(LocalPortKey, LocalPort.Text);

Preferences.Default.Set(RemotePortKey, RemotePort.Text);

Application.Current.MainPage = new NavigationPage(new MessagePage());

}

}

Полный текст программы представлен в приложении А.

**3.3 Функциональное тестирование**

Функциональное тестирование – это тестирование функций приложения на соответствие требованиям и проводится для выявления неполадок и недочетов программы на этапе ее сдачи в эксплуатацию.

Проведем тестирование проверки каждого пункта меню с целью проверки всех функций.

Тестирование программы будет производиться последовательно, переходя из одной части программы в другую. Во время теста будут проверяться все действия с программой, навигация пунктам меню, которые может произвести пользователь.

В таблице 3.7 представлены тест-кейсы, подготовленные для проведения функционального тестирования.

Таблица 3.7 – Тест-кейсы

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № | Модуль / Функция | Шаги выполнения | Ожидаемый результат | Фактический результат |
| 1 | Открытия соединения | 1. Перейти на вкладку «Подключение».  2. В поле «Ip адрес» ввести строку «192.168.0.108».  3. В поле «Внутренний порт» ввести строку «5001».  4. В поле «Внешний порт» ввести строку «5002».  5. Нажать кнопку «Подключение». | Открытие передачи данных и соединения для последующей передачи данных. | Открывается соединение по представленным портам и Ip адресу, для дальнейшей передачи данных. Результат представлен на рисунке Б.1. |
| 2 | Отправка сообщения | 1. Открыть соединение.  2. Открыть главное меню.  3. В поле «Сообщение» написать «Сообщение».  4. Нажать кнопку «Отправить сообщение». | Пользователь отправляет сообщение, и оно отображается на втором клиенте | Сообщение отправляется на подключенный клиент где и обрабатывается. Результат представлен на рисунке Б.2. |
| 3 | Отправить файл | 1. Открыть соединение.  2. Открыть главное меню.  3. Нажать кнопку отправить файл.  4.Выбрать необходимый файл. | Отправка файла и его скачивание на подключённом клиенте. | Отправляется часть файла. Подключены клиент обрабатывает пакет и отправляет подтверждение.Отправляется следующая часть файла и т.д. до полной загрузки файла. Результат представлен на рисунке Б.3. |

Продолжение таблицы 3.7

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № | Модуль / Функция | Шаги выполнения | Ожидаемый результат | Фактический результат |
| 4 | Отправка команды | 1. Открыть соединение.  2. Открыть главное меню.  3. В поле «Сообщение» написать «/explorer».  4. Нажать кнопку «Отправить сообщение». | На подключенном клиенте открывается проводник. | Подключенный клиент получает команду и выполняет ее, открывает проводник. Результат представлен на рисунке Б.4. |
| 5 | Отправка несуществующей команды | 1. Открыть соединение.  2. Открыть главное меню.  3. В поле «Сообщение» написать «/bebra».  4. Нажать кнопку «Отправить сообщение». | Команда не выполняется и получение сообщения что команда не распознана. | Подключенный клиент пытается выполнить команду и возвращает сообщение об ошибке выполнения. Результат представлен на рисунке Б.5. |
| 6 | Получение файла | 1. Открыть соединение.  2. Открыть главное меню.  3. Отправить с подключённого клиента файл. | Файл скачан в папку загрузки. | Клиент получает часть файла и отправляет файл, и так до полного получения файла. Результат представлен на рисунке Б.6. |

**4** **Применение**

**4.1 Назначение программного средства**

Программное средство «DeviceSync» разрабатывалось с целью создания приложения, с помощью которого люди смогут передавать данные между пользователями. Данная программа была разработана на ноутбуке Asus VivoBook с операционной системой Windows 10 Pro, со следующей конфигурацией:

* процессор AMD Ryzen 7 4700U;
* интегрированная видеокарта Radeon Vega Mobile Gfx 2.30 GHz;
* ОЗУ 16,00 ГБ.

Программа создана в средстве разработки Microsoft Visual Studio 2022 на языке программирования С#. Она может работать в средах операционных систем семейства Microsoft Windows, начиная с Windows 7.

Для работы данного программного средства необходима предварительная установка и настройка следующих программных продуктов:

* платформа [Newtonsoft.Json](https://github.com/JamesNK/Newtonsoft.Json) 13.0.3.

В состав программных входят:

* программа «DeviceSync.exe».

Объем установленного приложения не превышает 140 МБ, файлов – 60 МБ.

Программное средство требует внешнего Ip адреса и свободной пары для открытия соединения между пользователями.

**4.2 Условия применения**

Для работы с программой необходимо наличие программного обеспечения:

* операционная система, начиная с Windows 7;
* библиотека [Newtonsoft.Json](https://github.com/JamesNK/Newtonsoft.Json) 13.0.3;
* наличие интернета со скоростью не ниже 8-10 Мбит/с.

На случай редактирования проекта программы необходимо наличие программного обеспечения:

**Заключение**

В рамках курсового проекта на тему «Программа для синхронизации передачи данных "Device Sync"» было разработано программное средство «DeviceSync.exe».

Основными функциями приложения являются:

* Открытие UDP соединения между клиентами;
* Отправка текстовых сообщения;
* Отправка файлов;
* Отправка и выполнение команд.

Для достижения целей курсового проекта были решены следующие задачи:

* изучена предметная область;
* разработана физическая и логическая модель данных;
* разработано программное средство;
* описана область применения, созданного программное средство.

Разработка имеет интуитивно понятный графический интерфейс, позволяющий даже с минимальным знанием компьютера использовать данное программного средство.

Программа реализована в полном объеме и в соответствии с заданными требованиями, полностью отлажена и протестирована. Поставленные задачи выполнены.

К достоинствам программного средства можно отнести простоту использования, малые массивы данных.

В процессе разработки данного программного средства были применены и закреплены знания по уже изученному материалу, были отработаны навыки владения методами надежного программирования и эффективности разработки программного обеспечения в Microsoft Visual Studio 2022 с использованием языка программирования C#.

