ВВЕДЕНИЕ

Первое упоминание о передаче информации на дальние расстояния описано в древнегреческом мифе о Тесее (в случае победы над минотавром Тесей должен был поднять белый парус на своём корабле). Кроме таких визуальных сигналов, как дым, маяки, гелиограф, сигналы семафоров и флажков, использовались и аудиосигналы (бой барабанов, звуковые рожки и свистки). Сторожевые посты стали первыми системами неэлектрической связи.

Необходимость передавать не только сигналы тревоги, но и сообщения различного характера привела к созданию специальных кодов и обозначений.

Цифровой обмен данными начался задолго до создания телефонных сетей, социальных сетей, чатов, мессенджеров и иных средств передачи данных, популярных в наше время.

Одна из первых попыток создать средство связи с использованием электричества относится ко второй половине XVIII века, когда Ж.Л. Лесаж в 1774 году построил в Женеве электростатический телеграф.

Первый электромагнитный телеграф создал российский учёный Павел Львович Шиллинг в 1832 году. Павел Шиллинг также разработал оригинальный код, в котором каждой букве алфавита соответствовала определённая комбинация символов, которая могла проявляться чёрными и белыми кружками на телеграфном аппарате. В США электромагнитный телеграф запатентовал Сэмюэл Морзе в 1840 году. Большой заслугой Морзе является изобретение телеграфного кода, где буквы алфавита были представлены комбинацией коротких и длинных сигналов — «точек» и «тире» (код Морзе).

На текущий момент, методы передачи информации сохранили схожий принцип, ранее – «тире» и «точка», сейчас – «единица» и «ноль».

Но с развитием цифровой передачи данных и внедрением ее в обиход каждого человека, появилась нужда в защите информации. Незащищённые данные, передаваемые из точки «А» в точку «Б», стали подвергаться прослушиванию нежелательными лицами. Что создавало опасность утечки личной или секретной информации. Дабы избежать похищение данных, использовалось большое количество мер и средств, одни из них:

— создание выделенных каналов связи;

— открытая передача зашифрованных данных;

— комбинация двух вышеперечисленных методов.

Существует немалое количество программных средств, созданных для передачи информации в той или иной области. Одними из таких программных средств являются средства передачи зашифрованного трафика. Они предназначены для анонимной передачи данных, защищенной от третьих лиц.

Задачей дипломного проекта является разработка «Системы для защищённого обмена данными».

Главным преимуществом данной программы является то, что на сегодняшний момент имеется мало аналогов децентрализованных систем. Децентрализованность системы – это еще один столбов безопасности программного продукта. Чаще всего утечкой данных является плохая организованность и безответственное отношение к ним. Сервера хранят все информацию обо всех пользователях, что и создает главную опасность в безопасности, взлом одной системы рушит всю сеть.

Пояснительная записка состоит из восьми разделов, содержащих необходимую информацию по организации и эксплуатации программного продукта:

Раздел «Объектно-ориентированный анализ и проектирование системы» включает в себя исследование предметной области, а также определение круга задач, которые должны быть автоматизированы или модифицированы. Кроме этого, в нём даётся обоснование необходимости компьютерной обработки информации. В этом разделе также описываются UML диаграммы, которые были созданы при проектировании программы.

Раздел «Вычислительная система» – раздел, в котором описывается операционная система, с помощью которой будет разрабатываться программный продукт, ее краткое описание, а также все технологии и программные средства, которые будут использоваться, при разработке приложения.

Раздел «Проектирование задачи» содержит описание требований к программному продукту. Так же в разделе приводится логическая и физическая структура данных, состав модулей и принцип организации функций и процедур.

Раздел «Описание программного средства» содержит информацию о входящих в выходящих данных. Также содержит описание установки и функционирования программы.

Раздел «Методика испытаний» содержит перечень требований к информационной и программной совместимости, описание проверки каждого пункта меню, каждой операции, которую выполняет программное средство, а также моделирования всех возможных действий пользователя при работе с программой.

Раздел «Применение» состоит из описания области применения программного средства, возможностей данного средства, его основных характеристик. В данном разделе приводятся требования к необходимым для данного программного средства техническим и программным ресурсам.

Раздел «Охрана труда» – содержит описание вредных факторов при работе с ПЭВМ.

Раздел «Экономический раздел» содержит экономический расчет стоимости программного продукта, дополнительной и основной заработной платы, а также прочих затрат.

Раздел «Заключение», в котором подведены итоги дипломного проектирования. Описаны достоинства, недостатки и особенности «Средства для защищённого обмена данными».

1 ОБЪЕКТНО-ОРИЕНТИРОВАННЫЙ АНАЛИЗ И ПРОЕКТИРОВАНИЕ СИСТЕМЫ

1.1 Сущность задачи

Передача данных (обмен данными) — физический перенос цифрового битового потока в виде сигналов от точки к точке или от точки к нескольким точкам, средствами электросвязи по каналу передачи данных, как правило, для последующей обработки средствами вычислительной техники.

Шифрование — обратимое преобразование информации в целях сокрытия от неавторизованных лиц, с предоставлением, в это же время, авторизованным пользователям доступа к ней. Главным образом, шифрование служит задачей соблюдения конфиденциальности передаваемой информации.

Разработка программного средства для защищенного обмена данными, позволяет передавать зашифрованную последовательность данных между пользователями внутренней сети. Для достижения поставленной цели необходимо решить следующие взаимосвязанные задачи:

1. концепты и принципы передачи данных;
2. основы асинхронного шифрования и цифровой подписи;
3. изучить принципы ООП для грамотной реализации поставленной задачи;
4. основы создания интуитивно понятного интерфейса;
5. использовать основы дисперсионного концепта проектирования программного продукта.

Актуальность темы заключается в том, что в настоящее время различные виды средств передачи зашифрованного потока данных все еще используют сервера как контрольную точку между отношениями пользователей. Разрабатываемый продукт основывается на принципе децентрализации, что подразумевает собой отказ от типичной схемы передачи данных «клиент-сервер», и переходит на парадигму передачи «клиент-клиент», она же «сервер-сервер». Также в продукт будет включен концепт уникальных имен и цифровой подписи, о которых будет описано далее.

Из-за децентрализованной системы, контроль имен и идентификации пользователей между друг другом становится невозможным. Данная проблема решается в создании уникальных имен в виде последовательности 64 символов. Такая длина оптимальна на текущий момент, так как в сумме имеет уникальных имен, что также сильно уменьшает шанс возникновения коллизии.

Цифровая подпись, это уникальная возможность подтвердить однородность данных, а также установить реального получателя, данный метод идентификации позволяет избежать атаки «Maninthemiddle» (человек по центру). Так как подписать сообщения можно только своим не публичным ключом. Данная система может быть скомпрометирована только при условии утечки закрытого ключа. В таком случае необходимо пересоздать цифровую сигнатуру пользователя, ибо все данные получаемые от пользователя с скомпрометированным не публичным ключом более не имеет возможности доказать истинного отправителя.

1.2 Проектирование модели

Проектирование – процесс определения архитектуры, компонентов, интерфейсов и других характеристик системы или её части. Результатом проектирования является проект – целостная совокупность моделей, свойств или характеристик, описанных в форме, пригодной для реализации системы.

Проектирование, наряду с анализом требований, является частью большой стадии жизненного цикла системы, называемой определением системы. Результаты этой стадии являются входной информацией для стадии реализации системы.

Проектирование системы направлено на представление системы, соответствующее предусмотренной цели, принципам и замыслам; оно включает оценку и принятие решений по выбору таких компонентов системы, которые отвечают её архитектуре и укладываются в предписанные ограничения.

Язык проектирования – это средство для автоматизации проектирования систем и упрощения понимания предметной области.

Унификация языков проектирования позволяет обмениваться программными средствами или их компонентами, сокращает затраты на освоение языков и на технологические средства автоматизации их использования, способствует переносимости и повышению качества программных средств.

Один из унифицированных языков моделирования является язык UML, он выполняет задачу графического описания для объектного моделирования в области разработки программного обеспечения, моделирования бизнес-процессов, системного проектирования и отображения организационных структур.

UML является языком широкого профиля, использующий графические обозначения для создания абстрактной модели системы, называемой UML-моделью.

UML позволяет разработчикам программного обеспечения достигнуть соглашения в графических обозначениях для представления общих понятий и больше сконцентрироваться на проектировании и архитектуры.

При разработке специализированного приложения для организации поиска товара были разработаны три основные диаграммы, это диаграмма деятельности, диаграмма вариантов использования и диаграмма последовательности. Каждая из диаграмм описывают свою бизнес-логику.

Диаграмма деятельности – UML-диаграмма, на которой показаны действия, состояния которых описано на диаграмме состояний. Под деятельностью понимается спецификация исполняемого поведения в виде координированного последовательного и параллельного выполнения подчинённых элементов, вложенных видов деятельности и отдельных действий, соединённых между собой потоками, которые идут от выходов одного узла к входам другого, диаграмма изображена на рисунке В.1.

Диаграммы деятельности используются при моделировании бизнес-процессов, технологических процессов, последовательных и параллельных вычислений.

Диаграмма вариантов использования, отражает отношения между актёрами и прецедентами и являющаяся составной частью модели, диаграмма изображена на рисунке В.2.

Между элементами диаграммы вариантов использования могут существовать различные отношения, которые описывают взаимодействие экземпляров одних актеров и вариантов использования с экземплярами других актеров и вариантов. Один актер может взаимодействовать с несколькими вариантами использования. В этом случае этот актер обращается к нескольким сервисам данной системы. В свою очередь один вариант использования может взаимодействовать с несколькими актерами, предоставляя для всех них свой сервис.

В то же время два варианта использования, определенные в рамках одной моделируемой системы, также могут взаимодействовать друг с другом, однако характер этого взаимодействия будет отличаться от взаимодействия с актерами. Однако в обоих случаях способы взаимодействия элементов модели предполагают обмен сигналами или сообщениями, которые инициируют реализацию функционального поведения моделируемой системы.

Диаграмма последовательностей отображает взаимодействие объектов в динамике, диаграмма изображена на рисунке В.3.

В UML взаимодействие объектов понимается как обмен информацией между ними. При этом информация принимает вид сообщений. Кроме того, что сообщение несет какую-то информацию, оно, некоторым образом, также влияет на получателя. UML полностью соответствует основным принципам ООП, в соответствии с которыми информационное взаимодействие между объектами сводится к отправке и приему сообщений.

2 ВЫЧИСЛИТЕЛЬНАЯ СИСТЕМА

Для реализации задачи по разработке системы автоматизации была выбрана база данных «sqlite3», среда разработки «VisualStudio 2019» и высокоуровневый язык программирования «С#».

Выбранная среда разработки оснащена всем необходимым функционалом, который необходим для разработки автоматизированной системы. С помощью данной среды разработки, легко осуществляется проектирование различных форм, а также навигация всего приложения, это происходит с помощью предоставляемых компонентов среды разработки.

«Microsoft VisualStudio 2019» – это набор инструментов для создания программного обеспечения: от планирования до разработки пользовательского интерфейса, написания кода, тестирования, отладки, анализа качества кода и производительности, развертывания в средах клиентов и сбора данных телеметрии по использованию. Эти инструменты предназначены для максимально эффективной совместной работы; все они доступны в интегрированной среде разработки (IDE) «VisualStudio».

«VisualStudio» можно использовать для создания различных типов приложений, от простых приложений для магазина и мобильных клиентов до больших и сложных систем, обслуживающих предприятия и центры обработки данных. Можно создать:

1. приложения и игры, которые выполняются не только на платформе «Windows», но и на «Android» и «iOS»;
2. веб-сайты и веб-службы на основе «ASP.NET», «JQuery», «AngularJS» и других популярных платформ;
3. приложения для самых разных платформ и устройств, включая, но не ограничиваясь: «Office», «Sharepoint», «Hololens», «Kinect» и «Интернета вещей»;
4. игры и графические приложения для разных устройств «Windows», включая «Xbox», с поддержкой «DirectX».

По умолчанию «VisualStudio» обеспечивает поддержку «C#», «C» и «C++», «JavaScript», «F#» и «VisualBasic». «VisualStudio» хорошо работает и интегрируется со сторонними приложениями, например, «Unity» и «Apache Cordova», с помощью расширений средств «VisualStudio» для «Unity» инструментов «VisualStudio» для «Apache Cordova»с соответственно. Вы также можете самостоятельно расширить «VisualStudio», создав собственные инструменты для выполнения специализированных задач.

База данных разработанного приложения находится под «SQLITE3» – система управления реляционными базами данных (РСУБД), разработанная «Ричардом Хиппом». База данных является достоянием общественности. Основной используемый язык запросов– «SQL», создан совместно «Microsoft» и «Sybase». «Transact-SQL» является реализацией стандарта «ANSI/ISO» по структурированному языку запросов «SQL» с расширениями. Используется для работы с базами данных размером от персональных до крупных баз данных масштаба предприятия; конкурирует с другими «СУБД» в этом сегменте рынка.

Windows 10 — операционная система для персональных компьютеров и рабочих станций, разработанная корпорацией Microsoft в рамках семейства Windows NT. После Windows 8.1 система получила номер 10, минуя 9. Серверный аналог Windows 10 — Windows Server 2016.

Система призвана стать единой для разных устройств, таких как персональные компьютеры, планшеты, смартфоны и пр. Доступна единая платформа разработки и единый магазин универсальных приложений, совместимых со всеми поддерживаемыми устройствами. Windows 10 поставляется в качестве услуги с выпуском обновлений на протяжении всего цикла поддержки.

Согласно статистическим данным сайта W3Schools, Windows 10 занимает первое место в мире среди операционных систем, используемых для доступа к сети Интернет, опередив в апреле 2017 года предыдущего лидера — Windows 7.

3 ПРОЕКТИРОВАНИЕ ЗАДАЧИ

3.1 Требования к приложению

В системе передачи данных должен быть реализован понятный интерфейс, содержащий графические компоненты для взаимодействия с элементами создания подключения к другим экземплярам программы. Также должен быть реализован интерес для взаимодействия с элементами для упрощения работы.

В системе для защищённого обмена данными должна быть реализована технология цифровой подписи посредством асинхронного метода шифрования «RSA».

Поскольку программный продукт не должен иметь централизованную систему, должно быть организованно создание «Глобальных идентификаторов» пользователей, для возможности удостоверится в отправителе и получателе.

Объект или область объектов, предназначенная для хранения, создания, редактирования структур-контактов, созданных внутри программного средства.

Возможность иметь, как и прямое, так и множественное подключение к другим программным экземплярам в сети, также иметь возможность ветвления создаваемой внутренней подсети.

В программе должны быть реализованы основные функции, такие как:

1. добавление, удаление и сортировка контактов;
2. логгирования пересылаемых и получаемых данных в базе данных;
3. создание прямого подключения к другим пользователям;
4. шифрование пересылаемых данных асинхронным методом;
5. реализация методов цифровой подписи;
6. передача защищённых файлов;
7. создание группового подключения.

**3.2 Концептуальный прототип**

В «системе для защищённого обмена данными» взаимодействие с проектом будет происходить посредством элементов, представляющих из себе иллюстрацию – внешне, а функционально – кнопка. Пользователю будут доступны наборы управления, такие как:

Меню переключения слайдеров, включающее в себя:

1. кнопку перехода на главный слайдер;
2. кнопку перехода на слайдер с автоматической отправкой;
3. кнопку перехода на слайдер с настройками.

Меню взаимодействия с контактной книжкой:

1. панель взаимодействия включает в себя:
   * создание нового контакта;
   * создание новой папки хранения;
   * создание группы или сети.
2. контекстное меню включает в себя:
   * переименование записи;
   * удаление записи;
   * удаление содержимого записи;
   * создание папки (в случае клика на запись типа «папка»);
   * переход в диалог (в случае клика на запись типа«пользователь»).

Таким образом, была описана структура меню и доступные функций.

3.3 Организация данных

Сегодня практически любая задача требует работы с большим количеством данных — это и получение, и обработка, и хранение информации. Существует множество различных программных продуктов для работы с базами данных, но всех их объединяют одни и те же общие принципы.

При проектировании структуры базы данных используется схема «Сущность-связь». Эта схема выявляет основные объекты предметной области, которые значимы для решаемой задачи, а также показывает отношения между этими объектами. Любой фрагмент предметной области может быть представлен как множество сущностей, между которыми существует некоторое множество связей.

Основополагающими компонентами диаграммы сущность – связь являются следующие компоненты:

1. сущность — это класс однотипных объектов, информация о которых должна быть учтена в модели. Каждая сущность должна иметь наименование, выраженное существительным в единственном числе. Примерами сущностей могут быть такие классы объектов как «Пользователь», «Ключи», «Группа». Каждая сущность в модели изображается в виде прямоугольника с наименованием;
2. атрибут — это именованная характеристика, являющаяся некоторым свойством сущности. Наименование атрибута должно быть выражено существительным в единственном числе (возможно, с характеризующими прилагательными). Примерами атрибутов сущности «Пользователь» могут быть такие атрибуты как «Идентификационное имя», «Публичный ключ», «Отправленные данные», «Время отправки», «Адрес» и т.п. Атрибуты изображаются в пределах прямоугольника, определяющего сущность;
3. ключ — это не избыточный набор атрибутов, значения которых в совокупности являются уникальными для каждого экземпляра сущности. Не избыточность заключается в том, что удаление любого атрибута из ключа нарушается его уникальность. Сущность может иметь несколько различных ключей;
4. связь — это некоторая ассоциация между двумя сущностями. Одна сущность может быть связана с другой сущностью или сама с собою.

Схема «Сущность-связь» показана на рисунке В.4.

Структура таблиц базы данных, созданной в SQLite, содержит слудеющие таблицы: «Keys», «autosend\_mesg», «names», «tree\_views», «user\_info», «group», «user».

Структура таблиц базы данных показана в следующих таблицах.

Таблица 1 – Структура таблицы «Keys»

|  |  |  |  |
| --- | --- | --- | --- |
| Имя поля | Тип данных | Ключевое | Описание |
| id | integer | + | Идентификатор |
| global\_name | VARCHAR(65) |  | Глобальное имя |
| open\_key | TEXT |  | Публичный ключ |

Таблица 2 – Структура таблицы «autosend\_mesg»

|  |  |  |  |
| --- | --- | --- | --- |
| Имя поля | Тип данных | Ключевое | Описание |
| id | integer | + | Идентификатор |
| address | VARCHAR(30) |  | IP + PORT адрес |
| name | VARCHAR(30) |  | Имя получателя |
| message | TEXT |  | Сообщение |
| global\_name | VARCHAR(65) |  | Глобальное имя |

Таблица 3 – Структура таблицы «names»

|  |  |  |  |
| --- | --- | --- | --- |
| Имя поля | Тип данных | Ключевое | Описание |
| id | integer | + | Идентификатор |
| global\_name | VARCHAR(65) |  | Глобальное имя |
| name | VARCHAR(30) |  | Имя пользователя |

Таблица 4 – Структура таблицы «tree\_views»

|  |  |  |  |
| --- | --- | --- | --- |
| Имя поля | Тип данных | Ключевое | Описание |
| id | integer | + | Идентификатор |
| name | VARCHAR(30) |  | Навание книги |
| tree | BLOB |  | Компонет дерево |

Таблица 5 – Структура таблицы «user\_info»

|  |  |  |  |
| --- | --- | --- | --- |
| Имя поля | Тип данных | Ключевое | Описание |
| name | VARCHAR(30) |  | Глобальное имя |
|  |  |  |  |
| Продолжение таблицы 5 – Структура таблицы «user\_info» | | | |
| open\_key | TEXT |  | Наш открытый ключ |
| close\_key | TEXT |  | Наш закрытый ключ |

Таблица 6 – Структура таблицы «group»

|  |  |  |  |
| --- | --- | --- | --- |
| Имя поля | Тип данных | Ключевое | Описание |
| id | integer | + | Идентификатор |
| sender | VARCHAR(65) |  | Адрес |
| date | VARCHAR(30) |  | Дата |
| data | TEXT |  | Сообщение |

Таблица 7 – Структура таблицы «user»

|  |  |  |  |
| --- | --- | --- | --- |
| Имя поля | Тип данных | Ключевое | Описание |
| id | integer | + | Идентификатор |
| sender | VARCHAR(65) |  | Адрес |
| date | VARCHAR(30) |  | Дата |
| data | TEXT |  | Сообщение |
| signed | VARCHAR(65) |  | Цифровая подпись |

Схема базы данных, данного приложения, состоит из следующих сущностей:

1. сущность «Keys» - содержит публичные ключи шифрования пользователей;
2. сущность «autosend\_mesg» - cодержит в себе очередь сообщений, поставленных на автоматическую отправку;
3. сущность «names» - содержит имена других пользователей;
4. сущность «tree\_views» - хранит XML данные компонента tree\_view для воссоздания древовидной структуры контактов;
5. сущность «user\_info» - содержит данные о текущем пользователе, его глобальное имя, публичный и закрытый ключ шифрования;
6. сущность «%group%» - представляет собой динамически созданную таблицу, хранящую данные о полученных сообщениях в группе;
7. сущность «%global\_name%» - представляет собой динамически созданную таблицу, хранящую данные о полученных сообщения от пользователя.

Таким образов была описана организация данных в программе.

3.4 Функции: логическая и физическая организация

Проект — это специальная структура управления, которая осуществляет взаимодействие между программным кодом и визуальными объектами.

Модуль в программировании представляет собой функционально законченный фрагмент программы, оформленный в виде отдельного файла с исходным кодом непрерывной его части, предназначенный для использования в программах. Модули позволяют разбивать сложные задачи на более мелкие. «Система для защищённого обмена данными» состоит из следующих модулей:

* «Main» – данный модуль отвечает за меню программы;
* «main\_extension» – данный модуль отвечает за дополнительные методы в «Main»;
* «auto\_send» – данный модуль отвечает за работу функций автоматической отправки сообщений;
* «DataSending» – данный модуль отвечает за отправку данных.;
* «rsa» – данный модуль отвечает за шифрование и расшифровку сообщений;
* «SignedMessageForm» - данный модуль отвечает за проверку цифровой подписи;
* «Sqlite» - данный модель отвечает за управление базой данных;
* «TreeViewSerializer» - данный модуль отвечает за сериализацию компонента treeview;
* «Weapons» - данный модуль отвечает за дополнительные методы работы с сетью.

Таким образом, были описаны все модули разрабатываемого приложения, а также их предназначение.

**3.5 Функции и элементы управления**

В «Средстве для защищенного обмена данными» использовано большое количество элементов управления и функций:

1. private void Form1\_Load(object sender, EventArgs e) – метод ответственный за создание формы и инитиализации объектов;
2. private void Main\_Resize(object sender, EventArgs e) – метод ответственный за динамическое изменение размеров формы;
3. private void create\_table(string name) – метод ответственный за динамическое создание и добавление таблиц в базу данных;
4. private void connect\_result(bool con, bool first) – метод возвращающий результат подключение к другому экземпляру средства;
5. private void Display(string message) – метод, отвечающий за определение и реализацию получаемых сообщений;
6. private void textBox1\_KeyDown(object sender, KeyEventArgs e) – метод ответственный за отправку сообщений;
7. private void Node\_add\_Button\_Click(object sender, EventArgs e) – метод, ответственный за создание новых подключений;
8. private void treeView1\_NodeMouseDoubleClick(object sender, TreeNodeMouseClickEventArgs e) – метод реализующий подключение;
9. private void show\_dialog(object sender, EventArgs e) – метод, отвечающий за визуализацию сообщений;
10. internal void create\_new\_data\_base() – метод, динамически создающий базу данных.
11. public static byte[] RSADecrypt(byte[] DataToDecrypt, byte[] RSAKeyInfo, bool DoOAEPPadding) – метод, отвечающий за шифрование данных;
12. public static byte[] RSAEncrypt(byte[] DataToEncrypt, byte[] RSAKeyInfo, bool DoOAEPPadding) – метод, отвечающий за расшифровку данных;
13. internal void StartListening() – метод, отвечающий за получение сообщений.

Таким образом, в данном подразделе были описаны все основные функции и процедуры, которые присутствуют в данном разрабатываемом программном средстве.

4 ОПИСАНИЕ ПРОГРАММНОГО СРЕДСТВА

4.1 Общие сведения

Данное разрабатываемое программное средство было разработано на операционной системе «Windows10 Home» в среде программирования «MicrosoftVisualStudio 2019» на языке «C#».

В системе для защищённого обмена данными, исполняемым файлом является файл «Main.exe». Вспомогательными файлами являются таблицы базы данных, которые располагаются в папке c программой. Размер базы данных «DataBase.db3» составляет 35 «килобайт».

Для того чтобы скопировать данную программу себе на компьютер, пользователю необходимо расположить исполняемые и вспомогательные файлы и папки в одном месте и запустить исполняемый файл для создание уникальных значений пользователя. После проведения данной процедуры, можно полноценно использовать «Систему для защищённого обмена данными».

В данном подразделе была описана последовательность действий для установки описываемой программы.

4.2 Функциональное назначение

«Система для защищённого обмена данными» предназначена для создания децентрализованной сети и обмена зашифрованными данными между пользователями.

После запуска программы пользователь может создать новое подключение, что позволит обмениваться сообщениями и файлами с другим пользователем. Также пользователь может создать группу, в которой может обмениваться данными сразу с группой пользователей.

В окне «автоматической отправки сообщений», пользователь может добавить свое сообщение в очередь на отправку, данные манипуляции позволят отправить сообщение пользователю, который в данный момент не в сети, но как только таковой будет доступен, программа автоматически отправит ему сообщение.

4.3 Входные данные

Входные данные – это информация, которая вводится в приложение, посредством ввода данных в поля для текста, в выборе файлов и данных, над которыми будут производится манипуляции.

При помощи клавиатуры или других подобных устройств происходит ввод сообщений или выбор функций.

В разрабатываемом программном средстве, входными данными являются:

– данные для создания прямого подключения в сети (Глобальный идентификатор, адрес, открытый ключ);

– отправляемые и получаемые сообщения и файлы.

4.4 Выходные данные

Выходные данные **—** это информация, которую предоставляет программа. К выходным данным относиться информация, которая располагается в главной части окна. Так же, выходные данные — данные, которые находятся в файлах проектов.

К выходным данным относятся сообщения, которые будут нести информационный и предупреждающий характер. Информация, выводимая из базы данных. Также информация, выводимая на экран в результате отправки и получения сообщений или файлов.

Таким образом, были проанализированы виды выходных данных и способы отображения выходных данных.

**5 МЕТОДИКА ИСПЫТАНИЙ**

**5.1 Технические требования**

Для запуска и правильной работы «Системы для защищённого обмена данными» необходима операционная система «Microsoft Windows 7» или выше, а также «Microsoft DotNetFramework 4.8» или выше.

Не менее 500 свободных мегабайт оперативной памяти и не менее 1000 свободных мегабайт на жестком диске.

Программное средство не требовательно к частоте и количеству ядер процессора, вычисления сведены к разумному минимуму.

Необходимы является роутер или иной другой инструмент для объединения компьютеров в одной сети. Комфортная скорость соединения между компьютерами является скорость от 1Mb в секунду.

5.2 Порядок проведения испытаний

5.2.1 Функциональное тестирование

Функциональное тестирование — это тестирование ПО в целях проверки реализуемости функциональных требований, то есть способности ПО в определённых условиях решать задачи, нужные пользователям. То есть проведение функционального тестирования позволяет проверить способность информационной системы в определенных условиях решать задачи, необходимые пользователям.

После запуска программного средства, начали тестирование с основной функции – оправка сообщений. Для начала нажали кнопку создания нового пользователя, как показано на рисунке Б.1. Затем ввели данные для подключения, для подключения были использованы данные тестируемого экземпляра, т.е. было произведено подключение к себе, данные экземпляра, изображены на рисунке Б.2. Далее ввели полученные данные для подключения, как показано на рисунке Б.3. Далее нажали на кнопку «Добавить», после успешного создания элемента появилась запись о новом пользователе, как показано на рисунке Б.4. Далее нажали на ранее созданного пользователя левой кнопкой мыши, чтобы раскрылось дерево вкладок, в котором отобразился адрес, пример адреса изображен на рисунке Б.5. Далее щелкнули дважды левой кнопки мыши, по элементу адрес после началась инициализация подключения. Когда подключение прошло успешно, поле для ввода сообщений стало активным, в противном случае в поле для ввода сообщений отобразился текст, говорящий о том, что пользователь находится вне сети, пример неудачного подключения изображен на рисунке Б.6.

В случае удачного подключения, напечатали текст в поле для ввода сообщений, который отправили пользователю, пример написанного сообщения изображен на рисунке Б.7. Чтобы отправить сообщение, нажали на клавишу «Enter» на клавиатуре, пример успешно отправленного сообщения изображен на рисунке Б.8.

Далее отправили сообщение с цифровой подписью, для этого подключились к ранее созданному пользователю, как было описано выше, ввели сообщение в поле для ввода текста, как изображено на рисунке Б.7, и произвели нажатие на иконку «Карандаша» справа от поля для ввода текста, элемент для подписи сообщения изображен на рисунке Б.9. Когда режим цифровой подписи сообщения был успешно активирован, иконка карандаша приняла голубой цвет. Пример успешно активированного режима электронной подписи, изображен на рисунке Б.10. Далее нажали на клавишу «Enter», отпарвили подписанное сообщение, подписанные сообщения в отличии от обычных сообщений выделяются голубым цветом, что сигнализирует вам и оппоненту от том что отправленное сообщение – подписанное, пример отправленного подписанного сообщения, изображен на рисунке Б.11.

Чтобы проверить электронную подпись подписанного сообщения, сделали двойной щелчок левой кнопки мыши, по подписанному сообщению, после чего появилась новая форма, в которой присутствует три поля с текстом, пример данной формы изображён на рисунке Б.12. В одном из полей для текста, содержится открытый ключ пользователя, с которым происходила переписка, во втором поле с текстом, было непосредственно полученное сообщение и в третьем поле содержится цифровая подпись данного сообщение. Если подпись соответствовала ключу и сообщению, тогда над полем с цифровой подписью написано зеленым текстом «Подпись «Подтверждена»», пример подтвержденной подписи, изображён на рисунке Б.13. Когда подпись не соответствовала сообщению, текст надо полем с подписью стал красным и отображён как «Подпись «Несоответствие»», пример несоответствующей подписи, изображён на рисунке Б.14.

Далее протестировали функцию отправки файлов. Для отправки файла экземпляру программы, создали подключение с пользователем. Далее подключились к доступному пользователю. Чтобы отправить файл, нажали на иконку файла, которая находится левее иконки, активирующей отправку подписанных сообщений, как выглядит иконка отправки файла, можно увидеть на рисунке Б.15. После нажатия на иконку для отправки файла, появилось окно проводника, в котором выбрали файл, который был отправлен другому пользователю, как выглядит окно выбора файла можно посмотреть на рисунке Б.16. После успешного выбора файла, началась отправка файла. Отправка файла сопровождалась блокировкой поля для отправки сообщения и выводом текущего состояния отправки файла, которое можно увидеть на рисунке Б.17. После полной отправки файла, автоматически отправилось сообщение, которое уведомляло оппонента об отправленном файле, пример такого сообщения можно увидеть на рисунке Б.18. Пользователь которому отправлялся большой пакет данных, таких как файл или объёмное сообщение, мог увидеть состояние получаемых данных, на специальном компоненте, который находится в левом углу программного средства, пример такого монитора получаемых данных, изображён на рисунке Б.19. Полученный файл нашли в корневой директории, в папке «files», директория с полученным файлом, изображена на рисунке Б.20.

Далее протестировали компоненты окна, содержавшего элементы структур, как выглядит данное окно, можно увидеть на рисунке Б.1. Для начала нажали на иконку папки, находящуюся в панели инструментов над окном контактов, увидеть пример такой иконки можно на рисунке Б.21. После нажатия на иконку папки, в окне контактов появился элемент «Папка», что можно увидеть на рисунке Б.22.

Чтобы создать вложенную папку, в уже имеющейся папке, нажали правой кнопкой мыши по элементу «Папка», в которую вкладываем другую папку, пример контекстного меню папки изображен на рисунке Б.23. Далее нажали на элемент «Добавить папку», чтобы создать вложенную папку. После нажали на данный элемент, в элементе «папка», создался новый элемент «папка», что можно увидеть на рисунке Б.24. После всех манипуляций, была создана вложенная папка в элементе «папка».

Чтобы изменить название папки, также надо вызвать контекстное меню, которое изображено на рисунке Б.23, и выбрать элемент «Изменить». После нажатия, посредством анимации, появилось окно с тремя компонентами, одно поле для ввода текста и две кнопки, данное окно можно увидеть на рисунке Б.25. В поле для текста находилось текущее имя папки, чтобы изменить имя папки, изменили текст в поле для ввода текста, на тот, что был необходим. Далее нажали кнопку «Изменить», после чего имя папки изменилось, окно редактирования закрылось, результат показан на рисунке Б.26. Чтобы прервать операцию изменения имени папки, нажали на кнопку «Закрыть», после нажатия окно редактирования папки закрылось, имя папки не изменилось.

Чтобы удалить папку, вызвали контекстное меню папки посредством нажатия правой кнопки мыши по элементу «папка», после чего выбрали поле «Удалить», после нажатия кнопки папка и все ее содержимое было удалено.

Чтобы не удалять папку, но удалить все вложенные структуры в нее, вызвали контекстное меню папки, посредством правого клика мыши, после появилось контекстное меню, в котором выбрали «Удалить содержимое». После нажатия на кнопку, все содержимое папки было удалено, сама папка осталась на месте.

Чтобы изменить имя созданного пользователя, нажали правую кнопку мыши, вызвали контекстное меню элемента пользователь и выбрали пункт «Изменить», после нажатия на кнопку, появилось окно редактирования пользователя, в котором присутствовало пять компонентов, пример данного окна можно увидеть на рисунке Б.27. В данном окне было три поля для ввода текста, первое поле, изменяемое, это текущее имя пользователя в поле контактов, второе поле, неизменяемое, это глобальное имя пользователя во всей сети, третье поле, неизменяемое, это открытый ключ этого пользователя. Чтобы изменить имя пользователя, изменили текст в первом поле для текста на новый. После нажали на кнопку изменить. После нажатия на кнопку имя пользователя изменилось, окно редактирования закрылось. Чтобы прервать процесс изменения имени пользователя, нажали на кнопку «Закрыть». Пример изменённого пользователя можно увидеть на рисунке Б.28.

Чтобы перейти к переписке с пользователем, вызвали контекстное меню, посредством правого клика мыши по элементу «пользователь», после выбрали пункт меню «Перейти в диалог». После всех проделанных манипуляции, в окне отображения сообщений появилась текущая переписка, результат можно увидеть на рисунке Б.18.

Чтобы удалить пользователя, вызвали контекстное меню элемента «Пользователь», посредством нажатия правого клика мыши по элементу, после выбрали пункт меню «Удалить», после проделанных манипуляций пользователь был удален из окна контактов.

Чтобы создать группу, нажали на иконку, находящуюся на панели взаимодействия с окном контактов, вид и расположение данной иконки можно увидеть на рисунке Б.28. После нажатия по иконке создания группы, в окне контактов появился элемент «Группа», пример такого элемента можно увидеть на рисунке Б.29. Чтобы добавить пользователей в новую группу, создали элемент «Пользователь». Элемент пользователь переместили в элемент группа, тем самым создав структуру, изображенную на рисунке Б.30. После того как отправили сообщение в группу, его получили все пользователи, являющееся вложенными в элемент «Группа». При первом получении сообщение с новой группы, автоматически создался элемент «Группа», в которую сразу вложен был элемент «Пользователь», который отправил сообщение в группу.

Чтобы удалить группу, надо вызвали контекстное меню посредством правого клика мыши, после чего выбрали пункт «Удалить», после группа была удалена из окна контактов.

Таким образом, все функции «Средства для безопасной передачи данных» были протестированы, найденные ошибки были исправлены и программное средство готово к использованию.

5.2.2 Полное тестирование

Полное тестирование представляет собой ряд действий, которые возможны при работе с программой, то есть всевозможные пути и способы работы в ней. Достоверность работы программы можно показать на примере функции создания нового подключения.