**Список использованных источников**

1. Багласова, Т.Г. Методические указания по оформлению курсовых и дипломных проектов / Т.Г. Багласова, К.О. Якимович. – Минск: КБП, 2023.
2. Михалевич В.Ю. Методические указания к курсовому проектированию по учебному предмету «Конструирование программ и языки программирования» / В.Ю.Михалевич. – Минск: КБП, 2023.
3. Программа и методика испытаний. Требования к содержанию, оформлению и контролю качества: ГОСТ 19.301-2000. – Введ. 01.09.2001. – Минск: Межгос. совет по стандартизации, метрологии и сертификации, 2000. – 14 с
4. Diagrams [Электронный ресурс]. – UML, 2021. – Режим доступа: https://www.diagrams.net/. – Дата доступа: 10.05.2023.
5. Visual Studio [Электронный ресурс]. – Microsoft, 2020. – Режим доступа: https://visualstudio.microsoft.com/vs/. – Дата доступа: 01.09.2020.
6. GitHub официальная документация – github.com. – Режим доступа: https://github.com. – Дата доступа: 29.05.2023.
7. Newtonsoft.Json официальная документация – newtonsoft.com – Режим доступа: <https://www.newtonsoft.com/json/help/html/Introduction.htm>. – Дата доступа: 29.05.2023.
8. Сетевое программирование C# [Электронный ресурс] – metanit.com – Режим доступа: <https://metanit.com/sharp/net/1.1.php>. – Дата доступа: 01.05.2023.

**Приложение А**

(обязательное)

**Текст программных модулей**

namespace DeviceSync;

public partial class MainPage : ContentPage

{

private const string IpAddressKey = "IpAddress";

private const string LocalPortKey = "LocalPort";

private const string RemotePortKey = "RemotePort";

public MainPage()

{

InitializeComponent();

RemoteAddress.Text = Preferences.Default.Get(IpAddressKey, "192.168.0.1");

LocalPort.Text = Preferences.Default.Get(LocalPortKey, "5001");

RemotePort.Text = Preferences.Default.Get(RemotePortKey, "5002");

Address.Text = "Your IP Address: ...";

Task.Run(() =>

{

using var socket = new Socket(SocketType.Stream, ProtocolType.Tcp);

socket.Connect("example.com", 80);

string IPAdress = ((IPEndPoint)socket.LocalEndPoint).Address.ToString();

Dispatcher.Dispatch(() =>

{

Address.Text = "Your IP Address: " + IPAdress;

});

});

}

private void Message\_Clicked(object sender, EventArgs e)

{

Singleton.Instance.IpAddress = IPAddress.Parse(RemoteAddress.Text);

Singleton.Instance.LocalPort = Convert.ToInt32(LocalPort.Text);

Singleton.Instance.RemotePort = Convert.ToInt32(RemotePort.Text);

Preferences.Default.Set(IpAddressKey, RemoteAddress.Text);

Preferences.Default.Set(LocalPortKey, LocalPort.Text);

Preferences.Default.Set(RemotePortKey, RemotePort.Text);

Application.Current.MainPage = new NavigationPage(new MessagePage());

}

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

<ContentPage xmlns="http://schemas.microsoft.com/dotnet/2021/maui"

xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"

x:Class="DeviceSync.MainPage">

<ScrollView>

<VerticalStackLayout

Spacing="25"

Padding="30,0"

VerticalOptions="Center">

<Label x:Name="Address"/>

<Entry x:Name="RemoteAddress" Placeholder="Введите адрес" Text="192.168.0.1"/>

<Entry x:Name="LocalPort" Placeholder="Введите локальный порт" Text="5001"/>

<Entry x:Name="RemotePort" Placeholder="Введите удаленный порт" Text="5001"/>

<Button x:Name="Message" Text="Подключение" Clicked="Message\_Clicked"/>

</VerticalStackLayout>

</ScrollView>

</ContentPage>

using DeviceSync.Messages;

using Newtonsoft.Json;

using Newtonsoft.Json.Linq;

namespace DeviceSync;

public partial class MessagePage : ContentPage

{

private readonly UdpClient \_udpSender;

private readonly UdpClient \_udpReceiver;

private const int bufferSize = 8 \* 1024; // Maximum UDP packet size

public event EventHandler<Exception> ExceptionNotify;

private readonly AutoResetEvent waitHandler = new AutoResetEvent(true); // объект-событие

public MessagePage()

{

InitializeComponent();

ExceptionNotify += (sender, ex) => DisplayAlert("Exception", ex.Message + ex.Source + ex.StackTrace, "OK");

\_udpSender = new UdpClient();

\_udpSender.Connect(Singleton.Instance.IpAddress, Singleton.Instance.RemotePort);

\_udpReceiver = new UdpClient(Singleton.Instance.LocalPort);

ReceiverAsync().ConfigureAwait(false);

}

private async Task ReceiverAsync()

{

try

{

while (true)

{

var result = await \_udpReceiver.ReceiveAsync();

var receivedJson = Encoding.UTF8.GetString(result.Buffer);

JObject receivedObject = JObject.Parse(receivedJson);

PackageType messageType = receivedObject["Type"].ToObject<PackageType>();

switch (messageType)

{

case PackageType.Text:

{

StringMessage textMessage = receivedObject.ToObject<StringMessage>();

AddText($"--> {textMessage.Content}");

break;

}

case PackageType.FileChunk:

{

AddText("Плюс кусок");

waitHandler.Set();

FileChunk fileMessage = receivedObject.ToObject<FileChunk>();

await SaveFileAsync(fileMessage);

break;

}

case PackageType.Acknowledge:

{

AddText("Подтвержден");

AcknowledgePackage acknowledgePackage = receivedObject.ToObject<AcknowledgePackage>();

await SendChunkAsync(acknowledgePackage);

break;

}

}

}

}

catch (Exception ex)

{

Device.BeginInvokeOnMainThread(() => ExceptionNotify?.Invoke(this, ex));

}

}

private async Task SaveFileAsync(FileChunk file)

{

try

{

string fileName = Path.GetFileName(file.FileName);

#if \_\_ANDROID\_\_

var downloadsDir = Android.OS.Environment.GetExternalStoragePublicDirectory(Android.OS.Environment.DirectoryDownloads);

string folderPath = downloadsDir.AbsolutePath;

#else

string folderPath = Path.Combine(Environment.GetFolderPath(Environment.SpecialFolder.UserProfile), "Downloads");

#endif

string filePathToSave = Path.Combine(folderPath, fileName);

using (FileStream fs = new(filePathToSave, FileMode.Append, FileAccess.Write))