Создание подключения началось с открытия формы добавления подключения. Чтобы открыть форму нажали на иконку «человечка» на панели взаимодействия с окном контактов, пример изображен на рисунке Б.1. После открылась форма подключения, в которую ввели данные, в первое поле ввели глобальное имя пользователя, к которому создается подключение. В случае неправильно указанного имени, данные получаемые от такого пользователя недоступны по созданному объекту и при получении данных от этого пользователя, автоматически создался новый элемент, уже с корректными данными. Далее ввели «IP и PORT» пользователя, к которому создается подключения, в случае неправильно написанного адреса, программа игнорировала создание нового элемента, в случае отсутствия в сети пользователя, объект создался. Следующее поле это поле для ввода открытого ключа пользователя, если ввести неправильный ключ, то во время отправки и получения данных всплывали исключение, предупреждающее о том что введённый ключе не соответствовал формату, в случае если был установленный ключ не того пользователя, получение корректных данный в дальнейшем – было невозможно. Далее окне контактов появился новый элемент, который показан на рисунке Б.4, к которому уже можно подключится и отправлять сообщения. Максимальное количество отправляемых знаков ограничено в 750, в целях оптимизации объемов получаемых и отправляемых данных.

Таким образом, функции по отправке данных «Средства для безопасной передачи данных» были протестированы, найденные ошибки были исправлены и программное средство готово к использованию.

6 ПРИМЕНЕНИЕ

6.1 Назначение программы

Основное назначение «Средства для защищенного обмена данными» **—** это создание прямого защищенного подключения между пользователями или группой пользователей. Отправка зашифрованных сообщений и файлов, реализация системы цифровой подписи.

Средство предназначено для создания защищенной передачи внутри сети. Программное средство, может быть использовано как в корпоративной сети, так и на малых предприятиях, для расширения политики безопасной передачи данных.

Приложение может использовать любой желающий при наличии устройства с операционной системой «Windows 7» и выше, а также с «DotNetFramework 4.8» или выше.

6.2 Условия применения

Требуется наличие персонального компьютера с операционной системой Windows 7 или выше, а также устройств ввода в качестве клавиатуры и оптической мыши.

Минимальными системными требованиями для приложения являются – процессор 2 ядра, 1.4 GHz, видеокарта 512 мб, ОЗУ 1 гб, места на жестком диске от 500 мб.

Рекомендуемыми системными требованиями для приложения являются – процессор 4 ядра, 2.0 GHz, видеокарта 1 гб, ОЗУ 2 гб, место на жестком диске от 1 гб.

6.3 Справочная система

Справочная система предназначена для получения пользователем максимально релевантной информации по интересующей его теме. Обычно выбор статьи происходит по иерархии разделов справки. Справочные системы часто комбинируются с поисковыми, где выборка релевантных статей определяется по заданным ключевым словам или частью предложения.

«Средство для защищенного обмена данными» имеет справочную систему. Для запуска справочной системы, пользователю необходимо запустить приложение и нажать на кнопку вызывающую веб-страницу со справочной информацией.

После запуска справки отображается страница со справочной информацией, имеющая древовидное меню. С помощью этого меню Можно легко ознакомиться со всеми функциями программного средства.

В справочной системы описаны все компоненты и функции программного средства. Информация расположена в виде списка, от базовых компонентов и функций до функций, раскрывающих потенциал программного средства, справочное меню выглядит так как показано на рисунке Б.31.

Справка была реализована в виде WEB-страницы, при помощи языка разметки «HTML» и языка каскадных стилей «CSS», также был использован язык «JavaScript» для реализации логики древовидного меню, в котором элементы имеют ссылку на информацию о функциях и компонентах программного средства.

7 ОХРАНА ТРУДА. ВРЕДНЫЕ ФАКТОРЫ ПРИ РАБОТЕ С ПЭВМ

**Работа на ПЭВМ сопровождается постоянным и значительным напряжением функций зрительного анализатора**. Одной из основных особенностей является иной принцип чтения информации, чем при обычном чтении. При обычном чтении текст на бумаге, расположенный горизонтально на столе, считывается работником с наклоненной головой при падении светового потока на текст. При работе на **ПЭВМ** оператор считывает текст, почти не наклоняя голову, глаза смотрят прямо или почти прямо вперед, текст (источник — люминесцирующее вещество экрана) формируется по другую сторону экрана, поэтому пользователь не считывает отраженный текст, а смотрит непосредственно на источник света, что вынуждает глаза и орган зрения в целом работать в несвойственном ему стрессовом режиме длительное время.

**Расстройство органов зрения резко увеличивается при работе более четырех часов в день**. Всемирная организация здравоохранения (ВОЗ) ввела понятие «компьютерный зрительный синдром» (КЗС), типовыми симптомами которого являются жжение в глазах, покраснение век и конъюнктивы, чувство инородного тела или песка под веками, боли в области глазниц и лба, затуманивание зрения, замедленная перефокусировка с ближних объектов на дальние.

Нервно-эмоциональное напряжение при работе на **ПЭВМ** возникает вследствие дефицита времени, большого объема и плотности информации, особенностей диалогового режима общения человека и **ПЭВМ**, ответственности за безошибочность информации. Продолжительная работа на дисплее, особенно в диалоговом режиме, может привести к нервно-эмоциональному перенапряжению, нарушению сна, ухудшению состояния, снижению концентрации внимания и работоспособности, хронической головной боли, повышенной возбудимости нервной системы, депрессии.

**Повышенные статические и динамические нагрузки у пользователей** ПЭВИ **приводят к жалобам на боли в спине, шейном отделе позвоночника и руках**. Из всех недомоганий, обусловленных работой на компьютерах, чаще встречаются те, которые связаны с использованием клавиатуры. В период выполнения операций ввода данных количество мелких стереотипных движений кистей и пальцев рук за смену может превысить 60 тыс., что в соответствии с гигиенической классификацией труда относится к категории вредных и опасных. Поскольку каждое нажатие на клавишу сопряжено с сокращением мышц, сухожилия непрерывно скользят вдоль костей и соприкасаются с тканями, вследствие чего могут развиться болезненные воспалительные процессы.

Другой причиной возникновения болей в спине может быть длительное пребывание в положении «сидя», которое приводит к сильному перенапряжению мышц спины и ног, в результате чего возникают боли и неприятные ощущения в нижней части спины. Основной причиной перенапряжения мышц спины и ног являются нерациональная высота рабочей поверхности стола и сидения, отсутствие опорной спинки и подлокотников, неудобное размещение монитора, клавиатуры и документов, отсутствие подставки для ног.

Для существенного уменьшения боли и неприятных ощущений, возникающих у пользователей **ПЭВМ**, необходимы частые перерывы в работе и эргономические усовершенствования, в том числе оборудование рабочего места так, чтобы исключать неудобные позы и длительные напряжения.

К числу факторов, ухудшающих состояние здоровья пользователей компьютерной техники, относятся электромагнитное и электростатическое поля, акустический шум, изменение ионного состава воздуха и параметров микроклимата в помещении. Немаловажную роль играют эргономические параметры расположения экрана монитора (дисплея), состояние освещенности на рабочем месте, параметры мебели и характеристики помещения, где расположена компьютерная техника.

**8 ЭКОНОМИЧЕСКИЙ РАЗДЕЛ**

**8.1 Определение объёма и трудоёмкости программного обеспечения**

Программные средства вычислительной техники являются материальными объектами интеллектуальной специальной деятельности специалистов, состоящими из программных документально оформленных проектов, реализующих свои потребительские свойства и качества в составе функционирующих вычислительных систем или систем обработки данных.

По стоимости и срокам службы программное средство относится к основным производственным фондам предприятия. Программные средства вычислительной техники как товарная продукция могут быть двух видов:

научно-техническая продукция;

продукция производственно-технического назначения.

В современных рыночных условиях программные средства выступают преимущественно в виде продукции научно-технических организаций и реализуемые покупателям по рыночным отпускным ценам. Все современные разработки программного средства вычислительной техники являются научно-технической продукцией.

Само программное средство, как продукция производственно-технического назначения, представляет собой программное средство вычислительной техники. Данное средство разработано в соответствии с нормативными документами, прошедшее государственные испытания.

Широкое применение вычислительной техники требует постоянного обновления и совершенствования программного продукта. Выбор эффективных проектов программного продукта связан с их экономической оценкой и расчетом экономического эффекта.

Экономический эффект программного средства может определяться как у разработчика, так и у пользователя. У разработчика экономический эффект выступает в виде чистой прибыли, остающейся в распоряжении предприятия от реализации программы, а у пользователя – в виде экономии трудовых, материальных и финансовых ресурсов.

Стоимостная оценка программного продукта у разработчиков предполагает составление сметы затрат, которая включает следующие статьи:

* заработная плата исполнителей основная (ЗО) и дополнительная (ЗД);
* отчисления в фонд социальной защиты населения (Зсз) и в Белгосстрах (Збгс);
* материалы (М);
* специальное оборудование (РС);
* машинное время (РМ);
* расходы на научные командировки (РНК);
* прочие расходы (РП);
* накладные расходы (РН).

На основе общей суммы расходов по всем статьям (Ср) и результатов маркетинговых исследований на рынке ПО определяется плановая отпускная цена (Цo) с учетом прибыли (рентабельности) и налогов, включаемых в цену. Объем программы (в 1000 условных машинных команд) определяется путем подбора аналогов на основании классификации типов программного средства.

На основании имеющейся информации о функциях разрабатываемого программного средства определяется объем каждой функции для соответствующего типа электронно-вычислительных машин. Затем определяется тип разрабатываемого программного средства (по принятой классификации в каталоге аналогов) и уточняются объемы функций для данного программного средства вычислительной. Если значение объема той или иной функции разрабатываемого программного средства вычислительной техники не уточняется в зависимости от типа программного средства вычислительной техники, то это значение остается принятым в соответствии с «Каталогом функций программного средства вычислительной техники».

В таблице 1 приведены характеристики функций программного средства и их объем.

Таблица 1 – Характеристики функций и их объем

|  |  |  |
| --- | --- | --- |
| Номер функции | Содержание функции | Объем (условная машинная команда) |
| 101 | Организация ввода информации | 150 |
| 102 | Контроль, предварительная обработка и ввод информации | 150 |
| 111 | Управление вводом/выводом | 2400 |
| 203 | Формирование баз данных | 2180 |
| 204 | Обработка наборов и записи базы данных | 2670 |
| 205 | Обслуживание базы данных в пакетном режиме | 1260 |
| 206 | Обслуживание базы данных в интерактивном режиме | 6950 |
| 207 | Манипулирование данными | 9550 |
| 210 | Загрузки базы данных | 2780 |
| 305 | Обработка файлов | 720 |
| 308 | Управление файлами | 5750 |
| 309 | Формирование файла | 1020 |
| 506 | Обработка ошибочных и сбойных ситуаций | 410 |
| 507 | Обеспечение интерфейса между компонентами | 970 |
| 604 | Справка и обучение | 720 |
| Итого |  | 37 680 |

Оценка трудоемкости разработки программного средства осуществляется в два этапа. На первом этапе рассчитывается нормативная трудоемкость разработки программного средства (Тн). С учетом дополнительного коэффициента сложности (Ксл) рассчитывается общая трудоемкость программного средства по формуле:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| То = Тн \* Ксл \* Кт \* Кн | | | | (1) |
| где | То | – | общая трудоемкость программного средства; | |
| Тн | – | нормативная трудоемкость программного средства; | |
| Ксл | – | дополнительный коэффициент сложности программного средства; | |
| Кт | – | поправочный коэффициент, учитывающий степень использования при разработке стандартных модулей; | |
| Кн | – | степень новизны программного обеспечения. | |

Расчет общей трудоемкости:

|  |  |
| --- | --- |
| То = 792 \* 1,12 \* 0,7 \* 0,9 = 559 человеко-дней. |  |

Bce ПО принято подразделять на три категории сложности в зависимости от наличия (отсутствия) следующих характеристик:

* высокий уровень языкового интерфейса с пользователем;
* режим работы в реальном времени;
* управление удаленными объектами;
* машинная графика, многомашинные комплексы;
* существенное распараллеливание вычислений;
* нестандартная конфигурация технических средств;
* оптимизационные и особо сложные инженерные и научные
* расчеты;
* переносимость ПО.

Влияние фактора сложности на трудоемкость учитывается умножениемнормативной трудоемкости на соответствующий коэффициент сложности.

Коэффициент сложности (Kcл). Посредством коэффициента сложности учитываются дополнительные затраты труда, связанные со сложностью разрабатываемого программного продукта. Коэффициент сложности рассчитывается по формуле:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | | | | (2) |
| где | Кi | – | коэффициент, соответствующий степени повышения сложности ПО за счет конкретной характеристики. | |

Коэффициент, учитывающий степень использования при разработке ПО стандартных модулей (Kт). Степень использования в разрабатываемом ПО стандартных модулей определяется их удельным весом в общем объеме проектируемого продукта.

Коэффициент новизны разрабатываемого ПО (Kн). Сравнение характеристик разрабатываемого ПОс имеющимися аналогами позволяет определить экспертным путем степень его новизны. Если нет доступных аналогов, тоПО присваивается категория А. Степень новизны ПО категорий Б и В соответствует развитию уже имеющихся программных продуктов. При установлениикоэффициентов новизны учитываются степень новизны ПО и предназначениеего для новых или освоенных типов ПК, для новых или освоенных ОС. Влияние фактора новизны на трудоемкость учитывается путем умножения трудоемкости на соответствующий коэффициент новизны.

Расчет коэффициента сложности:

|  |  |
| --- | --- |
| Kcл= 1+ 0,12 = 1,12. |  |

При решении сложныхзадач с длительным периодом разработки ПО трудоемкость определяется постадиям разработки:

* техническое задание (ТЗ) − исследование;
* эскизный проект (ЭП) − анализ требований;
* технический проект (ТП) − проектирование;
* рабочий проект (РП) − разработка (кодирование, тестирование);
* внедрение (ВН) − ввод в действие.

При этом на основании нормативной трудоемкости рассчитывается общая трудоемкость с учетом распределения ее по стадиям (Tо):

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | | | | (3) |
| где | Тi | – | трудоемкость разработки ПО на i-й стадии (чел./дн.); | |
| n | – | количество стадий разработки. | |

Трудоемкость стадий определяется на основе нормативной трудоемкости  
с учетом сложности, новизны, степени использования в разработке стандартных модулей ПО и удельного веса трудоемкости каждой стадий в общей трудоемкости ПО:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Tyi= Tн \* dcтi \* Kcл \* Kт \* Kн | | | | (4) |
| где | Тyi | – | уточнённая трудоемкость разработки ПО на i-й стадии (технического задания, эскизного проекта, технического проекта, рабочего проекта и внедрения); | |
| dстi | – | удельный вес трудоемкости i-й стадии разработки ПО в общейтрудоемкости разработки ПО; | |
| Kсл | – | коэффициент, учитывающий сложность ПО, вводится на всех стадиях; | |
| Kт | – | коэффициент, учитывающий степень использования стандартных модулей ПО, вводится только на стадии рабочего проекта; | |
| Kн | – | коэффициент, учитывающий степень новизны ПО, вводится на всех стадиях. | |

Трудоемкость ПО по стадиям. Все стадии разработки ПО различаются трудоемкостью. Трудоемкость разработки стадий ПО (Tуз, Tуэ, Tут, Tуp, Tув) определяется с учетом удельного веса трудоемкости стадии в общей трудоемкостиПО (d), сложности (Kc), новизны ПО (Kн) и степени использования стандартных модулей (Kт). При этом коэффициент Kт используется только на стадии «Рабочий проект» при написании исходного кода (разработки программы).

Трудоемкость стадий ПО рассчитывается по следующим формулам:

|  |  |  |  |
| --- | --- | --- | --- |
| трудоемкость стадии ТЗ: | Tyз = Tн\*Kc\*dз \*Kн , |  | (5) |
| трудоемкость стадии ЭП: | Tyэ = Tн\*Kc\*dэ \*Kн , |  | (6) |
| трудоемкость стадии ТП: | Tyт = Tн **.**\* Kc \*dт\*Kн , |  | (7) |
| трудоемкость стадии РП: | Typ=Tн \* Kc\*dp\*Kн\*Kт, |  | (8) |
| трудоемкость стадии ВН: | Tyв = Tн  \*Kc \* dв \* Kн |  | (9) |

Общая трудоемкость определяется как сумма трудоемкостей по стадиям:

|  |  |
| --- | --- |
| Tу= Tyз+ Tyэ+ Tyт + Typ +Tyв . | (10) |

Таблица 2 – Расчет общей трудоемкости разработки ПО и численности исполнителей с учетом стадий

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Показатели | Стадии | | | | | Итого |
|  | ТЗ | ЭП | ТП | РП | ВН |
| 1.Коэффициенты удельных весов трудоемкости стадии разработки ПО (d) | 0,10 | 0,08 | 0,09 | 0,58 | 0,15 | 1,0 |
| 2.Распределение нормативной трудоемкости ПО(Тн) по стадиям, чел./дн. | 79,2 | 63,4 | 71,3 | 459,4 | 118,8 | 792 |
| 3.Коэффициент сложности ПО (Kcл) | 1,12 | 1,12 | 1,12 | 1,12 | 1,12 |  |
| 4.Коэффициент, учитывающий использование стандартных модулей |  |  |  | 0,7 |  |  |
| 5.Коэффициент, учитывающий новизну ПО (Кн) | 0,9 | 0,9 | 0,9 | 0,9 | 0,9 |  |
| 6.Общая трудоемкость ПО (Ty), чел./дн. | 79,8 | 63,9 | 71,9 | 324,25 | 119,8 | 659,6 |

**8.2 Расчет эффективного фонда времени и численности работников**

Эффективный фонд времени работника (Фэф) рассчитывается по формуле:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Фэф = Дг – Дп – Дв – Дo | | | | (11) |
| где | Дг | – | количество дней в году; | |
| Дп | – | количество праздничных дней в году; | |
| Дв | – | количество выходных дней в году; | |
| До | – | количество дней отпуска. | |
| Фэф = 366 – 10 – 101 – 24 = 231 дней. | | | |  |

На основании уточненной трудоемкости (Трудоемкость представляет собой затраты рабочего времени на производство единицы продукции) программного средства и установленного периода разработки рассчитаем продолжительность участия в разработке по формуле:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Туч = (Фэф / 12) \* Тпл | | | | (12) | |
| где | Туч | – | уточненная трудоемкость программного средства; | | |
|  | Тпл | – | плановая продолжительность разработки программного средства (месяцев); | | |
|  | Фэф | – | эффективный фонд времени одного работника. | | |
| Туч = (231 / 12) \* 3 = 58 дней. | | | | |  |

**8.3.1 Расчет основной заработной платы**

В соответствии с «Рекомендациями по применению Единой тарифной сетки рабочих и служащих народного хозяйства» тарифными разрядами и коэффициентами должностей руководителей научных организаций, вычислительных центров, бюджетных учреждений науки, непроизводственных отраслей народного хозяйства, каждому исполнителю устанавливается разряд и тарифный коэффициент. Месячная тарифная ставка каждого исполнителя (ТМ) определяется путем умножения действующей тарифной ставки первого разряда на тарифный коэффициент (ТК), соответствующий установленному тарифному разряду:

|  |  |
| --- | --- |
| Tм = Тм1 \* Tк | (13) |

Расчет месячной тарифной ставки программиста I категории:

|  |  |
| --- | --- |
| ТМ = 50 \* 3,25 = 162,5 рублей. |  |

Часовая тарифная ставка рассчитывается путем деления месячной тарифной ставки на установленный фонд рабочего времени, 160 часов:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Тч = Тм /Фр, | | | | (14) |
| где | Tч | – | часовая тарифная ставка; | |
| Tм | – | месячная тарифная ставка; | |
| Фр | – | фонд рабочего времени. | |
| рублей. | | | |  |

Основная заработная плата исполнителей на конкретное программное средство рассчитывается по формуле:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | | | | (15) |
| где | n | – | количество исполнителей, занятых разработкой конкретного ПО; | |
|  | Tчi | – | часовая тарифная ставка i-го исполнителя; | |
|  | Фп | – | плановый фонд рабочего времени i-го исполнителя (дн.); | |
|  | Tч | – | количество часов работы в день (ч); | |
|  | К | – | коэффициент премирования. | |

В соответствии со штатным расписанием на разработке будет занят программист II категории, что соответствует тарифному разряду − 12 и тарифному коэффициенту − 2,84. Расчет основной заработной платы представлен в таблице 3.

Таблица 3 – Основная заработная плата

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Исполнители | Тарифная ставка 1-го разряда (Тм1), тыс.р. | Тарифный коэффициент (Тк) | Тарифная ставка данного разряда (Тм), тыс.р. | Эффективный фонд работы за меяц (Нмес), час | Тарифная ставка часовая, (Тч) тыс.р. | Тарифная ставка дневная (Тдн.) тыс. р. | Продолжительность участия в разработке, дн. | Коэффициент премирования Кпр | Заработная плата основная |
| Программист первой категории | 50 | 3,25 | 162,5 | 160 | 1,02 | 8,2 | 58 | 1,5 | 709,9 |

**8.3.2 Расчет дополнительной заработной платы исполнителей**

Дополнительная зарплата на конкретное программное средство (Здi) включает выплаты, предусмотренные законодательством о труде (доплаты к отпускам, оплата льготных часов, оплата времени выполнения государственных обязанностей и другие выплаты, не связанные с основной деятельностью исполнителя) и определяются по нормативу в процентах к основной зарплате по формуле:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | | | | (16) |
| где | Здi | – | дополнительная заработная плата исполнителей на конкретное программное обеспечение; | |
|  | Нд | – | норматив дополнительной заработной платы; | |
|  | Зо | – | основная заработная плата в целом по организации. | |
| рубль. | | | |  |

**8.3.3 Расчет отчислений в фонд социальной защиты населения**

Отчисления в фонд социальной защиты определяются в соответствии с действующими законодательными актами по нормативу в процентном отношении к фонду основной и дополнительной зарплаты исполнителей, определенной по нормативу, установленному в целом по организации, согласно формуле:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | | | | (17) |
| где |  | – | норматив отчислений в фонд социальной защиты населения, 34%; | |
|  | Зо | – | заработная плата основная; | |
|  | Зд | – | заработная плата дополнительная. | |
| 265,5 рубля. | | | |  |

**8.3.4 Расчет отчислений в Белгосстрах**

Отчисления в Белгосстрах определяются в соответствии с действующими законодательными актами по нормативу в процентном отношении к фонду основной и дополнительной зарплаты исполнителей, определенной по нормативу, установленному в целом по организации, согласно формуле:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | | | | (18) |
| где | Нбгс | – | норматив отчислений в Белгосстрах, 0,1%; | |
|  | Зо | – | заработная плата основная; | |
|  | Зд | – | заработная плата дополнительная. | |
| рублей. | | | |  |

**8.4 Расчет затрат на материалы**

Расходы по статье «Материалы» (М) определяются на основании сметы затрат, разрабатываемой на ПО с учетом действующих нормативов. Нормы расхода материалов в суммарном выражении (Hм) определяются в расчете на 100 строк исходного кода или по нормативу в процентах к фонду основной заработной платы разработчиков (Нмз), который устанавливается организацией (3-5%). Сумма затрат на расходные материалы рассчитывается по формуле:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | | | | (19) |
| где | Нмз | – | норма расхода материалов от основной заработной платы (%); | |
|  | Зоi | – | заработная плата основная. | |
| Мi = (709,9 \* 3) / 100 = 21,3 рублей | | | |  |

**8.5 Расчет затрат на машинное время**

Расходы по статье «Машинное время» (Рмi), включают оплату машинного времени, необходимого для разработки и отладки программного средства, которое определяется по нормативам (машино-часах) на 100 условных команд (Нмв) машинного времени в зависимости от характера решаемых задач и типа электронно-вычислительных машин и рассчитывается по формуле:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Рмi=Цмi \* \* Нмв, | | | | (20) |
| где | Цмi | – | цена одного машино-часа; | |
|  | Voi | – | общий объем ПО (строк исходного кода); | |
|  | Нмв | – | норматив расхода машинного времени на отладку 100 строк исходного кода. | |
| рублей. | | | |  | |

**8.6 Расчет прочих затрат**

Расходы по статье «Прочие затраты» (Пзi)на конкретное ПО включают затраты на приобретение и подготовку специальной научно-технической информации и специальной литературы. Определяются по нормативу, разрабатываемому в целом по организации, в процентах к основной заработной плате:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | | | | (21) |
| где | Hпз | – | норматив прочих затрат в целом по организации; | |
|  | Пз | – | прочие затраты в целом по организации. | |
| Пзi = (709,9 \* 17) / 100 = 120,7 рублей | | | |  |

**8.7 Составление сметы затрат**

Составление калькуляции и сметы затрат преследует две основные цели:

* определение полной стоимости объекта или проекта для оценки эффективности финансовых вложений;
* определение статей, формирующих себестоимость, для принятия дальнейших управленческих решений по нахождению путей снижения себестоимости.

Общая сумма расходов по всем статьям сметы рассчитываются по формуле:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Сpi = Зоi + Здi + Зфсзн + Збгс + Мi + Рмi + Пзi | | | | (22) |
| где | Зoi | – | основная заработная плата исполнителя; | |
|  | Здi | – | дополнительная заработная плата; | |
|  | Зфсзнi | – | отчисления в фонд социальной защиты населения; | |
|  | Збгс | – | отчисления в Белгосстрах; | |
|  | Мi | – | расходы по статье «Материалы»; | |
|  | Pмi | – | расходы по статье «Машинное время»; | |
|  | Пзi | – | расходы по статье «Прочие затраты». | |
| Срi = 709,9 + 71 + 265,5 + 4,7 + 21,3 + 4521,6‬ + 120,7 = 5715 рублей. | | | |  |

Таблица 4 – Смета затрат

|  |  |  |  |
| --- | --- | --- | --- |
| Наименование статей затрат | | Обозначение | Сумма (руб.) |
| Основная заработная плата | | Зoi | 709,9 |
| Дополнительная заработная плата | | Здi | 71 |
| Отчисления в фонд социальной защиты населения | Зфсзнi | | 265,5 |
| Отчисления в Белгосстрах | Збгс | | 4,7 |
| Материалы | Мi | | 21,3 |
| Машинное время | Pмi | | 4521,6 |
| Прочие затраты | Пзi | | 120,7 |
| Итого |  | | 5715 |

Кроме того, организация-разработчик осуществляет затраты на сопровождение и адаптацию программного продукта (Pcаi), которые определяются по нормативу и рассчитываются по формуле:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | | | | (23) |
| где | Спр | – | производственная себестоимость; | |
|  | Нса | – | норматив отчислений на сопровождение и адаптацию программного средства. | |
| 628,7 рублей. | | | |  |

Таким образом полная себестоимость разработки программного средства определяется по формуле:

|  |  |
| --- | --- |
| Спол = Сpi + Рсi | (24) |
| Спол = 5715 + 628,7 = 6343,7 рублей. |  |

**8.8 Расчёт прогнозируемой отпускной цены**

Цена – это денежное выражение стоимости единицы товара. Отпускная цена производителя – это цена, по которой производитель реализует продукцию оптово-сбытовым организациям. Она включает издержки производства и реализации прибыль, налог на добавленную стоимость. Прогнозируемая отпускная цена рассчитывается по формуле:

|  |  |
| --- | --- |
| Ц=Спол+П+НДС | (25) |
| Ц = 6343,7 + 824,7 + 1433,7 = 8602‬ рублей |  |

Рентабельность и прибыль по создаваемому программному продукту определяется исходя из результатов анализа рыночных условий, переговоров с заказчиками (потребителями) и согласование с ним отпускной цены, включая дополнительно налог на добавленную стоимость, рассчитывается по формуле:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | | | | (26) |
| где | П | – | прибыль от реализации ПО заказчику; | |
|  | Ур | – | уровень рентабельности ПО (%); | |
|  | Спол | – | полная себестоимость ПО. | |
| П== 824,7 рублей. | | | |  |

Кроме этого, в цену включается налог на добавленную стоимость (Налог на добавленную стоимость (НДС) – это один из самых распространенных налогов на потребление. Его суть состоит в добавлении к цене при продаже определенного процента налога. Этот налог уплачивается покупателем продавцу, а тот переводит его в [бюджет государства](http://myfin.by/info/byudzhet-prozhitochnogo-minimuma). Взимается в большинстве государств мира. Поскольку налог относится к категории косвенных – то, по умолчанию, включается в стоимость практически любого товара или услуги), рассчитывается по формуле:

|  |  |
| --- | --- |
| НДС = | (27) |
| НДС== 1 433,7 рублей. |  |

Таблица 5 – Прогнозируемая отпускная цена

|  |  |  |
| --- | --- | --- |
| Наименование статей затрат | Обозначение | Сумма (тыс.руб.) |
| Полная производственная себестоимость | Спол | 6343,7 |
| Прибыль и рентабельность по создаваемому программному средству | П | 824,7 |
| Налог на добавленную стоимость | НДС | 1433,7 |
| Прогнозируемая отпускная цена | Ц | 8602,1 |

**8.9 Расчёт чистой прибыли и прибыли от реализации продукции**

Прибыль от реализации продукции (Прп) рассчитывается как выручка от реализации товаров (работ, услуг) за минусом налогов, включаемых в цену продукции и выплачиваемых из выручки, себестоимости реализованных товаров (работ, слуг), а также расходов на реализацию (если они не включены в себестоимость). Расчет производится по формуле:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Прп = Ц – Спол – НДС | | | | (28) |
| где | Ц | – | выручка от реализации продукции; | |
|  | С | – | затраты на производство и реализацию; | |
|  | НДС | – | налог на добавленную стоимость. | |

|  |  |
| --- | --- |
| Прп = 8602,1 – 6343,7 – 1433,7 = 824,7 рублей |  |

Налог на добавленную стоимость рассчитывается по формуле:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| НДС = Ц\*Сндс / (100 + Сндс), | | | | (29) |
| где | Ц | – | выручка от реализации продукции; | |
|  | Сндс | – | ставка налога на добавленную стоимость. | |
| НДС = 8602,1 \* 20 / (100 + 20) = 1433,7 рублей | | | |  |

Чистая прибыль (ЧП) рассчитывается как разница налогооблагаемой прибыли и суммы налога на прибыль по формуле:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ЧП=Прн-Ннпр, | | | | (30) |
| где | Прп | – | прибыль от реализации продукции; | |
|  | Ннпр | – | сумма налога на прибыль. | |
| ЧП = 824,7 – 148,4 = 676,3 рублей | | | |  |

Налог на прибыль рассчитывается по формуле:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Ннпр=Прп\*Снпр / 100, | | | | (31) |
| где | Прп | – | прибыль от реализации продукции; | |
|  | Снпр | – | ставка налога на прибыль (в соответствии с действующим законодательством). | |
| Ннрп = 824,7 \* 18 / 100 = 148,4 рублей | | | |  |

**8.10 Расчет показателей эффективности о внедрения программного средства**

К показателям эффективности от внедрения программного средства относятся**:**

* срок окупаемости пректа (Ток);
* коэффициент эффективности программного обеспечения;
* рентабельность затрат (себестоимость).

Срок окупаемости рассчитывается по формуле:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Ток=З/ЧП, | | | | (32) |
| где | З | – | затраты, связанные с разработкой и реализацией программного обеспечения (полная производственная себестоимость); | |
|  | ЧП | – | чистая прибыль. | |
| Ток = 6343,7 / 676,3 = 9,4 | | | |  |

Коэффициент эффективности программного обеспечения (Кэф) рассчитывается по формуле:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Кэф=1/Ток, | | | | (33) |
| где | Ток | – | срок окупаемости проекта. | |
| Кэф = 1 / 9,4 = 0,1 | | | |  |

Рентабельность затрат рассчитывается по формуле:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Рз = ЧП / З \* 100 | | | | (34) |
| где | ЧП | – | чистая прибыль; | |
|  | З | – | затраты, связанные с разработкой и реализацией программного обеспечения (полная производственная себестоимость). | |
| Рз = 676,3 / 6343,7 \*100 = 11% | | | |  |

ЗАКЛЮЧЕНИЕ

Задачей дипломного проекта было создание средства для защищенного обмена данными. Разрабатываемая система была выполнена в соответствии с техническим заданием. Все функции системы были протестированы, найденные ошибки были исправлены.

Программное средство было разработано в середе разработки Visual Studio 2019, в операционной системе «Windows 10». А также разработано с использованием программной платформы DotNetFramework и языка программирования C#.

В программном средстве реализована защищенная передача данных, между двумя или группой пользователей, посредством асинхронных ключей шифрования «RSA». Что позволяет защищенно передавать сообщения через разные сети и третьих лиц, не боясь о краже передаваемой информации. Также реализован алгоритм цифровой подписи, что позволяет быть уверенным в отправителе и полученных данных из любой точки отправления. Цифровая подпись позволяет точно узнать, что получаемые данные были отправлены конкретным пользователем.

В разработанном «Средстве для защищенного обмена данными» были реализованы следующие функции:

1. создание прямого подключения к другим пользователям;
2. создание множественных подключений;
3. отправка данных;
4. отправка файлов;
5. шифрование пересылаемых данных;
6. автоматическая отправка сообщений;
7. отправка данных с цифровой подписью;

В последующем в систему для защищенного обмена данными будет добавлена функция по созданию локальных хранилищ с возможностью скачивания для других пользователей, данная функция позволит создавать свои точки для распространения контента или информации.

Потенциал децентрализованных систем растет с каждым днем, это высокая отказа устойчивость таких систем, посредством того, что не существует единого центра, который может стать целью массивных атак и полностью остановить работу всей системы. Далее в децентрализованных системах не может существовать тотальная монополия на какие-либо виды услуг, также такие системы могут защищать пользователя от анонимизации посредством подключения к ресурсам через других пользователей, принцип работы схож с созданием цепочки VPN соединений. Также можно отметить что в разработанной системе нет обязательной статичности личных данных, это значит что в случая когда RSA ключи были скомпрометированы или при иной обходимости, контактные данные можно изменить за пару щелчков мыши, тем самым снова обезопасив работу в сети.

Данное программной средство соответствует всем требованиям, перечисленным в постановке задачи, и является готовым к применению.

ЛИТЕРАТУРА

1. С. Пугачев, А. Шериев, К. Кичинский. Разработка приложений для Windows 8 на языке C#. БХВ-Петербург, 2013. - 416 c;

2. Ю. С. Магда. NI Measurement Studio. Практика разработки систем измерения и управления на C#. ДМК Пресс, 2013. - 190 c;

3. Адам Фримен. ASP.NET MVC 4 с примерами на C# 5.0 для профессионалов. Вильямс, 2013. - 688 c;

4. Майкл Блюстайн. Изучаем MonoTouch. Создание приложений на платформе iOS с помощью C# и .NET. ДМК Пресс, 2012. - 336 c;

5. Кристиан Нейгел, Билл Ивьен, Джей Глинн, Карли Уотсон, Морган Скиннер. C# 4.0 и платформа .NET 4 для профессионалов (+ CD-ROM). Вильямс, 2011. - 1440 c;

6. Джеффри Рихтер. CLR via C#. Программирование на платформе Microsoft .NET Framework 4.0 на языке C#. Питер, 2012. - 928 c;

7. Саша Голдштейн, Дима Зурбалев, Идо Флатов. Оптимизация приложений на платформе .Net. ДМК Пресс, 2014. - 524 c;

8. Адам Фримен, Джозеф Раттц-мл.. LINQ. Язык интегрированных запросов в C# 2010 для профессионалов. Вильямс, 2011. - 656 c;

9. C# Language Pocket Reference.;

10. А. Л. Марченко. Основы программирования на С# 2.0. Интернет-университет информационных технологий,Бином. Лаборатория знаний, 2017. - 552 c;

11. В. А. Биллиг. Основы программирования на С#. Интернет-университет информационных технологий,Бином. Лаборатория знаний, 2012. - 488 c;

12. Джейсон Прайс, Майк Гандэрлой. Visual C# 2.0. Полное руководство. Век +, Корона-Век, Энтроп, 2007. - 736 c.