{

await fs.WriteAsync(file.Content.AsMemory(0, file.Content.Length));

var package = new AcknowledgePackage(file.FileName, file.CurrentChunk + 1);

await SendAsync(package);

}

await Task.Delay(250);

if (!waitHandler.WaitOne(0))

{

AddText($"Переотправил");

var package = new AcknowledgePackage(file.FileName, file.CurrentChunk + 1);

await SendAsync(package);

}

AddText($"Файл {fileName} сохранен! {filePathToSave}");

}

catch (Exception ex)

{

ExceptionNotify?.Invoke(this, ex);

}

}

private async Task SendAsync(Message package)

{

string jsonFragment = JsonConvert.SerializeObject(package);

byte[] dataToSend = Encoding.UTF8.GetBytes(jsonFragment);

await \_udpSender.SendAsync(dataToSend, dataToSend.Length);

}

private async void SendMessage\_Clicked(object sender, EventArgs e)

{

try

{

if (!string.IsNullOrEmpty(Message.Text))

{

if (Message.Text.StartsWith("/"))

{

string command = Message.Text.TrimStart('/');

StringMessage commandMessage = new StringMessage(PackageType.Command, command);

await SendAsync(commandMessage);

AddText("Console Command sent!");

}

else

{

StringMessage stringMessage = new StringMessage(PackageType.Text, Message.Text);

await SendAsync(stringMessage);

AddText("Message sent!");

}

}

}

catch (Exception ex)

{

ExceptionNotify?.Invoke(this, ex);

}

}

private void AddText(string message) =>

Device.BeginInvokeOnMainThread(() => { Chat.Text += $"{message}\t{DateTime.Now:HH:mm:ss:ffff}\n"; });

private async void SendFile\_Clicked(object sender, EventArgs e)

{

try

{

var result = await FilePicker.PickAsync();

if (result != null)

{

await SendChunkAsync(new AcknowledgePackage(result.FullPath, 0));

AddText("File send successfully!");

}

}

catch (Exception ex)

{

ExceptionNotify?.Invoke(this, ex);

}

}

private async Task SendChunkAsync(AcknowledgePackage package)

{

try

{

string filePath = package.Content;

int currentChunk = package.CurrentChunk; // Current chunk being sent

long fileSize = new FileInfo(filePath).Length; // Size of the file to be sent

int numChunks = (int)Math.Ceiling((double)fileSize / bufferSize); // Number of chunks needed to send the file

using FileStream fs = File.OpenRead(filePath);

if (currentChunk < numChunks)

{

fs.Position = currentChunk \* bufferSize;

currentChunk++;

byte[] buffer = new byte[bufferSize]; // Buffer for the data to be sent

while (fs.Position < fs.Length)

{

int bytesRead = await fs.ReadAsync(buffer.AsMemory(0, bufferSize));

await SendAsync(new FileChunk(PackageType.FileChunk, buffer.Take(bytesRead).ToArray(), filePath, currentChunk, numChunks));

++currentChunk;

}

}

}

catch (Exception ex)

{

Device.BeginInvokeOnMainThread(() => ExceptionNotify?.Invoke(this, ex));

}

}

}

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

<ContentPage xmlns="http://schemas.microsoft.com/dotnet/2021/maui"

xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"

x:Class="DeviceSync.MessagePage"

Title="MessagePage">

<VerticalStackLayout>

<ScrollView MinimumHeightRequest="50" MaximumHeightRequest="150" BackgroundColor="LightGray">

<Border>

<Label x:Name="Chat"/>

</Border>

</ScrollView>

<Entry x:Name="Message" Placeholder="Message" Text=""/>

<Button

x:Name="SendMessageButton"

Text="Send message"

Clicked="SendMessage\_Clicked"/>

<Button

x:Name="SendFileButton"

Text="Send file"

Clicked="SendFile\_Clicked"/>

</VerticalStackLayout>

</ContentPage>

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace DeviceSync

{

internal class Singleton

{

public static Singleton Instance { get; } = new Singleton();

public IPAddress IpAddress { get; set; }

public int RemotePort { get; set; }

public int LocalPort { get; set; }

}

}

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace DeviceSync.Messages

{

internal class AcknowledgePackage : StringMessage

{

public int CurrentChunk { get; set; }

public AcknowledgePackage(string path, int currentChunk) : base(PackageType.Acknowledge, path)

{

CurrentChunk = currentChunk;

}

}

}

namespace DeviceSync.Messages

{

internal class BytesMessage : Message

{

public byte[] Content { get; set; }

public string FileName { get; set; }

public BytesMessage(PackageType type, byte[] content, string fileName) : base(type)

{

Content = content;

FileName = fileName;

}

}

}

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace DeviceSync.Messages

{

internal class FileChunk : BytesMessage

{

public int CurrentChunk { get; set; }

public int NumChunk { get; set; }

public FileChunk(PackageType type, byte[] content, string fileName, int currentChunk, int numChunk)

: base(type, content, fileName)

{

CurrentChunk = currentChunk;

NumChunk = numChunk;

}

}

}

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace DeviceSync.Messages

{

internal abstract class Message

{

public PackageType Type = PackageType.Text;

public DateTime SendTime { get; set; }

public Message(PackageType type)

{

Type = type;

SendTime = DateTime.Now;

}

}

}

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace DeviceSync.Messages

{

internal enum PackageType

{

Text,

FileChunk,

Acknowledge,

KeyEvent,

Command

}

}

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace DeviceSync.Messages

{

internal class StringMessage : Message

{

public string Content { get; set; }

public StringMessage(PackageType type, string content) : base(type)

{

Content = content;

}

}

}

using DeviceSync.Messages;

using Microsoft.Win32;

using Newtonsoft.Json;

using Newtonsoft.Json.Linq;

using System;

using System.IO;

using System.Linq;

using System.Net.Sockets;

using System.Text;

using System.Threading.Tasks;

using System.Windows;

using DeviceSync;

using System.Diagnostics;

namespace App2

{

public partial class MessageWindow : Window

{

private System.Diagnostics.Process \_process;

private readonly UdpClient \_udpSender;

private readonly UdpClient \_udpReceiver;

private const int bufferSize = 8 \* 1024; // Maximum UDP packet size

public event EventHandler<Exception> ExceptionNotify;

private readonly System.Threading.AutoResetEvent waitHandler = new System.Threading.AutoResetEvent(true); // объект-событие

public MessageWindow()