Приложение А

(обязательное)

Текст программы

using System;

using System.Data;

using System.Drawing;

using System.IO;

using System.Net;

using System.Net.Sockets;

using System.Security.Cryptography;

using System.Text;

using System.Threading;

using System.Windows.Forms;

namespace ghost

{

public partial class Main : Form

{

SQLite sqlite = new SQLite();

Thread receive;

Thread conWindowAnimation;

Thread conn;

internal static readonly ManualResetEvent \_connectionDone =

new ManualResetEvent(false);

internal static readonly ManualResetEvent \_first\_connectionDone =

new ManualResetEvent(false);

internal static readonly ManualResetEvent file\_sending =

new ManualResetEvent(false);

internal static string local\_ip = "" + Dns.GetHostByName(Dns.GetHostName()).AddressList[Dns.GetHostByName(Dns.GetHostName()).AddressList.Length - 1];

Motoko motoko = new Motoko(local\_ip, 3000);

//Motoko\_autosend motoko\_as = new Motoko\_autosend(local\_ip, 3000, "127.0.0.1", 0);

internal static Color CustomForeColor = Color.White;//Color.FromArgb(120, 198, 200);

internal static Color FormBackColor = Color.FromArgb(29, 29, 36);

internal static Color CustomBackColor = Color.FromArgb(35, 35, 45);

Image[] imglist = new Image[6];

internal static string my\_global\_name;

internal static string public\_key;

internal static string private\_key;

string current\_connection;

string current\_dialog;

int current\_slider = 0;

bool isSign = false;

bool group = false; //Если пишем в группу будет тру если юзверю то фолс

TreeNode renaming\_node;

private int win\_anim\_offset;

private const int cGrip = 16; // Grip size

//===============================[Language Setings]=================================

//set massive for words (current language)

public static string[] Lang = new string[100];

private void LangInit()

{

//get lang file

using (StreamReader sr =

new StreamReader(@"./lang/lang.lang", System.Text.Encoding.UTF8))

{

string line;

int i = 0;

while ((line = sr.ReadLine()) != null)

{

Lang[i++] = line;

}

}

Node\_add\_Button.Text = Lang[3];

button1.Text = Lang[18];

button2.Text = Lang[11];

address\_textbox.Text = Lang[15];

address\_textbox.ForeColor = Color.Silver;

global\_name\_textbox.Text = Lang[16];

global\_name\_textbox.ForeColor = Color.Silver;

open\_key\_textbox.Text = "Публичный ключ";

open\_key\_textbox.ForeColor = Color.Silver;

}

//==================================================================================

public Main()

{

InitializeComponent();

LangInit();

this.FormBorderStyle = FormBorderStyle.None;

this.DoubleBuffered = true;

this.SetStyle(ControlStyles.ResizeRedraw, true);

try

{

receive = new Thread(new ThreadStart(motoko.StartListening));

motoko.GetMessageHandler(Display);

motoko.GetConnectionHandler(connect\_result);

autosend.GetConnectionHandler(autosend\_result);

motoko.GetIncomingPocketsHandler(increment\_pockets);

//motoko\_as.GetConnectionHandler(autosend\_result);

receive.Start();

}

catch (Exception ex)

{

MessageBox.Show(ex.ToString());

}

}

private void Form1\_Load(object sender, EventArgs e)

{

autosend\_timer.Enabled = true;

KeyPreview = true;

set\_treeView\_settings();

database\_connection();

set\_font\_and\_background();

set\_dgv\_settings();

DataSet ds = sqlite.Select\_DataSet("SELECT \* FROM user\_info");

my\_global\_name = ds.Tables[0].Rows[0].ItemArray[0].ToString();

public\_key = ds.Tables[0].Rows[0].ItemArray[1].ToString();

private\_key = ds.Tables[0].Rows[0].ItemArray[2].ToString();

conWindowAnimation = new Thread(new ThreadStart(conWinAnimation));

foreach (TreeNode node in treeView1.Nodes)

Set\_SelectedImageIndex\_in\_treeView\_like\_IamgeIndex(node);

textBox1.Text = my\_global\_name;

textBox2.Text = motoko.CallBackIPAddress.ToString() + ":" + motoko.CallBackPort.ToString();

textBox3.Text = public\_key;

file\_send\_panel.BackgroundImage = Image.FromFile("./images/document.png");

file\_send\_panel.BackgroundImageLayout = ImageLayout.Stretch;

}

//=========================[MainForm move and resize]===============================

//Не выпускай кракена

private void Main\_Resize(object sender, EventArgs e)

{

int top\_margin = 45;

int component\_margin = 15;

int bottom\_margin = 35;

int summary\_margin = component\_margin;

toolbox\_panel.Size = new Size(50, Height - bottom\_margin - component\_margin);

toolbox\_panel.Location = new System.Drawing.Point(component\_margin, top\_margin);

summary\_margin += toolbox\_panel.Size.Width + component\_margin;

treeView1.Size = new Size(Convert.ToInt32(Width \* 0.25), Height - bottom\_margin - component\_margin - 10 - 26);

treeView1.Location = new System.Drawing.Point(summary\_margin, top\_margin + 10 + 26);

panel1.Size = new Size(Convert.ToInt32(Width \* 0.25), Height - bottom\_margin - component\_margin - 60);

panel1.Location = new System.Drawing.Point(summary\_margin, top\_margin + 26);

panel2.Size = new Size(treeView1.Width, 24);

panel2.Location = new Point(summary\_margin, top\_margin);

panel2.BackColor = CustomBackColor;

add\_user\_panel.BackColor = CustomBackColor;

add\_user\_panel.BackgroundImage = Image.FromFile(@"./Images/ico/treeView/user.png");

add\_folder\_panel.BackColor = CustomBackColor;

add\_folder\_panel.BackgroundImage = Image.FromFile(@"./Images/ico/treeView/folder.png");

add\_group\_panel.BackColor = CustomBackColor;

add\_group\_panel.BackgroundImage = Image.FromFile(@"./Images/ico/treeView/group.png");

help\_panel.BackColor = CustomBackColor;

help\_panel.BackgroundImage = Image.FromFile(@"./Images/about.png");

summary\_margin += panel1.Width + component\_margin;

panel\_home.Location = new System.Drawing.Point(Convert.ToInt32((toolbox\_panel.Width - panel\_home.Width) / 2), 10);

panel\_autosend.Location = new Point(Convert.ToInt32((toolbox\_panel.Width - panel\_home.Width) / 2),

Convert.ToInt32(toolbox\_panel.Height - panel\_settings.Height - 10) - 46);

panel\_settings.Location = new Point(Convert.ToInt32((toolbox\_panel.Width - panel\_home.Width) / 2),

Convert.ToInt32(toolbox\_panel.Height - panel\_settings.Height - 10));

help\_panel.Location = new System.Drawing.Point(Convert.ToInt32((toolbox\_panel.Width - panel\_home.Width) / 2), 10 + 46);

send\_message\_text\_box.WordWrap = true;

autosend\_textbox.WordWrap = true;

panel3.Size = new Size(Width - summary\_margin - component\_margin, Height + 500);

panel3.Location = new Point(summary\_margin, top\_margin);

panel4.Size = new Size(Width - summary\_margin - component\_margin, Height - bottom\_margin - component\_margin);

panel4.Location = new Point(0, 0);

panel5.Size = new Size(Width - summary\_margin - component\_margin, Height - bottom\_margin - component\_margin);

panel5.Location = new System.Drawing.Point(panel4.Width, 0);

panel6.Size = new Size(Width - summary\_margin - component\_margin, Height - bottom\_margin - component\_margin);

panel6.Location = new System.Drawing.Point(panel4.Width \* 2, 0);

message\_view\_dgv.Size = new Size(Width - summary\_margin - component\_margin, panel4.Height - 50 - 25);

message\_view\_dgv.Location = new System.Drawing.Point(0, 25);

autosend\_dgv.Size = new Size(Width - summary\_margin - component\_margin, panel4.Height - 50 - 25);

autosend\_dgv.Location = new System.Drawing.Point(0, 25);

send\_message\_text\_box.Size = new Size(Width - summary\_margin - component\_margin - 20 - 46, 40);

send\_message\_text\_box.Location = new System.Drawing.Point(10, panel4.Height - 45);

functional\_send\_panel.Size = new Size(40, 40);

functional\_send\_panel.Location = new Point(panel4.Width - 50, panel4.Height - 45);

autosend\_textbox.Size = new Size(Width - summary\_margin - component\_margin - 20, 40);

autosend\_textbox.Location = new System.Drawing.Point(10, panel4.Height - 45);

global\_name\_textbox.Size = new Size(panel1.Width - 10, 27);

address\_textbox.Size = new Size(panel1.Width - 10, 27);

open\_key\_textbox.Size = new Size(panel1.Width - 10, 27);

button1.Size = new Size(Convert.ToInt32(panel1.Width / 2) - 10, 27);

button1.Location = new Point(2, 57 + 28);

button2.Size = new Size(Convert.ToInt32(panel1.Width / 2) - 10, 27);

button2.Location = new Point(button1.Width + 12, 57 + 28);

Node\_add\_Button.Size = new Size(Convert.ToInt32(panel1.Width / 2) - 10, 27);

Node\_add\_Button.Location = new Point(button1.Width + 12, 57 + 28);

form\_close\_panel.Location = new System.Drawing.Point(Width - 26 - component\_margin, 5);

fullscrin\_form\_panel.Location = new System.Drawing.Point(Width - 26 - component\_margin - 31, 5);

trey\_form\_panel.Location = new System.Drawing.Point(Width - 26 - component\_margin - (31 \* 2), 5);

timer\_label.Location = new System.Drawing.Point(Width - 20 - component\_margin - (31 \* 3), 4);

label1.Width = panel5.Width - 15;

label1.Location = new Point(0, 0);

label2.Width = panel5.Width - 15;

label2.Location = new Point(0, 0);

panel3.HorizontalScroll.Value = 0;

}

private void create\_table(string name)

{

if (!sqlite.Check\_table\_exist(name))

{

sqlite.ExecuteRequest("CREATE TABLE '" + name + "' (" +

"id INTEGER PRIMARY KEY AUTOINCREMENT, " +

"sender VARCHAR(30), " +

"date VARCHAR(30), " +

"data TEXT, " +

"sign TEXT )");

}

}

//==================================================================================

private void Form1\_FormClosing(object sender, FormClosingEventArgs e)

{

serialize\_tree\_view();

//stop receiving

File.Delete(@"./components/treeView/tree.xml");

receive.Abort();

receive.Join(500);

}

private void connect\_result(bool con, bool first)

{

if (first)

{

if (con)

{

//Create table in DataBase if not exist

create\_table(global\_name\_textbox.Text);

Action act = new Action(() =>

{

void final()

{

current\_connection = address\_textbox.Text;

address\_textbox.Text = Lang[15];

address\_textbox.ForeColor = Color.Silver;

global\_name\_textbox.Text = Lang[16];

global\_name\_textbox.ForeColor = Color.Silver;

open\_key\_textbox.Text = "Публичный ключ";

open\_key\_textbox.ForeColor = Color.Silver;

}

void new\_con()

{

TreeNode parent\_node = get\_parent\_node(global\_name\_textbox.Text); //<==

TreeNode con\_node = new TreeNode(address\_textbox.Text, 2, 2)

{

Tag = address\_textbox.Text

};

parent\_node.Nodes.Add(con\_node);

}

void new\_user()

{

TreeNode parent\_node = new TreeNode(get\_name\_by\_global\_name(global\_name\_textbox.Text), 3, 3)

{

Tag = global\_name\_textbox.Text

};

treeView1.Nodes.Insert(0, parent\_node);

}

//Check user existing in contact book

if (check\_on\_match\_in\_treeView(global\_name\_textbox.Text))

{

//If user exist

//Check IP Address in exisiting user note

//If Address exist

if (check\_on\_match\_in\_treeView(address\_textbox.Text))

{

//End cheking. Created nothing

final();

}

//If Address not exist in note

else

{

//Add new address in existing user note

new\_con();

final();

}

}

//If user not exist

else

{

//Add new user

new\_user();

//Add new address in new user

new\_con();

final();

}

//==============================================================

send\_message\_text\_box.Text = "";

send\_message\_text\_box.Enabled = true;

string text;

string key = public\_key;

decimal d = Convert.ToDecimal(key.Length) / 100;

for (int i = 0; i < Math.Ceiling(d); i++)

{

try

{

if (key.Length > 99)

{

text = key.Substring(0, 99);

key = key.Substring(99);

}

else

{

text = key;

}

if (i == 0)

{

motoko.crypted\_send\_message(text, "5");

}

else

{

motoko.crypted\_send\_message(text, "3");

}

}

catch (Exception ex)

{

MessageBox.Show(ex.ToString());

}

}

motoko.crypted\_send\_message("`" + my\_global\_name + "`#", "3");

motoko.crypted\_send\_message(motoko.CallBackIPAddress + ":" + motoko.CallBackPort, "0");

});

//==================================================

if (InvokeRequired)

Invoke(act);

else

act();

}

else

{

Action act = new Action(() =>

{

send\_message\_text\_box.Enabled = false;

send\_message\_text\_box.Text = "Пользователь вне сети..";

});

if (InvokeRequired)

Invoke(act);

else

act();

}

}

else

{

Action con\_action = new Action(() =>

{

if (con)

{

send\_message\_text\_box.Text = "";

send\_message\_text\_box.Enabled = true;

}

else

{

send\_message\_text\_box.Enabled = false;

send\_message\_text\_box.Text = "Пользователь вне сети..";

}

});

if (!group)

if (InvokeRequired)

Invoke(con\_action);

else

con\_action();

}

}

private void display\_code\_zero(string message)

{

Action act = new Action(() =>

{

string global\_name = message.Split('`')[2];

string ip = message.Split('`')[1];

TreeNode parent\_node = get\_parent\_node(global\_name);

if (parent\_node == null)

{

TreeNode node\_ = new TreeNode(get\_name\_by\_global\_name(global\_name), 3, 3)

{

Tag = global\_name

};

treeView1.Nodes.Add(node\_);

parent\_node = get\_parent\_node(global\_name);

TreeNode new\_node = new TreeNode(ip, 2, 2)

{

Tag = ip

};

if (InvokeRequired)

Invoke(new Action(() =>

{

parent\_node.ImageIndex = 3;

parent\_node.Nodes.Add(new\_node);

parent\_node.SelectedImageIndex = parent\_node.ImageIndex;

}));

else

{

parent\_node.ImageIndex = 3;

parent\_node.Nodes.Add(new\_node);

parent\_node.SelectedImageIndex = parent\_node.ImageIndex;

}

}

else

{

bool exist = false;

foreach (TreeNode node in parent\_node.Nodes)

{

if (node.Tag != null)

if (node.Tag.ToString() == ip)

{

exist = true;

break;

}

}

if (!exist)

{

TreeNode new\_node = new TreeNode(ip, 2, 2)

{

Tag = ip

};

//new\_node.SelectedImageIndex = new\_node.ImageIndex;

if (InvokeRequired)

Invoke(new Action(() => parent\_node.Nodes.Add(new\_node)));

else

parent\_node.Nodes.Add(new\_node);

}

}

create\_table(global\_name);

DataSet ds = sqlite.Select\_DataSet("SELECT open\_key FROM Keys WHERE global\_name LIKE '" + global\_name + "'");

if (ds.Tables[0].Rows.Count == 0)

{

motoko.remoteEP = new IPEndPoint(IPAddress.Parse(ip.Split(':')[0]), Convert.ToInt32(ip.Split(':')[1]));

string key = get\_public\_key\_by\_blobal\_name(global\_name);

motoko.user\_public\_key = key;

if (key != null)

{

motoko.connection();

\_connectionDone.WaitOne();

motoko.crypted\_send\_message(motoko.CallBackIPAddress + ":" +

motoko.CallBackPort + "`" + my\_global\_name, "4");

}

}

});

if (InvokeRequired)

Invoke(act);

else

act();

} //0

private void display\_code\_second(string message)

{

Action act = new Action(() =>

{

string address = message.Split('`')[1];

//MessageBox.Show("=> " + message);

string data = message.Split('`')[2];

string global\_name = message.Split('`')[3];

create\_table(global\_name);

sqlite.ExecuteRequest("INSERT INTO '" + global\_name + "' " +

"VALUES (NULL, '" + address + "', '" + DateTime.Now + "','" + data + "', NULL)");

if (current\_connection == address)

{

DataSet ds = sqlite.Select\_DataSet("SELECT id FROM '" + global\_name + "' ORDER BY id DESC LIMIT 1");

object[] row = new object[5];

if (ds != null)

row[0] = ds.Tables[0].Rows[0].ItemArray[0].ToString();

else

row[0] = "1";

row[1] = address;

string date = DateTime.Now.ToString().Split(' ')[1];

row[2] = date.Split(':')[0] + ":" + date.Split(':')[1];

row[3] = data;

//if (dataGridView1.Rows[dataGridView1.Rows.Count - 1].Cells[1])

message\_view\_dgv.Rows.Insert(message\_view\_dgv.Rows.Count, row);

message\_view\_dgv.Rows[message\_view\_dgv.Rows.Count - 1].Cells[3].Style.Alignment =

DataGridViewContentAlignment.MiddleLeft;

message\_view\_dgv.FirstDisplayedScrollingRowIndex = message\_view\_dgv.RowCount - 1;

}

else

{

mark\_node(global\_name, address);

}

});

if (InvokeRequired)

this.Invoke(act);

else

act();

} //2

private void display\_code\_twenty\_first(string message)

{

Action act = new Action(() =>

{

string address = message.Split('`')[1];

string data = message.Split('`')[2];

string sign = message.Split('#')[1];

string global\_name = message.Split('`')[message.Split('`').Length - 2];

create\_table(global\_name);

sqlite.ExecuteRequest("INSERT INTO '" + global\_name + "' " +

"VALUES (NULL, '" + address + "', '" + DateTime.Now + "','" + data + "', '" + sign + "')");

if (current\_connection == address)

{

DataSet ds = sqlite.Select\_DataSet("SELECT id FROM '" + global\_name + "' ORDER BY id DESC LIMIT 1");

object[] row = new object[5];

if (ds != null)

row[0] = ds.Tables[0].Rows[0].ItemArray[0].ToString();

else

row[0] = "1";

row[1] = address;

string date = DateTime.Now.ToString().Split(' ')[1];

row[2] = date.Split(':')[0] + ":" + date.Split(':')[1];

row[3] = data;

row[4] = sign;

//if (dataGridView1.Rows[dataGridView1.Rows.Count - 1].Cells[1])

message\_view\_dgv.Rows.Insert(message\_view\_dgv.Rows.Count, row);

message\_view\_dgv.Rows[message\_view\_dgv.Rows.Count - 1].Cells[3].Style.Alignment =

DataGridViewContentAlignment.MiddleLeft;

message\_view\_dgv.Rows[message\_view\_dgv.Rows.Count - 1].Cells[3].Style.Font =

message\_view\_dgv.RowsDefaultCellStyle.Font;

message\_view\_dgv.Rows[message\_view\_dgv.Rows.Count - 1].Cells[3].Style.ForeColor =

Color.FromArgb(120, 198, 200);

message\_view\_dgv.FirstDisplayedScrollingRowIndex = message\_view\_dgv.RowCount - 1;

}

else

{

mark\_node(global\_name, address);

}

});

if (InvokeRequired)

this.Invoke(act);

else

act();

} // 21

private TreeNode search\_existing\_groups(string node\_tag)

{

TreeNode check\_in(TreeNode node)

{

foreach (TreeNode \_node in node.Nodes)

{

if (\_node.Tag != null)

if (\_node.Tag.ToString() == node\_tag)

return \_node;

if (\_node.Nodes.Count > 0)

{

TreeNode node\_ = check\_in(\_node);

if (node\_ != null)

return node\_;

}

}

return null;

}

foreach (TreeNode node in treeView1.Nodes)

{

if (node.Tag != null)

if (node.Tag.ToString() == node\_tag)

return node;

if (node.Nodes.Count > 0)

{

TreeNode node\_ = check\_in(node);

if (node\_ != null)

return node\_;

}

}

return null;

}

private TreeNode searching\_node\_in\_node\_container(TreeNode node, string node\_tag)

{

foreach (TreeNode \_node in node.Nodes)

{

if (\_node.Tag.ToString() == node\_tag)

{

return \_node;

}

}

return null;

}

private void create\_user(TreeNode group\_node, string global\_name)

{

Action act = new Action(() =>

{

TreeNode user\_node = new TreeNode(get\_name\_by\_global\_name(global\_name), 3, 3)

{

Tag = global\_name

};

group\_node.Nodes.Add(user\_node);

if (!sqlite.Check\_table\_exist(global\_name))

{

sqlite.ExecuteRequest("CREATE TABLE '" + global\_name + "'(" +

"id INTEGER PRIMARY KEY AUTOINCREMENT, " +

"sender VARCHAR(30), " +

"date VARCHAR(30), " +

"data TEXT, " +

"sign TEXT)");

}

});

if (InvokeRequired) Invoke(act);

else act();

}

private void create\_group(string group\_name)

{

Action act = new Action(() =>

{

TreeNode group\_node = new TreeNode(group\_name, 4, 4)

{

Tag = group\_name

};

treeView1.Nodes.Add(group\_node);

if (!sqlite.Check\_table\_exist(group\_name))

{

sqlite.ExecuteRequest("CREATE TABLE '" + group\_name + "'(" +

"id INTEGER PRIMARY KEY AUTOINCREMENT, " +

"sender VARCHAR(65), " +

"date VARCHAR(30), " +

"data TEXT, " +

"sign TEXT)");

}

});

if (InvokeRequired) Invoke(act);

else act();

}

private void create\_address(TreeNode user\_node, string address)

{

Action act = new Action(() =>

{

TreeNode group\_node = new TreeNode(address, 2, 2)

{

Tag = address

};

user\_node.Nodes.Add(group\_node);

});

if (InvokeRequired) Invoke(act);

else act();

}

private void display\_code\_forth(string message) //4

{

string group = message.Split('`')[1];

string IP = message.Split('`')[2].Split(':')[0];

string port = message.Split('`')[2].Split(':')[1];

string msg = message.Split('`')[3];

string global\_name = message.Split('`')[4];

string user\_custom\_name = get\_name\_by\_global\_name(global\_name);

TreeNode node = search\_existing\_groups(group);

TreeNode user;

TreeNode address;

if (node != null)

{

user = searching\_node\_in\_node\_container(node, global\_name);

if (user != null)

{

address = searching\_node\_in\_node\_container(user, IP + ":" + port);

if (address != null)

{

;

}

else

{

create\_address(user, IP + ":" + port);

}

}

//else

//{

// create\_user(node, global\_name);

// user = searching\_node\_in\_node\_container(node, global\_name);

// create\_address(user, IP + ":" + port);

//}

}

else

{

create\_group(group);

node = search\_existing\_groups(group);

create\_user(node, global\_name);

user = searching\_node\_in\_node\_container(node, global\_name);

create\_address(user, IP + ":" + port);

}

node = search\_existing\_groups(group);

if (searching\_node\_in\_node\_container(node, global\_name) != null)

{

sqlite.ExecuteRequest("INSERT INTO `" + group + "` VALUES (NULL, '" + global\_name +

"', '" + DateTime.Now + "', '" + msg + "', NULL)");

}

if (current\_connection == group)

{

DataSet ds = sqlite.Select\_DataSet("SELECT id FROM '" + global\_name + "' ORDER BY id DESC LIMIT 1");

object[] row = new object[5];

if (ds != null)

row[0] = ds.Tables[0].Rows[0].ItemArray[0].ToString();

else

row[0] = "1";

row[1] = global\_name;

string date = DateTime.Now.ToString().Split(' ')[1];

row[2] = date.Split(':')[0] + ":" + date.Split(':')[1];

row[3] = msg;

//if (dataGridView1.Rows[dataGridView1.Rows.Count - 1].Cells[1])

message\_view\_dgv.Rows.Insert(message\_view\_dgv.Rows.Count, row);

message\_view\_dgv.Rows[message\_view\_dgv.Rows.Count - 1].Cells[3].Style.Alignment =

DataGridViewContentAlignment.MiddleLeft;

message\_view\_dgv.FirstDisplayedScrollingRowIndex = message\_view\_dgv.RowCount - 1;

}

else

{

;//Marking node group

}

TreeNode[] treeNodes = new TreeNode[0];

foreach (TreeNode \_node in node.Nodes)

{

if (\_node.Nodes.Count > 0)

{

check\_in(\_node, ref treeNodes, 2);

}

}

foreach (TreeNode \_node in treeNodes)

{

if (\_node.Tag.ToString() != (IP + ":" + port))

{

string ip = \_node.Tag.ToString().Split(':')[0];

int port\_ = Convert.ToInt32(\_node.Tag.ToString().Split(':')[1]);

motoko.remoteEP = new IPEndPoint(IPAddress.Parse(ip), port\_);

motoko.user\_public\_key = get\_public\_key\_by\_blobal\_name(\_node.Parent.Tag.ToString());

//send\_message\_text\_box.Text = "Подключение...";

//send\_message\_text\_box.Enabled = false;

motoko.connection();

Thread.Sleep(200);

//send\_message\_text\_box.Text = "Отправка...";

//send\_message\_text\_box.Enabled = false;

if (sending\_to\_group(msg, my\_global\_name, current\_dialog))

{

//there u can handle every sended message

}

//send\_message\_text\_box.Text = "";

//send\_message\_text\_box.Enabled = true;

}

}

}

private void display\_code\_fifth(string message)

{

string user\_public\_key = message.Split('`')[1] + "";

string global\_name = message.Split('`')[2] + "";

DataSet ds = sqlite.Select\_DataSet("SELECT open\_key FROM Keys WHERE global\_name LIKE '" + global\_name + "'");

if (ds.Tables[0].Rows.Count == 0)

{

sqlite.ExecuteRequest("INSERT INTO Keys VALUES (NULL, '" + global\_name + "', '" + user\_public\_key + "')");

}

} //5

private void display\_code\_sixth(string message) // 6

{

string IP = message.Split('`')[1].Split(':')[0];

string port = message.Split('`')[1].Split(':')[1];

string file\_name = message.Split('`')[2];

//if (InvokeRequired) Invoke(new Action(() => textBox3.Text = message.Split('`')[3])); else textBox3.Text = message.Split('`')[3];

byte[] file = Convert.FromBase64String(message.Split('`')[3]);

string global\_name = message.Split('`')[4];

MessageBox.Show("Address is " + IP + ":" + port + "\n" +

"file name is " + file\_name + "\n" +

"file size is " + file.Length + "\n" +

"global name is " + global\_name

);

File.WriteAllBytes("./files/" + file\_name, file);

}

private void Display(string message)

{

if (message.Split('`')[0] == "0")

{

display\_code\_zero(message);

}

else if (message.Split('`')[0] == "2")

{

display\_code\_second(message);

}

else if (message.Split('`')[0] == "21")

{

display\_code\_twenty\_first(message);

}

else if (message.Split('`')[0] == "4")

{

display\_code\_forth(message);

}

else if (message.Split('`')[0] == "5")

{

display\_code\_fifth(message);

}

else if (message.Split('`')[0] == "6")

{

display\_code\_sixth(message);

}

Action act = new Action(() =>

{

label8.Text = "0";

});

if (InvokeRequired)

Invoke(act);

else act();

}

private void textBox1\_KeyDown(object sender, KeyEventArgs e)

{

if (group)

{

send\_to\_group(e);

}

else

send\_to\_user(e);

}

//===================================[Buttons]======================================

private void button1\_Click(object sender, EventArgs e)

{

if (!conWindowAnimation.IsAlive)

{

if (treeView1.Location.Y > 55 + 26)

{

add\_user\_panel.BackgroundImage = Image.FromFile(@"./images/ico/treeView/user.png");

conWindowAnimation = new Thread(new ThreadStart(conWinAnimation));

conWindowAnimation.Start();

}

}

}

private string get\_name\_by\_global\_name(string global\_name)

{

DataSet ds = sqlite.Select\_DataSet("SELECT name FROM names WHERE global\_name LIKE '" + global\_name + "'");

if (ds.Tables[0] != null & ds.Tables[0].Rows.Count > 0)

{

return ds.Tables[0].Rows[0].ItemArray[0].ToString();

}

return global\_name;

}

private void Node\_add\_Button\_Click(object sender, EventArgs e)

{

void method()

{

try

{

IPAddress ip = IPAddress.Parse(address\_textbox.Text.Split(':')[0]);

int port = Int32.Parse(address\_textbox.Text.Split(':')[1]);

//Bing user address

motoko.remoteEP = new IPEndPoint(ip, port);

motoko.global\_name = global\_name\_textbox.Text;

try

{

DataSet ds = sqlite.Select\_DataSet("SELECT open\_key FROM Keys WHERE global\_name LIKE '" + global\_name\_textbox.Text + "'");

if (ds.Tables[0].Rows.Count == 0)

{

sqlite.ExecuteRequest("INSERT INTO Keys VALUES (NULL, '" + global\_name\_textbox.Text + "', '" + open\_key\_textbox.Text + "')");

}

motoko.user\_public\_key = get\_public\_key\_by\_blobal\_name(global\_name\_textbox.Text);

//Connecting to user

motoko.first\_connection();

}

catch (Exception ex)

{

MessageBox.Show(ex.ToString() + "\n\n" + ex.Message);

}

}

catch

{

//INCORRECT IP

}

}

if (global\_name\_textbox.Text.Length == 64 && address\_textbox.Text != "" && open\_key\_textbox.Text != "")

{

send\_message\_text\_box.Text = "Подключение...";

send\_message\_text\_box.Enabled = false;

conn = new Thread(new ThreadStart(method));

//Add new address in list (treeView1) by button

if (motoko.CallBackIPAddress == null)

conn.Start();

else if (motoko.CallBackIPAddress.Address.ToString() != address\_textbox.Text)

conn.Start();

}

}

//==========================[TreeView ContextMenu Events]===========================

//////===============================[Animation]=========================

private void conWinAnimation()

{

int i = 0;

int height = panel1.Height;

Action act\_show = new Action(() =>

{

treeView1.Height -= 1;

treeView1.Top += 1;

});

Action act\_hide = new Action(() =>

{

treeView1.Height += 1;

treeView1.Top -= 1;

});

if (treeView1.Location.Y == 55 + 26)

{

for (i = 11; i <= 90 + 33 + win\_anim\_offset; i++)

{

try

{

if (InvokeRequired)

this.Invoke(act\_show);

else

act\_show();

Thread.Sleep(10);

}

catch (ObjectDisposedException)

{

break;

}

}

}

else

{

for (i = 90 + 33 + win\_anim\_offset; i > 10; i--)

{

try

{

if (InvokeRequired)

this.Invoke(act\_hide);

else

act\_hide();

Thread.Sleep(10);

}

catch

{

break;

}

}

}

//conWindowAnimation.Abort();

}

private void new\_connection\_window\_showing(object sender, EventArgs e)

{

if (!conWindowAnimation.IsAlive)

{

if (treeView1.Location.Y == 55 + 26)

{

Node\_add\_Button.Visible = true;

address\_textbox.Visible = true;

button2.Visible = false;

open\_key\_textbox.Visible = true;

address\_textbox.Text = Lang[15];

address\_textbox.ForeColor = Color.Silver;

global\_name\_textbox.Text = Lang[16];

global\_name\_textbox.ForeColor = Color.Silver;

open\_key\_textbox.Text = "Публичный ключ";

open\_key\_textbox.ForeColor = Color.Silver;

add\_user\_panel.BackgroundImage = Image.FromFile(@"./images/ico/treeView/user\_press.png");

conWindowAnimation = new Thread(new ThreadStart(conWinAnimation));

conWindowAnimation.Start();

}

else

{

add\_user\_panel.BackgroundImage = Image.FromFile(@"./images/ico/treeView/user.png");

conWindowAnimation = new Thread(new ThreadStart(conWinAnimation));

conWindowAnimation.Start();

}

}

}

private void rename\_node\_window\_showing(object sender, EventArgs e)

{

if (treeView1.Location.Y == 55 + 26)

{

renaming\_node = treeView1.SelectedNode;

global\_name\_textbox.Text = renaming\_node.Text;

global\_name\_textbox.ForeColor = Color.Black;//Color.FromArgb(120, 198, 200);

if (renaming\_node.ImageIndex == 3) //user

{

address\_textbox.Text = renaming\_node.Tag.ToString();

DataSet ds = sqlite.Select\_DataSet("SELECT open\_key FROM Keys WHERE global\_name LIKE '" + global\_name\_textbox.Text + "'");

if (ds.Tables[0].Rows.Count > 0)

open\_key\_textbox.Text = ds.Tables[0].Rows[0].ItemArray[0].ToString();

address\_textbox.Visible = true;

address\_textbox.ReadOnly = true;

open\_key\_textbox.Visible = true;

open\_key\_textbox.ReadOnly = true;

button1.Location = new Point(2, 85); // 85

button2.Location = new Point(button1.Width + 12, 85);

Node\_add\_Button.Location = new Point(button1.Width + 12, 85);

win\_anim\_offset = 0;

}

else if (renaming\_node.ImageIndex == 2 || renaming\_node.ImageIndex == 4) //address

{

address\_textbox.Text = renaming\_node.Tag.ToString();

address\_textbox.Visible = true;

address\_textbox.ReadOnly = true;

open\_key\_textbox.Visible = false;

button1.Location = new Point(2, 85 - 26); // 85

button2.Location = new Point(button1.Width + 12, 85 - 26);

Node\_add\_Button.Location = new Point(button1.Width + 12, 85 - 26);

win\_anim\_offset = -26;

}

else if (renaming\_node.ImageIndex == 1) // folder

{

address\_textbox.Visible = false;

open\_key\_textbox.Visible = false;

button1.Location = new Point(2, 85 - 50); // 85

button2.Location = new Point(button1.Width + 12, 85 - 50);

Node\_add\_Button.Location = new Point(button1.Width + 12, 85 - 50);

win\_anim\_offset = -50;

}

Node\_add\_Button.Visible = false;

button2.Visible = true;

conWindowAnimation = new Thread(new ThreadStart(conWinAnimation));

conWindowAnimation.Start();

}

else

{

conWindowAnimation = new Thread(new ThreadStart(conWinAnimation));

conWindowAnimation.Start();

}

}

//////===================================================================

private void Add\_MenuItem\_click(object sender, EventArgs e)

{

if (treeView1.SelectedNode == null)

{

treeView1.Nodes.Add("0", Lang[14], 1, 1);

}

else if (treeView1.SelectedNode.ImageIndex == 1)

{

TreeNode newNode = new TreeNode(Lang[14], 1, 1);

treeView1.SelectedNode.Nodes.Add(newNode);

treeView1.SelectedNode.Expand();

}

}

private void rename\_node\_button\_click(object sender, EventArgs e)

{

if (global\_name\_textbox.Text != "")

{

renaming\_node.Text = global\_name\_textbox.Text;

conWindowAnimation = new Thread(new ThreadStart(conWinAnimation));

conWindowAnimation.Start();

if (renaming\_node.Tag != null)

{

string global\_name = renaming\_node.Tag.ToString();

DataSet ds = sqlite.Select\_DataSet("SELECT id FROM names WHERE global\_name LIKE '" + global\_name + "'");

if (ds.Tables[0].Rows != null & ds.Tables[0].Rows.Count > 0)

{

string id = ds.Tables[0].Rows[0].ItemArray[0].ToString();

sqlite.ExecuteRequest("UPDATE names SET name = '" + renaming\_node.Text + "' WHERE id LIKE " + id + " ");

}

else

{

sqlite.ExecuteRequest("INSERT INTO names VALUES (NULL, '" + renaming\_node.Tag + "', '" + renaming\_node.Text + "')");

}

}

address\_textbox.Text = Lang[15];

address\_textbox.ForeColor = Color.Silver;

global\_name\_textbox.Text = Lang[16];

global\_name\_textbox.ForeColor = Color.Silver;

}

}

private void Delete\_all\_MenuItem\_click(object sender, EventArgs e)