{

InitializeComponent();

ExceptionNotify += (sender, ex) => MessageBox.Show($"Exception: {ex.Message}\n{ex.Source}\n{ex.StackTrace}", "Exception");

\_udpSender = new UdpClient();

\_udpSender.Connect(Singleton.Instance.IpAddress, Singleton.Instance.RemotePort);

\_udpReceiver = new UdpClient(Singleton.Instance.LocalPort);

ReceiverAsync().ConfigureAwait(false);

}

private async Task ReceiverAsync()

{

try

{

while (true)

{

var result = await \_udpReceiver.ReceiveAsync();

var receivedJson = Encoding.UTF8.GetString(result.Buffer);

JObject receivedObject = JObject.Parse(receivedJson);

PackageType messageType = receivedObject["Type"]!.ToObject<PackageType>();

switch (messageType)

{

case PackageType.Text:

{

StringMessage? textMessage = receivedObject.ToObject<StringMessage>();

AddText($"--> {textMessage?.Content}");

break;

}

case PackageType.FileChunk:

{

AddText("Плюс кусок");

waitHandler.Set();

FileChunk? fileMessage = receivedObject.ToObject<FileChunk>();

await SaveFileAsync(fileMessage);

break;

}

case PackageType.Acknowledge:

{

AddText("Подтвержден");

AcknowledgePackage? acknowledgePackage = receivedObject.ToObject<AcknowledgePackage>();

await SendChunkAsync(acknowledgePackage);

break;

}

case PackageType.Command:

{

StringMessage commandMessage = receivedObject.ToObject<StringMessage>();

ExecuteCommand(commandMessage.Content);

break;

}

}

}

}

catch (Exception ex)

{

Application.Current.Dispatcher.Invoke(() => ExceptionNotify?.Invoke(this, ex));

}

}

private async Task ExecuteCommand(string command)

{

try

{

Process process = new Process();

process.StartInfo.FileName = "cmd.exe";

process.StartInfo.UseShellExecute = false;

process.StartInfo.RedirectStandardInput = true;

process.StartInfo.RedirectStandardOutput = true;

process.StartInfo.RedirectStandardError = true;

process.StartInfo.CreateNoWindow = true;

StringBuilder outputBuilder = new StringBuilder();

process.OutputDataReceived += (sender, e) =>

{

if (!string.IsNullOrEmpty(e.Data))

{

outputBuilder.AppendLine(e.Data);

SendCommandOutput(e.Data);

}

};

process.ErrorDataReceived += (sender, e) =>

{

if (!string.IsNullOrEmpty(e.Data))

{

outputBuilder.AppendLine(e.Data);

SendCommandOutput(e.Data);

}

};

process.Start();

process.BeginOutputReadLine();

process.BeginErrorReadLine();

process.StandardInput.WriteLine(command);

process.StandardInput.WriteLine("exit");

process.WaitForExit();

string output = outputBuilder.ToString();

AddText("Command executed!");

AddText(output);

}

catch (Exception ex)

{

ExceptionNotify?.Invoke(this, ex);

}

}

private async Task SendCommandOutput(string output)

{

if (!string.IsNullOrEmpty(output))

{

StringMessage stringMessage = new StringMessage(PackageType.Text, output);

await SendAsync(stringMessage);

}

}

private async Task SaveFileAsync(FileChunk? file)

{

try

{

string? fileName = Path.GetFileName(file?.FileName);

string folderPath = Environment.GetFolderPath(Environment.SpecialFolder.UserProfile);

string downloadsFolderPath = Path.Combine(folderPath, "Downloads");

if (fileName != null)

{

string filePathToSave = Path.Combine(downloadsFolderPath, fileName);

await using (FileStream fs = new(filePathToSave, FileMode.Append, FileAccess.Write))

{

await fs.WriteAsync(file.Content.AsMemory(0, file.Content.Length));

var package = new AcknowledgePackage(file.FileName, file.CurrentChunk + 1);

await SendAsync(package);

}

await Task.Delay(250);

if (!waitHandler.WaitOne(0))

{

AddText($"Переотправил");

var package = new AcknowledgePackage(file.FileName, file.CurrentChunk + 1);

await SendAsync(package);

}

AddText($"Файл {fileName} сохранен! {filePathToSave}");

}

}

catch (Exception ex)

{

ExceptionNotify?.Invoke(this, ex);

}

}

private async Task SendAsync(Message package)

{

string jsonFragment = JsonConvert.SerializeObject(package);

byte[] dataToSend = Encoding.UTF8.GetBytes(jsonFragment);

await \_udpSender.SendAsync(dataToSend, dataToSend.Length);

}

private async void SendMessage\_Clicked(object sender, RoutedEventArgs e)

{

try

{

if (!string.IsNullOrEmpty(Message.Text))

{

if (Message.Text.StartsWith("/"))

{

string command = Message.Text.TrimStart('/');

StringMessage commandMessage = new StringMessage(PackageType.Command, command);

await SendAsync(commandMessage);

AddText("Console Command sent!");

}

else

{

StringMessage stringMessage = new StringMessage(PackageType.Text, Message.Text);

await SendAsync(stringMessage);

AddText("Message sent!");

}

}

}

catch (Exception ex)

{

ExceptionNotify?.Invoke(this, ex);

}

}

private void AddText(string message) =>

Application.Current.Dispatcher.Invoke(() => { Chat.Text += $"{message}\t{DateTime.Now:HH:mm:ss:ffff}\n"; });

private async void SendFile\_Clicked(object sender, RoutedEventArgs e)

{

try

{

OpenFileDialog openFileDialog = new OpenFileDialog();

if (openFileDialog.ShowDialog() == true)

{

string filePath = openFileDialog.FileName;

await SendChunkAsync(new AcknowledgePackage(filePath, 0));

AddText("File send successfully!");

}

}

catch (Exception ex)

{

ExceptionNotify?.Invoke(this, ex);

}

}

private async Task SendChunkAsync(AcknowledgePackage? package)

{

try

{

string filePath = package.Content;

int currentChunk = package.CurrentChunk; // Current chunk being sent

long fileSize = new FileInfo(filePath).Length; // Size of the file to be sent

int numChunks = (int)Math.Ceiling((double)fileSize / bufferSize); // Number of chunks needed to send the file

using FileStream fs = File.OpenRead(filePath);

if (currentChunk < numChunks)