{

if (treeView1.SelectedNode != null)

treeView1.SelectedNode.Nodes.Clear();

treeView1.Nodes.Remove(treeView1.SelectedNode);

}

private void Delete\_MenuItem\_click(object sender, EventArgs e)

{

treeView1.SelectedNode.Nodes.Clear();

}

//==========================[TreeView additional methods]===========================

delegate void TreeViewCallBack(string ip);

public void TreeViewNodes(string ip)

{

if (InvokeRequired) // Вызов метода из другого потока

{

TreeViewCallBack callBack = new TreeViewCallBack(TreeViewNodes);

Invoke(callBack, new object[] { ip });

}

else // Вызов метода из текущего потока

{

TreeNode node = new TreeNode(ip, 2, 2)

{

Tag = ip

};

treeView1.Nodes.Add(node);

}

}

private void treeView1\_NodeMouseClick(object sender, TreeNodeMouseClickEventArgs e)

{

if (e.Button == MouseButtons.Left)

{

if (e.Node.ImageIndex == 1 ||

e.Node.ImageIndex == 3 ||

e.Node.ImageIndex == 4)

{

if (e.Node.IsExpanded)

e.Node.Collapse();

else

e.Node.Expand();

}

else if (e.Node.ImageIndex == 2)

{

label1.Text = e.Node.Text;

autosend\_textbox.Enabled = true;

}

}

if (e.Button == MouseButtons.Right)

{

ContextMenu m = new ContextMenu();

TreeNode node = treeView1.HitTest(e.X, e.Y).Node;

if (node.ImageIndex == 1)

{

m.MenuItems.Add(Lang[10]);

m.MenuItems.Add(Lang[11]);

m.MenuItems.Add(Lang[12]);

m.MenuItems.Add(Lang[13]);

m.MenuItems[0].Click += new EventHandler(Add\_MenuItem\_click);

m.MenuItems[1].Click += new EventHandler(rename\_node\_window\_showing);

m.MenuItems[2].Click += new EventHandler(Delete\_all\_MenuItem\_click);

m.MenuItems[3].Click += new EventHandler(Delete\_MenuItem\_click);

}

else if (node.ImageIndex == 2)

{

m.MenuItems.Add(Lang[11]);

m.MenuItems.Add(Lang[12]);

m.MenuItems[0].Click += new EventHandler(rename\_node\_window\_showing);

m.MenuItems[1].Click += new EventHandler(Delete\_all\_MenuItem\_click);

}

else if (node.ImageIndex == 3 || node.ImageIndex == 4)

{

m.MenuItems.Add("Перейти в диалог");

m.MenuItems.Add(Lang[11]);

m.MenuItems.Add(Lang[12]);

m.MenuItems.Add(Lang[13]);

m.MenuItems[0].Click += new EventHandler(show\_dialog);

m.MenuItems[1].Click += new EventHandler(rename\_node\_window\_showing);

m.MenuItems[2].Click += new EventHandler(Delete\_all\_MenuItem\_click);

m.MenuItems[3].Click += new EventHandler(Delete\_MenuItem\_click);

}

m.Show(treeView1, new Point(e.X, e.Y));

treeView1.SelectedNode = node;

}

e.Node.BackColor = CustomBackColor;

}

private void treeView1\_KeyPress(object sender, KeyPressEventArgs e)

{

////Connection by Enter on node

//if (current\_slider == 0)

// if (e.KeyChar == (char)13)

// {

// if (treeView1.SelectedNode != null)

// if (treeView1.SelectedNode.ImageIndex.ToString() == "2")

// {

// motoko.remoteEP = new IPEndPoint(

// IPAddress.Parse(treeView1.SelectedNode.Tag

// .ToString().Split(':')[0]),

// Int32.Parse(treeView1.SelectedNode.Tag

// .ToString().Split(':')[1])

// );

// motoko.connection();

// current\_connection = motoko.remoteEP.Address.ToString() + ":" + motoko.remoteEP.Port.ToString();

// }

// }

}

private void mark\_node(string global\_name, string value)

{

void method(TreeNode \_node)

{

\_node.BackColor = Color.FromArgb(120, 198, 200);

if (\_node.Parent != null)

method(\_node.Parent);

}

TreeNode node = get\_parent\_node(global\_name);

foreach (TreeNode n in node.Nodes)

{

if (n.Tag != null && n.Tag.ToString() == value)

n.BackColor = Color.FromArgb(120, 198, 200);

}

method(node);

}

private TreeNode get\_parent\_node(string global\_name)

{

TreeNode method(TreeNode \_node\_)

{

TreeNode \_\_node\_\_;

foreach (TreeNode \_node in \_node\_.Nodes)

{

if (\_node.Tag != null)

if (\_node.Tag.ToString() == global\_name)

{

return \_node;

}

if (\_node.Nodes.Count > 0)

{

\_\_node\_\_ = method(\_node);

if (\_\_node\_\_ != null)

return \_\_node\_\_;

}

}

return null;

}

//=====================================

TreeNode \_\_node;

foreach (TreeNode node in treeView1.Nodes)

{

if (node.Tag != null)

if (node.Tag.ToString() == global\_name)

{

return node;

}

if (node.Nodes.Count > 0)

{

\_\_node = method(node);

if (\_\_node != null)

return \_\_node;

}

}

return null;

}

int message\_offset = 0;

private void treeView1\_NodeMouseDoubleClick(object sender, TreeNodeMouseClickEventArgs e)

{

//connection to node by click on node in treeview

void method()

{

if (e.Node.ImageIndex == 2)

{

IPAddress ip = IPAddress.Parse(e.Node.Tag.ToString().Split(':')[0]);

int port = Int32.Parse(e.Node.Tag.ToString().Split(':')[1]);

motoko.remoteEP = new IPEndPoint(ip, port);

string global\_name = null;

try

{

//Добавить метод о начале создания подключения

Action act\_con = new Action(() =>

{

send\_message\_text\_box.Enabled = false;

send\_message\_text\_box.Text = "Подключение...";

});

if (InvokeRequired)

Invoke(act\_con);

else

act\_con();

Action act = new Action(() =>

{

current\_connection = e.Node.Tag.ToString();

if (e.Node.Parent.ImageIndex == 3)

{

global\_name = e.Node.Parent.Tag.ToString();

}

if (global\_name != null)

if (current\_dialog != global\_name)

{

treeView1.SelectedNode = e.Node.Parent;

show\_dialog(new object(), new EventArgs());

current\_dialog = global\_name;

}

label2.Text = e.Node.Tag.ToString();

e.Node.BackColor = CustomBackColor;

});

if (InvokeRequired)

Invoke(act);

else

act();

DataSet ds = sqlite.Select\_DataSet("SELECT open\_key FROM Keys WHERE global\_name LIKE '" + global\_name + "'");

motoko.user\_public\_key = ds.Tables[0].Rows[0].ItemArray[0].ToString();

//Подключение

motoko.connection();

conn.Join(100);

}

catch (Exception ex)

{

MessageBox.Show(ex.ToString());

conn.Join(100);

//conn.Abort();

}

}

}

if (current\_slider == 0)

if (e.Button == MouseButtons.Left)

{

if (treeView1.SelectedNode != null && e.Node.ImageIndex == 2)

{

try

{

conn = new Thread(new ThreadStart(method));

sign\_message\_panel.BackgroundImage = Image.FromFile(@"./images/pencil.png");

isSign = false;

if (motoko.remoteEP == null)

conn.Start();

else if (motoko.remoteEP.Address.ToString() != address\_textbox.Text)

conn.Start();

}

catch (ArgumentOutOfRangeException ex)

{

MessageBox.Show(ex.Message);

}

}

}

}

private bool check\_on\_match\_in\_treeView(string value)

{

bool check\_node(TreeNode node)

{

for (int i = 0; i < node.Nodes.Count; i++)

{

if (node.Nodes[i].ImageIndex != 4)

if (node.Nodes[i].Tag == null)

{

if (check\_node(node.Nodes[i]))

return true;

}

else if (node.Nodes[i].Nodes.Count > 0)

{

if (node.Nodes[i].Tag.ToString() == value)

return true;

else if (check\_node(node.Nodes[i]))

return true;

}

else

if (node.Nodes[i].Tag.ToString() == value)

return true;

}

return false;

}

foreach (TreeNode node in treeView1.Nodes)

{

if (node.ImageIndex != 4)

if (node.Tag == null)

{

if (check\_node(node))

return true;

}

else if (node.Nodes.Count > 0)

{

if (node.Tag.ToString() == value)

return true;

else if (check\_node(node))

return true;

}

else

if (node.Tag.ToString() == value)

return true;

}

return false;

}

private void serialize\_tree\_view()

{

TreeViewSerializer serializer = new TreeViewSerializer();

serializer.SerializeTreeView(this.treeView1, "./components/treeView/tree.xml");

sqlite.Insert\_BLOB("UPDATE treeViews SET tree = @file WHERE name LIKE 'book'");

}

private void sign\_message\_panel\_Click(object sender, EventArgs e)

{

if (isSign)

{

isSign = false;

sign\_message\_panel.BackgroundImage = Image.FromFile(@"./images/pencil.png");

}

else

{

isSign = true;

sign\_message\_panel.BackgroundImage = Image.FromFile(@"./images/pencil\_press.png");

}

}

private void message\_view\_dgv\_CellMouseDoubleClick(object sender, DataGridViewCellMouseEventArgs e)

{

if (e.Button == MouseButtons.Left)

if (e.ColumnIndex == 3 && message\_view\_dgv.Rows[e.RowIndex].Cells[3].Style.ForeColor != Color.White)

{

string username = current\_dialog;

int id = Convert.ToInt32(message\_view\_dgv.Rows[e.RowIndex].Cells[0].Value.ToString());

string Key;

DataSet ds = sqlite.Select\_DataSet("SELECT \* FROM '" + username + "' WHERE id LIKE " + id + " ");

if (ds.Tables[0].Rows[0].ItemArray[1].ToString() == "")

Key = public\_key;

else

{

DataSet key\_ = sqlite.Select\_DataSet("SELECT open\_key FROM Keys WHERE global\_name LIKE '" + username + "'");

Key = key\_.Tables[0].Rows[0].ItemArray[0].ToString();

}

string sign = ds.Tables[0].Rows[0].ItemArray[4].ToString();

string message = ds.Tables[0].Rows[0].ItemArray[3].ToString();

string date = ds.Tables[0].Rows[0].ItemArray[2].ToString();

SignedMessageShow form = new SignedMessageShow(Key, username, sign, message, date);

form.Show();

}

}

private void add\_group\_panel\_Click(object sender, EventArgs e)

{

Random rnd = new Random();

int id = rnd.Next(100000000, 999999999);

while (true)

{

if (sqlite.Check\_table\_exist(id.ToString()))

id = rnd.Next(100000000, 999999999);

else

{

sqlite.ExecuteRequest("CREATE TABLE '" + id.ToString() + "'(" +

"id INTEGER PRIMARY KEY AUTOINCREMENT, " +

"sender VARCHAR(65), " +

"date VARCHAR(30), " +

"data TEXT, " +

"sign TEXT)");

break;

}

}

TreeNode node = new TreeNode(id.ToString(), 4, 4)

{

Tag = id.ToString()

};

treeView1.Nodes.Add(node);

}

private void button3\_Click(object sender, EventArgs e)

{

motoko.listener.Close();

motoko = new Motoko(local\_ip, Convert.ToInt32(textBox4.Text));

textBox2.Text = local\_ip + ":" + motoko.CallBackPort;

receive.Abort();

receive.Join(500);

receive = new Thread(new ThreadStart(motoko.StartListening));

motoko.GetMessageHandler(Display);

motoko.GetConnectionHandler(connect\_result);

autosend.GetConnectionHandler(autosend\_result);

//motoko\_as.GetConnectionHandler(autosend\_result);

receive.Start();

}

private void button4\_Click(object sender, EventArgs e)

{

string path = @"./DataBases/DataBase.db3";

File.Delete(path);

DataBase db = new DataBase();

db.create\_new\_data\_base();

DataSet ds = sqlite.Select\_DataSet("SELECT \* FROM user\_info");

my\_global\_name = ds.Tables[0].Rows[0].ItemArray[0].ToString();

public\_key = ds.Tables[0].Rows[0].ItemArray[1].ToString();

private\_key = ds.Tables[0].Rows[0].ItemArray[2].ToString();

textBox1.Text = my\_global\_name;

textBox3.Text = public\_key;

}

Thread send\_file;

private void file\_send()

{

var open = new OpenFileDialog();

Action act = new Action(() =>

{

open.ShowDialog();

});

if (InvokeRequired)

Invoke(act);

else

act();

string path = open.FileName;

string file\_name = open.SafeFileName;

byte[] file = File.ReadAllBytes(path);

string message = Convert.ToBase64String(file);

motoko.connection();

Thread.Sleep(200);

string text;

decimal d = Convert.ToDecimal(message.Length) / 100;

if (message != "")

{

for (int i = 0; i < Math.Ceiling(d + 1); i++)

{

try

{

Action act1 = new Action(() =>

{

send\_message\_text\_box.Enabled = false;

send\_message\_text\_box.Text = "Отправка файла " + i + " частей из " + Math.Round(d + 1) + ".";

});

if (InvokeRequired)

Invoke(act1);

else

act1();

Thread.Sleep(100);

if (message.Length > 99)

{

text = message.Substring(0, 99);

message = message.Substring(99);

}

else

{

text = message;

}

if (i == 0)

{

motoko.crypted\_send\_message(

motoko.CallBackIPAddress + ":" +

motoko.CallBackPort + "`" + file\_name + "`", "6");

motoko.crypted\_send\_message(text, "3");

}

else

{

motoko.crypted\_send\_message(text, "3");

}

}

catch (Exception ex)

{

MessageBox.Show(ex.ToString());

return;

}

}

motoko.crypted\_send\_message("`" + my\_global\_name + "`#", "3");

}

motoko.crypted\_send\_message(motoko.CallBackIPAddress + ":" +

motoko.CallBackPort + "`" + "Отправил файл: " + file\_name, "2");

motoko.crypted\_send\_message("`" + my\_global\_name + "`#", "3");

file\_sending.WaitOne();

Action act2 = new Action(() =>

{

send\_message\_text\_box.Text = "";

send\_message\_text\_box.Enabled = true;

});

if (InvokeRequired)

Invoke(act2);

else

act2();

}

private void file\_send\_panel\_Click(object sender, EventArgs e)

{

send\_file = new Thread(new ThreadStart(file\_send));

send\_file.Start();

}

//==================================================================================

private void increment\_pockets()

{

Action act = new Action(() =>

{

label8.Text = "" + (Convert.ToInt32(label8.Text) + 1);

});

if (InvokeRequired)

Invoke(act);

else act();

}

private void panel1\_Click(object sender, EventArgs e)

{

}

private void label5\_Click(object sender, EventArgs e)

{

textBox4.Visible = true;

button3.Visible = true;

button4.Visible = true;

}

}

}

Приложение Б

(справочное)

Экранные формы

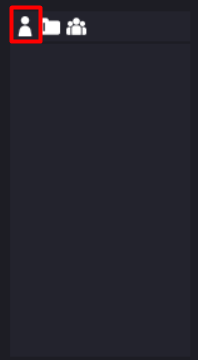


Рисунок Б.1 – Кнопка создания нового пользователя

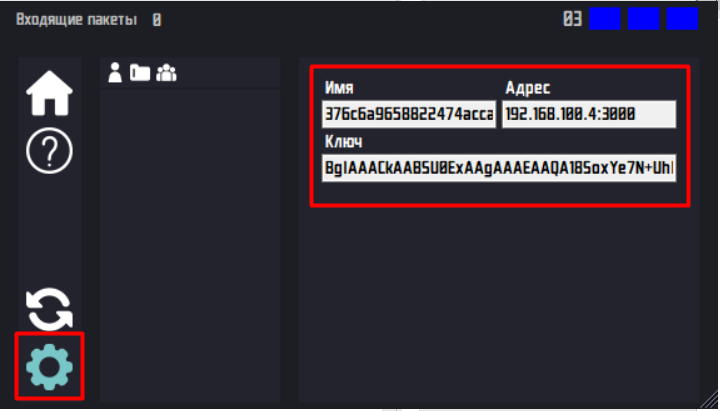


Рисунок Б.2 – Данные для подключения

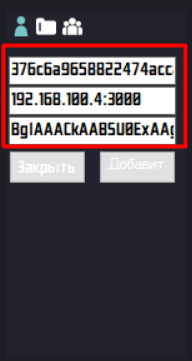


Рисунок Б.3 – Поля для ввода данных для подключения

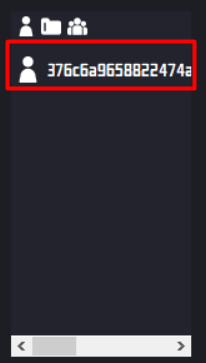


Рисунок Б.4 – Элемент «Пользователь»

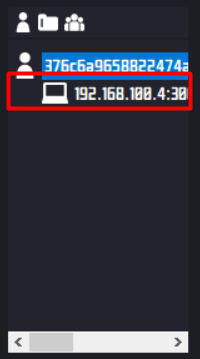


Рисунок Б.5 – Элемент «Адрес»



Рисунок Б.6 – Уведомление о недоступном пользователе

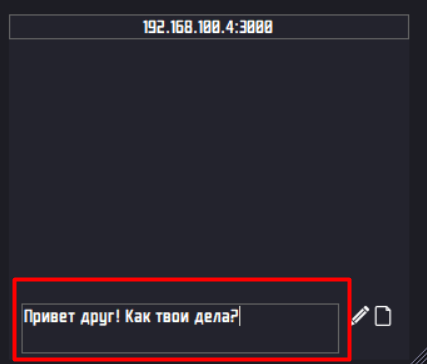


Рисунок Б.7 – Пример сообщения

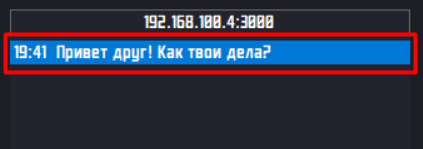


Рисунок Б.8 – Отправленное сообщение

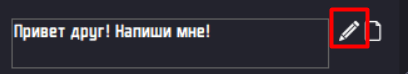


Рисунок Б.9 – Элемент для подписи сообщения

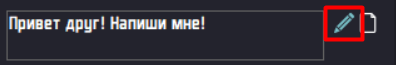


Рисунок Б.10 – Пример активированного элемента подписи сообщения

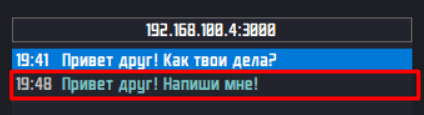


Рисунок Б.11 – Отправленное подписанное сообщение

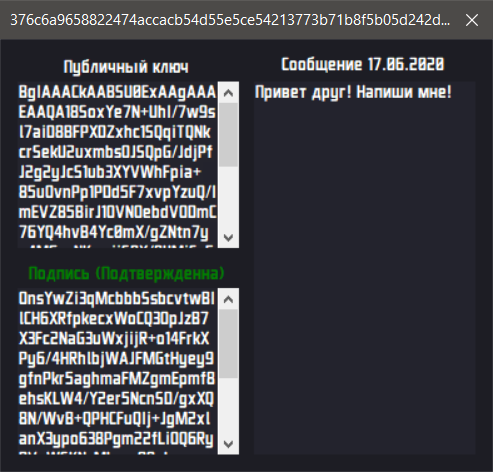


Рисунок Б.12 – Окно проверки цифровой подписи

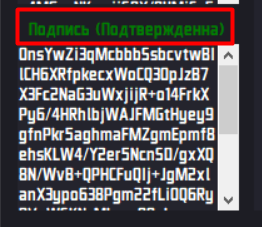


Рисунок Б.13 – Пример подтвержденной подписи

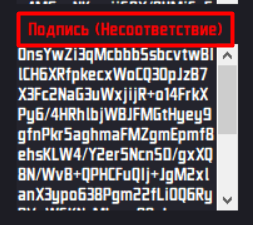


Рисунок Б.14 – Пример несоответствия подписи



Рисунок Б.15 – Элемент отправки файлов

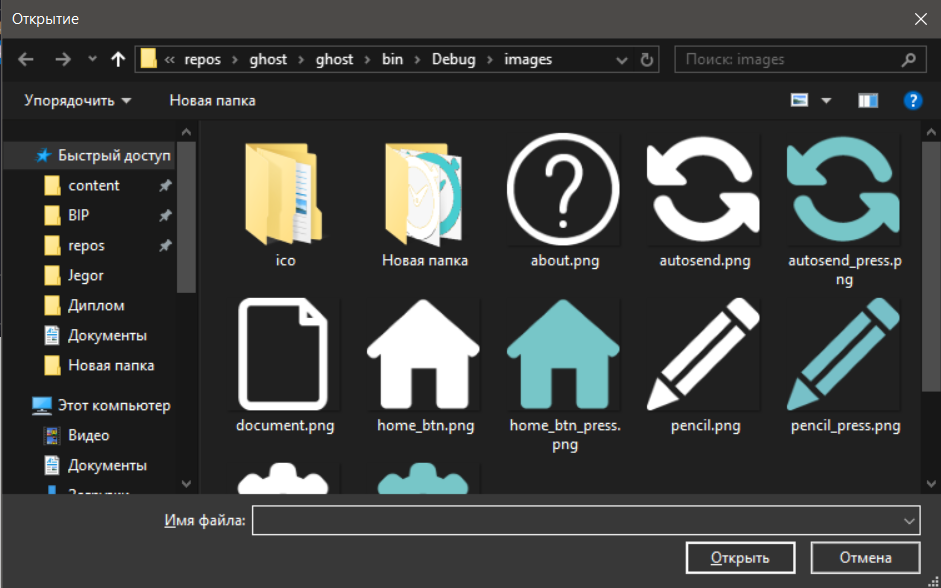


Рисунок Б.16 – Пример окна выбора файла

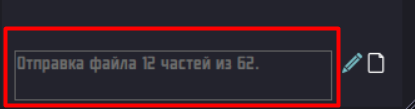


Рисунок Б.17 – Отправка файла

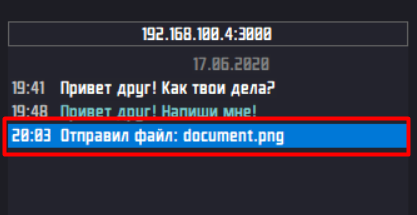


Рисунок Б.18 – Пример уведомления об отправленном файле

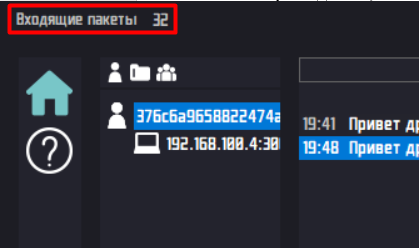


Рисунок Б.19 – Монитор входящих пакетов

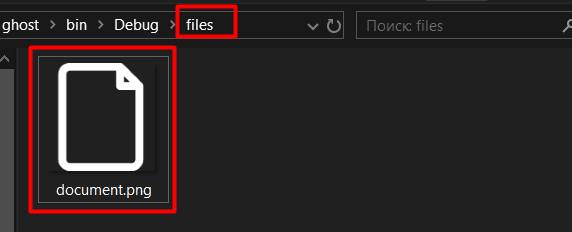


Рисунок Б.20 – Директория получаемых файлов

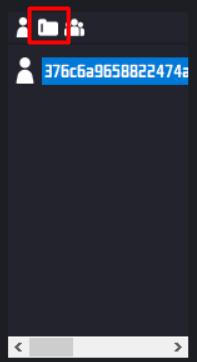


Рисунок Б.21 – Элемент создания папки

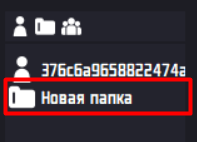
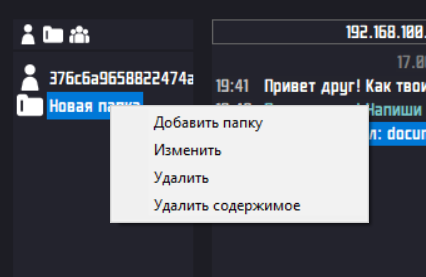


Рисунок Б.22 – Элемент папка

  
Рисунок Б.23 – Контекстное меню папки

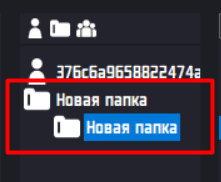


Рисунок Б.24 – Вложенная папка

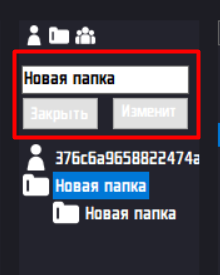


Рисунок Б.25 – Редактирование элемента папка

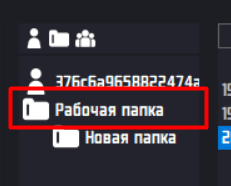


Рисунок Б.26 – Переименованная папка

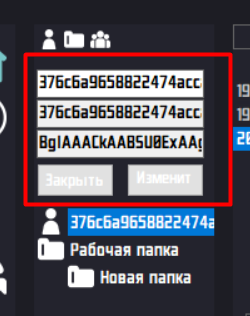


Рисунок Б.27 – Окно редактирования пользователя

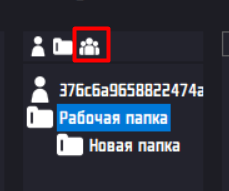


Рисунок Б.28 – Элемент создания группы

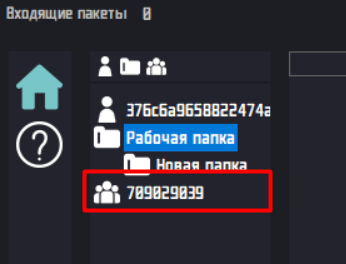


Рисунок Б.29 – Элемент группа

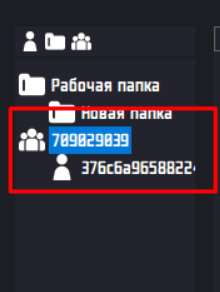


Рисунок Б.30 – Добавление пользователя в группу

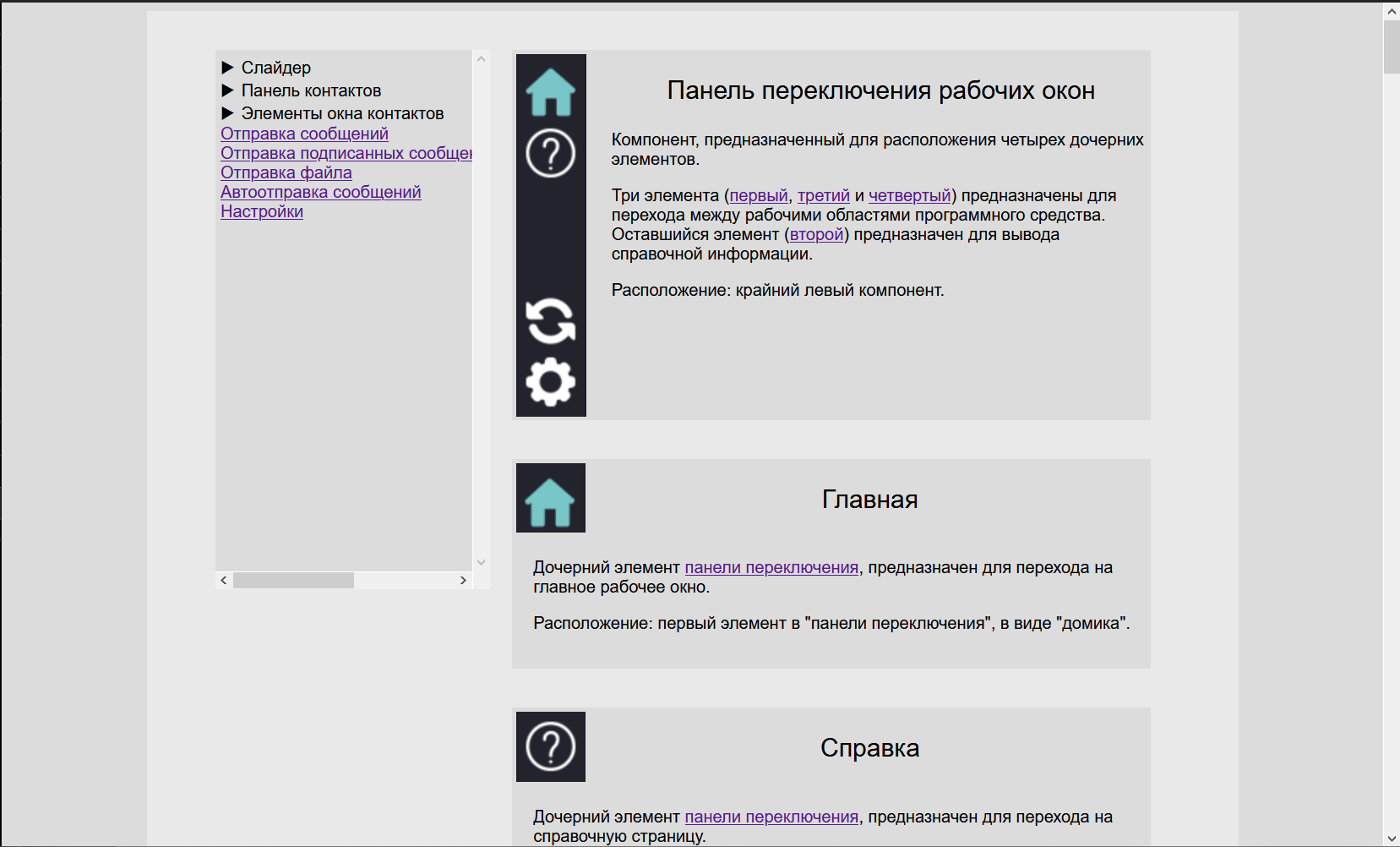


Рисунок Б.31 – Справочное меню

Приложение В

(справочное)

**Диаграммы**

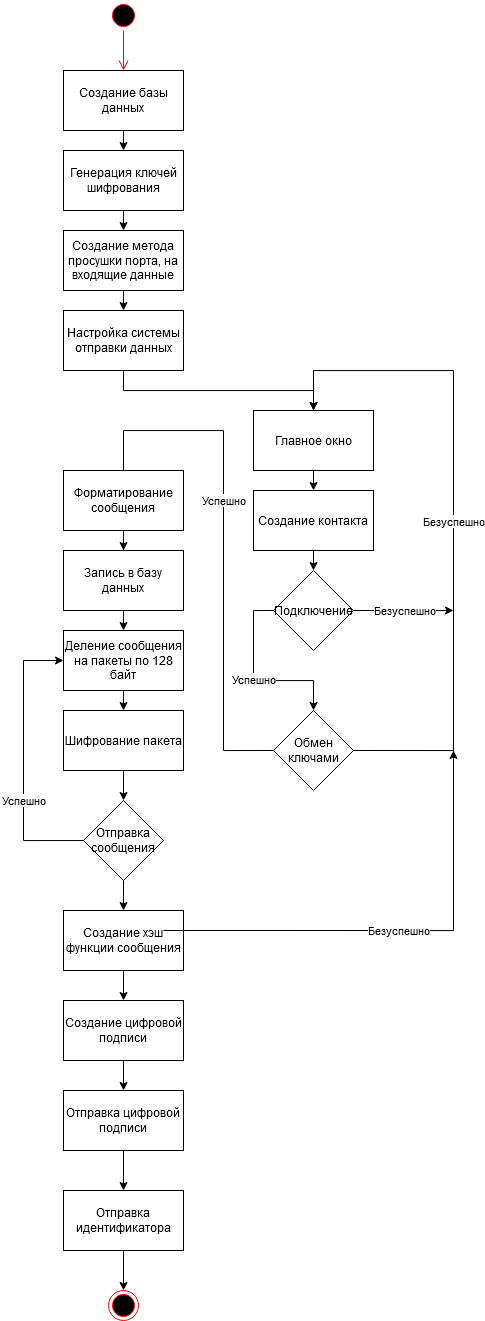
****

Рисунок В.1 – Диаграмма деятельности

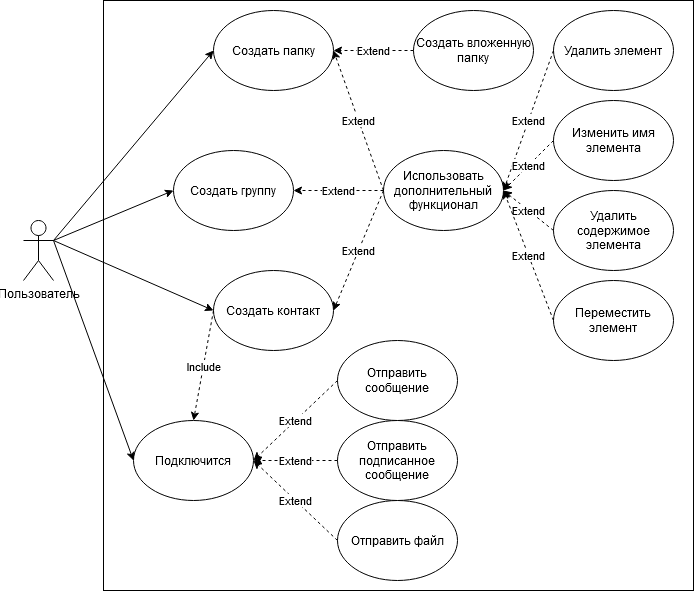


Рисунок В.2 – Диаграмма вариантов использования

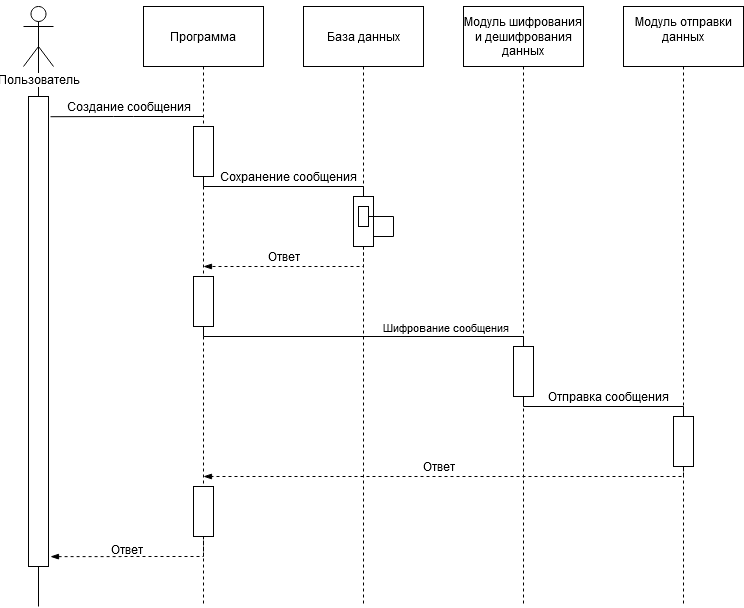


Рисунок В.3 – Диаграмма последовательности

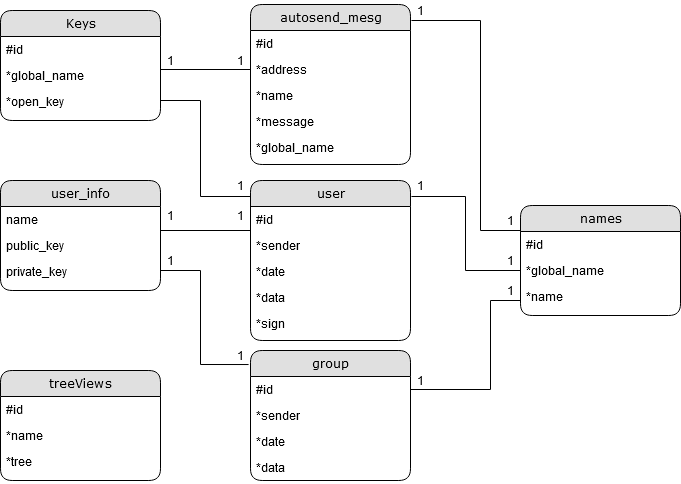


Рисунок В.4 – Диаграмма сущность-связь