{

fs.Position = currentChunk \* bufferSize;

currentChunk++;

byte[] buffer = new byte[bufferSize]; // Buffer for the data to be sent

while (fs.Position < fs.Length)

{

int bytesRead = await fs.ReadAsync(buffer.AsMemory(0, bufferSize));

await SendAsync(new FileChunk(PackageType.FileChunk, buffer.Take(bytesRead).ToArray(), filePath, currentChunk, numChunks));

++currentChunk;

}

}

}

catch (Exception ex)

{

Application.Current.Dispatcher.Invoke(() => ExceptionNotify?.Invoke(this, ex));

}

}

}

}

}

**Приложение Б**

(обязательное)

**Формы выходных документов**

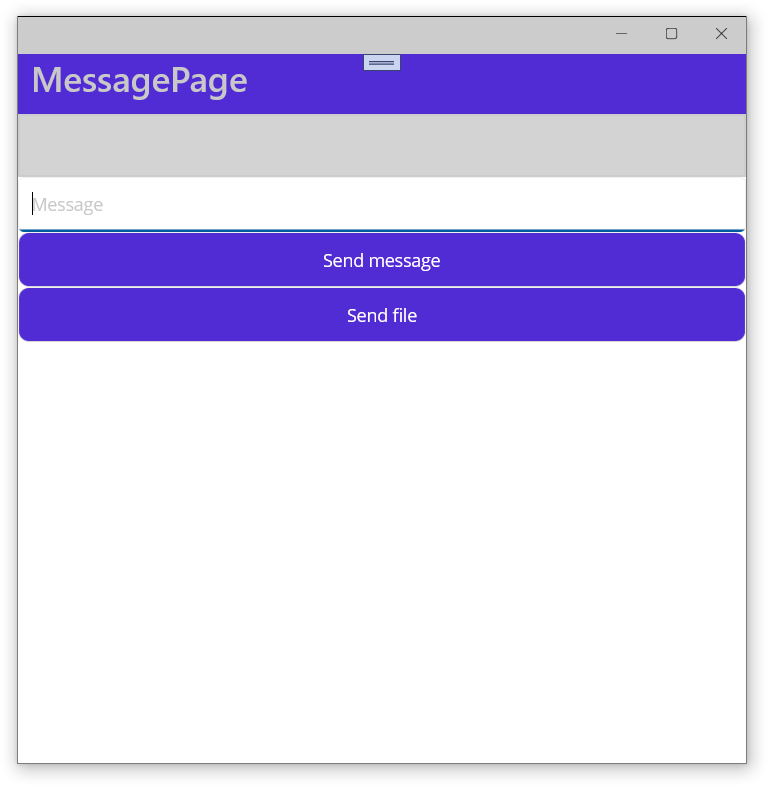


Рисунок Б.1 – Открытия соединения

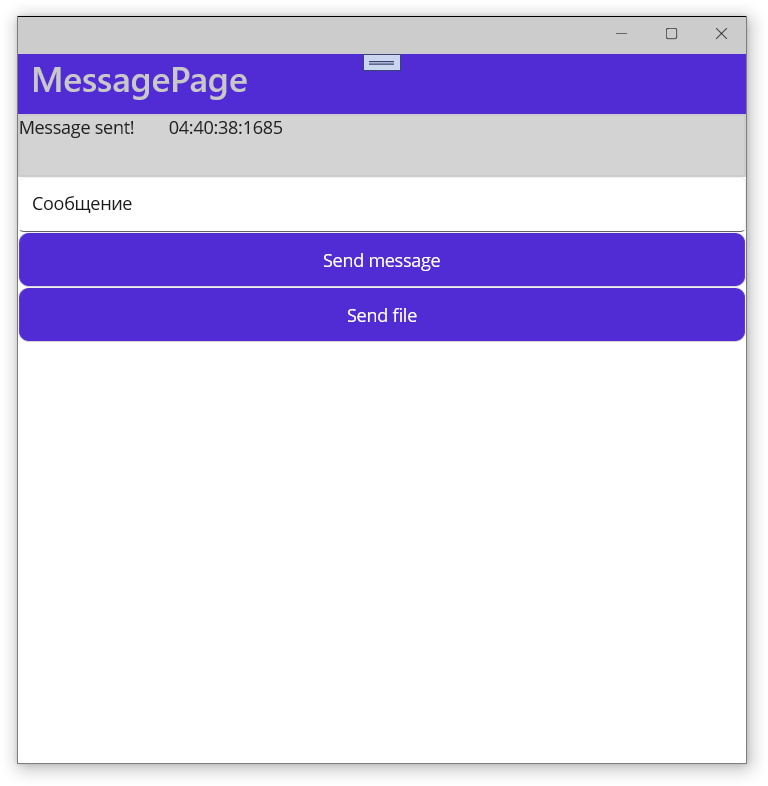


Рисунок Б.2 - Отправка сообщения

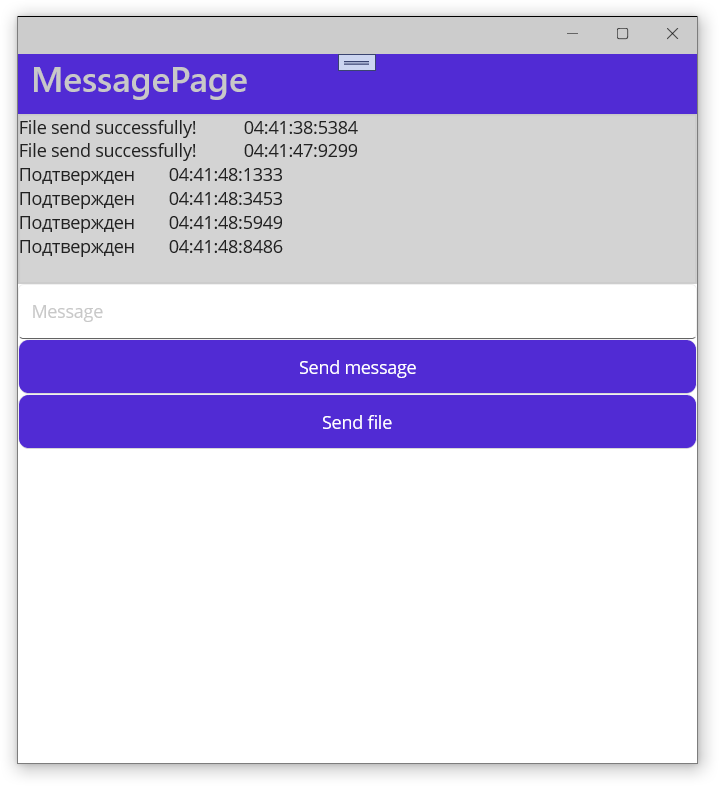


Рисунок Б.3 – Отправить файл

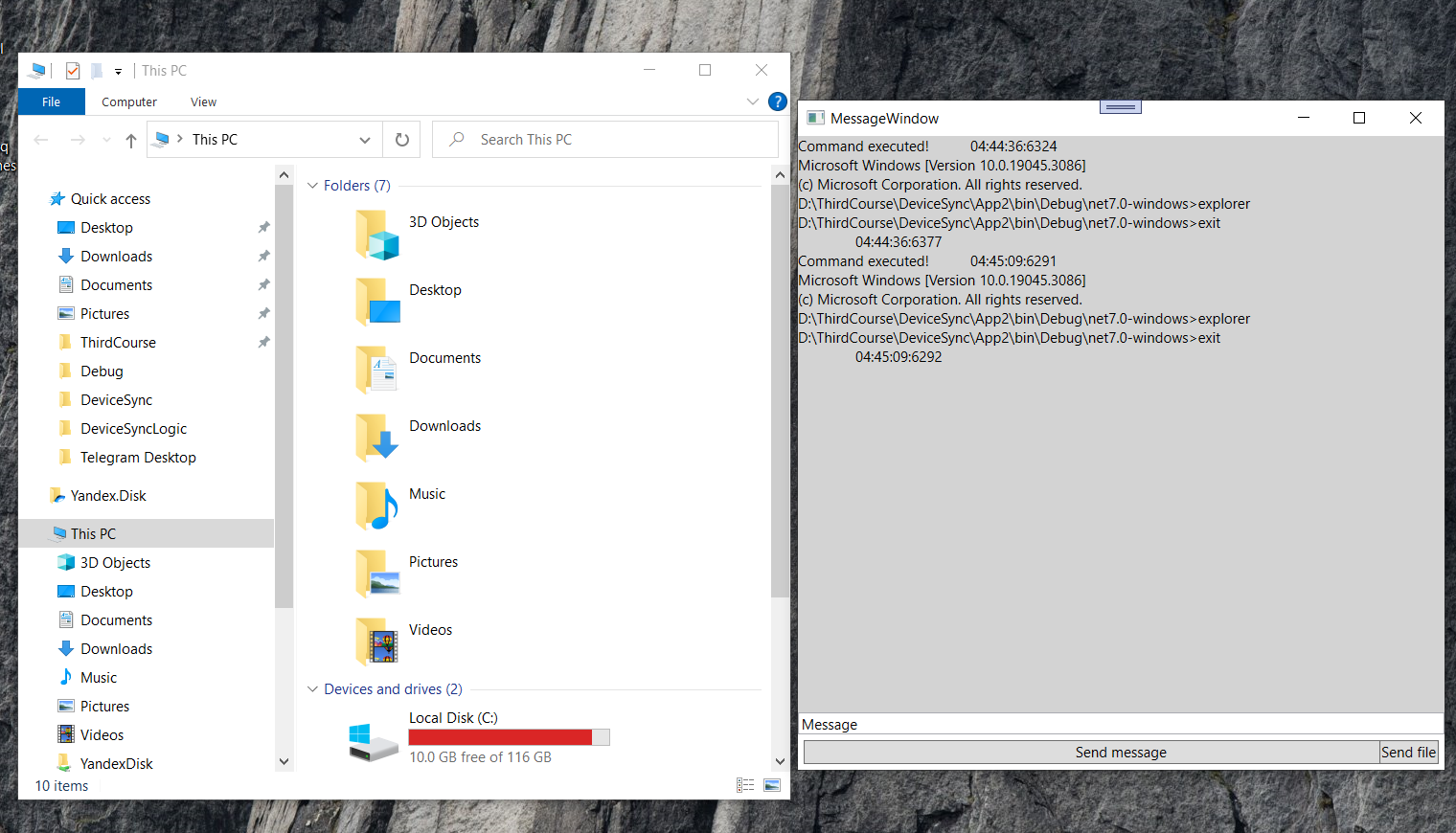


Рисунок Б.4 – Отправка команды

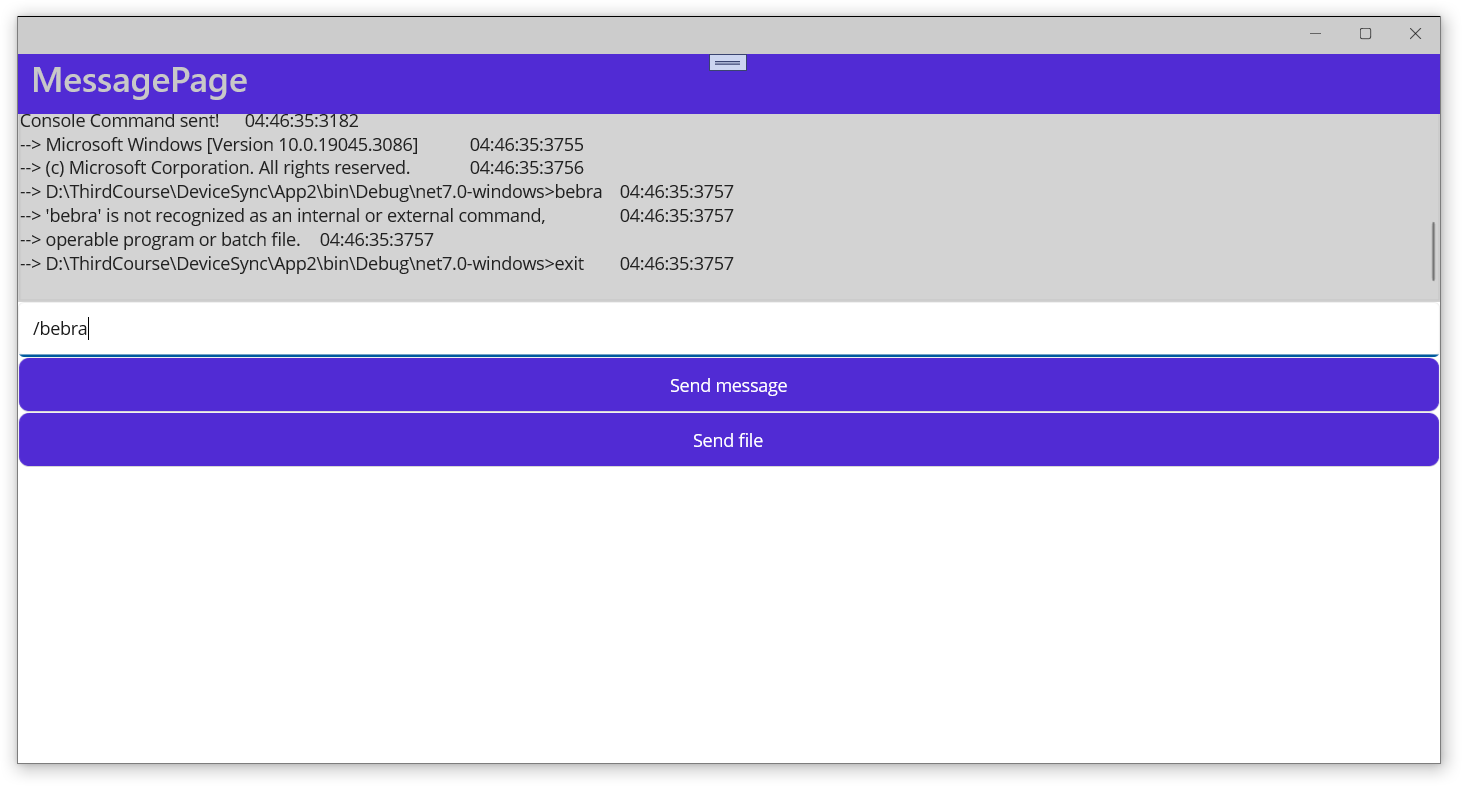


Рисунок Б.5 – Отправка несуществующей команды

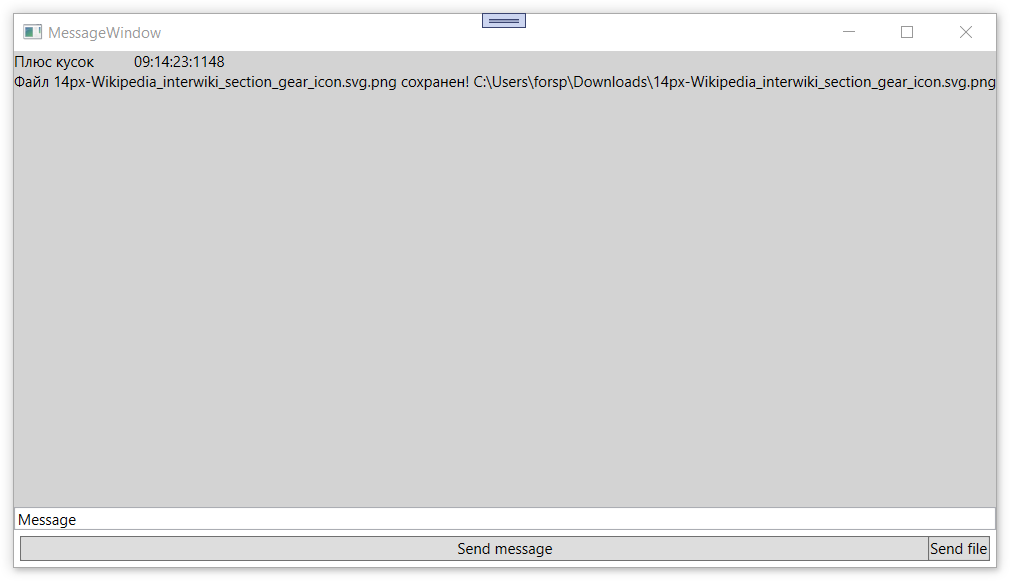


Рисунок Б.6 – Получение файла

КБиП

У

Т. Контр.

Разраб. Разраб.

Лукашонок В.А.

Провер. Провер.

Михалевич В.Ю.

Банцевич. С.В

Банцевич. С.В

Банцевич. С.В

Михалевич В.Ю.

Банцевич. С.В

Банцевич. С.В

Банцевич. С.В

Михалевич В.Ю.

Банцевич. С.В

Банцевич. С.В

Банцевич. С.В

Михалевич В.Ю.

Банцевич. С.В

Банцевич. С.В

Банцевич. С.В

Реценз.

Утверд.

Н. Контр.

Программа для синхронизации передачи данных "Device Sync"

№ докум.

Подпись

Дата

Изм.

Лист

Масса

Лист.

Масштаб

Лист 1

Листов 1

Инв.№подл.

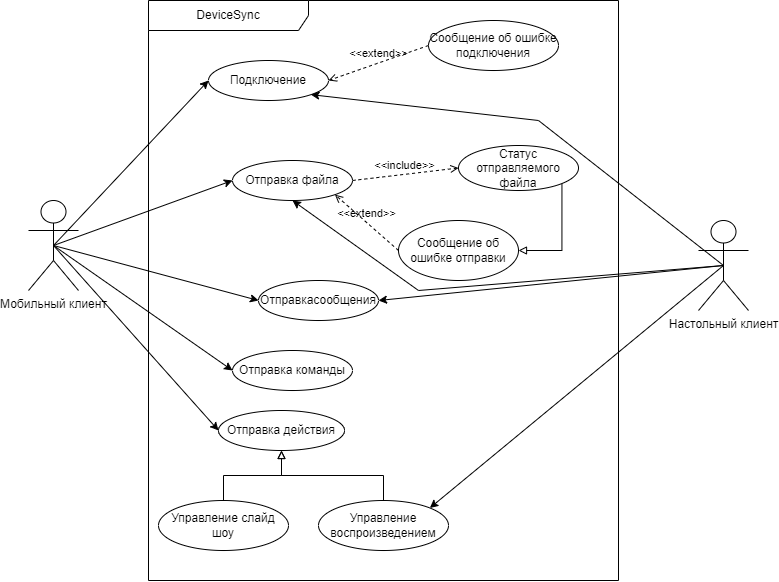
Подп. и дата

Взам.инв.№

Инв.№дубл.

Подп. и дата

КП Т.096020.401 ГЧ Т.ХХХХХХ.401 ГЧ



КП Т.096020.401 ГЧ

ERD-диаграмма вариантов использования



КП Т.096020.401 ГЧ

КБиП

У

Т. Контр.

Разраб. Разраб.

Лукашонок В.А.

Провер. Провер.

Михалевич В.Ю.

Банцевич. С.В

Банцевич. С.В

Банцевич. С.В

Михалевич В.Ю.

Банцевич. С.В

Банцевич. С.В

Банцевич. С.В

Михалевич В.Ю.

Банцевич. С.В

Банцевич. С.В

Банцевич. С.В

Михалевич В.Ю.

Банцевич. С.В

Банцевич. С.В

Банцевич. С.В

Реценз.

Утверд.

Н. Контр.

Программа для синхронизации передачи данных "Device Sync"

№ докум.

Подпись

Дата

Изм.

Лист

Масса

Лист.

Масштаб

Лист 1

Листов 1

Инв.№подл.

Подп. и дата

Взам.инв.№

Инв.№дубл.

Подп. и дата

КП Т.096020.401 ГЧ Т.ХХХХХХ.401 ГЧ

Диаграмма классов

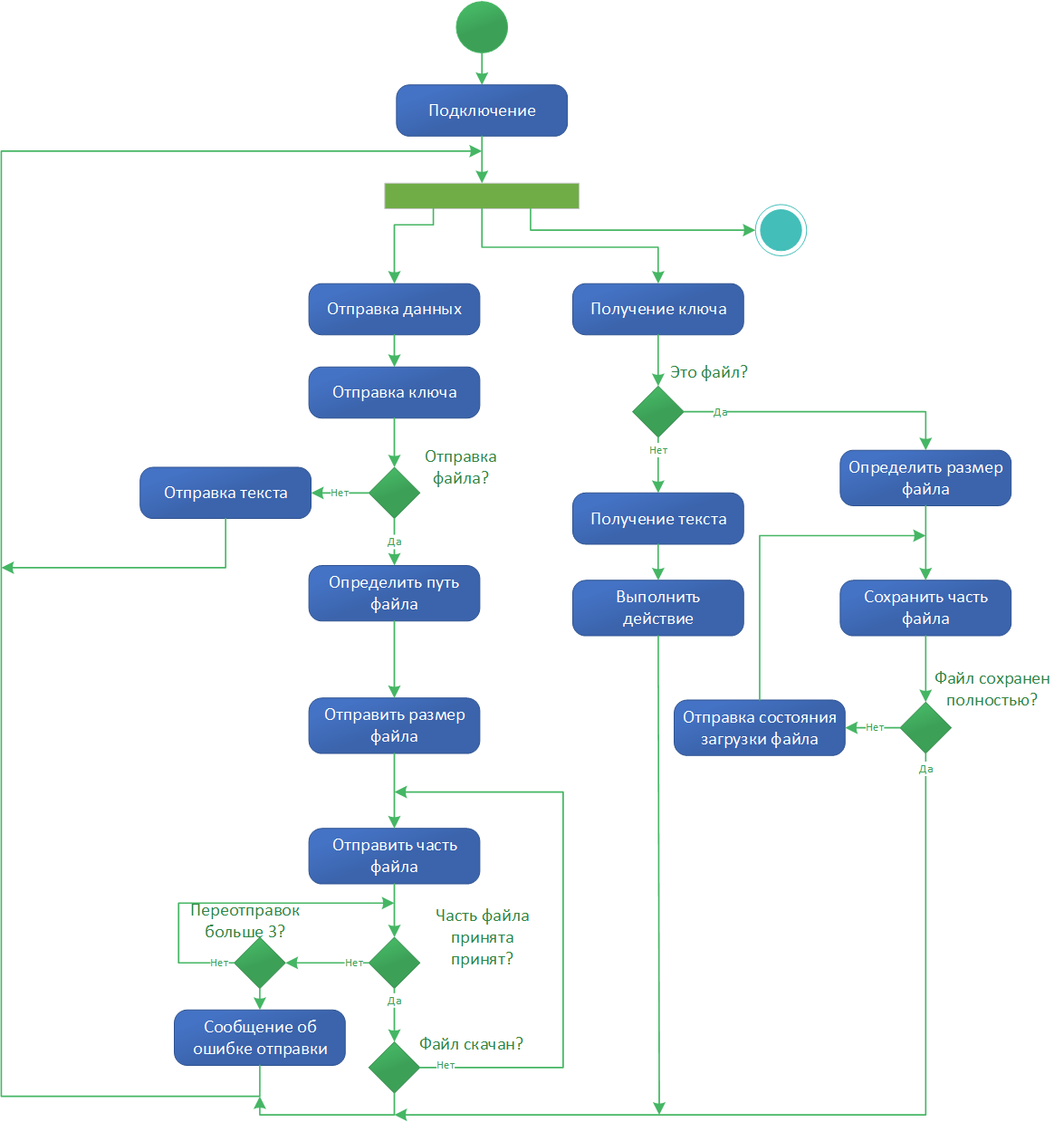


Диаграмма деятельности

КБиП

У

Т. Контр.

Разраб. Разраб.

Лукашонок В.А.

Провер. Провер.

Михалевич В.Ю.

Банцевич. С.В

Банцевич. С.В

Банцевич. С.В

Михалевич В.Ю.

Банцевич. С.В

Банцевич. С.В

Банцевич. С.В

Михалевич В.Ю.

Банцевич. С.В

Банцевич. С.В

Банцевич. С.В

Михалевич В.Ю.

Банцевич. С.В

Банцевич. С.В

Банцевич. С.В

Реценз.

Утверд.

Н. Контр.

Программа для синхронизации передачи данных "Device Sync"

№ докум.

Подпись

Дата

Изм.

Лист

Масса

Лист.

Масштаб

Лист 1

Листов 1

Инв.№подл.

Подп. и дата

Взам.инв.№

Инв.№дубл.

Подп. и дата

КП Т.096020.401 ГЧ Т.ХХХХХХ.401 ГЧ

КП Т.096020.401 ГЧ